**ADFSL2022 Microsoft Defender Will Be Defended MemoryRanger Prevents Blinding Windows AV**

|  |  |
| --- | --- |
| **Slide 1 Hello**  😊 Hi everyone! 😊  Thanks for inviting me. I am happy to be here.  Today I`ll show you how malware *can* bypass ... *can* disable Microsoft Defender by abusing system mechanisms and how to avoid this kind of attacks.  I think this topic is of crucial importance for endpoint security and anti-viruses’ experts. |  |
| **Slide 2 “WhoamI”** (fast: speed=3)  • This work was done in collaboration with one of my students from National Research Nuclear University.  Dénis investigated attacks on Microsoft Defender in his bachelor thesis under my supervision.  • I’m Igor. I’ve been exploring **/ˌəʊ ˈes/** OS security for more than 10 years.  I’ve been **curious** about discovering new attacks on the Windows kernel and find the ways to prevent them. |  |
| **Slide 3 “Agenda”** (fast: speed=2)   * Today, I`ll highlight that one of the recent malware trends is to disable endpoint security solutions. * I’ll show you the analysis of attacks on Microsoft Defender. * We`ll addr**é**ss the kernel attacks that can blind Microsoft Defender. * Finally, I will present you my MemoryRanger the tool I designed and customized to protect Microsoft Defender. |  |
| **Slide 4 “The Big Picture with Worm and Driver”**  After gaining an initial access cyber criminals always want |  |
| **Slide 5 “The Big Picture with Worm and Driver”**  to steal users’ credentials and perform lateral movement |  |
| **Slide 6 “The Big Picture with Worm and Driver”**  to install a backdoor and take the control for a long time |  |
| **Slide 7 “The Big Picture with Worm and Driver”**  to encrypt users’ files and demand a ransom payment. |  |
| **Slide 8 “The Big Picture with Worm and Driver”**  To achieve these goals attackers *can* download and install the malware.  To avoid signature-based detection on firewall stage attackers *can* use password-protected **/ ˈɑːrkaɪvs /** archives. |  |
| **Slide 9 “The Big Picture with Worm and Driver”**  Attackers *can* /**extrEct**/ extract the packed malware sample |  |
| **Slide 10 “The Big Picture with Worm and Driver”**  and launch malware to reach all desired goals. |  |
| **Slide 11 “Kernel Mode Restrictions”**  But anti-viruses and security solutions *can* detect all these malware samples and remove them easily. |  |
| **Slide 12 “Kernel Mode Restrictions”**  Attackers are always up to disable or blind security solutions. |  |
| **Slide 13 “Kernel Mode Restrictions”**  Such attacks result in disabling security solutions, and after that |  |
| **Slide 14 “Kernel Mode Restrictions”**  The malware can be installed without triggering any security reaction, because the security solutions are disabled. |  |
| **Slide 15 “Kernel Mode Restrictions”**  Criminals are always looking for the ways to attack anti-viruses using both. |  |
| **Slide 16 “Kernel Mode Restrictions”**  user-mode techniques. |  |
| **Slide 17 “Kernel Mode Restrictions”**  And kernel mode techniques.  This is a serious problem for security solutions |  |
| **Slide 18 “Kernel Mode Restrictions”**  even for Microsoft Defender. |  |
| **Slide 19 “Kernel Mode Restrictions”**  You know that, Microsoft Defender is installed as a primary anti-virus on more than half a billion devices.  The partner director at Microsoft underlined that Microsoft Defender is facing a **/hjuːdʒ/** **hu-uge** number of various attacks. |  |
| **Slide 20 “Attack vectors on Microsoft Defender-NULL\_SLIDE”**  Let me show you the current attack vectors on Microsoft Defender. |  |
| **Slide 21 “Attack vectors on Microsoft Defender”**  All attacks *can* be divided into two groups: user-mode and kernel-mode attacks. Regarding user-mode vectors: |  |
| **Slide 22 “Attack vectors on Microsoft Defender”**  Attackers *can* change the configurations of Microsoft Defender. |  |
| **Slide 23 “Attack vectors on Microsoft Defender”**  Attackers *can* implant the code into the white listed processes for example to access Internet without any **o**bstacles. |  |
