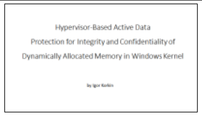
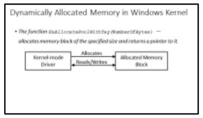
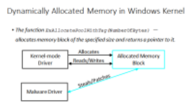
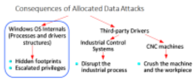
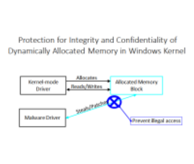
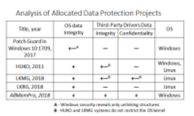
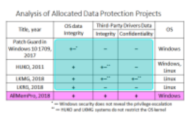
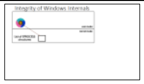
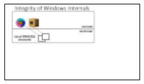
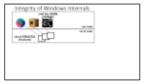
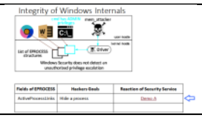
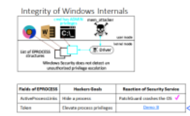
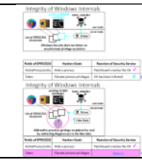
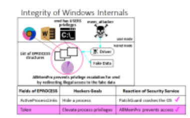
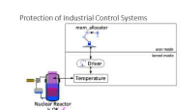
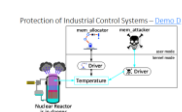
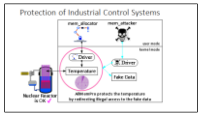
Hypervisor-Based Active Data Protection for Integrity and Confidentiality of Dynamically Allocated Memory in Windows Kernel

Slide 1 Hello Hi! I'm Igor. Thanks for coming. This is my first time in Texas and my 5th talk at the ADFSL Conference.	
Slide 1 Hello And today I'll tell you about my new project, which can protect allocated data.	
Slide 2 "Why do we focus on the dynamically allocated data in Windows kernel" First of all, what is a dynamically allocated data, and why we need to bother about it.	
Slide 3 "Dynamically allocated data in Windows kernel" Every time when drivers "make space" for the storage of data they call the function ExAllocatePoolWithTag. As a result, memory chunks with specified sizes are allocated "on the fly" during drivers run time. The dynamically allocated memory is very common in Windows Kernel.	
Slide 4 "Malware" And this allocated data can be easily attacked by malware driver. Let's look at the results of these malware attacks.	
Slide 5 "Allocated Data Attacks" Hackers can attack the allocated memory of Windows kernel and third-party drivers. By mani pulating Windows Kernel data, hackers can hide footprints and escalate privileges to steal as much data as they want.	
Slide 6 "Allocated Data Attacks" By mani pulating allocated data of third-party drivers' hackers can cause a real damage. Look at this. Nowadays there are a lot of Windows-based Industrial Control Systems. By attacking them malware can disrupt the industrial process. You remember, that in 2009 Stuxnet malware {a'} ravaged a nuclear facility in {a'} Iran.	
Slide 7 "Allocated Data Attacks" A similar target of attacks is the software for the machines with computer numerical control or	

Slide 8 “CNC machines” CNC machines. They manipulate machine shop tools using computer-programming inputs. Those machines are genius –	
Slide 9 “CNC machines” but hackers can cause the machine crushing and destroying the workpiece. We can prevent these risky situations –	
Slide 10 “” by focusing on protection of allocated data in memory.	
Slide 11 “” By preventing illegal access from a malware driver, we can guarantee` the integrity and confidentiality of allocated data. Nowadays there are a lot of projects, which protect allocated data.	
Slide 12 “Analysis of Allocated Data Protection Projects” Here is a summary table with some of those projects. Let’s start with the Windows built-in` security. Windows 10 has new security features but they do not fully protect allocated data in memory. For example, Patch Guard checks the integrity of some internal lists with allocated data. Windows does not protect allocated memory of third-party drivers. There are several research projects, which designed to protect the allocated data.	
Slide 13 “Analysis of Allocated Data Protection Projects” And you can see that there is no one Windows-based or Linux-based solution, which simultaneously deals with all these issues. I have developed AllMemPro to complete this gap. I am going to show how it works using two different scenarios.	
Slide 14 “” - Slide 15 “” - Slide 16 “Integrity of Windows Internals” Let’s start with the protection of allocated data in Windows kernel. Look. After launching a new process, a corresponding EeePROCESS structure is allocated in the kernel-mode memory. This structure includes a number of sensitive fields, for example ActiveProcessLinks and Token field.	
Slide 17 “Integrity of Windows Internals” Hackers can hide a process by changing the ActiveProcessLinks. Let’s look at Windows security response.	

