**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 tamper with security solution**s**, and we’ll give you a new attack that can blind vast majority of antiviruses including Microsoft Defender. * Finally, I will present you my MemoryRanger the tool I designed and customized to protect Microsoft Defender. |  |
| **Slide 4 “Disclaimer”** (fast: speed=2)  This research is designed to inform the world security community, without promotion any products and solutions. |  |
| **Slide 5 “The Big Picture with Worm and Driver”**  After gaining an initial access cyber criminals always want |  |
| **Slide 6 “The Big Picture with Worm and Driver”**  to steal users’ credentials and perform lateral movement |  |
| **Slide 7 “The Big Picture with Worm and Driver”**  to install a backdoor and take the control for a long time |  |
| **Slide 8 “The Big Picture with Worm and Driver”**  to encrypt users’ files and demand a ransom payment. |  |
| **Slide 9 “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 10 “The Big Picture with Worm and Driver”**  Attackers *can* /**extrEct**/ extract the packed malware sample |  |
| **Slide 11 “The Big Picture with Worm and Driver”**  and launch malware to reach all desired goals. |  |
| **Slide 12 “Kernel Mode Restrictions”**  But anti-viruses and security solutions *can* detect all these malware samples and remove them easily. |  |
| **Slide 13 “Kernel Mode Restrictions”**  Attackers are always up to disable or blind security solutions. |  |
| **Slide 14 “Kernel Mode Restrictions”**  Such attacks result in disabling security solutions, and after that |  |
| **Slide 15 “Kernel Mode Restrictions”**  The malware can be installed without triggering any security reaction, because the security solutions are disabled. |  |
| **Slide 16 “Kernel Mode Restrictions”**  Criminals are always looking for the ways to attack anti-viruses using both. |  |
| **Slide 17 “Kernel Mode Restrictions”**  user-mode techniques. |  |
| **Slide 18 “Kernel Mode Restrictions”**  And kernel mode techniques.  This is a serious problem for security solutions |  |
| **Slide 19 “Kernel Mode Restrictions”**  even for Microsoft Defender.  ~~Let’s have a look.~~ |  |
| **Slide 20 “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 21 “Disabling Microsoft Defender from Twitter”**  Many security experts share their ideas about how to disable and bypass the Defender.  They discussed: |  |
| **Slide 22 “Disabling Microsoft Defender from Twitter”**   * code injection; |  |
| **Slide 23 “Disabling Microsoft Defender from Twitter”**   * packing malware to bypass signature detection. |  |
| **Slide 24 “Disabling Microsoft Defender from Twitter”**   * **/məˌnɪpjʊˈleɪʃ(ə)n/**  mani**pu**lation with exclusion path of Microsoft Defender. |  |
| **Slide 25 “Disabling Microsoft Defender from Twitter”**   * abusing Trusted Installer to stop the WinDefender service. |  |
| **Slide 26 “Disabling Microsoft Defender from Twitter”**  Hackers l**é**verage these techniques in their malware. |  |
| **Slide 27 “Disabling Microsoft Defender: Malware”**  Here are the malware examples which are able to bypass Microsoft Defender. |  |
| **Slide 28 “Disabling Microsoft Defender: Malware”**   * Ragnarok ransomware – that modifies registry; |  |
| **Slide 29 “Disabling Microsoft Defender: Malware”**   * Zloader Banking Trojan – that uses PowerShell scripting; |  |
| **Slide 30 “Disabling Microsoft Defender: Malware”**   * Kraken Botnet – that adds itself in the exclusion path. |  |
| **Slide 31 “Disabling Microsoft Defender: Malware”**  Various modern malware programs are attempting to disable Microsoft Defender. |  |
| **Slide 32 “Academic papers and blogs about Attacks on Microsoft Defender”**  **/ˈɡləʊb(ə)l/** Gl**o-u**bal cyber security community widely discussed the attacks on Microsoft Defender.  Here are some recent academic papers, blogs and research talks that show how to bypass Microsoft Defender. |  |
| **Slide 33 “Attack vectors on Microsoft Defender-NULL\_SLIDE”**  Let me show you the current attack vectors on Microsoft Defender. |  |
| **Slide 34 “Attack vectors on Microsoft Defender”**  All attacks *can* be divided into two groups: user-mode and kernel-mode attacks |  |
| **Slide 35 “Attack vectors on Microsoft Defender”**  User-mode attacks target mostly user-space components, while kernel-mode attacks can tamper with kernel mode parts. Regarding user-mode vectors: |  |
