Final Report Group 1

Bartosz Głocki (2188953)
Kurt Ger (2046717)
Theodore Visoiu (1791346)
Deins Kovalcuks (1847031)

June 2025

Contents

1 Introduction and Methodology

Virtual machines (VMs) are pivotal to security applications, chiefly malware analysis. Due to isolation between virtualizing "host" and the virtualized "guest", researchers can execute samples to obtain results without fear of damage to the host. Although this isolation enables safe analysis, it also introduces potential attack vectors for malicious actors. For example, a student running their exam environment in a virtual machine could exploit the host system to cheat undetected. Therefore, it is essential to develop and test mechanisms that can reliably detect whether an environment is virtualized in such critical contexts.

This project as such started off by researching various methods of defeating the VM detection mechanisms employed by the Safe Exam Browser (SEB)[17]. Upon further analysis, we discovered that VirtualMachineDetector.cs contained within SEB and its dependencies provide the functionality mentioned above. The detection algorithm is trivial, based on checking vendor and hardware identifiers and their device descriptors. As a result, we decided to focus on patching a hypervisor to spoof these identifiers.

To begin, we had to choose a hypervisor suitable for spoofing the identifiers SEB[17] relies on. We prioritized widely used virtualization platforms to ensure good community support and documentation availability. After narrowing down our options to QEMU[16] and VirtualBox, we tested them and later explored existing forks that aim to minimize detectability.

Ultimately, we chose QEMU[16] as the foundation for our modifications. Its open-source nature made it ideal for the in-depth changes required to spoof hardware and vendor identifiers. We were able to bypass SEB's[17] checks relatively early, which allowed us to shift focus to more advanced anti-VM detection tools such as PAFish[14], Al-Khaser[3], and VMAware[19]. These tools helped us better understand common detection techniques and guided further improvements.

After successfully bypassing SEB[17] with modfied QEMU[16], we attempted to defeat detection in other tools as well. Our first attempt involved CloakBox[4], a fork of VirtualBox specifically modified to evade detection. Unfortunately, we encountered issues running CloakBox[4] and could not proceed with it. We then explored CAPEv2[10], which, while being a sandbox rather than a hypervisor, proved functional and provided valuable insights.

Finally, we discovered QEMU[16] fork Hypervisor-Phantom[7], a project that aligned precisely with our objectives. We spent the remainder of the time working on and experimenting with this solution.

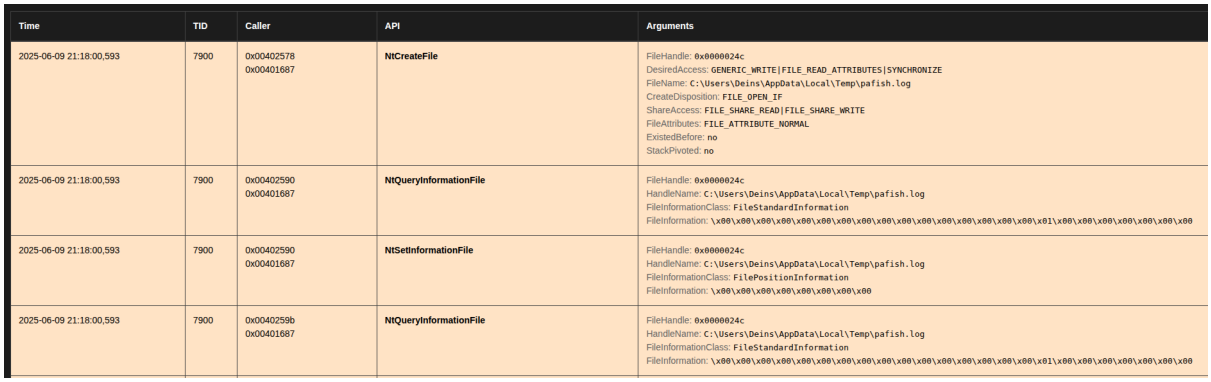
2 Cloakbox

We attempted to install CloakBox[4] on one of our Windows 11 systems, hoping to explore its features and test its functionality. However, we quickly discovered that the software was originally developed for Windows 10 by a Russian team, and that compatibility with newer operating systems had not been properly addressed. We used the VMBox version distributed by Vektor13, which is a popular community build, but ran into repeated issues when trying to create a virtual disk. The process would either fail silently or throw vague errors that were not documented anywhere in English.

Digging deeper, we realized that most of the user base and contributors were Russian, and nearly all available resources-blogs, forums, and video guides-were written in Russian as well. Even with translation tools, it was difficult to piece together a reliable installation guide or effective troubleshooting steps. Many of the blog posts were outdated. Overall, trying to get CloakBox running on Windows 11 turned into a frustrating experience due to compatibility issues.

3 CAPEv2

CAPEv2[10] is an automated malware analysis for GNU/Linux. CAPEv2 allows the user to submit malware samples in different formats, e.g., .exe, .dll, and then it handles the analysis for you. After the analysis, CAPEv2 generates an extensive report, where, among other metrics, it shows the API calls made by the sample (fig. 1).



Time	TID	Caller	API	Arguments
2025-06-09 21:18:00,593	7900	0x00402578 0x00401687	**NtCreateFile**	FileHandle: `0x0000024c` DesiredAccess: `GENERIC_WRITE|FILE_READ_ATTRIBUTES|SYNCHRONIZE` FileName: `C:\Users\Deins\AppData\Local\Temp\pafish.log` CreateDisposition: `FILE_OPEN_IF` ShareAccess: `FILE_SHARE_READ|FILE_SHARE_WRITE` FileAttributes: `FILE_ATTRIBUTE_NORMAL` ExistedBefore: `no` StackPivoted: `no`
2025-06-09 21:18:00,593	7900	0x00402590 0x00401687	**NtQueryInformationFile**	FileHandle: `0x0000024c` HandleName: `C:\Users\Deins\AppData\Local\Temp\pafish.log` FileInformationClass: `FileStandardInformation` FileInformation: `\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00`
2025-06-09 21:18:00,593	7900	0x00402590 0x00401687	**NtSetInformationFile**	FileHandle: `0x0000024c` HandleName: `C:\Users\Deins\AppData\Local\Temp\pafish.log` FileInformationClass: `FilePositionInformation` FileInformation: `\x00\x00\x00\x00\x00\x00\x00\x00`
2025-06-09 21:18:00,593	7900	0x0040259b 0x00401687	**NtQueryInformationFile**	FileHandle: `0x0000024c` HandleName: `C:\Users\Deins\AppData\Local\Temp\pafish.log` FileInformationClass: `FileStandardInformation` FileInformation: `\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00`

Figure 1: PAFish Behaviour Analysis example

CAPE was a fork of Cuckoo[5], which used *Python 2*, and later it was ported to *Python 3* and renamed to CAPEv2[10].
Note, that currently it only supports samples compiled for 32-bit architectures.

3.1 Installation

To install CAPEv2[10], we followed the guide[8]. It does not specify which exact Linux versions are supported, but we managed to install it on Ubuntu 22.04.5 LTS. Installation steps in appendix A give an overview and some key details. Please follow the installation guide[8] when doing the installation.

3.2 Software Architecture

CAPEv2's official documentation goes over the infrastructure topology[20], but it never describes the details of services and software modules used by CAPEv2[10]. [11] and [6] give an overview of the software architecture without concrete module names or APIs.

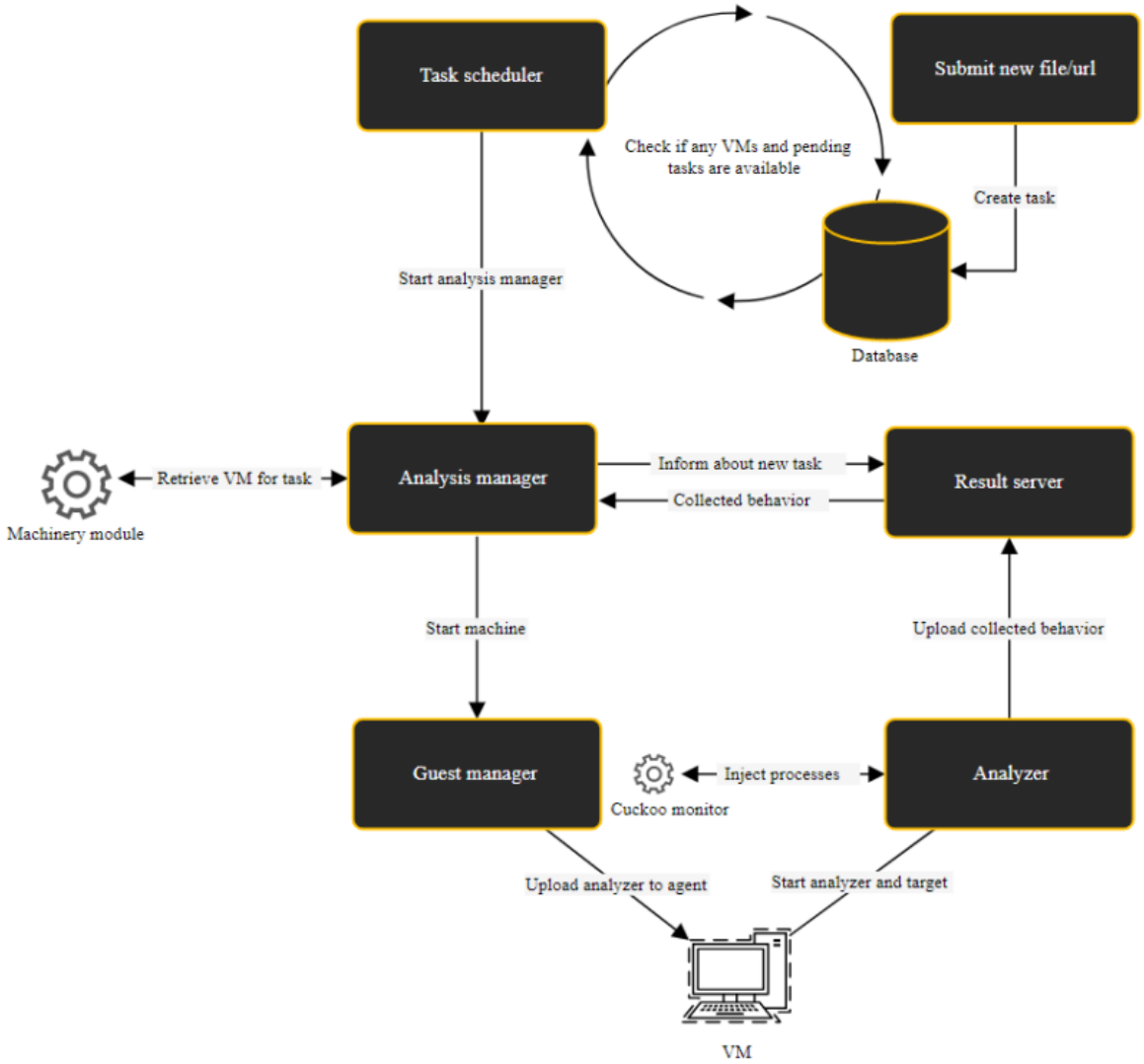


Figure 2: Analysis flow[6]

Just after the installation of CAPEv2, if you list the system services using '$systemctl list-units', you should see the following services running:

1. cape-processor.service (CAPE report processor)
2. cape-processor.service (CAPE rooter)
3. cape-web.service (CAPE WSGI app)
4. cape.service (CAPE)
5. postgresql@17-main.service

Cape-web.service runs a web server on localhost:8000 that serves as a web interface for interacting with cape.service. The sequence of events that occur internally when a malware sample is submitted is as follows:

1. The user submits the malware sample and analysis options via a form at localhost:8000/submit/.
2. cape-web.service processes it by downloading the submitted sample and adding it to the database as a pending task.
3. Meanwhile, cape.service is running a Scheduler, connected to the database, that periodically checks for new task entries.

4. When a new entry is detected because of our submission, a new instance of AnalysisManager is created.
5. AnalysisManager, in turn, creates the configuration file for the analysis and an instance of GuestManager, specifying the necessary vm configuration to run the sample, like the platform, architecture, vm name etc.
6. Once that is done, AnalysisManager calls start_analysis on GuestManager, which runs the vm and uploads the malware sample and files and directories inside analyzer/windows folder to the vm via POST requests to vm_ip/extract and vm_ip/store, respectively. Finally, it remotely executes analyzer.py using a POST request to vm_ip/execpy.
7. Meanwhile, the VM is running agent.py, which processes the incoming HTTP requests.
8. When agent.py receives the POST request at vm_ip/execpy, it executes analyzer.py.
9. analyzer.py imports and initializes auxhiliary modules (like screenshots and human) in new threads where the former periodically takes screenshots during the execution and the later moves the cursor to resemble human-like behaviour.
10. Next, analyzer invokes an analysis package with the malware sample. The analysis package is chosen depending on the type of the malware sample that is analyzed, for example, the analysis of .exe malware is done using exe.py analysis package.
11. exe.py creates a suspended process loaded with the malware sample and runs loader.exe.
12. loader.exe edits the Import Table of the malware sample process, adding an entry for capemon.dll.
13. loader.exe resumes the execution of malware sample process.
14. Windows Loader loads capemon.dll into the memory of the process and capemon.DllMain is executed.
15. capemon.dll registers a handler for debug exceptions using AddVectoredExceptionHandler.
16. Then, capemon.dll compiles yara rules and scans the image for matches with the rules and calls a callback on a match. (Debug registers can also be configured here.)
17. Next, it installs hooks for the API calls:
    (a) Gets the address of the API.
    (b) Saves the original function.
    (c) Overwrites the instruction at that address with a jump to the custom hook defined in capemon.dll.
18. After that, it installs a callback that is called on system calls.
19. Lastly, it sets breakpoints using the debug registers if they were supplied in analysis options in step 1 or were set during the yara rule scan.

The full (pseudo)execution trace can be found in the appendix B.

3.3 VM detection results

3.3.1 PAFish

```
[pafish] Start
[pafish] Windows version: 6.2 build 9200 (WoW64)
[pafish] CPU: GenuineIntel (HV: Microsoft Hv) 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
[pafish] CPU VM traced by checking the difference between CPU timestamp counters (rdtsc) forcing VM exit
[pafish] CPU VM traced by checking hypervisor bit in cpuid feature bits
[pafish] CPU VM traced by checking cpuid hypervisor vendor for known VM vendors
[pafish] Sandbox traced by missing mouse movement or supernatural speed
[pafish] Sandbox traced by missing or implausible dialog confirmation
[pafish] End
```