(Integrity of Windows Internals - Part 1/3 - Hiding a Process) I am going to show you an example of hiding process using command line or cmd. Look. I start cmd.exe. Now if we check the process list we can see that cmd is in the process list and its process ID is 2-1-9-2, let’s try to hide it. I am launching MemAttacker, which imitates hacker’s activity. The MemAttacker loads a driver to manipulate kernel-mode memory. To hide this process, I use hide command and process ID. Now, let’s check the process list again and we can see that there is no cmd, but it still active. Now let’s wait for Patch Guard response, which is designed to prevent illegal memory modifications. We’ve been waiting just for 5 minutes, which is not too long, and see, that Patch Guard has crashed the OS. It reveals the unauthorized memory manipulations, which hide a cmd process. This act of crashing is very important because it blocks users’ activity for example, paying online and protects the operating system.	
Slide 18 “Integrity of Windows Internals” Okay, Windows security prevents hiding processes. But can it detect the process privilege escalation? Hacker can elevate process privileges by manipulating with Token field. Let’s walk the talk.	
(Integrity of Windows Internals - Part 2/3 - Escalating Process Privileges) Now I will {e’liveit} elevate privileges for cmd without hiding it. Only privileges. Let’s see it. I start cmd. Next, by using this command, I get its process ID. Well. Copy its PID. We can see that cmd has users’ privileges, which is expected, and it is not enough to disable for example the security service. Now we try to elevate the cmd privileges. I launch MemAttacker again. I use priv command with process ID to escalate process privileges. Done. Checking the privileges again. Now cmd has the highest privileges and it can even disable the security service. The privileges have been escalated. Okay. And so let’s wait for Patch Guard reaction. You can imagine that this time we’ve been waiting for as long as 5 hours and almost nothing happened. We’ve got only small notification, which can also be disabled. We have no crushing the OS. It means that Patch Guard didn’t prevent this invasion and the OS became infected.	

Slide 19 “” Slide 20 “” We have just seen that Windows security can’t reveal process privilege escalation. Let me show you how AllMemPro is able to prevent this drawback of Windows Security.	
Integrity of Windows Internals - Part 3/3 - AllMemPro Prevents Escalation of Process Privileges. First of all, I start DbgView and set all flags to see AllMemPro output. Now you see that all flags have been set. I launch AllMemPro console app. It loads the driver and activates the hypervisor. After that, I repeat steps from the previous video. I start cmd, get its process ID, and check its privileges. As we expected cmd has users’ privileges and can’t disable the security service. Let’s try to elevate its privileges once again. We load our MemAttacker. And use priv command with process ID to escalate privileges. Done! Let’s check the privileges. Cmd has still users’ privileges and can’t disable security service. It means that MemAttacker failed. But why it fails? Let’s go to DbgView. Here we see that AllMemPro is loaded and catches the loading of cmd.exe (optionally return to the 00:33 and continue with 01:33). Let’s scroll down and see that AllMemPro prevents illegal write access and protects the EPROCESS structure by using fake data. This is how my AllMemPro protects the OS.	
Slide 21 “Integrity of Windows Internals” You have just seen that AllMemPro blocks process privilege elevation. Let’s move on to the second scenario dealing with protection of Industrial Control Systems.	
Slide 22 “Protection of Industrial Control Systems” I use MemAllocator as an example of industrial software that controls a hypothetical nuclear reactor. MemAllocator loads a driver, which allocates the memory and sets an initial sensor reading. Let’s assume that this value is used as a temperature of our nuclear reactor.	
Slide 23 “Protection of Industrial Control Systems” Next, I launch MemAttacker to read and increase this temperate. You will see that this memory can be changed, without any security alerts.	

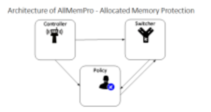
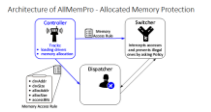
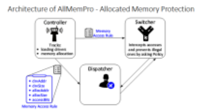
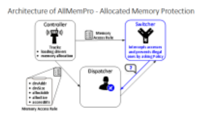
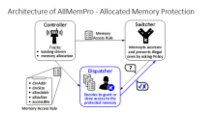
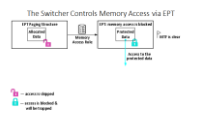
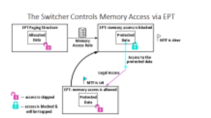
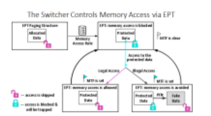
(Protection of Industrial Control Systems - Part 1/2 - Unauthorized Modification of Dynamically Allocated Memory) In the first place, I load DbgView to see the output from drivers. I set all flags to see debug outputs. Now, I start MemAllocator. This app plays the role of industrial software and sets the temperature of a nuclear reactor. To set the initial temperature I use starttemp – command. I use ABCD as a two-byte hexadecimal value. AB – is one byte, CD – is the second one. We can see that the software is mo`nitoring the temperature. Now hackers are launching MemAttacker in order to read this temperature. But, they need to know the corresponded memory address. Let’s assume they do know this address. We see that hackers are able to read the temperature. Here is the first byte and now they are trying to read the second byte. MemAttacker has read the second byte as well. It means that allocated data can be stolen. But can hackers change(?) the temperature? Let’s have a look. We assume that hackers want to change the temperature from CD-value to FF-value. They use write1 command with FF value. And we see that data has been changed. Now they try to modify the first byte. Let’s see the output. The data has been changed again. Now let’s read the temperate using the console of industrial software. We can see, that the data has been changed indeed. What about the next address? It also has been changed. Okay. Let’s check if Patch Guard causes BSOD in order to protects us. We had started timer and have been waited for 5 hours. 5 hours have passed – a BSOD hasn’t appeared, it means that OS is infected and our nuclear reactor is in {ai} danger. Let’s see what can happen. Here you can see a nuclear power plant and here is our hypothetical nuclear reactor. Look the temperature is {‘ins-sing} increasing and …	
Slide 24 “Protection of Industrial Control Systems” Let’s see how AllMemPro can prevent this kind of unwanted industrial disasters.	