| **Slide 36 “Attack vectors on Microsoft Defender”**  Attackers *can* change the configurations of Microsoft Defender and even terminate its process. |  |
| **Slide 37 “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 38 “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 39 “Attack vectors on Microsoft Defender”**  Attackers *can* use various techniques to bypass file scanning engine of Microsoft Defender: such as packing and obfuscation. |  |
| **Slide 40 “Attack vectors on Microsoft Defender”**  Attackers *can* manipulate with NT symb**ó**lic links, which pushes Microsoft Defender to follow the wrong path. |  |
| **Slide 41 “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 42 “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 43 “Attack vectors on Microsoft Defender”**  But this attack *can* be blocked by Windows feature named ‘trust label’. |  |
| **Slide 44 “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 45 “Attack vectors on Microsoft Defender”**  One more kernel attack *can* blind Microsoft Defender by attacking Event Tracing for Windows, EeeTeeW. |  |
| **Slide 46 “Attack vectors on Microsoft Defender”**  Attackers *can* disable Protection Process Light, PeePeeL to terminate Microsoft Defender. |  |
| **Slide 47 “Attack vectors on Microsoft Defender”**  Attackers can blind security solutions by removing kernel callbacks. |  |
| **Slide 48 “Attack vectors on Microsoft Defender”**  Attackers *can* terminate the Microsoft Defender process by calling the termination routine from a kernel driver. |  |
| **Slide 49 “Attack vectors on Microsoft Defender”**  To show the details of the discovered attack let me highlight some recent trends in driver-based attacks. |  |
| **Slide 50 “”**  Burum-burum |  |
| **Slide 51 “”**  Using the first technique hackers can /ɪnˈst**ɔː**l/ inst**a**ll any signed **vulnerable** driver and exploit it**s** bugs to access kernel. |  |
| **Slide 52 “”**  Researchers from Rapid7 showed about 30 malware examples that exploit various legítimate drivers. |  |
| **Slide 53 “”**  Attackers *can* also use leaked digital certificates to sign their malware drivers. Let’s look at some cases. |  |
| **Slide 54 “”**  The notorious hacker’s group targeted NVIDIA to steal confidential information. |  |
| **Slide 55 “”**  The data leakage has been confirmed. |  |
| **Slide 56 “”**  And the stolen digital certificates have already been used to sign malware. |  |
| **Slide 57 “”**  Hackers *can* easily buy leaked certificates using Dark Web Market. |  |
| **Slide 58 “”**  Now, I want to **/raize/** raise concern about Windows Hardware Cert**í-**f**í**cation program. |  |
| **Slide 59 “”**  You know that developers can submit their drivers to the Windows Hardware Quality Labs that checks whether this driver is compatible. If the driver passes the check, it will be signed using a special digital signature.  But Microsoft has accidentally signed malware drivers, such as |  |
| **Slide 60 “”**  Netfilter driver {pause} and |  |
| **Slide 61 “”**  FiveSys driver. Both of them are malware. |  |
| **Slide 62 “”**  Another big trend is malware implants working on firmware-level. |  |
| **Slide 63 “”**  Such bugs in UEFI firmware can impact millions of devices. |  |
| Slide 64 “”  We can conclude that driver-based attacks are still very important. |  |
| **Slide 65 “”**  let me introduce you Microsoft Defender. |  |
| **Slide 66 “”**  Microsoft Defender has about 20 years old history /releas~~z~~e/ releasing as a free anti-spyware program and |  |
| **Slide 67 “”**  nowadays, it is a leader in the endpoint protection platforms. |  |
| **Slide 68 “”**  let me uncover Microsoft Defender components. |  |
| **Slide 69 “”**  Microsoft Defender includes User mode applications such as processes and services and kernel mode drivers. |  |
| **Slide 70 “”**  All in all, there are about 10 apps and 6 drivers. |  |
| **Slide 71 “”**  In this research we analyzed Microsoft Malware Protection Engine process MsMpEng and  Microsoft Defender file filter driver. |  |
| **Slide 72 “”**  The process is just a normal windows app that *can* detect the malware |  |
| **Slide 73 “”**  on the disk |  |
| **Slide 74 “”**  and in the memory. |  |