Figure 3: pafish.log contents with automated interaction enabled

3.3.2 Al-khaser (log.txt) output

'1' means that VM was detected.

```
[Sat Jun 21 02:59:02 2025] [*] TLS process attach callback  -> 0
[Sat Jun 21 02:59:02 2025] [*] TLS thread attach callback  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking IsDebuggerPresent API  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking PEB.BeingDebugged  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking CheckRemoteDebuggerPresent API  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking PEB.NtGlobalFlag  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking ProcessHeap.Flags  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking ProcessHeap.ForceFlags  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking Low Fragmentation Heap -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking NtQueryInformationProcess with ProcessDebugPort  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking NtQueryInformationProcess with ProcessDebugFlags  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking NtQueryInformationProcess with ProcessDebugObject  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking WudfIsAnyDebuggerPresent API  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking WudfIsKernelDebuggerPresent API  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking WudfIsUserDebuggerPresent API  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking NtSetInformationThread with ThreadHideFromDebugger  -> 0
[Sat Jun 21 02:59:02 2025] [*] Checking CloseHandle with an invalide handle  -> 0
[Sat Jun 21 02:59:03 2025] [*] Checking NtSystemDebugControl -> 1
[Sat Jun 21 02:59:03 2025] [*] Checking UnhandledExcepFilterTest  -> 0
[Sat Jun 21 02:59:03 2025] [*] Checking OutputDebugString  -> 0
[Sat Jun 21 02:59:03 2025] [*] Checking Hardware Breakpoints  -> 0
[Sat Jun 21 02:59:03 2025] [*] Checking Software Breakpoints  -> 0
[Sat Jun 21 02:59:03 2025] [*] Checking Interupt 0x2d  -> 0
[Sat Jun 21 02:59:03 2025] [*] Checking Interupt 1  -> 0
[Sat Jun 21 02:59:03 2025] [*] Checking trap flag -> 0
[Sat Jun 21 02:59:03 2025] [*] Checking Memory Breakpoints PAGE GUARD  -> 0
[Sat Jun 21 02:59:03 2025] [*] Checking If Parent Process is explorer.exe  -> 1
[Sat Jun 21 02:59:03 2025] [*] Checking SeDebugPrivilege  -> 1
[Sat Jun 21 02:59:03 2025] [*] Checking NtQueryObject with ObjectTypeInformation  -> 0
[Sat Jun 21 02:59:03 2025] [*] Checking NtQueryObject with ObjectAllTypesInformation  -> 0
[Sat Jun 21 02:59:04 2025] [*] Checking NtYieldExecution  -> 0
[Sat Jun 21 02:59:04 2025] [*] Checking CloseHandle protected handle trick  -> 0
[Sat Jun 21 02:59:04 2025] [*] Checking NtQuerySystemInformation with SystemKernelDebuggerInformation  ->
    0
[Sat Jun 21 02:59:04 2025] [*] Checking SharedUserData->KdDebuggerEnabled  -> 0
[Sat Jun 21 02:59:04 2025] [*] Checking if process is in a job  -> 1
[Sat Jun 21 02:59:04 2025] [*] Checking VirtualAlloc write watch (buffer only)  -> 0
[Sat Jun 21 02:59:04 2025] [*] Checking VirtualAlloc write watch (API calls)  -> 0
[Sat Jun 21 02:59:04 2025] [*] Checking VirtualAlloc write watch (IsDebuggerPresent)  -> 1
[Sat Jun 21 02:59:04 2025] [*] Checking VirtualAlloc write watch (code write)  -> 0
[Sat Jun 21 02:59:04 2025] [*] Checking for page exception breakpoints  -> 0
[Sat Jun 21 02:59:04 2025] [*] Checking for API hooks outside module bounds  -> 0
[Sat Jun 21 02:59:05 2025] [*] Enumerating modules with EnumProcessModulesEx [32-bit]  -> 0
[Sat Jun 21 02:59:05 2025] [*] Enumerating modules with EnumProcessModulesEx [64-bit]  -> 0
[Sat Jun 21 02:59:05 2025] [*] Enumerating modules with EnumProcessModulesEx [ALL]  -> 0
[Sat Jun 21 02:59:05 2025] [*] Enumerating modules with ToolHelp32  -> 0
[Sat Jun 21 02:59:05 2025] [*] Enumerating the process LDR via LdrEnumerateLoadedModules  -> 0
[Sat Jun 21 02:59:05 2025] [*] Enumerating the process LDR directly  -> 0
[Sat Jun 21 02:59:05 2025] [*] Walking process memory with GetModuleInformation  -> 1
```

```
[Sat Jun 21 02:59:05 2025] [*] Walking process memory for hidden modules  -> 1
[Sat Jun 21 02:59:05 2025] [*] Walking process memory for .NET module structures  -> 0
[Sat Jun 21 02:59:05 2025] [*] Checking if process loaded modules contains: avghookx.dll  -> 0
[Sat Jun 21 02:59:05 2025] [*] Checking if process loaded modules contains: avghooka.dll  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process loaded modules contains: snxhk.dll  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process loaded modules contains: sbiedll.dll  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process loaded modules contains: dbghelp.dll  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process loaded modules contains: api_log.dll  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process loaded modules contains: dir_watch.dll  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process loaded modules contains: pstorec.dll  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process loaded modules contains: vmcheck.dll  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process loaded modules contains: wpespy.dll  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process loaded modules contains: cmdvrt64.dll  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process loaded modules contains: cmdvrt32.dll  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process file name contains: sample.exe  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process file name contains: bot.exe  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process file name contains: sandbox.exe  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process file name contains: malware.exe  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process file name contains: test.exe  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process file name contains: klavme.exe  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process file name contains: myapp.exe  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process file name contains: testapp.exe  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if process file name looks like a hash: al-khaser_x86  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : CurrentUser  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : Sandbox  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : Emily  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : HAPUBWS  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : Hong Lee  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : IT-ADMIN  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : Johnson  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : Miller  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : milozs  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : Peter Wilson  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : timmy  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : user  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : sand box  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : malware  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : maltest  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : test user  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : virus  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if username matches : John Doe  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if hostname matches : SANDBOX  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if hostname matches : 7SILVIA  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if hostname matches : HANSPETER-PC  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if hostname matches : JOHN-PC  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if hostname matches : MUELLER-PC  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if hostname matches : WIN7-TRAPS  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if hostname matches : FORTINET  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking if hostname matches : TEQUILABOOMBOOM  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking whether username is 'Wilber' and NetBIOS name starts with 'SC' or
    'SW'  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking whether username is 'admin' and NetBIOS name is 'SystemIT'  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking whether username is 'admin' and DNS hostname is 'KLONE_X64-PC'  ->
    0
[Sat Jun 21 02:59:06 2025] [*] Checking whether username is 'John' and two sandbox files exist  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking whether four known sandbox 'email' file paths exist  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking whether three known sandbox 'foobar' files exist  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking Number of processors in machine  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking Interupt Descriptor Table location  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking Local Descriptor Table location  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking Global Descriptor Table location  -> 0
[Sat Jun 21 02:59:06 2025] [*] Checking Store Task Register  -> 0
[Sat Jun 21 02:59:17 2025] [*] Checking Number of cores in machine using WMI  -> 1
[Sat Jun 21 02:59:17 2025] [*] Checking hard disk size using WMI  -> 0
[Sat Jun 21 02:59:17 2025] [*] Checking hard disk size using DeviceIoControl  -> 0
[Sat Jun 21 02:59:17 2025] [*] Checking SetupDi_diskdrive  -> 0
[Sat Jun 21 02:59:22 2025] [*] Checking mouse movement  -> 0
[Sat Jun 21 02:59:22 2025] [*] Checking lack of user input  -> 0
[Sat Jun 21 02:59:22 2025] [*] Checking memory space using GlobalMemoryStatusEx  -> 0
[Sat Jun 21 02:59:22 2025] [*] Checking disk size using GetDiskFreeSpaceEx  -> 0
[Sat Jun 21 02:59:22 2025] [*] Checking if CPU hypervisor field is set using cpuid(0x1) -> 1
[Sat Jun 21 02:59:22 2025] [*] Checking hypervisor vendor using cpuid(0x40000000) -> 1
[Sat Jun 21 03:00:22 2025] [*] Check if time has been accelerated  -> 0
```

```
[Sat Jun 21 03:00:22 2025] [*] VM Driver Services   -> 0
[Sat Jun 21 03:00:22 2025] [*] Checking SerialNumber from BIOS using WMI  -> 0
[Sat Jun 21 03:00:22 2025] [*] Checking Model from ComputerSystem using WMI  -> 0
[Sat Jun 21 03:00:22 2025] [*] Checking Manufacturer from ComputerSystem using WMI  -> 0
[Sat Jun 21 03:00:23 2025] [*] Checking Current Temperature using WMI  -> 1
[Sat Jun 21 03:00:27 2025] [*] Checking ProcessId using WMI  -> 0
[Sat Jun 21 03:00:27 2025] [*] Checking power capabilities  -> 1
[Sat Jun 21 03:00:27 2025] [*] Checking CPU fan using WMI  -> 1
[Sat Jun 21 03:00:27 2025] [*] Checking NtQueryLicenseValue with Kernel-VMDetection-Private  -> 0
[Sat Jun 21 03:00:27 2025] [*] Checking Win32_CacheMemory with WMI  -> 1
[Sat Jun 21 03:00:28 2025] [*] Checking Win32_PhysicalMemory with WMI  -> 0
[Sat Jun 21 03:00:28 2025] [*] Checking Win32_MemoryDevice with WMI  -> 0
[Sat Jun 21 03:00:28 2025] [*] Checking Win32_MemoryArray with WMI  -> 0
[Sat Jun 21 03:00:28 2025] [*] Checking Win32_VoltageProbe with WMI  -> 1
[Sat Jun 21 03:00:28 2025] [*] Checking Win32_PortConnector with WMI  -> 1
[Sat Jun 21 03:00:28 2025] [*] Checking Win32_SMBIOSMemory with WMI  -> 0
[Sat Jun 21 03:00:44 2025] [*] Checking ThermalZoneInfo performance counters with WMI  -> 1
[Sat Jun 21 03:00:44 2025] [*] Checking CIM_Memory with WMI  -> 1
[Sat Jun 21 03:00:44 2025] [*] Checking CIM_Sensor with WMI  -> 1
[Sat Jun 21 03:00:44 2025] [*] Checking CIM_NumericSensor with WMI  -> 1
[Sat Jun 21 03:00:44 2025] [*] Checking CIM_TemperatureSensor with WMI  -> 1
[Sat Jun 21 03:00:45 2025] [*] Checking CIM_VoltageSensor with WMI  -> 1
[Sat Jun 21 03:00:47 2025] [*] Checking CIM_PhysicalConnector with WMI  -> 1
[Sat Jun 21 03:00:49 2025] [*] Checking CIM_Slot with WMI  -> 1
[Sat Jun 21 03:00:49 2025] [*] Checking if Windows is Genuine  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking Services\Disk\Enum entries for VM strings  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking Enum\IDE and Enum\SCSI entries for VM strings  -> 1
[Sat Jun 21 03:00:49 2025] [*] Checking SMBIOS tables   -> 1
[Sat Jun 21 03:00:49 2025] [*] Checking reg key HARDWARE\Description\System - Identifier is set to VBOX ->
     0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key HARDWARE\Description\System - SystemBiosVersion is set to
    VBOX -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key HARDWARE\Description\System - VideoBiosVersion is set to
    VIRTUALBOX -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key HARDWARE\Description\System - SystemBiosDate is set to
    06/23/99 -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking VirtualBox Guest Additions directory  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\drivers\VBoxMouse.sys  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\drivers\VBoxGuest.sys  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\drivers\VBoxSF.sys  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\drivers\VBoxVideo.sys  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxdisp.dll  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxhook.dll  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxmrxnp.dll  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxogl.dll  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxoglarrayspu.dll  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxoglcrutil.dll  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxoglerrorspu.dll  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxoglfeedbackspu.dll  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxoglpackspu.dll  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxoglpassthroughspu.dll  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxservice.exe  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\vboxtray.exe  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking file C:\Windows\System32\VBoxControl.exe  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key HARDWARE\ACPI\DSDT\VBOX__  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key HARDWARE\ACPI\FADT\VBOX__  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key HARDWARE\ACPI\RSDT\VBOX__  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key SOFTWARE\Oracle\VirtualBox Guest Additions  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\VBoxGuest  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\VBoxMouse  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\VBoxService  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\VBoxSF  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\VBoxVideo  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking Mac Address start with 08:00:27  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking MAC address (Hybrid Analysis)  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking device \\.\VBoxMiniRdrDN  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking device \\.\VBoxGuest  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking device \\.\pipe\VBoxMiniRdDN  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking device \\.\VBoxTrayIPC  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking device \\.\pipe\VBoxTrayIPC  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking VBoxTrayToolWndClass / VBoxTrayToolWnd  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking VirtualBox Shared Folders network provider  -> 0
[Sat Jun 21 03:00:49 2025] [*] Checking VirtualBox process vboxservice.exe  -> 0
```

```
[Sat Jun 21 03:00:49 2025] [*] Checking VirtualBox process vboxtray.exe  -> 0
[Sat Jun 21 03:00:55 2025] [*] Checking Win32_PnPDevice DeviceId from WMI for VBox PCI device  -> 0
[Sat Jun 21 03:00:56 2025] [*] Checking Win32_PnPDevice Name from WMI for VBox controller hardware  -> 0
[Sat Jun 21 03:00:59 2025] [*] Checking Win32_PnPDevice Name from WMI for VBOX names  -> 0
[Sat Jun 21 03:01:01 2025] [*] Checking Win32_Bus from WMI  -> 0
[Sat Jun 21 03:01:02 2025] [*] Checking Win32_BaseBoard from WMI  -> 0
[Sat Jun 21 03:01:04 2025] [*] Checking MAC address from WMI  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking NTEventLog from WMI  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking SMBIOS firmware   -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking ACPI tables   -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\Target Id
    0\Logical Unit Id 0 -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key HARDWARE\DEVICEMAP\Scsi\Scsi Port 1\Scsi Bus 0\Target Id
    0\Logical Unit Id 0 -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key HARDWARE\DEVICEMAP\Scsi\Scsi Port 2\Scsi Bus 0\Target Id
    0\Logical Unit Id 0 -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SYSTEM\ControlSet001\Control\SystemInformation -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SYSTEM\ControlSet001\Control\SystemInformation -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SOFTWARE\VMware, Inc.\VMware Tools  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmnet.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmmouse.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmusb.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vm3dmp.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmci.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmhgfs.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmmemctl.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmx86.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmrawdsk.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmusbmouse.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmkdb.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmnetuserif.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vmnetadapter.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking MAC starting with 00:05:69 -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking MAC starting with 00:0c:29 -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking MAC starting with 00:1C:14 -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking MAC starting with 00:50:56 -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking VMWare network adapter name  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking device \\.\HGFS  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking device \\.\vmci  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking VMWare directory  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking SMBIOS firmware   -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking ACPI tables   -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking Virtual PC processes VMSrvc.exe  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking Virtual PC processes VMUSrvc.exe  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SOFTWARE\Microsoft\Virtual Machine\Guest\Parameters  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\Target Id
    0\Logical Unit Id 0  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key HARDWARE\Description\System  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking qemu processes qemu-ga.exe  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking qemu processes vdagent.exe  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking qemu processes vdservice.exe  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking QEMU directory C:\Program Files\qemu-ga  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking QEMU directory C:\Program Files\SPICE Guest Tools  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking SMBIOS firmware   -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking ACPI tables   -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking Citrix Xen process xenservice.exe -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking Mac Address start with 08:16:3E  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\balloon.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\netkvm.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\pvpanic.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\viofs.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\viogpudo.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vioinput.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\viorng.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vioscsi.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\vioser.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking file C:\Windows\System32\drivers\viostor.sys  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\vioscsi  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\viostor  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\VirtIO-FS Service  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\VirtioSerial  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\BALLOON  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\BalloonService  -> 0
```

```
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SYSTEM\ControlSet001\Services\netkvm  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking KVM virio directory  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking Wine via dll exports  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking reg key SOFTWARE\Wine  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking Parallels processes: prl_cc.exe -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking Parallels processes: prl_tools.exe -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking Mac Address start with 00:1C:42  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking for Hyper-V driver objects  -> 0
[Sat Jun 21 03:01:06 2025] [*] Checking for Hyper-V global objects  -> 1
```

3.3.3 VMAware

We were unable to gather the results from VMAware[19] since, during analysis, it opened a terminal window for a split second and then immediately closed it without dropping any files. The same behaviour was observed when running 32 and 64 bit version of VMAware outside the sandbox environment, in the same VM. This suggests that the issue is not related to CAPEv2.

3.4 Results

PAFish detects that something is off using the rdtsc instruction and forcing the VM to exit. Moreover, the fact that PAFish was run in the VM is also given away by the hypervisor bit and vendor ID, returned by cpuid instruction. Some of the mouse activity checks fail despite automated interaction being enabled.

Al-khaser[3] detects the presence of a debugger with *NtSystemDebugControl* and *SeDebugPrivilege* checks. *GetModuleInformation* check fails because the injected dll has a suspicious path. It is not entirely clear what causes the check to fail for hidden modules because the injected dll should appear legitimate as it was injected via IAT. Windows registry has entries with hypervisor clues. It also detected that SMBIOS tables are too short.

3.5 Discussion

You cannot intercept cpu instructions like rdtsc or cpuid using API hooks. A solution to this could be to set hardware breakpoints at the relevant instructions and manipulate the control flow or the program state to avoid the detection.

Some checks like NtSystemDebugControl can be circumvented by hooking into the API call and returning the result indicating that the debugger is inactive.

Windows registry and SMBIOS table detection can be avoided by hypervisor patching.

We made an attempt to explore the interactive mode of CAPEv2[10] with the hope that it might allow us to more easily execute interactive programs like Safe Exam Browser[17]. To this end, we followed the installation instructions [9], where we installed guacamole, added VNC to our VM and set up nginx. When we tried to connect to the VM via the browser interface, after submitting a sample with interactive mode enabled, we could not interact with the VM. The browser page was blank, only showing the 'end session' button.

3.6 Conclusion

The malware samples executed by CAPEv2[10] can without much effort detect the virtualization.

To avoid detection, several paths could be explored:

1. The custom QEMU Fork or Hypervisor phantom could potentially be used together with CAPEv2[10]. Then, QEMU[16] patches would provide additional stealthiness while CAPEv2[10] would provide the tooling for analyzing the malware samples.
2. Instructions that indicate virtualized environment could be handled by the debugger, and if that does not work, by a hypervisor patch.
3. API calls that can lead to detection could be intercepted by defining hooks and a safe value could be returned instead.

Moreover, as a further work, it could be explored:

1. Why VMAware[19] is not running properly on the VM.
2. Why hidden modules check in Al-khaser[3] detects virtualization.
3. Why interactive desktop mode in CAPEv2[10] is not working.

4 Custom QEMU Fork

One of our implementations is based on a fork of QEMU 10, the latest version at the time of writing. We replaced all references to "QEMU" in the codebase, replacing both strings and the associated vendor and hardware identifiers.

For configuring and running the virtual machine, we used libvirt, an open-source toolkit for managing virtualization platforms, including QEMU. This allowed us to try different hardware configurations more easily than using QEMU command-line flags, which tend to get quite verbose when the number of overrides grows. Our configuration builds on top of the modifications done to QEMU by providing fake SMBIOS information, disabling CPU feature flags such as the virtualization flag, and using our real display's EDID information (previously dumped using `dump-edid`).

As part of the trials, we tested multiple video drivers with following results:

- **QXL** - Could not find a way to spoof vendor ID `0x1b36`
- **VGA** - Passes SEB VM checks but fails to show up as a display (display count = 0), hence SEB refuses to start using most exam configurations
- **VIRTIO** - Same as VGA
- **BOCHS** - Passes VM checks; passes display count tests; can set custom (realistic) resolution
- **RAMFB** - Not tested

These changes have allowed us to successfully spoof the Safe Exam Browser, enabling us to take an exam while having access to the entire Linux host operating system for "additional exam assistance".

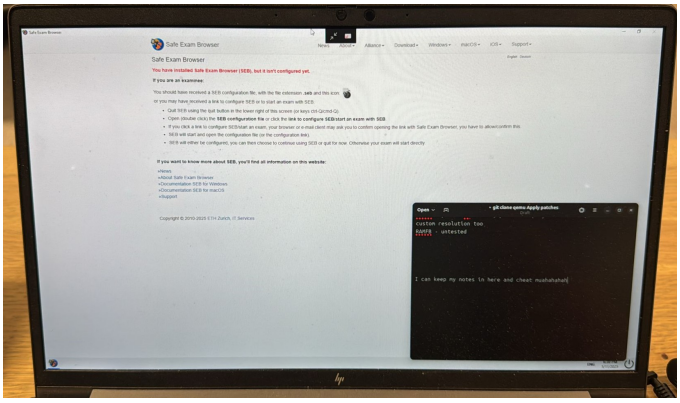


Figure 4: Proof of the Safe Exam Browser bypass

We then continued our investigation by analyzing how effective our detection evasion mechanisms were.

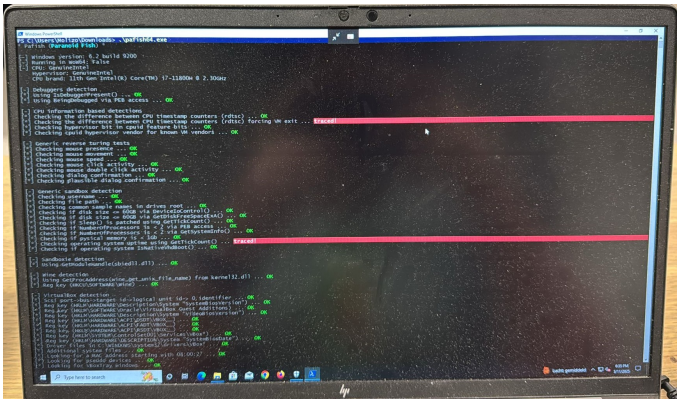


Figure 5: PAFish results (part 1)

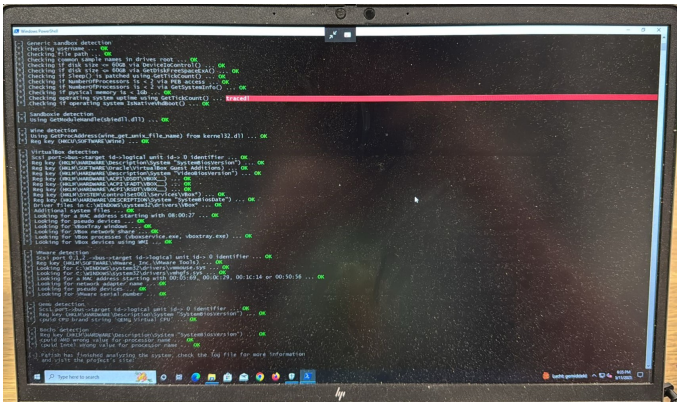


Figure 6: PAFish results (part 2)

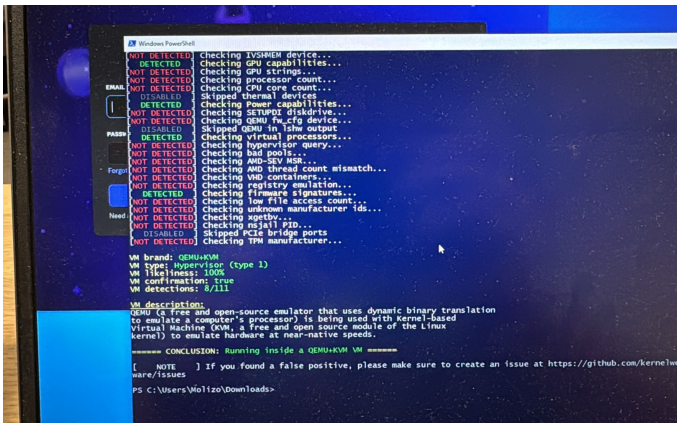


Figure 7: VMAware results (partial)

As depicted in fig. 5 and fig. 6, our VM passes all checks except the `RDTSC` exit-based VM one. This check uses the `RDTSC` CPU instruction, which performs **VM-exit** and then returns the number of cycles that have been executed since the CPU was initially turned on. In virtualized environments, the returned number is noticeably higher than on bare metal, as besides VM the host OS also increments the same counter. Our early research shows that a patch to the host operating system's KVM module will be needed. The patch will have to intercept all `RDTSC` instructions from the VM and return a spoofed CPU cycle count, within the realistic range for a bare metal operating system. The `GetTickCount()` check is not reliable and can be circumvented by letting the VM run for ~15 minutes before running PAFish (pictures were taken too early).

In fig. 7, one of the failing checks is again related to `RDTSC` and `GetTickCount()`. We expect that the other checks can be compromised in the following way:

- **GPU capabilities** - Online forums recommend that we pass a GPU through, but this is not practical for a laptop-based examination environment. Furthermore, those checks could be trivially improved, as most laptops are equipped with integrated graphics. Not passing integrated graphics through to a VM but passing a dedicated GPU can be easily detected using the CPU's identifiers, as laptop CPUs are typically equipped with integrated graphics for driving the laptop's display. Passing both the CPU's and GPU the dedicated GPU is not practical on a laptop platform, as the host OS would have no way of displaying anything on the screen, including the virtualized environment's own display.
- **Power capabilities** - One approach could involve using a tiny USB-dongle-like microcontroller reporting fake battery percentages from a simulated uninterruptible power supply (UPS). This USB dongle could then be passed through to the VM to pass this check. Alternatively, a spoofed battery implemented with ACPI tables could also be used.
- **Virtual processors** - This check is typically tricked by having an improper virtualized CPU count, which does not align with the real CPU's SKU. One bypass approach could involve passing through all host CPU cores to the guest, although this could trigger system instability and reduced responsiveness on the host.
- **Firmware signatures** - This is probably caused by Al-Khaser detecting unique thumbprints in the virtualized Seabios BIOS ROM. A possible approach could involve creating a fork of Seabios or OVMF, removing all traces of them using a similar approach to our QEMU fork, and then using this new BIOS ROM for our VM. The new ROM would then inherently have a new fingerprint that would no longer match any of the fingerprints specified in Al-Khaser's code. We will return to this in the next chapter.

Configuration instructions and notes on replicating our virtualization environment can be found in appendix C.1. The code for our custom build of QEMU is available upon request, as the repository is currently private.

