Disclaimer:

The translation of the original post in Russian is done as courtesy contribution of Rubos, Inc, and does not represents Rubos, Inc. opinion on the matters discussed, and neither Rubos, Inc. has any association with the post author.

We used Google auto-translation to produce bulk text and then corrected it manually.

Translator's remarks:

1. We fixed obvious typos or incorrectness of the original text, and its language as well, to make it more readable for English speaking audience. We corrected original, but not essential, text to make it clear what the author means. Such text translated "as it is" would really confuse our audience and definitely will not help in delivering of the author's message.
2. The post describes relatively old findings related to Intel chipset S5000X features used in S5000/S7000 based server motherboards, and which have been manufactured in China according to the label "Assembled in China" on these motherboards.
3. Our comments to the original text are identified as (comment: <our comment text>)

Published:12/26/2011

Chinese Add-ons: True Stories of virtualization, information security and computer spying

The post author remark: "it means that in the BMC flash memory of the server boards from China, and having Intel label, there is undeclared software module that acts as a host hypervisor, and which is embedded during manufacturing/production of the boards."

vladimir_krm

Current post source: http://xakep.ru/2011/12/26/58104/

I am not a professional in the field of information security; my area of interest is high-performance computing systems. The matters discussed below I discovered by a chance not a choice. I think that this true story highlights the problems associated with hardware virtualization much better than a dry presentation of facts.

I decided to use new Intel processors with the support of hardware virtualization even before the official announcement (in early 2007), and to create a unified computing system based on multiple servers, which would provide a single computational unit with SMP-architecture for the OS and applications. It required writing compact hypervisor with non-standard functionality. The main feature of it would not be a single computing resource sharing settings between different operating systems, but vice versa, pooling of resources of several computers into a single complex, which would be run by the same OS. In this case, the OS does not even have to guess whether it is dealing with a single system or with multiple servers. Hardware virtualization provides such an opportunity, although it was not originally intended for such purpose. Actually, the system in which the virtualization equipment would be used for High Performance Computation (HPC), is not created even until now. When I did that, I was a pioneer in this field.

Hypervisor, for this task of course, was written from scratch. It is essential to boot the OS on already virtualized platform, so even the first commands of the OS boot loader would be executed in a virtual environment. Then the hypervisor has to virtualize the CPU real mode and all other operating modes, and run virtualization immediately after hardware platform initialization and prior to OS boot.

Virtualization solution for this purpose should be a fully autonomous compact program module (code size up to 40-60 KB). Thus, it was overstatement to call that as "hypervisor", and I began to use the term "hyper-driver" because it more accurately conveyed the essence of the functional purpose of the software. Standard motherboards with hardware virtualization at this time were not yet available, however, thanks to the cooperation with the company Kraftway (comment: the company location is Russian Federation, see http://www.kraftway.com) I had access to pre-production models of processors and motherboards with the support of virtualization that are not yet officially released (so called "samples" were provided to Intel business partners). Therefore, the work began on this "sample" equipment.

The system was built, hyper-driver written, everything worked as planned. I should say that at this time the equipment was very virtualization vise "raw", and sometimes it refused to work as written in the documentation. We had to deal with literally every assembly instruction. Virtualization commands themselves were written in machine codes as there was no compiler which would support virtualization commands.

I was proud of the results, felt almost like a master of virtual worlds ... but my euphoria did not last long, only a month. I had to assemble a system based on first production motherboards, which had just appeared. However, the system did not work.

I began to investigate and finally realized that my system hangs while executing hardware virtualization commands. I had the impression that they either do not work at all or work somehow outside of standards. Freeze occurred only when the virtualized hardware was working in real mode, but if my system were to start from the protected mode after the OS loading, everything worked normally.

Professionals know that the first revision of Intel hardware virtualization did not support the CPU in real mode. This required an additional and large enough layer to emulate virtual x86. Since my hyper-driver runs before the operating system so that it can fully "believe" in the new virtual configuration, a small piece of the OS boot code runs in real mode of the processor. The system was dying right in the emulators of the real-mode handlers in hyper-driver. At first I thought it was a mistake somewhere; something was not understood, something forgotten. I checked every bit in the code, but did not find errors and began to blame not myself but colleagues from the abroad.