| **Slide 75 “”**  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 76 “”**  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 77 “”**  Purum- Purum |  |
| **Slide 78 “”**  Each time any app for example Microsoft Defender tries to access the object for example open a file. |  |
| **Slide 79 “”**  the Security Reference Monitor checks whether this access is allowed. |  |
| **Slide 80 “”**  It was originally based on Discretionary Access Control, DeeACee |  |
| **Slide 81 “”**  Mandatory Integrity Control, MICee has been added in addition to DeeACee.  MIC was designed to isolate untrusted apps from the rest of the OS. |  |
| **Slide 82 “”**  Now we have two different access checks. |  |
| **Slide 83 “”**  DeeACee checks privileges using access token and security descriptor. |  |
| **Slide 84 “”**  While MICee reads and compares “Integrity Level” from access token and “Integrity Label” from object descriptor. |  |
| **Slide 85 “”**  Windows first checks an access request against MIC |  |
| **Slide 86 “”**  and if it passes, then it checks DAC. |  |
| **Slide 87 “”**  To **/allÁw/** allow access it has passed both checks. But if MIC check didn’t pass the access will be blocked. |  |
| **Slide 88 “”**  But attackers can patch integrity level for the Defender so that any Defender’s access attempt will be blocked.  To demonstrate this attack let me first focus on Integrity Levels. |  |
| **Slide 89 “”**  Purum- Purum |  |
| **Slide 90 “”**  Windows supports several Integrity Levels to limit access to the resource: |  |
| **Slide 91 “”**  Untrusted and Low Integrity Levels are usually used by browsers to isolate web threats from the OS. |  |
| **Slide 92 “”**  Most apps that we use are running with Medium Integrity Level, such as explorer, and notepad. |  |
| **Slide 93 “”**  High Integrity Level is assigned with apps launched by User Account Control, UAC. |  |
| **Slide 94 “”**  System Integrity Level is used only by system-level apps, such as Microsoft Defender, to protect them from other apps. |  |
| **Slide 95 “”**  Here is an example of Integrity Levels for the running processes and files on the disk. |  |
| **Slide 96 “”**  MIC implements the Bell-LaPadula access control model, you know it.  For example, an app tries to get write access to the file. |  |
| **Slide 97 “”**  According to Bell-LaPadula model, access check compares integrity levels of app and a file. |  |
| **Slide 98 “”**  The access will be granted if the app’s integrity level is equal to {pause} or higher than {pause} the file’s integrity level. |  |
| **Slide 99 “”**  Therefore, apps with Low Integrity Levels cannot overwrite the most OS objects.  Let me show you how Integrity Levels are implemented in Windows. |  |
| **Slide 100 “”**  Burum-burum. |  |
| **Slide 101 “”**  For each app Windows allocates a separate data structure in kernel memory named EPROCESS.  1– |  |
| **Slide 102 , Slide 103**  2–3. |  |
| **Slide 104 “”**  This structure contains Process Name, Process ID, and other process attributes. |  |
| **Slide 105 “”**  This structure points to the Token structure, |  |
| **Slide 106 “”**  that includes all information about process privileges.  The set of process privileges is stored using Security identifiers, SIDs.  To support MIC the following two new fields have been added: |  |
| **Slide 107 “”**   * IntegrityLevelIndex and the corresponding SID\_AND\_ATTRIBUTES structure. * IntegrityLevelIndex is just an index into the array pointed by the UserAndGroups. |  |
| **Slide 108 “”**  This entry points to one of the integrity level SID. Examples of SIDs are here.  Let me show you how Windows gets these Integrity Levels. |  |
| **Slide 109 “”**  Burum-burum |  |
| **Slide 110 “”**  An app *can* get (or read) the integrity level of the target app by calling GetTokenInformation() with the parameter. |  |
| **Slide 111 “”**  The control goes to the NtQueryInformationToken, from ntdll and then to the kernel. We have a list of function calls. |  |
| **Slide 112 “”**  Finally, the control goes to the function SepLocateTokenIntegrity.  Let’s see inside it. |  |
| **Slide 113 “”**  The function receives the pointer to the token structure.  And it returns the pointer to the SID\_AND\_ATTRIBUTES. |  |
| **Slide 114 “”**  The function reads the IntegrityLevelIndex value. |  |
| **Slide 115 “”**  If it is minus one, the function returns zero. |  |