5 Hypervisor-Phantom

In light of the shortcomings of our custom QEMU fork, together with the issues raised by Ubuntu's AppArmor and file permissions when launching the custom-built binary, we decided to investigate the approaches taken by automated scripts available on the Internet. One such script is Hypervisor-Phantom (H-P)[7], developed by Scrut1ny. H-P[7] builds on our approach with the custom QEMU build and adds additional types of spoofing, such as:

- `RDTSC` handling: H-P[7] patches the host's kernel to intercept and return spoofed (artificially lowered) `RDTSC` value to the guest OS, fixing VM-exit timing anomalies. It must be adapted depending on the host's CPU clock speed. For example, at a 3.2Ghz clock speed, the `RDTSC` value had to be divided by 16, at 3.5Ghz by 20 and at 4.5Ghz by 40.
- Support for GPU passthrough: While testing our own QEMU build we discovered that passing through host's own GPU contributed significantly to the host system's instability. This was the case because of poor IOMMU groupings on the TU/e-provided HP ZBook Power G8. H-P fixes this issue by providing scripts to patch the host's bootloader and kernel to aid with GPU passthrough.
- Custom UEFI BIOS for QEMU: H-P builds and installs a custom EDK2/OVMF loader with all references to QEMU/virtualized environments patched out.
- ACPI tables for a fake battery: H-P provides a custom ACPI table that emulates a fake battery, spoofing one of the tests VMAware and Al-Khaser employ.

At the start of the project, Hypervisor-Phantom[7] supported versions of QEMU up to v9.2.4. Noticing this, we tried contributing to the project by upgrading its patches to QEMU 10, using the patches we previously implemented. Unfortunately, during development, H-P[7] published updated QEMU 10 patches, so we had to abandon this line of research.

Next, we pivoted to investigating the QEMU 10 version of H-P[7]. With a few patches to the auto-generated libvirt XML we were able to replicate their test scores across all three test suites:

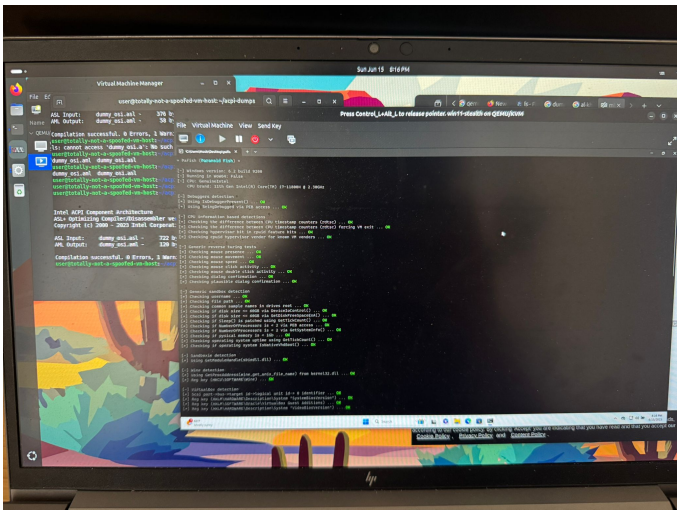


Figure 8: PAFish results (part 1)

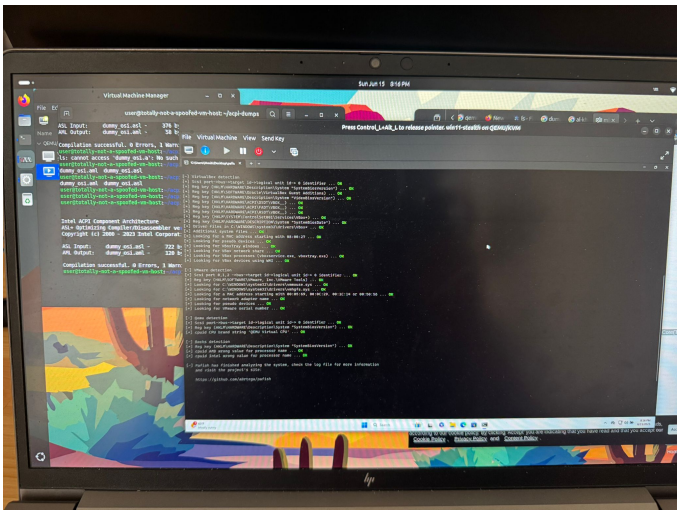


Figure 9: PAFish results (part 2)

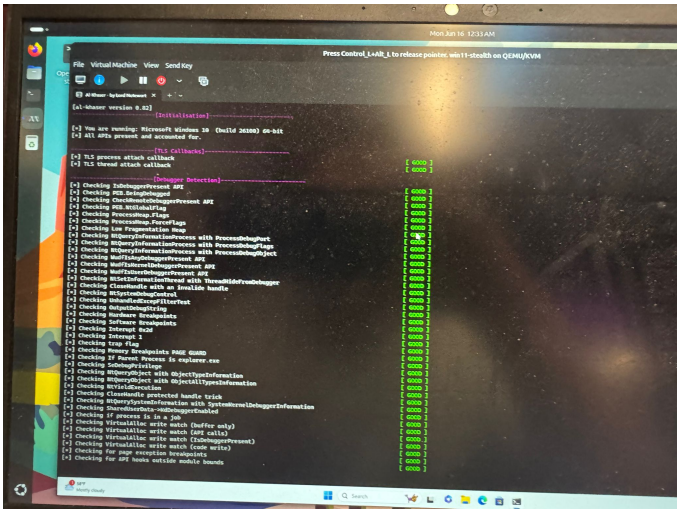


Figure 10: Al-Khaser results (part 1)

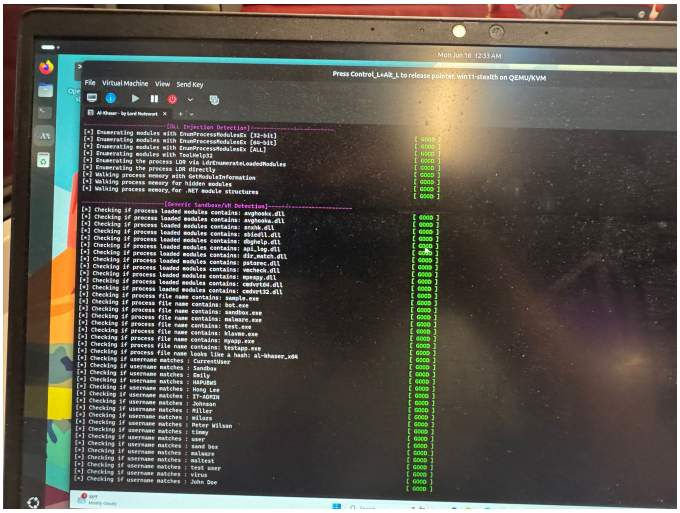


Figure 11: Al-Khaser results (part 2)

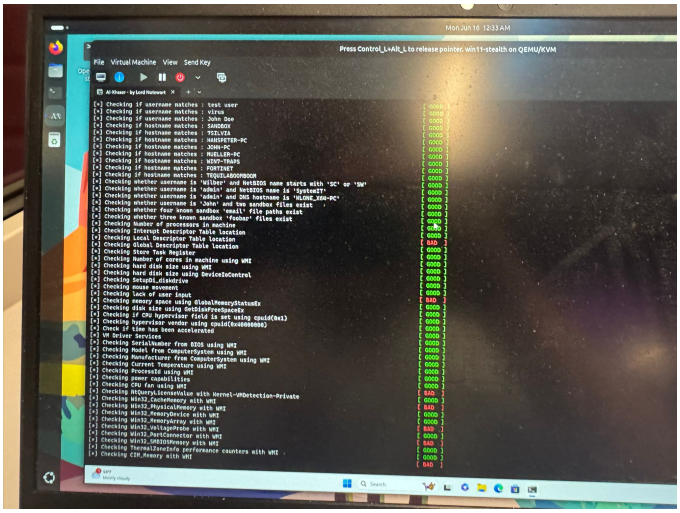


Figure 12: Al-Khaser results (part 3)

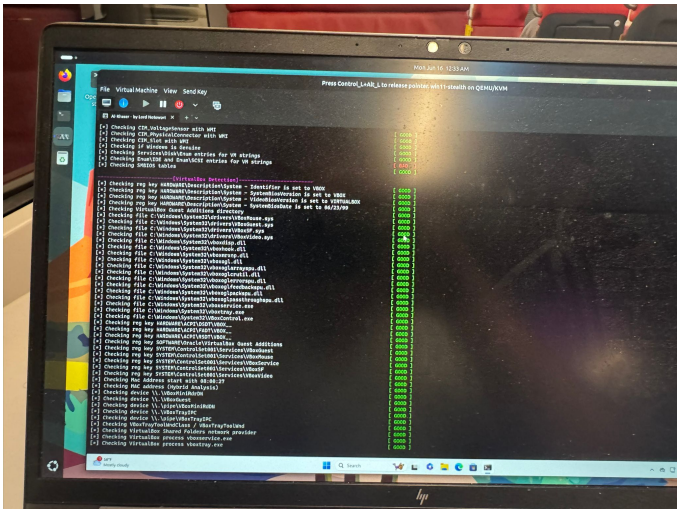


Figure 13: Al-Khaser results (part 4)

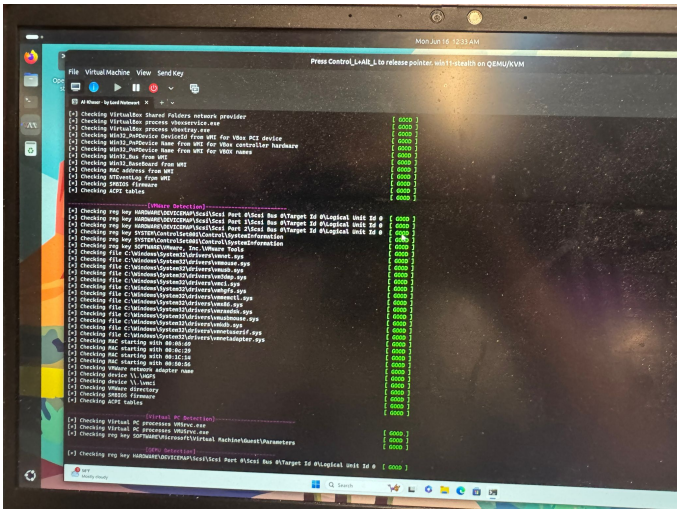


Figure 14: Al-Khaser results (part 5)

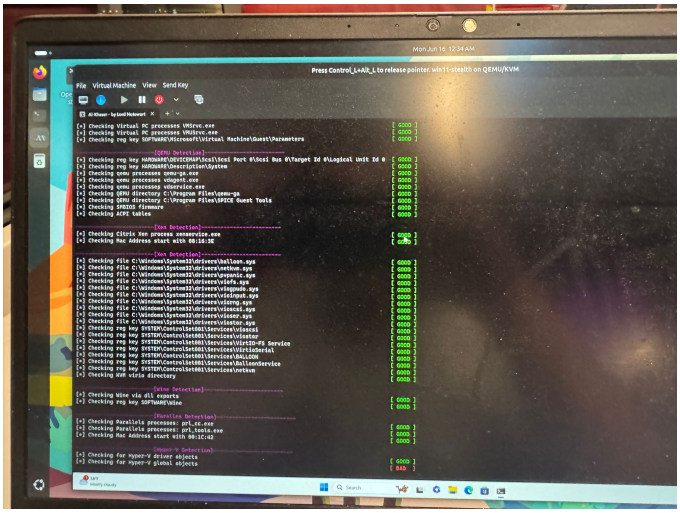


Figure 15: Al-Khaser results (part 6)

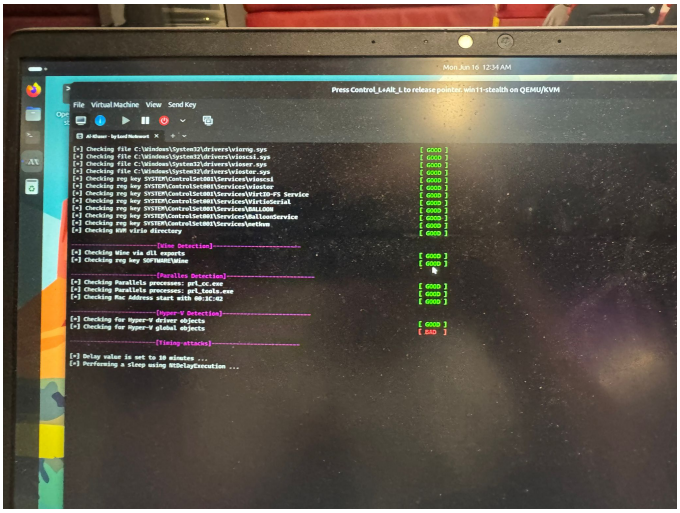


Figure 16: Al-Khaser results (part 7)

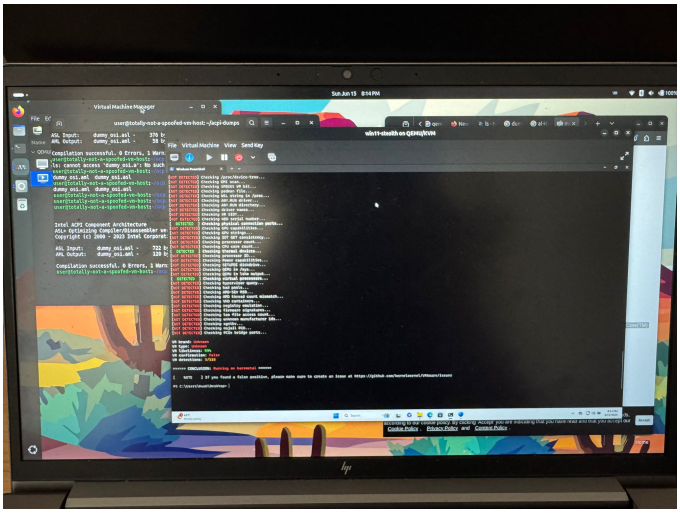
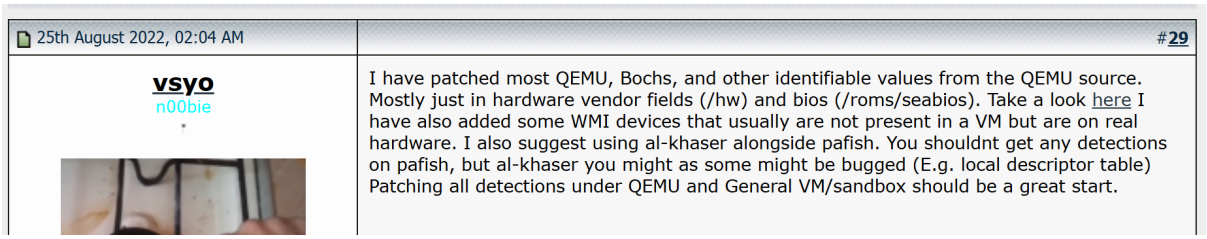


Figure 17: VMAware results

As seen in fig. 8 through fig. 17 all test suites report that they are running on bare metal. Still, some of them remain suspicious (Al-Khaser[3] and VMAware[19]), primarily because of CPU thermal readouts and other missing Windows Management Interface (WMI) devices. Furthermore, Al-Khaser[3] has been out of development since September 2024, with developers agreeing that some of its checks raise false positive results, as seen in fig. 18


25th August 2022, 02:04 AM

#29

vsyo

n00bie

I have patched most QEMU, Bochs, and other identifiable values from the QEMU source. Mostly just in hardware vendor fields (/hw) and bios (/roms/seabios). Take a look here I have also added some WMI devices that usually are not present in a VM but are on real hardware. I also suggest using al-khaser alongside pafish. You shouldnt get any detections on pafish, but al-khaser you might as some might be bugged (E.g. local descriptor table) Patching all detections under QEMU and General VM/sandbox should be a great start.

Figure 18: Al-Khaser bug with local descriptor table checks[12]

Following that, we implemented custom Advanced Configuration and Power Interface (ACPI)[2] tables in ACPI Scripting Language (ASL)[18] to attempt to spoof thermal zones and other measurements. An additional ACPI table was also implemented to spoof the `_OSI` method, which returns what each

operating system supports. The source code for them can be found in appendix D.1 and appendix D.2. These tables (and the methods they implement) are lacking in the default OVMF build H-P[7] provides. Furthermore, host system-provided tables (SSDT) were also dumped from `/sys/firmware/acpi/tables`, decompiled, inspected, recompiled, and injected into QEMU. Not all of them were usable due to HP ZBook Power G8-specific bytecode, but over 75% were used to successfully spoof the test suites.

The custom ACPI tables yielded an improvement of 24% confidence in VMAware's checks, as shown in fig. 20.

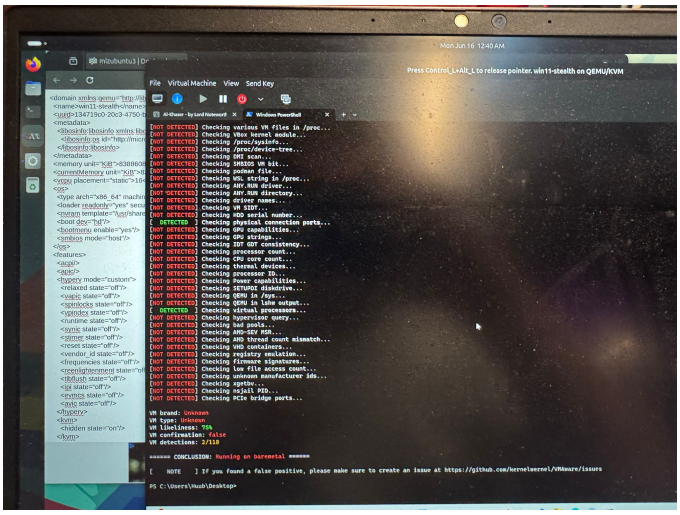


Figure 19: VMAware results

The custom Hypervisor-Phantom[7] build has been made available to our supervisor in the form of a bootable external SSD.

Even with all the checks passing for now, there are still ways they can be improved upon to detect stealth QEMU VMs. During testing, we noticed that QEMU[16] makes extensive use of virtual PCI-to-PCI or PCI-to-PCIe adapter cards. The presence of these on any modern system should be a red flag, given that most modern PCs have solely PCIe devices. An even bigger giveaway is the presence of a virtual PCI graphics card. Historically, graphics cards used the AGP[1] bus, while PCI was reserved for more general-purpose expansion cards. Having a modern system that uses a PCI graphics card instead of the CPU's integrated graphics or a more modern PCIe graphics card points to either a questionable choice of PC parts or running in a VM.

6 QEMU (Hypervisor-Phantom) on Windows

Most of the methods covered so far involve using Linux (Ubuntu) as the host OS, which might be unintuitive to less skilled attackers, such as students trying to bypass SEB. This chapter describes our efforts in building our own QEMU[16] forks on Windows.

Our first attempt at building QEMU[16] was on Linux, first using our own custom QEMU[16] fork, and later the patched sources used by Hypervisor-Phantom. Unfortunately, both cross-compilation attempts failed due to issues. MinGW-w64 installed without errors and was able to build a minimal version of QEMU[16], but issues started to arise when virtualized device packages were needed and no cross-compiled version was available.

The next attempts tried to build QEMU[16] directly on Windows, using Minimal SYStem 2 (MSYS2)[13], a virtualized UNIX-like environment. Our build toolchain closely resembles the one depicted in [15]. Similarly, we started by building our own custom build of QEMU[16] first, but this proved unsuccessful due to a myriad of build issues, missing dependencies, and MSYS2 DLL versioning mishaps. We then switched to building Hypervisor-Phantom's[7] version of QEMU[16], but we first needed to get its patching utility running on Windows. To patch QEMU[16], Hypervisor-Phantom[7]. First downloads a fresh copy of the QEMU[16] source, then applies its own Git patches, and finally goes through every placeholder their Git

patches contain and replaces them with a randomized string from a pool of hardware manufacturers, vendor IDs, and the like. This ensures that every Hypervisor-Phantom[7] install is unique and prevents detection. Unfortunately, it also means we could not simply apply the patch ourselves; we had to run the full Hypervisor-Phantom script.

Luckily, this proved to be quite easy, after commenting out some irrelevant lines, disabling all the other scripts outside the QEMU[16] ones, and replacing the dependency checks with MSYS2-appropriate dependencies. Soon enough, we got a Hypervisor-Phantom[7] build of QEMU[16] on Windows. We then adapted the template libvirt XML into command line (CLI) flags for QEMU[16] (as libvirt is Linux exclusive, but CLI flags are not), and we were able to get it running.