The first step was to replace the processors, but it did not help. By this time motherboard hardware virtualization was only in the BIOS, where it is initialized during the server power on, so I started to compare the BIOS in the motherboards (the same type of production boards with samples) - all was the same up to the last byte and the version number of the BIOS. I fell into a stupor, and, not knowing what to do, used a last resort – random, almost stupid, actions. I was not thinking, just stupidly probing, and finally get the BIOS code from the official Intel website and reloaded it in the motherboard, and then system started working ...

I was extremely surprised: BIOS number was the same, the images of the BIOS match byte by byte, but for some reason the “standard” China-made motherboard started to work only when I uploaded the BIOS taken from Intel site. Hence, the problem is in the motherboards, isn't it? But the only difference was in the labeling: the samples were labeled “Assembled Canada”, and the serial boards - “Assembled China”. It became clear that the boards from China contain additional software modules, embedded in the BIOS, and the standard analysis software does not see them. Apparently, they also worked with hardware virtualization and thus were able to hide the true contents of the BIOS. Now the reason of hanging of the hyper-driver became clear: these Chinese boards had two software systems working simultaneously on the same hardware virtualization level, which does not allow sharing of resources. I wanted to investigate this malicious BIOS software without a thought about a "backdoor", "undocumented features", etc. It was just an academic interest, and nothing more.

It must be said that, in parallel with the introduction of hardware virtualization, Intel radically changed the chipset. This chipset, having the number 5000X, is available in several versions so far. I/O Controller Hub 631xESB/632xESB of the chipset, (comment: which is also known as “ESB2 or “South Bridge”, and which name we will use in the text as well) is connected to the flash memory chip with its BIOS. It was almost unchanged since 2007, and is used as the base chip in a two-socket (CPU) systems. I downloaded the datasheet for the I/O Controller, have read the description and was blown away. It turns out that this new I/O Controller connects three flash memory devices: the first is the standard BIOS, the second is for the network controller programs, and the third is designed for the integration of Baseboard Management Controller (BMC) in the ESB2.

BMC (IMPI/BMC) is used for remote control and computer monitoring. It is indispensable for large server rooms, where, because of noise, heat, etc., is impossible for one to stay for a long time.

The fact that the BMC has own processor and thus, flash memory for its programs, of course, not a news, but until now, BMC with its processor and memory were put on add-on board, which could be installed in the motherboard; if you want – put it in. Now Intel has implemented these components inside of South Bridge, in fact, connecting this ESB2 to the system bus instead of using a dedicated network link (as IPMI standard describes BMC functions) for network service, and tunneled all service network traffic to the core network adapters. Then I found out from the documentation that the programs in the BMC flash memory (comment: – actually – flash for ESB2) are encrypted by special hardware cryptographic module, which is also integrated into the ESB2. Thus, decryption as well is done by this hardware module inside the ESB2. I had not come across such BMC implementation yet.

Below, there are some references concerning ESB2 from its documentation:

- ARC4 processor working at 62.5 MHz speed

- Interface to both LAN ports of Intel ® 631xESB/632xESB I / O Controller Hub allowing direct connection to the net and access to all LAN registers.

- Cryptographic module, supporting AES and RC4 encryption algorithms and SHA1 and MD5 authentication algorithms.

- Secured mechanism for loadable Regulated FW”.

However, the use of foreign cryptographic key with more than 40 bits is prohibited by a law in Russia, and here we see that each Intel server has encryption hardware with unknown keys of 256 bits. Moreover, these keys are used to encrypt the programs embedded in the flash memory chip of the motherboard at the production stage.

It means that in Russia the BMC part of ESB2, should be disabled. However, these units (comment: ESB2 and BMC in it), are always up and working, even if the computer is down; to power up ESB2 is enough if power cable is connected to external power outlet. That all seemed to me at this time as of secondary importance, since it was necessary to begin to figure out in which of the flash chips was a software module that works with hardware virtualization and interfering with my hyper-driver, and I began to experiment with firmware.

