Detecting Sandboxes Without Syscalls

pentest.party/posts/2024/detecting-sandboxes-without-syscalls

19.04.2024 · dadevel

The PEB, TEB and KUSER_SHARED_DATA structs are mapped into the memory space of every process. They provide a wealth of information to the process and can be accessed without having to perform syscalls. Using them for anti-debugging is widely known and documented for example by CheckPoint. But they can also be used for stealthy anti-sandbox and anti-vm checks.

The following sandbox indicators might be interesting for both malware developers and sandbox vendors. The code assumes the struct definitions from VX-API.

```
constexpr uint32_t TICKS_PER_SECOND = 10'000'000;

const PEB* peb = GetPeb();
const KUSER_SHARED_DATA* ksd = GetKUserSharedData();

const uint32_t boot_count = ksd->BootId;

const uint32_t cpu_core_count1 = peb->NumberOfProcessors;
const uint32_t cpu_core_count2 = ksd->ActiveProcessorCount;

const double ram_size = static_cast<double>(ksd->NumberOfPhysicalPages) * 4096 / 1024 / 1024 / 1024;  // in gigabyte

const LARGE_INTEGER time1 = { .LowPart = ksd->InterruptTime.LowPart, .HighPart = ksd->InterruptTime.High2Time };
const uint32_t uptime = time1.QuadPart / TICKS_PER_SECOND / 60 / 60;  // in hours

const uint32_t os_major_version1 = ksd->NtMajorVersion;
const uint32_t os_major_version2 = peb->OSMajorVersion;

const bool license_valid = ksd->SystemExpirationDate.QuadPart == 0;

const bool secure_boot_enabled = ksd->DbgSecureBootEnabled;

const wchar_t* filepath = peb->ProcessParameters->ImagePathName.Buffer;

const LARGE_INTEGER time2 = { .LowPart = ksd->TimeZoneBias.LowPart, .HighPart = ksd->TimeZoneBias.High2Time };
const double timezone_offset = -1 * static_cast<double>(time2.QuadPart) / TICKS_PER_SECOND / 60 / 60;  // in hours from UTC

const wchar_t* env = peb->ProcessParameters->Environment;
const wchar_t* computername = GetEnvVar(env, L"COMPUTERNAME");
const wchar_t* userdomain = GetEnvVar(env, L"USERDOMAIN");
const wchar_t* username = GetEnvVar(env, L"USERNAME");

const wchar_t* workdir = peb->ProcessParameters->CurrentDirectory.DosPath.Buffer;
```

I implemented a small proof of concept that collects these indicators and performs a DNS query for each, so that they show up in the sandbox network log. The results from VirusTotal are shown below.

Sandbox	Boot Count	CPU Cores	RAM Size	OS Version	Licensed	Secure Boot	File Renamed	Uptime	Timezone	COMPUTERNAME	USERDOMAIN
My Gaming PC	523	16	32GB	10	yes	yes	no	74h	GMT+2	gamingstation	gamingstation
VirusTotal CAPE	64	4	4GB	10	yes	no	no	0h	GMT-7	desktop-RANDOM	desktop-RANDOM
VirusTotal Zenbox	62-76	4	8GB	10	yes	no	no	0h	GMT-7	desktop-RANDOM	desktop-RANDOM
Unknown Sandbox 1	24	2	2GB	10	yes	no	no	0h	GMT-7	laptop-RANDOM	laptop-RANDOM
Unknown Sandbox 2	5	1	4GB	10	yes	no	no	0h	GMT+2	horst-pc	horst-pc

> **Note:** The OS version can not be used to differentiate between Windows 10 and 11. It is set to 10 in both cases.

After testing various sandboxes it seems that boot count, Secure Boot status and uptime are strong generic sandbox indicators. Besides that there is still T1480/001 aka Environmental Keying to hide the payload itself from analysis.

The overall bottom line: Don't trust the analysis results of sandboxes too much.