| **Slide 116 “”**  Or else the function returns the address of the structure, that points to the Integrity Level SID. |  |
| **Slide 117 “”**  The return value is used by the function SepCopyTokenIntegrity.  Let’s see how. |  |
| **Slide 118 “”**  The function receives the pointer to the token structure, and it fills the output pointer. |  |
| **Slide 119 “”**  The function first calls SepLocateTokenIntegrity to get pointer to the SID\_AND\_ATTRIBUTES structure. |  |
| **Slide 120 “”**  If the return value is non-zero the corresponding data are copied to the output. |  |
| **Slide 121 “”**  Or else, {pause} for the zero value, the untrusted SID and its Attributes are copied to the output. |  |
| **Slide 122 “”**  It means that by setting -1 to the IntegrityLevelIndex attackers can mark the target process as an untrusted.  Let’s see how it can happen. |  |
| **Slide 123 “”**  Burum-burum. |  |
| **Slide 124 “”**  At first, attackers can |  |
| **Slide 125 “”**  load kernel driver |  |
| **Slide 126 “”**  to overwrite the IntegrityLevelIndex. |  |
| **Slide 127 “”**  And after that they extract and launch a malware sample. |  |
| **Slide 128 “”**  We will see if Defender will be able to detect malware files |  |
| **Slide 129 “”**  and malware process. |  |
| **Slide 130 “”**  A few words about the infection part. |  |
| **Slide 131 “”**  The Mimikatz app will be used as a malware. We initially zipped mimikatz with password. |  |
| **Slide 132 “”**  Here is the batch script that extracts the Mimikatz from the archive file and launch it.  Let’s see how it can happen. |  |
| **Slide 133 “”**  <demo> |  |
| **Slide 134 “”**  Microsoft Defender fails to remove the file, but it is still able to block the running process. |  |
| **Slide 135 “”**  We’ve just seen some Microsoft Defender internals. It uses pending deletion state to remove the malware file.  Also, we’ve seen a new **/ˈsté-ɪtəs/** status that blocks creating a malware process. |  |
| **Slide 136 “”**  Let me show you the driver that returns this status. |  |
| **Slide 137 “”**  My research has revealed that this status is returned by Microsoft Defender file-filter driver. |  |
| **Slide 138 “”**  This driver registers a mini-filter to monitor file operations using pre- and post- processing callbacks.  I have discovered that post-create callback function returns this status for the malware apps. |  |
| **Slide 139 “”**  It means that Microsoft Defender can still detect malware in apps memory. But how? |  |
| **Slide 140 “”**  Burum-burum |  |
| **Slide 141 “”**  We know that malware detection engine is located in the Microsoft Defender process.  Each time the Defender process gains the access to the running process. |  |
| **Slide 142 “”**  Windows checks whether the privileges are enough using |  |
| **Slide 143 “”**  access token and security descriptor. |  |
| **Slide 144 “”**  However, the Defender process has Debug Privilege |  |
| **Slide 145 “”**  As we know, for process with Debug privilege SRM always **/allÁw/** allows **fúll** access without any checks. |  |
| **Slide 146 “”**  Attackers *can* revoke debug privilege from Microsoft Defender to restrict this access. |  |
| **Slide 147 “”**  The mentioned privilege is stored in token structure in memory. And *can* be patched by attackers.  Let’s see who it can happen. |  |
| **Slide 148 “”**  Burum-burum |  |
| **Slide 149 “”**  At first, attackers |  |
| **Slide 150 “”**  load the kernel driver to perform both actions: |  |
| **Slide 151 “”**  to overwrite the IntegrityLevelIndex and revoke process privilege |  |
| **Slide 152 “”**  And after that they extract and launch a malware sample. |  |
| **Slide 153 “”**  Let’s see whether these manipulations will be enough to disable Microsoft Defender. |  |
| **Slide 154 “”**  <demo> |  |
| **Slide 155 “”**  We *can* see that Microsoft Defender has been completely disabled without triggering any Patch Guard reaction. |  |
| **Slide 156 “”**   * 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. * Microsoft Defender has been disabled. What about other popular anti-viruses? |  |
| **Slide 157 “”**  Burum-burum |  |
| **Slide 158 “”**  Here are the results of checking the given attack against top Windows anti-viruses.  We can see that vast majority of AVs has been successfully disabled.  Let me show how this attack can be blocked. |  |