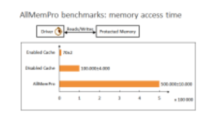
Protection of Industrial Control Systems - Part 2/2 - AllMemPro Prevents Illegal Access to the Allocated Memory

I’m starting DbgView to see all the debug output.
I’m launching AllMemPro. We can see that AllMemPro has been activated.
I’m starting MemAllocator, which plays the role of industrial software and activates a demo reactor with ABCD temperature.
If we check the debug output we can see that AllMemPro has trapped loading of MemAllocator driver and it has also trapped calling memory allocation routines.
Next I’ll copy the address with the temperature, which will be used by attackers.
We see that attackers <u>are trying</u> to read this data. But, this time attackers have read only <u>a zero value</u>.
Let’s see the output.
While MemAllocator is reading the ABCD value, MemAttacker is reading a zero value instead.
But why it reads a zero value?
We see that AllMemPro can prevent illegal READ access to every byte of protected data. AllMemPro is foisting the fake data to the attacker instead of the real one.
What about illegal WRITE access to the temperature?
I’m using write command. Let’s see the output.
You can see the attackers couldn’t change it, the temperature is still ABCD.
So AllMemPro protects data from illegal WRITE access <u>as well</u>.
May AllMemPro also block <u>any</u> access to this data even the <u>legal</u> one?
Let’s read this data legally, using the console app of industrial software.
We see that legal READ access works correctly.
What about legal WRITE access? It also works in a proper way.
We can conclude that AllMemPro allows legal access and blocks the unauthorized one.
The operating system and the nuclear reactor is fully protected.

Slide 25 “”

Let me show you how AllMemPro functions.

Slide 26 “AllMemPro Architecture” AllMemPro applies hardware virtualization technology and is has three components: the controller, the switcher, and the dispatcher.	
Slide 27 “AllMemPro Architecture” The Controller tracks the loading drivers and processes. It also tracks memory allocation routine calls. It decides which drivers and processes need to be protected using list of their names.	
Slide 28 “AllMemPro Architecture” Each time the protected driver allocates memory, the controller sends the following memory control rule to the Switcher and to the Dispatcher. This rule includes the loading addre’ss and size of the driver; the address and the size of the data, which this driver allocates.	
Slide 29 “AllMemPro Architecture” The switcher changes access to the protected memory areas. After receiving the control rule, it blocks the access to this memory. Then, the switcher intercepts access to this area and grants or prevents access by asking the Dispatcher.	
Slide 30 “AllMemPro Architecture” The Dispatcher decides to grant or to block memory access to the protected data. It saves each memory control rule from the Controller to the white list. Next, responding to the switcher, the dispatcher checks if driver allows accessing to this memory.	
Slide 31 “The Switcher” Let’s move on to the Switcher. It applies Extended Page Tables technology to control memory access. After the Switcher receives a rule from the controller, it resets access bits to the memory. As a result, any access to this memory always causes access violations. The switcher processes all violations according to the Dispatcher decision.	
Slide 32 “The Switcher” If this access is legal, the controller just allows it and sets Monitor Trap Flag. Because of this flag after the legal driver reads or writes the protected data the control goes to the switcher again. It resets Monitor Trap Flag and blocks any access to the sensitive data to be ready to protect it again.	
Slide 33 “The Switcher” However, for an illegal access, the Switcher changes PFN value of the protected page to the fake one, and allows access to this fake data. Finally, it sets Monitor Trap Flag. As a result, after the malware driver	

reads the fake data the control goes to the switcher again. Now the controller {ou-lar} resets Monitor Trap Flag and restores PFN value. The thing is that the switcher intercepts all access attempts to the protected data using only one EPT table. That is why the switcher causes significant {sorry} performance overhead.	
Slide 34 “Benchmarks Analysis” Look. I’ve measured the memory access duration to the protected data in three cases: with and without AllMemPro. And without AllMemPro I’ve measured the duration with enabled and disabled cache. You can see that the memory access duration with AllMemPro is 5 times higher than with disabled cache. But I am sure that this overhead can be reduced by using separate EPT structures for each driver.	
Slide 35 “Conclusions and Future Plans” Anyway, AllMemPro protects each byte of the allocated memory. If access attempts to the memory are not so frequent, my AllMemPro is applicable. For example, to prevent privilege escalation attacks. It isolates data from any driver access even from the Windows Kernel. AllMemPro causes a time overhead, but it seems to be able to prevent stealing protected data using the Spectre and Meltdown attacks. This research is still ongoing.	
Slide 36 “Thank you” Thank you! That’s it. Do you have any questions?