Running a custom build of QEMU[16] on Windows has one big advantage and one big disadvantage: because it uses the Tiny Code Generator (TCG) engine, all `RDTSC` instructions are captured and handled directly by QEMU[16], which returns realistic numbers as it's doing software emulation instead of hardware. That also means that QEMU[16] is *excruciatingly* slow, beyond any practical use by a malware researcher or fraudulent student.

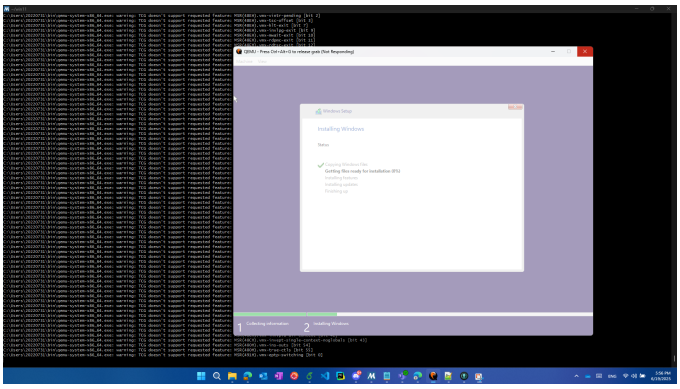


Figure 20: QEMU+TCG freezing while installing Windows

Enabling Windows Hypervisor Platform (WHPX), the Windows equivalent of Linux's KVM brings negligible speed improvements at the detriment of now failing `RDTSC` checks (unless a hypothetical kernel patch is applied). Enabling WHPX still did not bring enough speed improvements to make the VM usable.

7 Summary

This report describes various methods for bypassing Virtual Machine detection mechanisms, commonly used by malware authors to make their code harder to analyze or reverse engineer. As a testing ground, we evaluated tools designed to detect virtualization, including Safe Exam Browser[17], Al-Khaser[3], PAFish[14], and VMAware[19]. The project began with modifying QEMU[16] to spoof hardware identifiers, which allowed us to successfully evade SEB's[17] simple detection logic.

We then explored alternatives such as CloakBox[4], which proved unstable, and CAPEv2[10], a sandbox that offered valuable insights into detection techniques. These insights were later used to improve evasion capabilities of our custom QEMU[16] and Hypervisor-Phantom (H-P)[7].

H-P[7] extended beyond QEMU[16] XML modification. With features like RDTSC spoofing, GPU passthrough support, and custom ACPI/UEFI firmware to improve stealth. Additionally we developed patches, custom ACPI tables that emulated thermal zones, and allowed us to improve detection evasion scores. Although we successfully ported H-P[7] to Windows using MSYS2[13], its reliance on QEMU's[16] software-based TCG engine turned out too slow. As of right now, Hypervisor-Phantom[7] on Windows does not pose a significant threat to SEB[17], but future improvements in performance and integration may make it a viable vector for bypassing VM-based defenses.

A Installation of CAPEv2

In short, to install CAPEv2, you have to:

1. Clone the git repository (we cloned the commit with hash 5cd3819bc65e4ef05b7157478706525bbfe0ac3e).
2. Go to /CAPEv2/installer and open kvm-qemu.sh.
3. Replace the $\langle WOOT \rangle$ tags with the following:
   (a) PEN_REPLACER="ELAN Touchscreen"
   (b) SCSI_REPLACER="WDC WD10EZEX-08WN4A0"
   (c) ATAPI_REPLACER="HL-DT-ST DVDRAM GH24NSD5"
   (d) MICRODRIVE_REPLACER="Hitachi Microdrive"
   (e) BOCHS_BLOCK_REPLACER="INTEL SSDPEKNW512G8"
   (f) BOCHS_BLOCK_REPLACER2="INTEL SSDPEKNW512G8"
   (g) BOCHS_BLOCK_REPLACER3="INTEL SSDPEKNW512G8"
4. Run kvm-qemu.sh, which installs dependencies and removes hardware clues from QEMU and SeaBIOS used by QEMU to avoid detection.
5. Run cape2.sh, which clones a new copy of CAPEv2 repository at /opt/CAPEv2, installs some more dependencies and runs the necessary services.
6. Specify the ip of the host in /opt/CAPEv2/conf/cuckoo.conf under '[resultserver]' and the ip of the VM in /opt/CAPEv2/conf/kvm.conf under '[cuckoo1]'.
7. Then, you create a new virtual machine with the name 'cuckoo1' using Virtual Machine Manager, and install Windows. We used Windows 10 Enterprise, version 22H2 64-bit ([21])
8. Next, copy agent.py to the VM.
9. Install 32-bit Python 3.11.0 on the VM.
10. Rename agent.py to agent.pyw so that it runs headless and does not interfere with human behaviour simulator.
11. Configure Task Scheduler to run agent.pyw with admin privileges on log in.
12. Disable all Virus & Threat Protection settings.
13. Save the snapshot of your VM in this state. (CAPEv2 will automatically use the latest snapshot.)
14. Now you are ready to submit your first malware sample.

B (pseudo) Execution Trace of CAPEv2

```
cape-web.service ->
    manage.py #runs django
    re_path(r"^submit/", include(submission)) ->
        re_path(r"^$", views.index, name="submission") ->
            download_file(**details) ->
                db.demux_sample_and_add_to_db()

cape.service ->
    cuckoo.py ->
        init_database ->
            _connect_database(postgresql) ->
        Scheduler(max_analysis_count) ->
            start() ->
                do_main_loop_work() ->
                    find_next_serviceable_task()
                    AnalysisManager() ->
                        start()
                        build_options() # generates analysis config analysis.conf
                        GuestManager() ->
```

```
                            start_analysis() ->
                                upload_analyzer() #"""Upload the analyzer to the Virtual Machine.""" ->
                                    self.post("/extract", files={"zipfile": zip_data}, data=data)
                                add_config() ->
                                    self.post("/store", files={"file": "\n".join(config)}, data=data)
                                post("/store", files=files, data=data) #upload the malware sample to the VM
                                post("/execpy", data=data) # Execute the analyzer that we just uploaded.

agent.py ->
    do_extract()
    do_store()
    do_execpy() ->
        analyzer.py ->
            self.config = Config(cfg="analysis.conf")
            self.options = self.config.get_options()
            # sets monitorDLL to capemon.dll and loader to loader.exe
            PipeServer(PipeDispatcher, self.config.pipe);
            PipeServer(PipeForwarder, self.config.logpipe);
            #imports and starts modules
            exe.py ->
                start() ->
                    execute() ->
                        Process() ->
                            execute() #Creates a suspended process with the malware sample
                            inject() ->
                                write_monitor_config() #creates {pid}.ini on the vm
                                subprocess.run([bin_name, "inject", str(self.pid), str(thread_id), dll]) -> #runs loader with these arguments
                                    loader.exe
                            resume() # resumes the suspended thread
                            -> capemon.dll

loader.exe ->
    GrantDebugPrivileges() # you need these to read and write memory of other processes
    ReadConfig(ProcessId, DllName) # read the monitor config
    InjectDll(ProcessId, ThreadId, DllName); ->
        OpenProcess(PROCESS_ALL_ACCESS, FALSE, ProcessId) # get a process handle with every possible permission, e.g., read/write memory, start/suspend/resume threads
        GetProcessPeb(ProcessHandle, &Peb); -> #Returns the address of the PEB in the target process's memory,
            ReadProcessMemory(ProcessHandle, ProcessBasicInformation.PebBaseAddress)
        InjectDllViaIAT(ProcessHandle, ThreadHandle, DllPath, Peb) #dll is then injected via IAT ->
            # Reads and validates DOS header of the PE
            # Reads and validates NT header of the PE
            # Validates AddressOfEntryPoint
            # Sizes the IMPORT_DIRECTORY
            # Allocates memory for NT header + old(IMPORT_DIRRECTORY size) + IMAGE_IMPORT_DESCRIPTOR + 4 * IMAGE_THUNK_DATAXX + DllPathLength
            # Computes the total size of all headers of the PE
            # Section headers are located at (PBYTE)BaseAddress + SizeOfHeaders + sizeof(SectionHeader) * i
            # Creates new import directory, where 1 descriptor is new, then we have the old descriptors, thunks, new dll name and NT headers are appended at the end/
            # The newly created table is loaded into the target process memory.
            # The newly created NT headers are loaded into the target process memory.

capemon.dll ->
    CAPE_init(); ->
        read_config() # reads {pid}.ini
        AddVectoredExceptionHandler(1, capemon_exception_handler);
            capemon_exception_handler ->
                CAPEExceptionFilter(ExceptionInfo);
        YaraInit(); #compiles and saves the rules
        YaraScan(ImageBase, GetAccessibleSize(ImageBase)) ->
            yr_rules_scan_mem(Rules, Address, Size, Flags, YaraCallback, Address, Timeout) # scans the rules and calls the callback on a match
        YaraCallBack() #You can specify custom options "dump"  and "clear" in the metadata (cape_options) of the yara rule and capemon.dll will act upon those. So, you dont have to pass them during sample submission.
    add_all_dlls_to_dll_ranges();
    set_hooks() ->
        hook_api(hooks+i, g_config.hook_type) ->
```

```
            # Get the address of the to be hooked function
            # Choose the right hook type, i.e., how begining of the function is overwritten0.
            # hook_create_trampoline(unsigned char *addr, int len, unsigned char *tramp) #save overwritten
      bytes of original function to tramp and add jump to the rest of the original function
            # insert the hook (jump from the api to the pre-trampoline)
            hook_create_pre_tramp(h)
            # pre-trampoline:
                1. push the pointer to arguments onto the stack
                1. Call enter_hook(h)
                2. If eax == 0 -> jump to original function (h->tramp)
                3. If eax == 1 -> jump to h->new_func
            hook() # install it
            *h->old_func = h->hookdata->tramp; #make sure that Old_##apiname == tramp
            # you use HOOKDEF macro to get the correct signature for the hook. Then you use it in hooks.h
     to declare a hook and in hook_{name}.c to define it.

        register_dll_notification_manually(&New_DllLoadNotification); # get notified whenever a new dll is
       loaded
    CAPE_post_init() ->
        NirvanaInit() ->
            pNtSetInformationProcess(GetCurrentProcess(), (ULONG)ProcessInstrumentationCallback, &Nirvana,
      sizeof(Nirvana)) # installs a callback called on system calls
        InitialiseDebugger() ->
        SetInitialBreakpoints() ->
            SetBreakpoint(Register, 0, BreakpointVA, Type, HitCount, BreakpointCallback)
        UnpackerInit() # Adds monitor image base to tracked regions.
```

C Custom QEMU build resources

C.1 Custom QEMU setup guide

Install Ubuntu 24.04.2 LTS

Install updates (`apt update & apt upgrade`)

Reboot (mandatory; AppArmor needs to use new rules)

Check if your system is ready

```
$ sudo apt install cpu-checker -y
$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

Install dependencies

```
$ sudo apt install qemu-kvm virt-manager virtinst libvirt-clients bridge-utils libvirt-
    daemon-system -y
```

Enable newly-installed libvirtd

```
$ sudo systemctl enable --now libvirtd
$ sudo systemctl start libvirtd
```

Add yourself to the kvm/libvirt users

```
$ sudo usermod -aG kvm $USER
$ sudo usermod -aG libvirt $USER
```

Start VMM

```
$ sudo virt-manager
```

[Optional] Test the installed QEMU/KVM/libvirt install by using virt-manager or virsh to boot the Windows installer.

Reboot (may not be needed; but experience suggests it is beneficial)

Enable APT sources (on Ubuntu open Software & Updates and tick the "Source code" source)

Install QEMU build deps

```
$ sudo apt build-dep qemu -y
```

Clone custom QEMU build

```
$ git clone https://github.com/Molizo/2IC80-qemu.git
```

Clean QEMU build folder (only do it on subsequent builds)

```
$ make distclean
```

Build QEMU

```
$ ./configure --enable-kvm --enable-vnc --enable-spice --enable-gtk --target-list=x86_64
    -softmmu
$ sudo make install -j$(nproc)
```

Create the VM config file

```
$ sudo cat /etc/libvirt/qemu/win10-stealth2.xml}
```

See the full configuration file: VM Config Appendix.

Run the VM with Windows 10 Enterprise edition
We advise that the reader try to make the VM appear more authentically used by, for instance, using their own name for the Windows username, installing some apps using Ninite, and browsing the Internet for some time to build up browser history.

Download and run SEB and Pafish to verify that the VM is not detected.

C.2 Custom QEMU 10 patch

```
diff --git a/block/vhdx.c b/block/vhdx.c
index b2a4b81..f4876b1 100644
--- a/block/vhdx.c
+++ b/block/vhdx.c
@@ -2020,7 +2020,7 @@ vhdx_co_create(BlockdevCreateOptions *opts, Error **errp)

     /* The creator field is optional, but may be useful for
      * debugging / diagnostics */
-    creator = g_utf8_to_utf16("QEMU v" QEMU_VERSION, -1, NULL,
+    creator = g_utf8_to_utf16("DELL v" QEMU_VERSION, -1, NULL,
                               &creator_items, NULL);
     signature = cpu_to_le64(VHDX_FILE_SIGNATURE);
     ret = blk_co_pwrite(blk, VHDX_FILE_ID_OFFSET, sizeof(signature), &signature,
diff --git a/block/vvfat.c b/block/vvfat.c
index 91d69b3..4a5cb5e 100644
--- a/block/vvfat.c
+++ b/block/vvfat.c
@@ -1176,7 +1176,7 @@ static int vvfat_open(BlockDriverState *bs, QDict *options, int flags,
         }
         memcpy(s->volume_label, label, label_length);
     } else {
-        memcpy(s->volume_label, "QEMU VVFAT", 10);
+        memcpy(s->volume_label, "DELL VVFAT", 10);
     }

     if (floppy) {
diff --git a/chardev/msmouse.c b/chardev/msmouse.c
index 1a55755..05a6e33 100644
--- a/chardev/msmouse.c
+++ b/chardev/msmouse.c
@@ -172,7 +172,7 @@ static int msmouse_chr_write(struct Chardev *s, const uint8_t *buf, int len)
 }
```

```
 static const QemuInputHandler msmouse_handler = {
-    .name  = "QEMU Microsoft Mouse",
+    .name  = "DELL Microsoft Mouse",
     .mask  = INPUT_EVENT_MASK_BTN | INPUT_EVENT_MASK_REL,
     .event = msmouse_input_event,
     .sync  = msmouse_input_sync,
diff --git a/chardev/wctablet.c b/chardev/wctablet.c
index 0dc6ef0..de3d126 100644
--- a/chardev/wctablet.c
+++ b/chardev/wctablet.c
@@ -179,7 +179,7 @@ static void wctablet_input_sync(DeviceState *dev)
 }

 static const QemuInputHandler wctablet_handler = {
-    .name  = "QEMU Wacom Pen Tablet",
+    .name  = "Wacom Pen Tablet",
     .mask  = INPUT_EVENT_MASK_BTN | INPUT_EVENT_MASK_ABS,
     .event = wctablet_input_event,
     .sync  = wctablet_input_sync,
diff --git a/contrib/vhost-user-gpu/vhost-user-gpu.c b/contrib/vhost-user-gpu/vhost-user-gpu.c
index bb41758..6e136dd 100644
--- a/contrib/vhost-user-gpu/vhost-user-gpu.c
+++ b/contrib/vhost-user-gpu/vhost-user-gpu.c
@@ -1254,7 +1254,7 @@ main(int argc, char *argv[])
     QTAILQ_INIT(&g.reslist);
     QTAILQ_INIT(&g.fenceq);

-    context = g_option_context_new("QEMU vhost-user-gpu");
+    context = g_option_context_new("DELL vhost-user-gpu");
     g_option_context_add_main_entries(context, entries, NULL);
     if (!g_option_context_parse(context, &argc, &argv, &error)) {
         g_printerr("Option parsing failed: %s\n", error->message);
diff --git a/hw/acpi/aml-build.c b/hw/acpi/aml-build.c
index f8f93a9..867618c 100644
--- a/hw/acpi/aml-build.c
+++ b/hw/acpi/aml-build.c
@@ -1723,11 +1723,11 @@ void acpi_table_begin(AcpiTable *desc, GArray *array)
     build_append_int_noprefix(array, 0, 4); /* Length */
     build_append_int_noprefix(array, desc->rev, 1); /* Revision */
     build_append_int_noprefix(array, 0, 1); /* Checksum */
-    build_append_padded_str(array, desc->oem_id, 6, '\0'); /* OEMID */
+    build_append_padded_str(array, ACPI_BUILD_APPNAME6, 6, '\0'); /* OEMID */
     /* OEM Table ID */
-    build_append_padded_str(array, desc->oem_table_id, 8, '\0');
+    build_append_padded_str(array, ACPI_BUILD_APPNAME8, 8, '\0');
     build_append_int_noprefix(array, 1, 4); /* OEM Revision */
-    g_array_append_vals(array, ACPI_BUILD_APPNAME8, 4); /* Creator ID */
+    g_array_append_vals(array, "PIL ", 4); /* Creator ID */
     build_append_int_noprefix(array, 1, 4); /* Creator Revision */
 }

diff --git a/hw/arm/sbsa-ref.c b/hw/arm/sbsa-ref.c
index deae5cf..3d6e32f 100644
--- a/hw/arm/sbsa-ref.c
+++ b/hw/arm/sbsa-ref.c
@@ -894,7 +894,7 @@ static void sbsa_ref_class_init(ObjectClass *oc, const void *data)
     };
```

```
     mc->init = sbsa_ref_init;
-    mc->desc = "QEMU 'SBSA Reference' ARM Virtual Machine";
+    mc->desc = "DELL 'SBSA Reference' ARM Machine";
     mc->default_cpu_type = ARM_CPU_TYPE_NAME("neoverse-n2");
     mc->valid_cpu_types = valid_cpu_types;
     mc->max_cpus = 512;
diff --git a/hw/arm/virt.c b/hw/arm/virt.c
index 9a6cd08..4fb6c5d 100644
--- a/hw/arm/virt.c
+++ b/hw/arm/virt.c
@@ -112,7 +112,7 @@ static void arm_virt_compat_set(MachineClass *mc)
         MachineClass *mc = MACHINE_CLASS(oc); \
         arm_virt_compat_set(mc); \
         MACHINE_VER_SYM(options, virt, __VA_ARGS__)(mc); \
-        mc->desc = "QEMU " MACHINE_VER_STR(__VA_ARGS__) " ARM Virtual Machine"; \
+        mc->desc = "DELL " MACHINE_VER_STR(__VA_ARGS__) " ARM Machine"; \
         MACHINE_VER_DEPRECATION(__VA_ARGS__); \
         if (latest) { \
             mc->alias = "virt"; \
@@ -1702,13 +1702,13 @@ static void virt_build_smbios(VirtMachineState *vms)
     uint8_t *smbios_tables, *smbios_anchor;
     size_t smbios_tables_len, smbios_anchor_len;
     struct smbios_phys_mem_area mem_array;
-    const char *product = "QEMU Virtual Machine";
+    const char *product = "DELL Machine";

     if (kvm_enabled()) {
-        product = "KVM Virtual Machine";
+        product = "DELL Machine";
     }

-    smbios_set_defaults("QEMU", product, mc->name);
+    smbios_set_defaults("DELL", product, mc->name);

     /* build the array of physical mem area from base_memmap */
     mem_array.address = vms->memmap[VIRT_MEM].base;
diff --git a/hw/audio/hda-codec.c b/hw/audio/hda-codec.c
index 66edad2..3566750 100644
--- a/hw/audio/hda-codec.c
+++ b/hw/audio/hda-codec.c
@@ -118,7 +118,7 @@ static void hda_codec_parse_fmt(uint32_t format, struct audsettings *as)