I was not even surprised when found out that the "add-on" hypervisor starts working just after reloading software in flash memory of BMC. Further, I understood that reverse engineering of the code (comment: including the illegal hypervisor) was blocked by the encryption, and was not possible at all without special hardware. I did not find documentation on the internal architecture of the integrated BMC either. The data sheet on ESB2 (South Bridge) by Intel described only interface registers to control it using standard access methods, and my efforts resulted in facing typical "black box" situation.

This collection of facts raised the troubling paranoid thoughts in the style of spy novels. These facts clearly stated the following: in new series of server boards based on Intel 5000 chip set, **there are "add-on" programs in flash memory working with BMC and executed on the CPU, and these programs work with CPU hardware virtualization level.**

Images of flash memory software from Intel's website do not contain such software modules, thus software modules preventing my hyper-driver from working properly were illegally embedded in the motherboard flash memory during the production stage.

Therefore, BMC flash memory contains encrypted BMC program modules that are impossible to load in the flash memory without the knowledge of the encryption keys (comment: actually, it is possible first to encrypt and then to upload for following decryption before execution), and the one who put these illegal software modules, known encryption keys, that is, in fact, had access to confidential information.

I informed the management of "Kraftway" about the problem with the BMC flash memory firmware and questionable, in legal terms, situation with the new Intel chipsets (comment: concerning embedded encryption), and received rather expected response in the style of "no buzzing, that may hurt business." I had to calm down not to disturb my relationship with the employer.

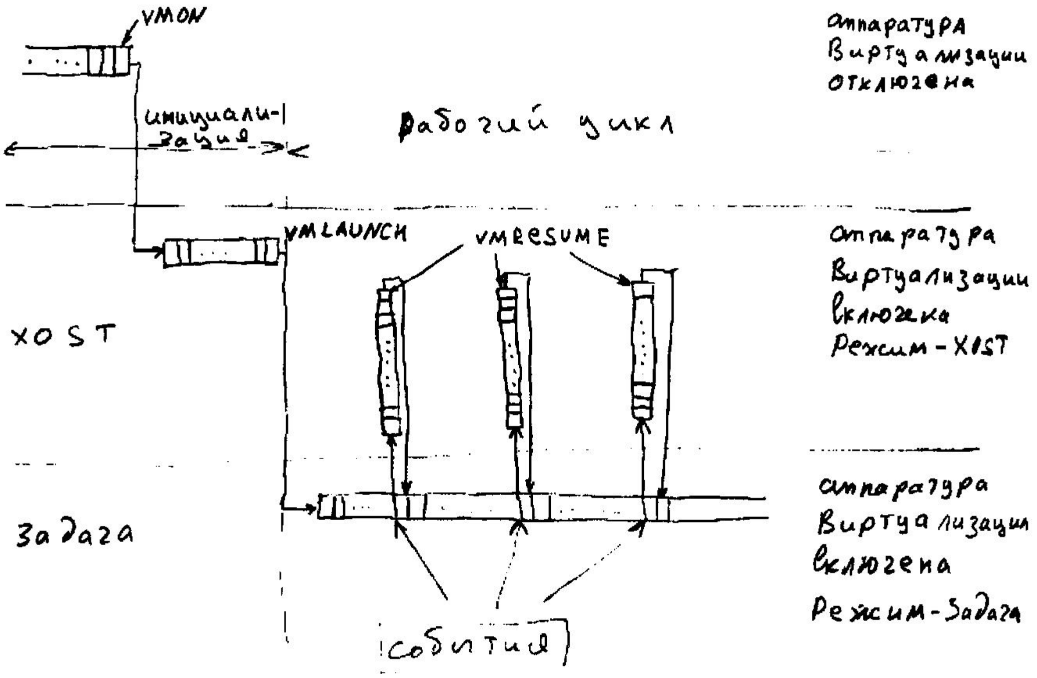
Thoughts concerning the "add-on" software did not give me a rest. It was not clear why all these done and how. If one has the opportunity to place a software in BMC memory, then why such troubles with the CPU were needed (comment: with ground level virtualization controlling hardware)? Reasonable cause and answer could be just that it was required to control the context of the CPU computing process. Obviously, it is impossible to keep track of the processed information to the main computer system using only a peripheral low-speed processor with the frequency of 60 MHz (comment:

meaning the BMC) . Thus, it seems, the goal of this illegal system was collecting the information processed in the main computer by means of hardware virtualization. It is obvious that implementing a remote control of such illegal system is much easier from BMC, because it has its own independent access to the network adapters on the motherboard and its own MAC and IP-address. "How is it done?" was easier to answer - somebody had managed to create a hypervisor, which is able to share resources with hardware virtualization, and does that all correctly except the CPU real mode. Now such systems are of no surprise, but then, five years ago, they were seen as a miracle, and, in addition, the emulation speed - a software emulating host without significant loss in performance seemed as impossible.

To explain I need to go a little further into the theory. Virtualization architecture of both Intel and AMD does not constitute the platform of several hypervisors, but, after starting, the first hypervisor can emulate work on real virtualization hardware for hypervisors that run after. In this case, all hypervisors running after the first run in an emulated host environment. This principle I call the "right of the first night." It can be easily implemented with a special handler to the root host, with secondary host hypervisors will be run as secondary processes for the root host. Emulation is not difficult to implement, but the performance may be an issue. Hardware virtualization works primarily with VMCB (VMCS) (comment: Virtual Machine Control Block - memory block/page), the host program constantly refer to this block, and for each such access requires 0.4-0.7 microseconds. So, hiding a software host's emulation within Intel virtualization is almost impossible, because it requires too many commands for software virtualization, meaning going to root host instead of executing on real virtualization hardware.

Let me tell you a bit about the differences between virtualization architectures. System hardware virtualization from Intel and AMD are different. The main architectural difference of these systems is how the host operates. The AMD system works with disabled hardware virtualization, i.e. its programs are executed on real CPU. Thus, the virtualization of the secondary host requires only VMRUN command virtualization (we can assume that other commands do not exist). Access to VMCB-block in AMD architecture occurs through the standard commands accessing memory. That allows the control by a secondary host only the execution of VMRUN commands and change, if necessary, VMCB-block before entering in the task execution mode. Lengthening of the event execution twice is possible, and on the AMD platform such emulation is still viable. Intel virtualization system is more complicated. Special commands VMREAD and VMLOAD are used to access the VMCB-block, and such command should be virtualized. Usually host handlers access to VMCB- block fields dozens if not hundreds times, and each operations must be emulated. In this case the execution speed drops by an order, and it is very inefficient.

It became clear that to emulate the virtualization as above, the unknown colleagues used more efficient mechanism, and I found the documentation concerning that. Virtual Host at Intel is a virtual environment itself, and nothing, in fact, is different from the environment of the task execution and simply controlled by another VMCB (see the diagram below).


VMON
инициализация
рабогий цикл
аппаратура виртуализации отклюгена
VMLAUNCH
VMRESUME
ХОST
аппаратура виртуализации вклюгена режим-ХОST
задага
аппаратура виртуализации вклюгена режим-задага
события

Pic.1: The author tries to explain how multi=level Intel virtualization works

In addition, the documentation describes the concept of "dual monitor" for System Management Mode (SMM) virtualization, when, in fact, the two hosts are active at once, and therefore two VMCB blocks as well. Host virtualizing system management mode controls the main host as a task/process, but only to call system management interrupt.

Circumstantial evidences suggest that Intel hardware virtualization probably has a control mechanism of multiple secondary hosts, managed by the "root" host, though it is not described anywhere. Also, my system worked exactly as described, and I do not have other explanation for almost imperceptible actions of root hypervisor. And it became even more interesting as it looked like someone had access to these undocumented features and used them in practice.