| **Slide 24 “Attack vectors on Microsoft Defender”**  Attackers *can* confuse Microsoft Defender by modifying the files content on the disk after the image has been mapped. |  |
| **Slide 25 “Attack vectors on Microsoft Defender”**  Attackers *can* use various techniques to bypass file scanning engine of Microsoft Defender: such as packing and obfuscation. |  |
| **Slide 26 “Attack vectors on Microsoft Defender”**  Attackers *can* manipulate with NT symb**ó**lic links, which pushes Microsoft Defender to follow the wrong path. |  |
| **Slide 27 “Attack vectors on Microsoft Defender”**  Attackers *can* bypass kernel callbacks to spoof the process name, so that Microsoft Defender will receive a fake app name. |  |
| **Slide 28 “Attack vectors on Microsoft Defender”**  Attackers *can* sandbox … *can* isolate Microsoft Defender so that it cannot access any file or process.  This attack was published just a couple months ago. |  |
| **Slide 29 “Attack vectors on Microsoft Defender”**  But this attack *can* be blocked by Windows feature named ‘trust label’. |  |
| **Slide 30 “Attack vectors on Microsoft Defender”**  My student Dénis has independently discovered the kernel attack that produced the same result and cannot be blocked by ‘trust labels’. |  |
| **Slide 31 “Attack vectors on Microsoft Defender”**  One more kernel attack *can* blind Microsoft Defender by attacking Event Tracing for Windows, EeeTeeW. |  |
| **Slide 32 “Attack vectors on Microsoft Defender”**  Attackers *can* disable Protection Process Light, PeePeeL to terminate Microsoft Defender. |  |
| **Slide 33 “Attack vectors on Microsoft Defender”**  Attackers can blind security solutions by removing kernel callbacks. |  |
| **Slide 34 “Attack vectors on Microsoft Defender”**  Attackers *can* terminate the Microsoft Defender process by calling the termination routine from a kernel driver. |  |
| **Slide 35 “Attack vectors on Microsoft Defender”**  To show the details of the discovered attack let me introduce you Microsoft Defender. |  |
| **Slide 36 “”** |  |
| **Slide 37 “”**  Microsoft Defender has about 20 years old history /releas~~z~~e/ releasing as a free anti-spyware program and |  |
| **Slide 38 “”**  nowadays, it is a leader in the endpoint protection platforms. |  |
| **Slide 39 “”**  let me uncover Microsoft Defender components. |  |
| **Slide 40 “”**  Microsoft Defender includes User mode applications such as processes and services and kernel mode drivers. |  |
| **Slide 41 “”**  All in all, there are about 10 apps and 6 drivers. |  |
| **Slide 42 “”**  In this research we analyzed Microsoft Malware Protection Engine process MsMpEng and  Microsoft Defender file filter driver. |  |
| **Slide 43 “”**  The process is just a normal windows app that *can* detect the malware |  |
| **Slide 44 “”**  on the disk |  |
| **Slide 45 “”**  and in the memory. |  |
| **Slide 46 “”**  One of its DLL modules named MpEngine.dll implements the signature based-detection. It is the brain of anti-virus engine that decides whether the input is malicious or not. |  |
| **Slide 47 “”**  An idea of the attack is to block the brain from receiving input data.  I will show how Windows built-**in** mechanism *can* be used to disable Microsoft Defender.  Let’s look at the M**á**ndat**ó**ry Integrity Control (MIC) |  |
| **Slide 48 “”**  Purum- Purum |  |
| **Slide 49 “”**  Mandatory Integrity Control, MICee has been added in addition to DeeACee.  MIC was designed to isolate untrusted apps from the rest of the OS.  Here is an example of Integrity Levels for the running processes and files on the disk. |  |
| **Slide 50 “”**  Each time any app tries to access the object for example open a file.  The Security Reference Monitor checks whether this access is allowed. |  |
| **Slide 51 “”**  To **/allÁw/** allow access it has passed both checks. But if MIC check didn’t pass the access will be blocked. |  |
| **Slide 52 “”**  But attackers can patch integrity level for the Defender so that any Defender’s access attempt will be blocked.  We have tested this attack. |  |