 /* some defines */

-#define QEMU_HDA_ID_VENDOR  0x1af4
+#define QEMU_HDA_ID_VENDOR  0x0951 /* Kingston */
 #define QEMU_HDA_PCM_FORMATS (AC_SUPPCM_BITS_16 |       \
                               0x1fc /* 16 -> 96 kHz */)
 #define QEMU_HDA_AMP_NONE    (0)
diff --git a/hw/char/escc.c b/hw/char/escc.c
index afe4ca4..078c6d3 100644
--- a/hw/char/escc.c
+++ b/hw/char/escc.c
@@ -1037,7 +1037,7 @@ static void sunmouse_sync(DeviceState *dev)
 }

 static const QemuInputHandler sunmouse_handler = {
```

```
-    .name  = "QEMU Sun Mouse",
+    .name  = "DELL Mouse",
     .mask  = INPUT_EVENT_MASK_BTN | INPUT_EVENT_MASK_REL,
     .event = sunmouse_handle_event,
     .sync  = sunmouse_sync,
diff --git a/hw/display/edid-generate.c b/hw/display/edid-generate.c
index 2cb8196..09ec548 100644
--- a/hw/display/edid-generate.c
+++ b/hw/display/edid-generate.c
@@ -394,10 +394,10 @@ void qemu_edid_generate(uint8_t *edid, size_t size,
     /* =============== set defaults  =============== */

     if (!info->vendor || strlen(info->vendor) != 3) {
-        info->vendor = "RHT";
+        info->vendor = "DEL";
     }
     if (!info->name) {
-        info->name = "QEMU Monitor";
+        info->name = "DELL Monitor";
     }
     if (!info->prefx) {
         info->prefx = 1280;
@@ -449,7 +449,7 @@ void qemu_edid_generate(uint8_t *edid, size_t size,
     uint16_t vendor_id = ((((info->vendor[0] - '@') & 0x1f) << 10) |
                           (((info->vendor[1] - '@') & 0x1f) <<  5) |
                           (((info->vendor[2] - '@') & 0x1f) <<  0));
-    uint16_t model_nr = 0x1234;
+    uint16_t model_nr = 0xB443;
     uint32_t serial_nr = info->serial ? atoi(info->serial) : 0;
     stw_be_p(edid +  8, vendor_id);
     stw_le_p(edid + 10, model_nr);
diff --git a/hw/i386/acpi-build.c b/hw/i386/acpi-build.c
index f40ad06..ce016a1 100644
--- a/hw/i386/acpi-build.c
+++ b/hw/i386/acpi-build.c
@@ -2599,6 +2599,14 @@ void acpi_build(AcpiBuildTables *tables, MachineState *machine)
         g_array_append_vals(tables_blob, u, len);
     }

+    /* Disable BGRT (UEFI Logo) */
+    acpi_add_table(table_offsets, tables_blob);
+    AcpiTable table = { .sig = "BGRT", .rev = 1,
+                        .oem_id = x86ms->oem_id, .oem_table_id = x86ms->oem_table_id };
+    acpi_table_begin(&table, tables_blob);
+    build_append_int_noprefix(tables_blob, 0x00000000, 4);
+    acpi_table_end(tables->linker, &table);
+
     /* RSDT is pointed to by RSDP */
     rsdt = tables_blob->len;
     build_rsdt(tables_blob, tables->linker, table_offsets,
diff --git a/hw/i386/fw_cfg.c b/hw/i386/fw_cfg.c
index 5c0bcd5..afd3c27 100644
--- a/hw/i386/fw_cfg.c
+++ b/hw/i386/fw_cfg.c
@@ -228,7 +228,7 @@ void fw_cfg_add_acpi_dsdt(Aml *scope, FWCfgState *fw_cfg)
     Aml *dev = aml_device("FWCF");
     Aml *crs = aml_resource_template();
```

```
-    aml_append(dev, aml_name_decl("_HID", aml_string("QEMU0002")));
+    aml_append(dev, aml_name_decl("_HID", aml_string("DELL0002")));

     /* device present, functioning, decoding, not shown in UI */
     aml_append(dev, aml_name_decl("_STA", aml_int(0xB)));
diff --git a/hw/i386/pc.c b/hw/i386/pc.c
index 7065615..a0aa11b 100644
--- a/hw/i386/pc.c
+++ b/hw/i386/pc.c
@@ -76,9 +76,9 @@
  * depending on QEMU versions up to QEMU 2.4.
  */
 #define PC_CPU_MODEL_IDS(v) \
-    { "qemu32-" TYPE_X86_CPU, "model-id", "QEMU Virtual CPU version " v, },\
-    { "qemu64-" TYPE_X86_CPU, "model-id", "QEMU Virtual CPU version " v, },\
-    { "athlon-" TYPE_X86_CPU, "model-id", "QEMU Virtual CPU version " v, },
+    { "intel32-" TYPE_X86_CPU, "model-id", "GenuineIntel CPU version " v, },\
+    { "intel64-" TYPE_X86_CPU, "model-id", "GenuineIntel CPU version " v, },\
+    { "athlon-" TYPE_X86_CPU, "model-id", "AuthenticAMD Virtual CPU version " v, },

 GlobalProperty pc_compat_10_0[] = {};
 const size_t pc_compat_10_0_len = G_N_ELEMENTS(pc_compat_10_0);
diff --git a/hw/ide/atapi.c b/hw/ide/atapi.c
index a42b748..40b7dae 100644
--- a/hw/ide/atapi.c
+++ b/hw/ide/atapi.c
@@ -798,8 +798,8 @@ static void cmd_inquiry(IDEState *s, uint8_t *buf)
         buf[5] = 0;     /* reserved */
         buf[6] = 0;     /* reserved */
         buf[7] = 0;     /* reserved */
-        padstr8(buf + 8, 8, "QEMU");
-        padstr8(buf + 16, 16, "QEMU DVD-ROM");
+        padstr8(buf + 8, 8, "DELL");
+        padstr8(buf + 16, 16, "DELL DVD-ROM");
         padstr8(buf + 32, 4, s->version);
         idx = 36;
     }
diff --git a/hw/ide/core.c b/hw/ide/core.c
index b14983e..337fab9 100644
--- a/hw/ide/core.c
+++ b/hw/ide/core.c
@@ -2639,20 +2639,20 @@ int ide_init_drive(IDEState *s, IDEDevice *dev, IDEDriveKind kind, Error **e
         pstrcpy(s->drive_serial_str, sizeof(s->drive_serial_str), dev->serial);
     } else {
         snprintf(s->drive_serial_str, sizeof(s->drive_serial_str),
-                 "QM%05d", s->drive_serial);
+                 "DELL%05d", s->drive_serial);
     }
     if (dev->model) {
         pstrcpy(s->drive_model_str, sizeof(s->drive_model_str), dev->model);
     } else {
         switch (kind) {
         case IDE_CD:
-            strcpy(s->drive_model_str, "QEMU DVD-ROM");
+            strcpy(s->drive_model_str, "DELL DVD-ROM");
             break;
         case IDE_CFATA:
-            strcpy(s->drive_model_str, "QEMU MICRODRIVE");
+
```

```
+            strcpy(s->drive_model_str, "DELL MICRODRIVE");
             break;
         default:
-            strcpy(s->drive_model_str, "QEMU HARDDISK");
+            strcpy(s->drive_model_str, "DELL HARDDISK");
             break;
         }
     }
diff --git a/hw/input/adb-kbd.c b/hw/input/adb-kbd.c
index 507557d..ea48d6e 100644
--- a/hw/input/adb-kbd.c
+++ b/hw/input/adb-kbd.c
@@ -356,7 +356,7 @@ static void adb_kbd_reset(DeviceState *dev)
 }

 static const QemuInputHandler adb_keyboard_handler = {
-    .name  = "QEMU ADB Keyboard",
+    .name  = "DELL Keyboard",
     .mask  = INPUT_EVENT_MASK_KEY,
     .event = adb_keyboard_event,
 };
diff --git a/hw/input/adb-mouse.c b/hw/input/adb-mouse.c
index 373ef3f..8c1e122 100644
--- a/hw/input/adb-mouse.c
+++ b/hw/input/adb-mouse.c
@@ -94,7 +94,7 @@ static void adb_mouse_handle_event(DeviceState *dev, QemuConsole *src,
 }

 static const QemuInputHandler adb_mouse_handler = {
-    .name  = "QEMU ADB Mouse",
+    .name  = "DELL Mouse",
     .mask  = INPUT_EVENT_MASK_BTN | INPUT_EVENT_MASK_REL,
     .event = adb_mouse_handle_event,
     /*
diff --git a/hw/input/hid.c b/hw/input/hid.c
index 76bedc1..9435e69 100644
--- a/hw/input/hid.c
+++ b/hw/input/hid.c
@@ -511,20 +511,20 @@ void hid_free(HIDState *hs)
 }

 static const QemuInputHandler hid_keyboard_handler = {
-    .name  = "QEMU HID Keyboard",
+    .name  = "DELL HID Keyboard",
     .mask  = INPUT_EVENT_MASK_KEY,
     .event = hid_keyboard_event,
 };

 static const QemuInputHandler hid_mouse_handler = {
-    .name  = "QEMU HID Mouse",
+    .name  = "DELL HID Mouse",
     .mask  = INPUT_EVENT_MASK_BTN | INPUT_EVENT_MASK_REL,
     .event = hid_pointer_event,
     .sync  = hid_pointer_sync,
 };

 static const QemuInputHandler hid_tablet_handler = {
-    .name  = "QEMU HID Tablet",
```

```
+    .name  = "DELL HID Tablet",
     .mask  = INPUT_EVENT_MASK_BTN | INPUT_EVENT_MASK_ABS,
     .event = hid_pointer_event,
     .sync  = hid_pointer_sync,
diff --git a/hw/input/ps2.c b/hw/input/ps2.c
index 7f7b1fc..b8f8a8d 100644
--- a/hw/input/ps2.c
+++ b/hw/input/ps2.c
@@ -1232,7 +1232,7 @@ static const VMStateDescription vmstate_ps2_mouse = {
 };

 static const QemuInputHandler ps2_keyboard_handler = {
-    .name  = "QEMU PS/2 Keyboard",
+    .name  = "DELL PS/2 Keyboard",
     .mask  = INPUT_EVENT_MASK_KEY,
     .event = ps2_keyboard_event,
 };
@@ -1243,7 +1243,7 @@ static void ps2_kbd_realize(DeviceState *dev, Error **errp)
 }

 static const QemuInputHandler ps2_mouse_handler = {
-    .name  = "QEMU PS/2 Mouse",
+    .name  = "DELL PS/2 Mouse",
     .mask  = INPUT_EVENT_MASK_BTN | INPUT_EVENT_MASK_REL,
     .event = ps2_mouse_event,
     .sync  = ps2_mouse_sync,
diff --git a/hw/input/virtio-input-hid.c b/hw/input/virtio-input-hid.c
index d986c3c..a178370 100644
--- a/hw/input/virtio-input-hid.c
+++ b/hw/input/virtio-input-hid.c
@@ -16,10 +16,10 @@

 #include "standard-headers/linux/input.h"

-#define VIRTIO_ID_NAME_KEYBOARD     "QEMU Virtio Keyboard"
-#define VIRTIO_ID_NAME_MOUSE        "QEMU Virtio Mouse"
-#define VIRTIO_ID_NAME_TABLET       "QEMU Virtio Tablet"
-#define VIRTIO_ID_NAME_MULTITOUCH   "QEMU Virtio MultiTouch"
+#define VIRTIO_ID_NAME_KEYBOARD     "DELL Keyboard"
+#define VIRTIO_ID_NAME_MOUSE        "DELL Mouse"
+#define VIRTIO_ID_NAME_TABLET       "DELL Tablet"
+#define VIRTIO_ID_NAME_MULTITOUCH   "DELL Touchscreen"

 /* ---------------------------------------------------------------------- */

diff --git a/hw/loongarch/virt.c b/hw/loongarch/virt.c
index 7ad7fb6..e21aaac 100644
--- a/hw/loongarch/virt.c
+++ b/hw/loongarch/virt.c
@@ -130,13 +130,13 @@ static void virt_build_smbios(LoongArchVirtMachineState *lvms)
     MachineClass *mc = MACHINE_GET_CLASS(lvms);
     uint8_t *smbios_tables, *smbios_anchor;
     size_t smbios_tables_len, smbios_anchor_len;
-    const char *product = "QEMU Virtual Machine";
+    const char *product = "DELL Machine";

     if (!lvms->fw_cfg) {
         return;
```

```
     }

-    smbios_set_defaults("QEMU", product, mc->name);
+    smbios_set_defaults("DELL", product, mc->name);

     smbios_get_tables(ms, SMBIOS_ENTRY_POINT_TYPE_64,
                       NULL, 0,
@@ -1183,7 +1183,7 @@ static void virt_class_init(ObjectClass *oc, const void *data)
     mc->init = virt_init;
     mc->default_cpu_type = LOONGARCH_CPU_TYPE_NAME("la464");
     mc->default_ram_id = "loongarch.ram";
-    mc->desc = "QEMU LoongArch Virtual Machine";
+    mc->desc = "DELL LoongArch Machine";
     mc->max_cpus = LOONGARCH_MAX_CPUS;
     mc->is_default = 1;
     mc->default_kernel_irqchip_split = false;
diff --git a/hw/nvme/ctrl.c b/hw/nvme/ctrl.c
index fd93550..d911de5 100644
--- a/hw/nvme/ctrl.c
+++ b/hw/nvme/ctrl.c
@@ -8786,7 +8786,7 @@ static void nvme_init_ctrl(NvmeCtrl *n, PCIDevice *pci_dev)

     id->vid = cpu_to_le16(pci_get_word(pci_conf + PCI_VENDOR_ID));
     id->ssvid = cpu_to_le16(pci_get_word(pci_conf + PCI_SUBSYSTEM_VENDOR_ID));
-    strpadcpy((char *)id->mn, sizeof(id->mn), "QEMU NVMe Ctrl", ' ');
+    strpadcpy((char *)id->mn, sizeof(id->mn), "DELL NVMe Ctrl", ' ');
     strpadcpy((char *)id->fr, sizeof(id->fr), QEMU_VERSION, ' ');
     strpadcpy((char *)id->sn, sizeof(id->sn), n->params.serial, ' ');

diff --git a/hw/nvram/fw_cfg.c b/hw/nvram/fw_cfg.c
index 237b9f7..83c3a35 100644
--- a/hw/nvram/fw_cfg.c
+++ b/hw/nvram/fw_cfg.c
@@ -56,7 +56,7 @@
 #define FW_CFG_DMA_CTL_SELECT  0x08
 #define FW_CFG_DMA_CTL_WRITE   0x10

-#define FW_CFG_DMA_SIGNATURE 0x51454d5520434647ULL /* "QEMU CFG" */
+#define FW_CFG_DMA_SIGNATURE 0x4153532620444647ULL /* "QEMU CFG" */

 struct FWCfgEntry {
     uint32_t len;
diff --git a/hw/pci-host/gpex.c b/hw/pci-host/gpex.c
index b806a22..ba7d50c 100644
--- a/hw/pci-host/gpex.c
+++ b/hw/pci-host/gpex.c
@@ -243,7 +243,7 @@ static void gpex_root_class_init(ObjectClass *klass, const void *data)
     DeviceClass *dc = DEVICE_CLASS(klass);

     set_bit(DEVICE_CATEGORY_BRIDGE, dc->categories);
-    dc->desc = "QEMU generic PCIe host bridge";
+    dc->desc = "DELL PCIe host bridge";
     dc->vmsd = &vmstate_gpex_root;
     k->vendor_id = PCI_VENDOR_ID_REDHAT;
     k->device_id = PCI_DEVICE_ID_REDHAT_PCIE_HOST;
diff --git a/hw/ppc/e500plat.c b/hw/ppc/e500plat.c
index 775b9d8..9d0085d 100644
--- a/hw/ppc/e500plat.c
```

```
+++ b/hw/ppc/e500plat.c
@@ -22,7 +22,7 @@

 static void e500plat_fixup_devtree(void *fdt)
 {
-    const char model[] = "QEMU ppce500";
+    const char model[] = "DELL ppce500";
     const char compatible[] = "fsl,qemu-e500";

     qemu_fdt_setprop(fdt, "/", "model", model, sizeof(model));
diff --git a/hw/scsi/mptconfig.c b/hw/scsi/mptconfig.c
index 19d01f3..d6600ad 100644
--- a/hw/scsi/mptconfig.c
+++ b/hw/scsi/mptconfig.c
@@ -189,12 +189,12 @@ static
 size_t mptsas_config_manufacturing_0(MPTSASState *s, uint8_t **data, int address)
 {
     return MPTSAS_CONFIG_PACK(0, MPI_CONFIG_PAGETYPE_MANUFACTURING, 0x00,
-                              "s16s8s16s16s16",
-                              "QEMU MPT Fusion",
+                              "s11s4s51s41s91",
+                              "DELL MPT Fusion",
                               "2.5",
-                              "QEMU MPT Fusion",
+                              "DELL MPT Fusion",
                               "QEMU",
-                              "0000111122223333");
+                              "145343919810000");
 }

 static
diff --git a/hw/scsi/scsi-bus.c b/hw/scsi/scsi-bus.c
index 70be4a7..74b141f 100644
--- a/hw/scsi/scsi-bus.c
+++ b/hw/scsi/scsi-bus.c
@@ -698,8 +698,8 @@ static bool scsi_target_emulate_inquiry(SCSITargetReq *r)
         r->buf[3] = 2 | 0x10; /* HiSup, response data format */
         r->buf[4] = r->len - 5; /* Additional Length = (Len - 1) - 4 */
         r->buf[7] = 0x10 | (r->req.bus->info->tcq ? 0x02 : 0); /* Sync, TCQ.  */
-        memcpy(&r->buf[8], "QEMU    ", 8);
-        memcpy(&r->buf[16], "QEMU TARGET     ", 16);
+        memcpy(&r->buf[8], "DELL    ", 8);
+        memcpy(&r->buf[16], "DELL TARGET     ", 16);
         pstrcpy((char *) &r->buf[32], 4, qemu_hw_version());
     }
     return true;
diff --git a/hw/scsi/scsi-disk.c b/hw/scsi/scsi-disk.c
index cb4af1b..e743255 100644
--- a/hw/scsi/scsi-disk.c
+++ b/hw/scsi/scsi-disk.c
@@ -2568,7 +2568,7 @@ static void scsi_realize(SCSIDevice *dev, Error **errp)
         s->version = g_strdup(qemu_hw_version());
     }
     if (!s->vendor) {
-        s->vendor = g_strdup("QEMU");
+        s->vendor = g_strdup("DELL");
     }
     if (s->serial && strlen(s->serial) > MAX_SERIAL_LEN) {
```

```
         error_setg(errp, "The serial number can't be longer than %d characters",
@@ -2632,7 +2632,7 @@ static void scsi_hd_realize(SCSIDevice *dev, Error **errp)
     s->qdev.blocksize = s->qdev.conf.logical_block_size;
     s->qdev.type = TYPE_DISK;
     if (!s->product) {
-        s->product = g_strdup("QEMU HARDDISK");
+        s->product = g_strdup("DELL HARDDISK");
     }
     scsi_realize(&s->qdev, errp);
 }
@@ -2659,7 +2659,7 @@ static void scsi_cd_realize(SCSIDevice *dev, Error **errp)
     s->qdev.type = TYPE_ROM;
     s->features |= 1 << SCSI_DISK_F_REMOVABLE;
     if (!s->product) {
-        s->product = g_strdup("QEMU CD-ROM");
+        s->product = g_strdup("DELL CD-ROM");
     }
     scsi_realize(&s->qdev, errp);
 }
diff --git a/hw/scsi/spapr_vscsi.c b/hw/scsi/spapr_vscsi.c
index 20f70fb..269e08a 100644
--- a/hw/scsi/spapr_vscsi.c
+++ b/hw/scsi/spapr_vscsi.c
@@ -713,8 +713,8 @@ static void vscsi_inquiry_no_target(VSCSIState *s, vscsi_req *req)
     resp_data[3] = 0x02;   /* Resp data format */
     resp_data[4] = 36 - 5; /* Additional length */
     resp_data[7] = 0x10;   /* Sync transfers */
-    memcpy(&resp_data[16], "QEMU EMPTY      ", 16);
-    memcpy(&resp_data[8], "QEMU    ", 8);
+    memcpy(&resp_data[16], "DELL EMPTY      ", 16);
+    memcpy(&resp_data[8], "DELL    ", 8);

     req->writing = 0;
     vscsi_preprocess_desc(req);
diff --git a/hw/smbios/smbios.c b/hw/smbios/smbios.c
index ad4cd67..45b5e64 100644
--- a/hw/smbios/smbios.c
+++ b/hw/smbios/smbios.c
@@ -573,7 +573,7 @@ static void smbios_build_type_0_table(void)

     t->bios_characteristics = cpu_to_le64(0x08); /* Not supported */
     t->bios_characteristics_extension_bytes[0] = 0;
-    t->bios_characteristics_extension_bytes[1] = 0x14; /* TCD/SVVP | VM */
+    t->bios_characteristics_extension_bytes[1] = 0x08; /* TCD/SVVP | VM */
     if (smbios_type0.uefi) {
         t->bios_characteristics_extension_bytes[1] |= 0x08; /* |= UEFI */
     }
diff --git a/hw/usb/dev-audio.c b/hw/usb/dev-audio.c
index 26af709..fb78dfd 100644
--- a/hw/usb/dev-audio.c
+++ b/hw/usb/dev-audio.c
@@ -73,8 +73,8 @@ enum usb_audio_strings {
 };

 static const USBDescStrings usb_audio_stringtable = {
-    [STRING_MANUFACTURER]       = "QEMU",
-    [STRING_PRODUCT]            = "QEMU USB Audio",
+    [STRING_MANUFACTURER]       = "DELL",
```