| **Slide 159 “”**  Burum-burum |  |
| **Slide 160 “”**  In Windows OS kernel drivers share the same memory space with the OS kernel. |  |
| **Slide 161 “”**  There is no built-in mechanism to restrict illegal access attempts in kernel memory. |  |
| **Slide 162 “”→ 163 “”→ 164 “”**  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 165 “”**  This scheme helps to block illegal access attempts with affordable performance degradation. |  |
| **Slide 166 “”**  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 167 “”**  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 168 “”**  Without MemoryRanger attacker’s driver can get access to the Token structure. |  |
| **Slide 169 “”**  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 170 “”**  <demo> |  |
| **Slide 171 “”**  Let’s discuss briefly what we’ve seen.  Without MemoryRanger attackers disable Microsoft Defender and install malware without any problem.  Microsoft Defender fails to remove the malware. |  |
| **Slide 172 “”**  But with MemoryRanger the situation is different. |  |
| **Slide 173 “”**  MemoryRanger allocates the default enclave for the OS kernel and all previously loaded drivers. |  |
| **Slide 174 “”**  MemoryRanger traps the loading of attacker’s driver and allocates a separate memory enclave for this driver. |  |
| **Slide 175 “”**  In this enclave MemoryRanger restricts any access attempt to the Token of Microsoft Defender. |  |
| **Slide 176 “”**  As a result, attackers have failed to disable Microsoft Defender.  This is the way how MemoryRanger protects Microsoft Defender. |  |
| **Slide 177 “”**  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 178 “”**  Thank you! |  |

Msmpeng has a child process named MsMpEngCP.exe, which is a kind of sandbox for software. Kernel mode drivers include the following components: file-filter, device filter, network filter, Early Launch AntiMalware, ELAM Driver etc.

Integrity level associated with processes, but what about objects? Objects also have an integrity level stored as part of their security descriptor, in a structure that is called the mandatory label.

**ETC**

|  |  |
| --- | --- |
| **Slide 179 “”**  An implant named MoonBounce *can* load a malicious driver. |  |
| **Slide 180 “Why kernel attacks are still important”**  let me first raise the question Why the kernel mode attacks are still dangerous? |  |
| **Slide 181 “Kernel Mode Restrictions”**  You remember that in Windows XP hackers *could* loaded any driver easily. |  |
| **Slide 182 “Kernel Mode Restrictions”**  Thanks for the Driver Signature Enforcement only digitally signed drivers could be loaded. |  |
| **Slide 183 “Kernel Mode Restrictions”**  But attackers had analyzed this security feature and discover the way how to bypass it.  Finally, they could load their own drivers without any **o**bstacles. |  |
| **Slide 184 “Kernel Mode Restrictions”**  Windows experts made another step ahead to protect Driver Signature Enforcement using Kernel Patch Protector and Kernel Data Protection technology that is based on hardware virtualization. |  |
| **Slide 185 “Kernel Mode Restrictions”**  But hackers are not going to give up. |  |
| **Slide 186 “”**  There are lot of vulnerable drivers with digital signatures, that can be used to evade security products. |  |
| **Slide 187 “”**  For example, a recent data-wiping malware abuses such driver. |  |
| **Slide 188 “”**  And yet Windows experts are fightinging with these security challenges. |  |
| **Slide 189 “”**  Microsoft launches a special web portal where users *can* submit suspicious drivers for analysis. |  |
| **Slide 190 “”**  Microsoft Defender feature named Vulnerable Driver Blocklist protects against malicious and exploitable drivers. |  |
| **Slide 191 “”**  One more newly introduced security feature provides blacklist-based validation for each attempt to load a driver.  We *can* see that attacks based on kernel drivers are still dangerous even for newest Windows OS. |  |
| **Slide 192 “”**  Here is the list of **all** Integrity Level SIDs.  It was interesting for me to reveal that Chromium browser designed by Google defines its own Integrity Levels. |  |
| **Slide 193 “”**  Here are the most popular Integrity Levels.  Let’s see how Windows gets these Integrity Levels. |  |

// Igór – Igo0301r highlight and press ALT+X