| **Slide 53 “”**  However, this manipulation is not enough and Microsoft Defender blocks creating a malware process using this  **/ˈsté-ɪtəs/** status. |  |
| **Slide 54 “”**  My research has revealed that this status is returned by Microsoft Defender file-filter driver.  It means that Microsoft Defender can still detect malware in apps memory. But how? |  |
| **Slide 55 “”**  Burum-burum |  |
| **Slide 56 “”**  Microsoft Defender process has Debug Privilege  As we know, for process with Debug privilege SRM always **/allÁw/** allows **fúll** access without any checks. |  |
| **Slide 57 “”**  Attackers *can* revoke debug privilege from Microsoft Defender to restrict this access. |  |
| **Slide 58 “”**  The mentioned privilege is stored in token structure in memory. And *can* be patched by attackers.  Let’s see who it can happen. |  |
| **Slide 59 “”**  Burum-burum |  |
| **Slide 60 “”**  At first, attackers |  |
| **Slide 61 “”**  load the kernel driver to perform both actions: |  |
| **Slide 62 “”**  to overwrite the IntegrityLevelIndex and revoke process privilege |  |
| **Slide 63 “”**  And after that they extract and launch a malware sample. |  |
| **Slide 64 “”**  Let’s see whether these manipulations will be enough to disable Microsoft Defender. |  |
| **Slide 65 “”**  A few words about the infection part. |  |
| **Slide 66 “”**  The Mimikatz app will be used as a malware. We initially zipped mimikatz with password. |  |
| **Slide 67 “”**  Here is the batch script that extracts the Mimikatz from the archive file and launch it.  Let’s see how it can happen. |  |
| **Slide 68 “”**  <demo> |  |
| **Slide 69 “”**  We *can* see that Microsoft Defender has been completely disabled without triggering any Patch Guard reaction. |  |
| **Slide 70 “”**   * This attack results in sandboxing or isolating Microsoft Defender from the rest of the OS. * The key point is that we do not terminate Microsoft Defender * It’s still running but it cannot detect malware on the disk and in the memory. * Let me show how this attack can be blocked. |  |
| **Slide 71 “”**  Burum-burum |  |
| **Slide 72 “”**  In Windows OS kernel drivers share the same memory space with the OS kernel. |  |
| **Slide 73 “”**  There is no built-in mechanism to restrict illegal access attempts in kernel memory. |  |
| **Slide 74 “”→ 75 “”→ 76 “”**  I designed MemoryRanger to solve this problem by allocating a separate kernel enclave for each newly loaded driver.  2→3  As a result, drivers are running inside separate enclaves with different memory access permissions. |  |
| **Slide 77 “”**  This scheme helps to block illegal access attempts with affordable performance degradation. |  |
| **Slide 78 “”**  Originally, I designed MemoryRanger as a tool, as a framework to monitor access to the kernel data.  Now it can hook kernel functions; it can locate various sensitive data in kernel; and it can be easily customized.  Let’s have a look. |  |
| **Slide 79 “”**  Here are the examples of MemoryRanger customizations that block various new attacks on Windows kernel.  In this research MemoryRanger has been customized in the following way. |  |
| **Slide 80 “”**  Without MemoryRanger attacker’s driver can get access to the Token structure. |  |
| **Slide 81 “”**  MemoryRanger traps loading of attacker’s driver and moves it to the separate allocated enclave.  In this enclave the Token structure cannot be accessed. Let’s see how it can happen. |  |
| **Slide 82 “”**  <demo> |  |
| **Slide 83 “”**  Let me /ˈriːkap/ ré-cáp very briefly on what we have discussed so far.   1. First of all, driver-based attacks are crucial. 2. Intruders are plotting to bypass endpoint security solutions and their target number one is Microsoft Defender. 3. Hackers can apply M**á**ndatory Integrity Control and Token Privilege against Microsoft Defender to isolate it. 4. This is a very serious attack, because it blinds vast majority of anti-viruses without triggering any Patch Guard reaction. 5. I have presented my MemoryRanger, that customized version can prevent this kind of attacks. |  |
| **Slide 84 “”**  Thank you! |  |