```
+    [STRING_PRODUCT]            = "DELL USB Audio",
     [STRING_SERIALNUMBER]       = "1",
     [STRING_CONFIG]             = "Audio Configuration",
     [STRING_USBAUDIO_CONTROL]   = "Audio Device",
@@ -1005,7 +1005,7 @@ static void usb_audio_class_init(ObjectClass *klass, const void *data)
     dc->vmsd          = &vmstate_usb_audio;
     device_class_set_props(dc, usb_audio_properties);
     set_bit(DEVICE_CATEGORY_SOUND, dc->categories);
-    k->product_desc   = "QEMU USB Audio Interface";
+    k->product_desc   = "DELL USB Audio Interface";
     k->realize        = usb_audio_realize;
     k->handle_reset   = usb_audio_handle_reset;
     k->handle_control = usb_audio_handle_control;
diff --git a/hw/usb/dev-hid.c b/hw/usb/dev-hid.c
index 54d064e..c132912 100644
--- a/hw/usb/dev-hid.c
+++ b/hw/usb/dev-hid.c
@@ -63,17 +63,17 @@ enum {
 };

 static const USBDescStrings desc_strings = {
-    [STR_MANUFACTURER]     = "QEMU",
-    [STR_PRODUCT_MOUSE]    = "QEMU USB Mouse",
-    [STR_PRODUCT_TABLET]   = "QEMU USB Tablet",
-    [STR_PRODUCT_KEYBOARD] = "QEMU USB Keyboard",
+    [STR_MANUFACTURER]     = "DELL",
+    [STR_PRODUCT_MOUSE]    = "DELL USB Mouse",
+    [STR_PRODUCT_TABLET]   = "DELL USB Tablet",
+    [STR_PRODUCT_KEYBOARD] = "DELL USB Keyboard",
     [STR_SERIAL_COMPAT]    = "42",
     [STR_CONFIG_MOUSE]     = "HID Mouse",
     [STR_CONFIG_TABLET]    = "HID Tablet",
     [STR_CONFIG_KEYBOARD]  = "HID Keyboard",
-    [STR_SERIAL_MOUSE]     = "89126",
-    [STR_SERIAL_TABLET]    = "28754",
-    [STR_SERIAL_KEYBOARD]  = "68284",
+    [STR_SERIAL_MOUSE]     = "89122",
+    [STR_SERIAL_TABLET]    = "28755",
+    [STR_SERIAL_KEYBOARD]  = "68288",
 };

 static const USBDescIface desc_iface_mouse = {
@@ -805,7 +805,7 @@ static void usb_tablet_class_initfn(ObjectClass *klass, const void *data)
     USBDeviceClass *uc = USB_DEVICE_CLASS(klass);

     uc->realize        = usb_tablet_realize;
-    uc->product_desc   = "QEMU USB Tablet";
+    uc->product_desc   = "DELL USB Tablet";
     dc->vmsd = &vmstate_usb_ptr;
     device_class_set_props(dc, usb_tablet_properties);
     set_bit(DEVICE_CATEGORY_INPUT, dc->categories);
@@ -827,7 +827,7 @@ static void usb_mouse_class_initfn(ObjectClass *klass, const void *data)
     USBDeviceClass *uc = USB_DEVICE_CLASS(klass);

     uc->realize        = usb_mouse_realize;
-    uc->product_desc   = "QEMU USB Mouse";
+    uc->product_desc   = "DELL USB Mouse";
     dc->vmsd = &vmstate_usb_ptr;
```

```
     device_class_set_props(dc, usb_mouse_properties);
     set_bit(DEVICE_CATEGORY_INPUT, dc->categories);
@@ -850,7 +850,7 @@ static void usb_keyboard_class_initfn(ObjectClass *klass, const void *data)
     USBDeviceClass *uc = USB_DEVICE_CLASS(klass);

     uc->realize        = usb_keyboard_realize;
-    uc->product_desc   = "QEMU USB Keyboard";
+    uc->product_desc   = "DELL USB Keyboard";
     dc->vmsd = &vmstate_usb_kbd;
     device_class_set_props(dc, usb_keyboard_properties);
     set_bit(DEVICE_CATEGORY_INPUT, dc->categories);
diff --git a/hw/usb/dev-hub.c b/hw/usb/dev-hub.c
index a19350d..f1e00a6 100644
--- a/hw/usb/dev-hub.c
+++ b/hw/usb/dev-hub.c
@@ -104,9 +104,9 @@ enum {
 };

 static const USBDescStrings desc_strings = {
-    [STR_MANUFACTURER] = "QEMU",
-    [STR_PRODUCT]      = "QEMU USB Hub",
-    [STR_SERIALNUMBER] = "314159",
+    [STR_MANUFACTURER] = "DELL",
+    [STR_PRODUCT]      = "DELL USB Hub",
+    [STR_SERIALNUMBER] = "314259",
 };

 static const USBDescIface desc_iface_hub = {
@@ -676,7 +676,7 @@ static void usb_hub_class_initfn(ObjectClass *klass, const void *data)
     USBDeviceClass *uc = USB_DEVICE_CLASS(klass);

     uc->realize        = usb_hub_realize;
-    uc->product_desc   = "QEMU USB Hub";
+    uc->product_desc   = "DELL USB Hub";
     uc->usb_desc       = &desc_hub;
     uc->find_device    = usb_hub_find_device;
     uc->handle_reset   = usb_hub_handle_reset;
diff --git a/hw/usb/dev-mtp.c b/hw/usb/dev-mtp.c
index ce45c9c..24bb8ec 100644
--- a/hw/usb/dev-mtp.c
+++ b/hw/usb/dev-mtp.c
@@ -247,8 +247,8 @@ OBJECT_DECLARE_SIMPLE_TYPE(MTPState, USB_MTP)

 /* --------------------------------------------------------------------- */

-#define MTP_MANUFACTURER  "QEMU"
-#define MTP_PRODUCT       "QEMU filesharing"
+#define MTP_MANUFACTURER  "DELL"
+#define MTP_PRODUCT       "DELL filesharing"
 #define MTP_WRITE_BUF_SZ  (512 * KiB)

 enum {
@@ -264,11 +264,11 @@ enum {
 static const USBDescStrings desc_strings = {
     [STR_MANUFACTURER] = MTP_MANUFACTURER,
     [STR_PRODUCT]      = MTP_PRODUCT,
-    [STR_SERIALNUMBER] = "34617",
+    [STR_SERIALNUMBER] = "34217",
```

```
      [STR_MTP]          = "MTP",
-     [STR_CONFIG_FULL]  = "Full speed config (usb 1.1)",
-     [STR_CONFIG_HIGH]  = "High speed config (usb 2.0)",
-     [STR_CONFIG_SUPER] = "Super speed config (usb 3.0)",
+     [STR_CONFIG_FULL]  = "Full speed (USB 1.1)",
+     [STR_CONFIG_HIGH]  = "High speed (USB 2.0)",
+     [STR_CONFIG_SUPER] = "Super speed (USB 3.0)",
  };

  static const USBDescIface desc_iface_full = {
diff --git a/hw/usb/dev-network.c b/hw/usb/dev-network.c
index 81cc09d..f1ab170 100644
--- a/hw/usb/dev-network.c
+++ b/hw/usb/dev-network.c
@@ -99,15 +99,15 @@ enum usbstring_idx {
  #define ETH_FRAME_LEN                   1514 /* Max. octets in frame sans FCS */

  static const USBDescStrings usb_net_stringtable = {
-     [STRING_MANUFACTURER]       = "QEMU",
+     [STRING_MANUFACTURER]       = "DELL",
      [STRING_PRODUCT]            = "RNDIS/QEMU USB Network Device",
-     [STRING_ETHADDR]            = "400102030405",
-     [STRING_DATA]               = "QEMU USB Net Data Interface",
-     [STRING_CONTROL]            = "QEMU USB Net Control Interface",
-     [STRING_RNDIS_CONTROL]      = "QEMU USB Net RNDIS Control Interface",
-     [STRING_CDC]                = "QEMU USB Net CDC",
-     [STRING_SUBSET]             = "QEMU USB Net Subset",
-     [STRING_RNDIS]              = "QEMU USB Net RNDIS",
+     [STRING_ETHADDR]            = "400144524406",
+     [STRING_DATA]               = "DELL USB Net Data Iface",
+     [STRING_CONTROL]            = "DELL USB Net Control Iface",
+     [STRING_RNDIS_CONTROL]      = "DELL USB Net RNDIS Control Iface",
+     [STRING_CDC]                = "DELL USB Net CDC",
+     [STRING_SUBSET]             = "DELL USB Net Subset",
+     [STRING_RNDIS]              = "DELL USB Net RNDIS",
      [STRING_SERIALNUMBER]       = "1",
  };

@@ -717,7 +717,7 @@ static int ndis_query(USBNetState *s, uint32_t oid,

      /* mandatory */
      case OID_GEN_VENDOR_DESCRIPTION:
-         pstrcpy((char *)outbuf, outlen, "QEMU USB RNDIS Net");
+         pstrcpy((char *)outbuf, outlen, "DELL USB RNDIS Net");
          return strlen((char *)outbuf) + 1;

      case OID_GEN_VENDOR_DRIVER_VERSION:
@@ -1417,7 +1417,7 @@ static void usb_net_class_initfn(ObjectClass *klass, const void *data)
      USBDeviceClass *uc = USB_DEVICE_CLASS(klass);

      uc->realize        = usb_net_realize;
-     uc->product_desc   = "QEMU USB Network Interface";
+     uc->product_desc   = "DELLd USB Network Interface";
      uc->usb_desc       = &desc_net;
      uc->handle_reset   = usb_net_handle_reset;
      uc->handle_control = usb_net_handle_control;
diff --git a/hw/usb/dev-serial.c b/hw/usb/dev-serial.c
index 1c116d8..c68642f 100644
```

```
--- a/hw/usb/dev-serial.c
+++ b/hw/usb/dev-serial.c
@@ -119,10 +119,10 @@ enum {
 };

 static const USBDescStrings desc_strings = {
-    [STR_MANUFACTURER]     = "QEMU",
-    [STR_PRODUCT_SERIAL]   = "QEMU USB SERIAL",
-    [STR_PRODUCT_BRAILLE]  = "QEMU USB BAUM BRAILLE",
-    [STR_SERIALNUMBER]     = "1",
+    [STR_MANUFACTURER]     = "DELL",
+    [STR_PRODUCT_SERIAL]   = "DELL USB SERIAL",
+    [STR_PRODUCT_BRAILLE]  = "DELL USB BAUM BRAILLE",
+    [STR_SERIALNUMBER]     = "3952935",
 };

 static const USBDescIface desc_iface0 = {
@@ -663,7 +663,7 @@ static void usb_serial_class_initfn(ObjectClass *klass, const void *data)
     DeviceClass *dc = DEVICE_CLASS(klass);
     USBDeviceClass *uc = USB_DEVICE_CLASS(klass);

-    uc->product_desc   = "QEMU USB Serial";
+    uc->product_desc   = "DELL USB Serial";
     uc->usb_desc       = &desc_serial;
     device_class_set_props(dc, serial_properties);
 }
@@ -683,7 +683,7 @@ static void usb_braille_class_initfn(ObjectClass *klass, const void *data)
     DeviceClass *dc = DEVICE_CLASS(klass);
     USBDeviceClass *uc = USB_DEVICE_CLASS(klass);

-    uc->product_desc   = "QEMU USB Braille";
+    uc->product_desc   = "DELL USB Braille";
     uc->usb_desc       = &desc_braille;
     device_class_set_props(dc, braille_properties);
 }
diff --git a/hw/usb/dev-storage.c b/hw/usb/dev-storage.c
index b13fe34..703a0bf 100644
--- a/hw/usb/dev-storage.c
+++ b/hw/usb/dev-storage.c
@@ -47,12 +47,12 @@ enum {
 };

 static const USBDescStrings desc_strings = {
-    [STR_MANUFACTURER] = "QEMU",
-    [STR_PRODUCT]      = "QEMU USB HARDDRIVE",
-    [STR_SERIALNUMBER] = "1",
-    [STR_CONFIG_FULL]  = "Full speed config (usb 1.1)",
-    [STR_CONFIG_HIGH]  = "High speed config (usb 2.0)",
-    [STR_CONFIG_SUPER] = "Super speed config (usb 3.0)",
+    [STR_MANUFACTURER] = "DELL",
+    [STR_PRODUCT]      = "DELL USB HARDDRIVE",
+    [STR_SERIALNUMBER] = "12395723-94",
+    [STR_CONFIG_FULL]  = "Full speed (USB 1.1)",
+    [STR_CONFIG_HIGH]  = "High speed (USB 2.0)",
+    [STR_CONFIG_SUPER] = "Super speed (USB 3.0)",
 };

 static const USBDescIface desc_iface_full = {
```

```
@@ -590,7 +590,7 @@ static void usb_msd_class_initfn_common(ObjectClass *klass, const void *data)
     DeviceClass *dc = DEVICE_CLASS(klass);
     USBDeviceClass *uc = USB_DEVICE_CLASS(klass);

-    uc->product_desc   = "QEMU USB MSD";
+    uc->product_desc   = "DELL USB MSD";
     uc->usb_desc       = &desc;
     uc->cancel_packet  = usb_msd_cancel_io;
     uc->handle_attach  = usb_desc_attach;
diff --git a/hw/usb/dev-uas.c b/hw/usb/dev-uas.c
index 21cc283..0edea24 100644
--- a/hw/usb/dev-uas.c
+++ b/hw/usb/dev-uas.c
@@ -171,11 +171,11 @@ enum {
 };

 static const USBDescStrings desc_strings = {
-    [STR_MANUFACTURER] = "QEMU",
+    [STR_MANUFACTURER] = "DELL",
     [STR_PRODUCT]      = "USB Attached SCSI HBA",
-    [STR_SERIALNUMBER] = "27842",
-    [STR_CONFIG_HIGH]  = "High speed config (usb 2.0)",
-    [STR_CONFIG_SUPER] = "Super speed config (usb 3.0)",
+    [STR_SERIALNUMBER] = "278242",
+    [STR_CONFIG_HIGH]  = "High speed (USB 2.0)",
+    [STR_CONFIG_SUPER] = "Super speed (USB 3.0)",
 };

 static const USBDescIface desc_iface_high = {
diff --git a/hw/usb/dev-wacom.c b/hw/usb/dev-wacom.c
index f4b71a2..d15126b 100644
--- a/hw/usb/dev-wacom.c
+++ b/hw/usb/dev-wacom.c
@@ -64,9 +64,9 @@ enum {
 };

 static const USBDescStrings desc_strings = {
-    [STR_MANUFACTURER]     = "QEMU",
+    [STR_MANUFACTURER]     = "DELL",
     [STR_PRODUCT]          = "Wacom PenPartner",
-    [STR_SERIALNUMBER]     = "1",
+    [STR_SERIALNUMBER]     = "11394085",
 };

 static const uint8_t qemu_wacom_hid_report_descriptor[] = {
@@ -231,7 +231,7 @@ static int usb_mouse_poll(USBWacomState *s, uint8_t *buf, int len)

     if (!s->mouse_grabbed) {
         s->eh_entry = qemu_add_mouse_event_handler(usb_mouse_event, s, 0,
-                                                   "QEMU PenPartner tablet");
+                                                   "DELL tablet");
         qemu_activate_mouse_event_handler(s->eh_entry);
         s->mouse_grabbed = 1;
     }
@@ -269,7 +269,7 @@ static int usb_wacom_poll(USBWacomState *s, uint8_t *buf, int len)

     if (!s->mouse_grabbed) {
         s->eh_entry = qemu_add_mouse_event_handler(usb_wacom_event, s, 1,
```

```
-                        "QEMU PenPartner tablet");
+                        "DELL tablet");
         qemu_activate_mouse_event_handler(s->eh_entry);
         s->mouse_grabbed = 1;
     }
@@ -425,7 +425,7 @@ static void usb_wacom_class_init(ObjectClass *klass, const void *data)
     DeviceClass *dc = DEVICE_CLASS(klass);
     USBDeviceClass *uc = USB_DEVICE_CLASS(klass);

-    uc->product_desc   = "QEMU PenPartner Tablet";
+    uc->product_desc   = "DELL Tablet";
     uc->usb_desc       = &desc_wacom;
     uc->realize        = usb_wacom_realize;
     uc->handle_reset   = usb_wacom_handle_reset;
@@ -433,7 +433,7 @@ static void usb_wacom_class_init(ObjectClass *klass, const void *data)
     uc->handle_data    = usb_wacom_handle_data;
     uc->unrealize      = usb_wacom_unrealize;
     set_bit(DEVICE_CATEGORY_INPUT, dc->categories);
-    dc->desc = "QEMU PenPartner Tablet";
+    dc->desc = "DELL Tablet";
     dc->vmsd = &vmstate_usb_wacom;
 }

diff --git a/hw/usb/u2f-emulated.c b/hw/usb/u2f-emulated.c
index ace5ece..49e39c4 100644
--- a/hw/usb/u2f-emulated.c
+++ b/hw/usb/u2f-emulated.c
@@ -385,7 +385,7 @@ static void u2f_emulated_class_init(ObjectClass *klass, const void *data)
     kc->realize = u2f_emulated_realize;
     kc->unrealize = u2f_emulated_unrealize;
     kc->recv_from_guest = u2f_emulated_recv_from_guest;
-    dc->desc = "QEMU U2F emulated key";
+    dc->desc = "DELL U2F key";
     device_class_set_props(dc, u2f_emulated_properties);
 }

diff --git a/hw/usb/u2f-passthru.c b/hw/usb/u2f-passthru.c
index fa8d9cd..7915f14 100644
--- a/hw/usb/u2f-passthru.c
+++ b/hw/usb/u2f-passthru.c
@@ -528,7 +528,7 @@ static void u2f_passthru_class_init(ObjectClass *klass, const void *data)
     kc->realize = u2f_passthru_realize;
     kc->unrealize = u2f_passthru_unrealize;
     kc->recv_from_guest = u2f_passthru_recv_from_guest;
-    dc->desc = "QEMU U2F passthrough key";
+    dc->desc = "DELL U2Fp key";
     dc->vmsd = &u2f_passthru_vmstate;
     device_class_set_props(dc, u2f_passthru_properties);
     set_bit(DEVICE_CATEGORY_MISC, dc->categories);
diff --git a/hw/usb/u2f.c b/hw/usb/u2f.c
index b051a99..7a4e5e3 100644
--- a/hw/usb/u2f.c
+++ b/hw/usb/u2f.c
@@ -46,11 +46,11 @@ enum {
 };

 static const USBDescStrings desc_strings = {
-    [STR_MANUFACTURER]     = "QEMU",
```

```
+    [STR_MANUFACTURER]     = "DELL",
     [STR_PRODUCT]          = "U2F USB key",
-    [STR_SERIALNUMBER]     = "0",
-    [STR_CONFIG]           = "U2F key config",
-    [STR_INTERFACE]        = "U2F key interface"
+    [STR_SERIALNUMBER]     = "3028497408",
+    [STR_CONFIG]           = "U2F kc",
+    [STR_INTERFACE]        = "U2F kiface"
 };

 static const USBDescIface desc_iface_u2f_key = {
diff --git a/include/hw/acpi/aml-build.h b/include/hw/acpi/aml-build.h
index c18f681..a69b5f8 100644
--- a/include/hw/acpi/aml-build.h
+++ b/include/hw/acpi/aml-build.h
@@ -4,8 +4,8 @@
 #include "hw/acpi/acpi-defs.h"
 #include "hw/acpi/bios-linker-loader.h"

-#define ACPI_BUILD_APPNAME6 "BOCHS "
-#define ACPI_BUILD_APPNAME8 "BXPC    "
+#define ACPI_BUILD_APPNAME6 "INTEL "
+#define ACPI_BUILD_APPNAME8 "PC8086  "

 #define ACPI_BUILD_TABLE_FILE "etc/acpi/tables"
 #define ACPI_BUILD_RSDP_FILE "etc/acpi/rsdp"
diff --git a/include/hw/pci/pci.h b/include/hw/pci/pci.h
index c2fe6ca..4be9213 100644
--- a/include/hw/pci/pci.h
+++ b/include/hw/pci/pci.h
@@ -51,25 +51,25 @@ extern bool pci_available;
 #define PCI_DEVICE_ID_REALTEK_8029       0x8029

 /* Xilinx (0x10ee) */
-#define PCI_DEVICE_ID_XILINX_XC2VP30     0x0300
+#define PCI_DEVICE_ID_XILINX_XC2VP30     0x8086

 /* Marvell (0x11ab) */
 #define PCI_DEVICE_ID_MARVELL_GT6412X    0x4620

 /* QEMU/Bochs VGA (0x1234) */
-#define PCI_VENDOR_ID_QEMU               0x1234
-#define PCI_DEVICE_ID_QEMU_VGA           0x1111
-#define PCI_DEVICE_ID_QEMU_IPMI          0x1112
+#define PCI_VENDOR_ID_QEMU               0x8086
+#define PCI_DEVICE_ID_QEMU_VGA           0x8086
+#define PCI_DEVICE_ID_QEMU_IPMI          0x8086

 /* VMWare (0x15ad) */
-#define PCI_VENDOR_ID_VMWARE             0x15ad
-#define PCI_DEVICE_ID_VMWARE_SVGA2       0x0405
-#define PCI_DEVICE_ID_VMWARE_SVGA        0x0710
-#define PCI_DEVICE_ID_VMWARE_NET         0x0720
-#define PCI_DEVICE_ID_VMWARE_SCSI        0x0730
-#define PCI_DEVICE_ID_VMWARE_PVSCSI      0x07C0
-#define PCI_DEVICE_ID_VMWARE_IDE         0x1729
-#define PCI_DEVICE_ID_VMWARE_VMXNET3     0x07B0
+#define PCI_VENDOR_ID_VMWARE             0x8086
```

```
+#define PCI_DEVICE_ID_VMWARE_SVGA2       0x8086
+#define PCI_DEVICE_ID_VMWARE_SVGA        0x8086
+#define PCI_DEVICE_ID_VMWARE_NET         0x8086
+#define PCI_DEVICE_ID_VMWARE_SCSI        0x8086
+#define PCI_DEVICE_ID_VMWARE_PVSCSI      0x8086
+#define PCI_DEVICE_ID_VMWARE_IDE         0x8086
+#define PCI_DEVICE_ID_VMWARE_VMXNET3     0x8086

 /* Intel (0x8086) */
 #define PCI_DEVICE_ID_INTEL_82551IT      0x1209
@@ -77,19 +77,19 @@ extern bool pci_available;
 #define PCI_DEVICE_ID_INTEL_82801IR      0x2922

 /* Red Hat / Qumranet (for QEMU) -- see pci-ids.txt */
-#define PCI_VENDOR_ID_REDHAT_QUMRANET    0x1af4
-#define PCI_SUBVENDOR_ID_REDHAT_QUMRANET 0x1af4
-#define PCI_SUBDEVICE_ID_QEMU            0x1100
+#define PCI_VENDOR_ID_REDHAT_QUMRANET    0x8029
+#define PCI_SUBVENDOR_ID_REDHAT_QUMRANET 0x8029
+#define PCI_SUBDEVICE_ID_QEMU            0x8029

 /* legacy virtio-pci devices */
-#define PCI_DEVICE_ID_VIRTIO_NET         0x1000
-#define PCI_DEVICE_ID_VIRTIO_BLOCK       0x1001
-#define PCI_DEVICE_ID_VIRTIO_BALLOON     0x1002
-#define PCI_DEVICE_ID_VIRTIO_CONSOLE     0x1003
-#define PCI_DEVICE_ID_VIRTIO_SCSI        0x1004
-#define PCI_DEVICE_ID_VIRTIO_RNG         0x1005
-#define PCI_DEVICE_ID_VIRTIO_9P          0x1009
-#define PCI_DEVICE_ID_VIRTIO_VSOCK       0x1012
+#define PCI_DEVICE_ID_VIRTIO_NET         0x8029
+#define PCI_DEVICE_ID_VIRTIO_BLOCK       0x8029
+#define PCI_DEVICE_ID_VIRTIO_BALLOON     0x8030
+#define PCI_DEVICE_ID_VIRTIO_CONSOLE     0x8030
+#define PCI_DEVICE_ID_VIRTIO_SCSI        0x8031
+#define PCI_DEVICE_ID_VIRTIO_RNG         0x8032
+#define PCI_DEVICE_ID_VIRTIO_9P          0x8033
+#define PCI_DEVICE_ID_VIRTIO_VSOCK       0x8033

 /*
  * modern virtio-pci devices get their id assigned automatically,
@@ -100,28 +100,28 @@ extern bool pci_available;
  */
 #define PCI_DEVICE_ID_VIRTIO_10_BASE     0x1040

-#define PCI_VENDOR_ID_REDHAT             0x1b36
-#define PCI_DEVICE_ID_REDHAT_BRIDGE      0x0001
-#define PCI_DEVICE_ID_REDHAT_SERIAL      0x0002
-#define PCI_DEVICE_ID_REDHAT_SERIAL2     0x0003
-#define PCI_DEVICE_ID_REDHAT_SERIAL4     0x0004
-#define PCI_DEVICE_ID_REDHAT_TEST        0x0005
-#define PCI_DEVICE_ID_REDHAT_ROCKER      0x0006
-#define PCI_DEVICE_ID_REDHAT_SDHCI       0x0007
-#define PCI_DEVICE_ID_REDHAT_PCIE_HOST   0x0008
-#define PCI_DEVICE_ID_REDHAT_PXB         0x0009
-#define PCI_DEVICE_ID_REDHAT_BRIDGE_SEAT 0x000a
-#define PCI_DEVICE_ID_REDHAT_PXB_PCIE    0x000b
-#define PCI_DEVICE_ID_REDHAT_PCIE_RP     0x000c
```

```
-#define PCI_DEVICE_ID_REDHAT_XHCI        0x000d
-#define PCI_DEVICE_ID_REDHAT_PCIE_BRIDGE 0x000e
-#define PCI_DEVICE_ID_REDHAT_MDPY        0x000f
-#define PCI_DEVICE_ID_REDHAT_NVME        0x0010
-#define PCI_DEVICE_ID_REDHAT_PVPANIC     0x0011
-#define PCI_DEVICE_ID_REDHAT_ACPI_ERST   0x0012
-#define PCI_DEVICE_ID_REDHAT_UFS         0x0013
-#define PCI_DEVICE_ID_REDHAT_RISCV_IOMMU 0x0014
-#define PCI_DEVICE_ID_REDHAT_QXL         0x0100
+#define PCI_VENDOR_ID_REDHAT             0x8081
+#define PCI_DEVICE_ID_REDHAT_BRIDGE      0x8082
+#define PCI_DEVICE_ID_REDHAT_SERIAL      0x8083
+#define PCI_DEVICE_ID_REDHAT_SERIAL2     0x8083
+#define PCI_DEVICE_ID_REDHAT_SERIAL4     0x8083
+#define PCI_DEVICE_ID_REDHAT_TEST        0x8085
+#define PCI_DEVICE_ID_REDHAT_ROCKER      0x9021
+#define PCI_DEVICE_ID_REDHAT_SDHCI       0x9021
+#define PCI_DEVICE_ID_REDHAT_PCIE_HOST   0x9023
+#define PCI_DEVICE_ID_REDHAT_PXB         0x9056
+#define PCI_DEVICE_ID_REDHAT_BRIDGE_SEAT 0x9034
+#define PCI_DEVICE_ID_REDHAT_PXB_PCIE    0x3958
+#define PCI_DEVICE_ID_REDHAT_PCIE_RP     0x3394
+#define PCI_DEVICE_ID_REDHAT_XHCI        0x3932
+#define PCI_DEVICE_ID_REDHAT_PCIE_BRIDGE 0x3394
+#define PCI_DEVICE_ID_REDHAT_MDPY        0x3932
+#define PCI_DEVICE_ID_REDHAT_NVME        0x9034
+#define PCI_DEVICE_ID_REDHAT_PVPANIC     0x3958
+#define PCI_DEVICE_ID_REDHAT_ACPI_ERST   0x3394
+#define PCI_DEVICE_ID_REDHAT_UFS         0x9034
+#define PCI_DEVICE_ID_REDHAT_RISCV_IOMMU 0x8085
+#define PCI_DEVICE_ID_REDHAT_QXL         0x3932

 #define FMT_PCIBUS                      PRIx64

diff --git a/include/hw/pci/pci_ids.h b/include/hw/pci/pci_ids.h
index 33e2898..eecf3d1 100644
--- a/include/hw/pci/pci_ids.h
+++ b/include/hw/pci/pci_ids.h
@@ -284,7 +284,7 @@
 #define PCI_VENDOR_ID_TEWS               0x1498
 #define PCI_DEVICE_ID_TEWS_TPCI200       0x30C8

-#define PCI_VENDOR_ID_VMWARE             0x15ad
+#define PCI_VENDOR_ID_VMWARE             0x8086
 #define PCI_DEVICE_ID_VMWARE_PVRDMA      0x0820

 #define PCI_VENDOR_ID_SYNOPSYS           0x16C3
diff --git a/include/standard-headers/linux/qemu_fw_cfg.h b/include/standard-headers/linux/qemu_fw_c
index cb93f66..5b20172 100644
--- a/include/standard-headers/linux/qemu_fw_cfg.h
+++ b/include/standard-headers/linux/qemu_fw_cfg.h
@@ -71,7 +71,7 @@ struct fw_cfg_file {
 #define FW_CFG_DMA_CTL_SELECT        0x08
 #define FW_CFG_DMA_CTL_WRITE        0x10

-#define FW_CFG_DMA_SIGNATURE    0x51454d5520434647ULL /* "QEMU CFG" */
+#define FW_CFG_DMA_SIGNATURE    0x4153532620444647ULL /* "QEMU CFG" */
```

```
 /* Control as first field allows for different structures selected by this
  * field, which might be useful in the future
diff --git a/migration/rdma.c b/migration/rdma.c
index 2d839fc..1b64a22 100644
--- a/migration/rdma.c
+++ b/migration/rdma.c
@@ -220,7 +220,7 @@ static const char *control_desc(unsigned int rdma_control)
          [RDMA_CONTROL_NONE] = "NONE",
          [RDMA_CONTROL_ERROR] = "ERROR",
          [RDMA_CONTROL_READY] = "READY",
-         [RDMA_CONTROL_QEMU_FILE] = "QEMU FILE",
+         [RDMA_CONTROL_QEMU_FILE] = "DELL FILE",
          [RDMA_CONTROL_RAM_BLOCKS_REQUEST] = "RAM BLOCKS REQUEST",
          [RDMA_CONTROL_RAM_BLOCKS_RESULT] = "RAM BLOCKS RESULT",
          [RDMA_CONTROL_COMPRESS] = "COMPRESS",
diff --git a/pc-bios/optionrom/optionrom.h b/pc-bios/optionrom/optionrom.h
index 7bcdf0e..c09ae36 100644
--- a/pc-bios/optionrom/optionrom.h
+++ b/pc-bios/optionrom/optionrom.h
@@ -43,7 +43,7 @@
 #define FW_CFG_DMA_CTL_SELECT  0x08
 #define FW_CFG_DMA_CTL_WRITE   0x10

-#define FW_CFG_DMA_SIGNATURE 0x51454d5520434647ULL /* "QEMU CFG" */
+#define FW_CFG_DMA_SIGNATURE 0x4153532620444617ULL /* "QEMU CFG" */

 #define BIOS_CFG_DMA_ADDR_HIGH  0x514
 #define BIOS_CFG_DMA_ADDR_LOW   0x518
diff --git a/pc-bios/s390-ccw/virtio-scsi.h b/pc-bios/s390-ccw/virtio-scsi.h
index c5612e1..3baf491 100644
--- a/pc-bios/s390-ccw/virtio-scsi.h
+++ b/pc-bios/s390-ccw/virtio-scsi.h
@@ -25,7 +25,7 @@
 #define VIRTIO_SCSI_S_OK                        0x00
 #define VIRTIO_SCSI_S_BAD_TARGET                0x03

-#define QEMU_CDROM_SIGNATURE "QEMU CD-ROM     "
+#define QEMU_CDROM_SIGNATURE "DELL CD-ROM     "

 enum virtio_scsi_vq_id {
     VR_CONTROL  = 0,
diff --git a/qapi/ui.json b/qapi/ui.json
index c536d4e..71e49da 100644
--- a/qapi/ui.json
+++ b/qapi/ui.json
@@ -831,13 +831,13 @@
 #     -> { "execute": "query-mice" }
 #     <- { "return": [
 #              {
-#                 "name":"QEMU Microsoft Mouse",
+#                 "name":"DELL Microsoft Mouse",
 #                 "index":0,
 #                 "current":false,
 #                 "absolute":false
 #              },
 #              {
-#                 "name":"QEMU PS/2 Mouse",
+#                 "name":"DELL PS/2 Mouse",
```

```
 #                 "index":1,
 #                 "current":true,
 #                 "absolute":true
diff --git a/qga/vss-win32/vss-handles.h b/qga/vss-win32/vss-handles.h
index 1a7d842..1b8647b 100644
--- a/qga/vss-win32/vss-handles.h
+++ b/qga/vss-win32/vss-handles.h
@@ -3,7 +3,7 @@

 /* Constants for QGA VSS Provider */

-#define QGA_PROVIDER_NAME "QEMU Guest Agent VSS Provider"
+#define QGA_PROVIDER_NAME "DELL VSS Provider"
 #define QGA_PROVIDER_LNAME L(QGA_PROVIDER_NAME)
 #define QGA_PROVIDER_VERSION L(QEMU_VERSION)
 #define QGA_PROVIDER_REGISTRY_ADDRESS "SYSTEM\\CurrentControlSet"\
diff --git a/roms/config.vga-qxl b/roms/config.vga-qxl
index d393f0c..aa5a22d 100644
--- a/roms/config.vga-qxl
+++ b/roms/config.vga-qxl
@@ -2,5 +2,5 @@ CONFIG_BUILD_VGABIOS=y
 CONFIG_VGA_BOCHS=y
 CONFIG_VGA_PCI=y
 CONFIG_OVERRIDE_PCI_ID=y
-CONFIG_VGA_VID=0x1b36
-CONFIG_VGA_DID=0x0100
+CONFIG_VGA_VID=0x1c36
+CONFIG_VGA_DID=0x0200
diff --git a/target/i386/kvm/kvm.c b/target/i386/kvm/kvm.c
index c9a3c02..fea03ff 100644
--- a/target/i386/kvm/kvm.c
+++ b/target/i386/kvm/kvm.c
@@ -2207,7 +2207,7 @@ int kvm_arch_init_vcpu(CPUState *cs)
         abort();
 #endif
     } else if (cpu->expose_kvm) {
-        memcpy(signature, "KVMKVMKVM\0\0\0", 12);
+        memcpy(signature, "GenuineIntel", 12);
         c = &cpuid_data.entries[cpuid_i++];
         c->function = KVM_CPUID_SIGNATURE | kvm_base;
         c->eax = KVM_CPUID_FEATURES | kvm_base;
diff --git a/ui/spice-core.c b/ui/spice-core.c
index 0326c63..e549703 100644
--- a/ui/spice-core.c
+++ b/ui/spice-core.c
@@ -808,7 +808,7 @@ static void qemu_spice_init(void)

     qemu_opt_foreach(opts, add_channel, &tls_port, &error_fatal);

-    spice_server_set_name(spice_server, qemu_name ?: "QEMU " QEMU_VERSION);
+    spice_server_set_name(spice_server, qemu_name ?: "DELL " QEMU_VERSION);
     spice_server_set_uuid(spice_server, (unsigned char *)&qemu_uuid);

     seamless_migration = qemu_opt_get_bool(opts, "seamless-migration", 0);
diff --git a/ui/spice-input.c b/ui/spice-input.c
index a5c5d78..2413fbd 100644
--- a/ui/spice-input.c
+++ b/ui/spice-input.c
```

```
@@ -39,7 +39,7 @@ static uint8_t kbd_get_leds(SpiceKbdInstance *sin);

 static const SpiceKbdInterface kbd_interface = {
     .base.type          = SPICE_INTERFACE_KEYBOARD,
-    .base.description   = "qemu keyboard",
+    .base.description   = "DELL keyboard",
     .base.major_version = SPICE_INTERFACE_KEYBOARD_MAJOR,
     .base.minor_version = SPICE_INTERFACE_KEYBOARD_MINOR,
     .push_scan_freg     = kbd_push_key,
```

C.3 libvirt XML

```
<!--
WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE
OVERWRITTEN AND LOST. Changes to this xml configuration should be made using:
  virsh edit win10-stealth2
or other application using the libvirt API.
-->

<domain type='kvm'>
  <name>win10-stealth2</name>
  <uuid>550e8400-e29b-41d4-a716-446622440001</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://microsoft.com/win/10"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit='KiB'>4194304</memory>
  <currentMemory unit='KiB'>4194304</currentMemory>
  <vcpu placement='static'>8</vcpu>
  <sysinfo type='smbios'>
    <bios>
      <entry name='vendor'>American Megatrends Inc.</entry>
      <entry name='version'>P1.20</entry>
      <entry name='date'>02/01/2025</entry>
    </bios>
    <system>
      <entry name='manufacturer'>DELL COMPUTER INC.</entry>
      <entry name='product'>XPS 15 7590</entry>
      <entry name='serial'>1230582347856</entry>
      <entry name='uuid'>550e8400-e29b-41d4-a716-446622440001</entry>
    </system>
  </sysinfo>
  <os>
    <type arch='x86_64' machine='pc-q35-10.1'>hvm</type>
    <bootmenu enable='yes'/>
    <smbios mode='host'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <hyperv mode='custom'>
      <relaxed state='on'/>
      <vapic state='on'/>
      <spinlocks state='on' retries='8191'/>
      <vendor_id state='on' value='GenuineIntel'/>
    </hyperv>
```

```
  <kvm>
    <hidden state='on'/>
  </kvm>
  <vmport state='off'/>
  <smm state='on'/>
  <ioapic driver='kvm'/>
</features>
<cpu mode='host-passthrough' check='partial' migratable='on'>
  <topology sockets='1' dies='1' cores='4' threads='2'/>
  <feature policy='disable' name='hypervisor'/>
  <feature policy='require' name='invtsc'/>
  <feature policy='disable' name='rdtscp'/>
</cpu>
<clock offset='localtime'>
  <timer name='rtc' tickpolicy='catchup'/>
  <timer name='pit' tickpolicy='delay'/>
  <timer name='hpet' present='no'/>
  <timer name='hypervclock' present='yes'/>
</clock>
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>destroy</on_crash>
<pm>
  <suspend-to-mem enabled='no'/>
  <suspend-to-disk enabled='no'/>
</pm>
<devices>
  <emulator>/usr/local/bin/qemu-system-x86_64</emulator>
  <disk type='file' device='cdrom'>
    <driver name='qemu' type='raw'/>
    <source file='/var/lib/libvirt/boot/win10enterprise_22h2.iso'/>
    <target dev='sda' bus='sata'/>
    <readonly/>
    <boot order='2'/>
    <address type='drive' controller='0' bus='0' target='0' unit='0'/>
  </disk>
  <disk type='file' device='disk'>
    <driver name='qemu' type='qcow2'/>
    <source file='/var/lib/libvirt/images/win10-stealth.qcow2'/>
    <target dev='sdb' bus='sata'/>
    <serial>SSD83257489</serial>
    <boot order='1'/>
    <address type='drive' controller='0' bus='0' target='0' unit='1'/>
  </disk>
  <controller type='usb' index='0' model='qemu-xhci'>
    <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
  </controller>
  <controller type='sata' index='0'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x1f' function='0x2'/>
  </controller>
  <controller type='pci' index='0' model='pcie-root'/>
  <controller type='pci' index='1' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='1' port='0x10'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0' multifunction='on'/>
  </controller>
  <controller type='pci' index='2' model='pcie-root-port'>
    <model name='pcie-root-port'/>
```

```
      <target chassis='2' port='0x11'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x1'/>
    </controller>
    <controller type='pci' index='3' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='3' port='0x12'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x2'/>
    </controller>
    <controller type='pci' index='4' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='4' port='0x13'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x3'/>
    </controller>
    <controller type='pci' index='5' model='pcie-to-pci-bridge'>
      <model name='pcie-pci-bridge'/>
      <address type='pci' domain='0x0000' bus='0x03' slot='0x00' function='0x0'/>
    </controller>
    <interface type='network'>
      <mac address='3c:97:0e:aa:bb:cc'/>
      <source network='default'/>
      <model type='e1000e'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
    </interface>
    <input type='mouse' bus='ps2'/>
    <input type='keyboard' bus='ps2'/>
    <graphics type='spice' autoport='yes' listen='0.0.0.0'>
      <listen type='address' address='0.0.0.0'/>
    </graphics>
    <sound model='ich9'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1b' function='0x0'/>
    </sound>
    <audio id='1' type='spice'/>
    <video>
      <model type='bochs' vram='16384' heads='1' primary='yes'>
        <resolution x='1920' y='1080'/>
      </model>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x0'/>
    </video>
    <hostdev mode='subsystem' type='pci' managed='yes'>
      <source>
        <address domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
      </source>
      <address type='pci' domain='0x0000' bus='0x04' slot='0x00' function='0x0'/>
    </hostdev>
    <watchdog model='itco' action='reset'/>
    <memballoon model='virtio'>
      <address type='pci' domain='0x0000' bus='0x02' slot='0x00' function='0x0'/>
    </memballoon>
  </devices>
</domain>
```

D Hypervisor-Phantom resources

D.1 dummy_osi.asl

```
DefinitionBlock ("", "SSDT", 2, "OEMID", "OEMTABLE", 0x00000001)
{
    Scope (\_SB)
```

```
    {
        Method (_OSI, 1, Serialized) // _OSI takes one argument (String)
        {
            // Local0 is the passed parameter (the string)
            If (Arg0 == "Windows 10") {
                // Return a buffer indicating Windows 10 support
                Local0 = Buffer(4) { "Win10" }
            } ElseIf (Arg0 == "Windows 11") {
                // Return a buffer indicating Windows 11 support
                Local0 = Buffer(4) { "Win11" }
            } Else {
                // If the OS is unknown, return an empty buffer
                Local0 = Buffer(1) { 0x00 }
            }

            Return (Local0)  // Return the buffer
        }
    }
}
```

D.2 thermal_zone_ssdt.asl

```
DefinitionBlock ("thermal_zone_ssdt.aml", "SSDT", 2, "QEMU  ", "TZSSDTS1", 0x01)
{
    Scope (\_TZ)
    {
        ThermalZone (TZ00)
        {
            /* Current temperature (300.0 K → 3000) */
            Method (_TMP, 0, NotSerialized) { Return (3000) }

            /* Passive-cooling trip point (320.0 K) */
            Method (_PSV, 0, NotSerialized) { Return (3200) }

            /* Hibernation trip point (340.0 K) */
            Method (_HOT, 0, NotSerialized) { Return (3400) }

            /* Critical-shutdown trip point (360.0 K) */
            Method (_CRT, 0, NotSerialized) { Return (3600) }

            /* Active-cooling (fan) trip point: none */
            Method (_AC0, 0, NotSerialized) { Return (0) }
            Name   (_AL0, Package() {})      /* No fans listed */

            /* Passive-cooling device list: none */
            Name   (_PSL, Package() {})

            /* Thermal constants */
            Name   (_TC1, 4)
            Name   (_TC2, 3)

            /* Sampling period (15 s → 150 tenths of seconds) */
            Name   (_TSP, 150)

            /* No polling required */
            Name   (_TZP, 0)

            /* No extra notifications */
```

```
            Method (_NTT, 0, NotSerialized) { Return (0) }

            /* Trip-info callback (unused) */
            Method (_DTI, 1, NotSerialized) { /* no-op */ }

            /* Set cooling policy: just echo back */
            Method (_SCP, 1, NotSerialized) { Return (Arg0) }
        }
    }
}
```

D.3 libvirt XML

In order to successfully use this XML make sure that you have the custom-built ACPI tables together with table dumps from the host system. This configuration was tested on an HP ZBook Power G8, and might need configuration changes for other systems.

```
<domain xmlns:qemu="http://libvirt.org/schemas/domain/qemu/1.0" type="kvm">
  <name>win11-stealth</name>
  <uuid>134719c0-20c3-4750-bb77-22ab4e866a1f</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://microsoft.com/win/11"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit="KiB">8388608</memory>
  <currentMemory unit="KiB">8388608</currentMemory>
  <vcpu placement="static">16</vcpu>
  <os>
    <type arch="x86_64" machine="pc-q35-9.2">hvm</type>
    <loader readonly="yes" secure="yes" type="pflash" format="qcow2">/usr/share/edk2/x64/OVMF_CODE.s
    <nvram template="/usr/share/edk2/x64/OVMF_VARS.4m.qcow2" format="qcow2">/var/lib/libvirt/qemu/nv
    <boot dev="hd"/>
    <bootmenu enable="yes"/>
    <smbios mode="host"/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <hyperv mode="custom">
      <relaxed state="off"/>
      <vapic state="off"/>
      <spinlocks state="off"/>
      <vpindex state="off"/>
      <runtime state="off"/>
      <synic state="off"/>
      <stimer state="off"/>
      <reset state="off"/>
      <vendor_id state="off"/>
      <frequencies state="off"/>
      <reenlightenment state="off"/>
      <tlbflush state="off"/>
      <ipi state="off"/>
      <evmcs state="off"/>
      <avic state="off"/>
    </hyperv>
    <kvm>
      <hidden state="on"/>
    </kvm>
```

```
  <pmu state="off"/>
  <vmport state="off"/>
  <smm state="on"/>
  <ioapic driver="kvm"/>
  <msrs unknown="fault"/>
</features>
<cpu mode="host-passthrough" check="none" migratable="on">
  <topology sockets="1" dies="1" cores="8" threads="2"/>
  <cache mode="passthrough"/>
  <feature policy="require" name="vmx"/>
  <feature policy="require" name="invtsc"/>
  <feature policy="disable" name="vmx-vnmi"/>
  <feature policy="disable" name="hypervisor"/>
  <feature policy="disable" name="ssbd"/>
  <feature policy="disable" name="amd-ssbd"/>
  <feature policy="disable" name="virt-ssbd"/>
  <feature policy="disable" name="rdpid"/>
</cpu>
<clock offset="localtime">
  <timer name="tsc" present="yes" tickpolicy="discard" mode="native"/>
  <timer name="hpet" present="yes"/>
  <timer name="rtc" present="no"/>
  <timer name="pit" present="no"/>
  <timer name="kvmclock" present="no"/>
  <timer name="hypervclock" present="no"/>
</clock>
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>destroy</on_crash>
<pm>
  <suspend-to-mem enabled="yes"/>
  <suspend-to-disk enabled="yes"/>
</pm>
<devices>
  <emulator>/usr/local/bin/qemu-system-x86_64</emulator>
  <disk type="file" device="disk">
    <driver name="qemu" type="qcow2"/>
    <source file="/var/lib/libvirt/images/win10.qcow2"/>
    <target dev="sda" bus="sata"/>
    <address type="drive" controller="0" bus="0" target="0" unit="0"/>
  </disk>
  <controller type="sata" index="0">
    <address type="pci" domain="0x0000" bus="0x00" slot="0x1f" function="0x2"/>
  </controller>
  <controller type="pci" index="0" model="pcie-root"/>
  <controller type="pci" index="1" model="pcie-root-port">
    <model name="pcie-root-port"/>
    <target chassis="1" port="0x8"/>
    <address type="pci" domain="0x0000" bus="0x00" slot="0x01" function="0x0"
    multifunction="on"/>
  </controller>
  <controller type="pci" index="2" model="pcie-root-port">
    <model name="pcie-root-port"/>
    <target chassis="2" port="0x9"/>
    <address type="pci" domain="0x0000" bus="0x00" slot="0x01" function="0x1"/>
  </controller>
  <controller type="pci" index="3" model="pcie-to-pci-bridge">
    <model name="pcie-pci-bridge"/>
```

```
  <address type="pci" domain="0x0000" bus="0x02" slot="0x00" function="0x0"/>
</controller>
<controller type="pci" index="4" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="4"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x01" function="0x0"/>
</controller>
<controller type="pci" index="5" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="5"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x02" function="0x0"/>
</controller>
<controller type="pci" index="6" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="6"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x03" function="0x0"/>
</controller>
<controller type="pci" index="7" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="7"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x04" function="0x0"/>
</controller>
<controller type="pci" index="8" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="8"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x05" function="0x0"/>
</controller>
<controller type="pci" index="9" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="9"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x06" function="0x0"/>
</controller>
<controller type="pci" index="10" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="10"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x07" function="0x0"/>
</controller>
<controller type="pci" index="11" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="11"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x08" function="0x0"/>
</controller>
<controller type="pci" index="12" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="12"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x09" function="0x0"/>
</controller>
<controller type="pci" index="13" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="13"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x0a" function="0x0"/>
</controller>
<controller type="pci" index="14" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="14"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x0b" function="0x0"/>
</controller>
<controller type="pci" index="15" model="pci-bridge">
```

```
  <model name="pci-bridge"/>
  <target chassisNr="15"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x0c" function="0x0"/>
</controller>
<controller type="pci" index="16" model="pci-bridge">
  <model name="pci-bridge"/>
  <target chassisNr="16"/>
  <address type="pci" domain="0x0000" bus="0x03" slot="0x0d" function="0x0"/>
</controller>
<controller type="pci" index="17" model="pcie-root-port">
  <model name="pcie-root-port"/>
  <target chassis="17" port="0xa"/>
  <address type="pci" domain="0x0000" bus="0x00" slot="0x01" function="0x2"/>
</controller>
<controller type="pci" index="18" model="pcie-root-port">
  <model name="pcie-root-port"/>
  <target chassis="18" port="0xb"/>
  <address type="pci" domain="0x0000" bus="0x00" slot="0x01" function="0x3"/>
</controller>
<controller type="pci" index="19" model="pcie-root-port">
  <model name="pcie-root-port"/>
  <target chassis="19" port="0xc"/>
  <address type="pci" domain="0x0000" bus="0x00" slot="0x01" function="0x4"/>
</controller>
<controller type="usb" index="0" model="ich9-ehci1">
  <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x7"/>
</controller>
<controller type="usb" index="0" model="ich9-uhci1">
  <master startport="0"/>
  <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x0" multifunction="on"/>
</controller>
<controller type="usb" index="0" model="ich9-uhci2">
  <master startport="2"/>
  <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x1"/>
</controller>
<controller type="usb" index="0" model="ich9-uhci3">
  <master startport="4"/>
  <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x2"/>
</controller>
<interface type="network">
  <mac address="02:6b:89:0c:c0:d1"/>
  <source network="default"/>
  <model type="e1000e"/>
  <link state="up"/>
  <address type="pci" domain="0x0000" bus="0x01" slot="0x00" function="0x0"/>
</interface>
<input type="mouse" bus="ps2"/>
<input type="keyboard" bus="ps2"/>
<tpm model="tpm-tis">
  <backend type="emulator" version="2.0"/>
</tpm>
<graphics type="spice" autoport="yes">
  <listen type="address"/>
  <image compression="off"/>
  <gl enable="no"/>
</graphics>
<sound model="ich9">
  <address type="pci" domain="0x0000" bus="0x00" slot="0x1b" function="0x0"/>
```

```
  </sound>
  <audio id="1" type="none"/>
  <video>
    <model type="bochs" vram="16384" heads="1" primary="yes"/>
    <address type="pci" domain="0x0000" bus="0x12" slot="0x00" function="0x0"/>
  </video>
  <hostdev mode="subsystem" type="pci" managed="yes">
    <source>
      <address domain="0x0000" bus="0x01" slot="0x00" function="0x0"/>
    </source>
    <address type="pci" domain="0x0000" bus="0x11" slot="0x00" function="0x0"/>
  </hostdev>
  <watchdog model="itco" action="reset"/>
  <memballoon model="none"/>
  <shmem name="looking-glass">
    <model type="ivshmem-plain"/>
    <size unit="M">32</size>
    <address type="pci" domain="0x0000" bus="0x10" slot="0x02" function="0x0"/>
  </shmem>
</devices>
<qemu:commandline>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT1"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT5"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT6"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT8"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT9"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT10"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT12"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT14"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT15"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT16"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT17"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT18"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT19"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT20"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT21"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT22"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT23"/>
  <qemu:arg value="-acpitable"/>
  <qemu:arg value="file=/home/user/acpi-dumps/SSDT24"/>
  <qemu:arg value="-acpitable"/>
```

```
    <qemu:arg value="file=/home/user/acpi-dumps/SSDT25"/>
    <qemu:arg value="-acpitable"/>
    <qemu:arg value="file=/home/user/dummy_osi.aml"/>
    <qemu:arg value="-acpitable"/>
    <qemu:arg value="file=/home/user/thermal_zone_ssdt.aml"/>
    <qemu:arg value="-acpitable"/>
    <qemu:arg value="file=/home/user/fake_battery.aml"/>
    <qemu:arg value="-smbios"/>
    <qemu:arg value="type=0,version=20250601,date=06/01/2025,uefi=true"/>
    <qemu:arg value="-smbios"/>
    <qemu:arg value="type=1,serial=20250601-10238,uuid=0634f858-4d05-4b56-95f6-d22f91052ee0"/>
    <qemu:arg value="-smbios"/>
    <qemu:arg value="type=2,serial=20250601-13452382"/>
    <qemu:arg value="-smbios"/>
    <qemu:arg value="type=3,serial=20250601-345666"/>
    <qemu:arg value="-smbios"/>
    <qemu:arg value="type=4,sock_pfx=U3E1,manufacturer=Intel(R) Corporation,version=11th Gen Intel(R
    <qemu:arg value="-smbios"/>
    <qemu:arg value="type=8,internal_reference=J1,external_reference=P1,connector_type=3,port_type=2
    <qemu:arg value="-smbios"/>
    <qemu:arg value="type=17,loc_pfx=Controller0-ChannelA-DIMM0,bank=BANK 0,manufacturer=Micron,seri
    <qemu:arg value="-smbios"/>
    <qemu:arg value="type=17,loc_pfx=Controller1-ChannelA-DIMM0,bank=BANK 0,manufacturer=Micron,seri
  </qemu:commandline>
  <qemu:override>
    <qemu:device alias="sata0-0-0">
      <qemu:frontend>
        <qemu:property name="rotation_rate" type="unsigned" value="1"/>
        <qemu:property name="discard_granularity" type="unsigned" value="0"/>
      </qemu:frontend>
    </qemu:device>
  </qemu:override>
</domain>
```

E QEMU (Hypervisor-Phantom) on Windows resources

E.1 Hypervisor-Phantom on MSYS2 patch

```
diff --git a/Hypervisor-Phantom/functions/spoof_qemu_patch.sh b/Hypervisor-Phantom/functions/spoof_q
index 2418f57..58c0f62 100755
--- a/Hypervisor-Phantom/functions/spoof_qemu_patch.sh
+++ b/Hypervisor-Phantom/functions/spoof_qemu_patch.sh
@@ -27,17 +27,7 @@ readonly FAKE_BATTERY_ACPITABLE="${PATCH_DIR}/fake_battery.dsl"

 REQUIRED_PKGS_Arch=(
   # Basic Build Dependencie(s)
-  acpica base-devel dmidecode glib2 ninja python-packaging
-  python-sphinx python-sphinx_rtd_theme gnupg
-
-  # Spice Dependencie(s)
-  spice gtk3
-
-  # USB passthrough Dependencie(s)
-  libusb
-
-  # USB redirection Dependencie(s)
-  usbredir
```

```
+  git curl python3 mingw-w64-x86_64-spice  mingw-w64-x86_64-spice-gtk mingw-w64-x86_64-gtk4 mingw-w
 )

 REQUIRED_PKGS_Debian=(
@@ -420,13 +410,14 @@ compile_qemu() {
               --enable-usb-redir \
               --enable-spice \
               --enable-spice-protocol \
-              --disable-werror &>> "$LOG_FILE"
+              --disable-werror \
+              --prefix=C:/Users/20220731/bin

   fmtr::log "Building QEMU"
-  make -j"$(nproc)" &>> "$LOG_FILE"
+  make -j"$(nproc)"

   fmtr::log "Installing QEMU"
-  sudo make install &>> "$LOG_FILE"
+  make install
   fmtr::info "Compilation finished!"

 }
```

References

[1] *Accelerated Graphics Port*. URL: https://en.wikipedia.org/wiki/Accelerated_Graphics_Port. (accessed: 17/06/2025).

[2] *Advanced Configuration and Power Interface (ACPI) Specification, Version 6.4*. Accessed: June 20, 2025. UEFI Forum. Beaverton, OR, Jan. 2021. URL: https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/.

[3] *Al-Khaser: Public malware techniques used in the wild: Virtual Machine, Emulation, Debuggers, Sandbox detection.* URL: https://github.com/ayoubfaouzi/al-khaser. (accessed: 18/06/2025).

[4] *CloakBox: Bypass of virtual machine detection using a fork of VirtualBox*. URL: https://github.com/Batlez/CloakBox. (accessed: 12/06/2025).

[5] *cuckoosandbox/cuckoo: Cuckoo Sandbox is an automated dynamic malware analysis system*. URL: https://github.com/cuckoosandbox. (accessed: 20/06/2025).

[6] *Hatching - Automated malware analysis solutions*. en. Mar. 2019. URL: https://www.hatching.io (visited on 06/20/2025).

[7] *Hypervisor-Phantom, Advanced Malware Analysis Tool*. URL: https://github.com/Scrut1ny/Hypervisor-Phantom. (accessed: 2/06/2025).

[8] *Installing CAPE — CAPE Sandbox v2.2 Book*. URL: https://capev2.readthedocs.io/en/latest/installation/host/installation.html (visited on 06/20/2025).

[9] *Interactive session — CAPE Sandbox v2.2 Book*. URL: https://capev2.readthedocs.io/en/latest/usage/interactive_desktop.html (visited on 06/20/2025).

[10] *kevoreilly/CAPEv2: Malware Configuration And Payload Extraction*. URL: https://github.com/kevoreilly/CAPEv2. (accessed: 20/06/2025).

[11] Bc Dominik Kouba. "Analyzing the execution of malware in a sandbox using hierarchical multiple instance learning". en. In: (). URL: https://dspace.cvut.cz/bitstream/handle/10467/95327/F3-DP-2021-Kouba-Dominik-malware_analysis.pdf?sequence=-1&isAllowed=y.

[12] *KVM setup for tarkov2*. URL: https://www.unknowncheats.me/forum/escape-from-tarkov/500926-kvm-setup-tarkov-2.html. (accessed: 18/06/2025).

[13] *Minimal System 2*. URL: https://www.msys2.org/. (accessed: 15/06/2025).

[14] *PAFish: a testing tool that uses different techniques to detect virtual machines and malware analysis environments in the same way that malware families do*. URL: https://github.com/a0rtega/pafish. (accessed: 18/06/2025).

[15] *PolyOS Developer Guides: How to build QEMU on Windows 10/11*. URL: https://polyos.iscas.ac.cn/en/docs/developer-guides/build-qemu/on-windows/. (accessed: 18/06/2025).

[16] *QEMU generic and open source machine & userspace emulator and virtualizer*. URL: https://github.com/qemu/qemu.git. (accessed: 20/06/2025).

[17] *Safe Exam Browser, Version 3.x*. URL: https://github.com/SafeExamBrowser/seb-win-refactoring. (accessed: 27/05/2025).

[18] UEFI Forum. "ACPI Machine Language (AML) Specification". In: *Advanced Configuration and Power Interface (ACPI) Specification, Version 6.4*. Accessed: June 20, 2025. Beaverton, OR: UEFI Forum, Jan. 2021. Chap. 20. URL: https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/20_AML_Specification/AML_Specification.htm.

[19] *VMAware: VM detection library and tool*. URL: https://github.com/kernelwernel/VMAware. (accessed: 18/06/2025).

[20] *What is CAPE? — CAPE Sandbox v2.2 Book*. URL: https://capev2.readthedocs.io/en/latest/introduction/what.html#architecture (visited on 06/20/2025).

[21] *Windows 10 Enterprise — Microsoft Evaluation Center*. en-US. URL: http://www.microsoft.com/en-us/evalcenter/evaluate-windows-10-enterprise (visited on 06/20/2025).