About six months before the end of the project with Kraftway I took the position of a passive observer, however, continuing regularly run their system on new series of motherboard shipments from China, and new samples (comment: boards not from China). All samples continued to work steadily. When I switched to the Chinese boards, I saw more and more miracles. It seemed that colleagues from abroad are actively improving the operations of their root hypervisor. Last shipments of suspicious boards behaved almost normally, i.e. first boot of my hyper-driver leaded to the system restart when the OS itself starts, but all subsequent booting of the hyper-driver and OS did not have any problems. Finally, it happened what I was waiting for a long time - motherboards from new shipment did not hang my hyper-driver at all. I was in doubt if I was paranoid in my suspicions, but new case has strengthened my opinion.

It should be noted that Intel continuously improves hardware virtualization. I started to work when it was revision 7, and the described situation occurred on the 11th revision, that is, approximately two years later considering two revisions in a year, because even numbers are not used.

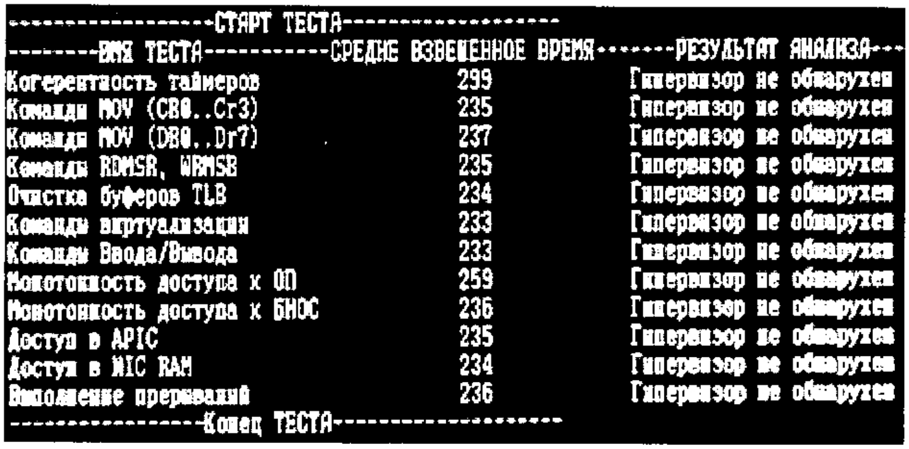
So, in revision 11 conditions of entry in virtual host for hardware virtualization have been improved significantly, whereby a new control field n VMCB-block has been introduced. When I got new samples (chip sets) of revision 11, I wanted to test new features. I added new features of the hardware virtualization in my hyper-driver, and installed new chip set in standard boards from China, in which everything had worked without any problems, and started debugging. New features of the equipment did not work, and again I felt into prostration, sinning on samples and documentation. After a while the motherboards were needed for other tasks, and I, resuming the experiments, moved new revision 11 chip set to Canadian sample motherboards. And new features started working!

At first I thought that I did something wrong with serial motherboard from China, because new hardware virtualization features are in CPU, and have nothing in common with a motherboard.

To test that again, I reinstalled sample processor on serial (China) board, and all again stopped working. So, I was not mistaken, but the problem lays in the fact that the motherboard in some way influenced the new hardware virtualization features of the processor. Given my suspicions, there is only one conclusion possible that colleagues from abroad put in the motherboard flash memory an illegal "root host", and did not know about the new revision of hardware virtualization. When that unaware of new hardware virtualization root host started working, it stopped passing correctly the exit from a task in my secondary host via its own event handler. Already knowing how to deal with this problem, I downloaded in the standard motherboard firmware for the BMC from Intel site. I was sure that everything will work, but to my great surprise, it did not. Hanging was still there. This was something new.

According to my theory, illegal hypervisor is confident of its invulnerability. Apparently, its authors thought that the debugging phase of their child was done, and there is no need any more to mask their software as BIOS faults. After enabling a protection of the code from being overwritten in the flash memory, uploading of correct (original Intel) software became not possible.

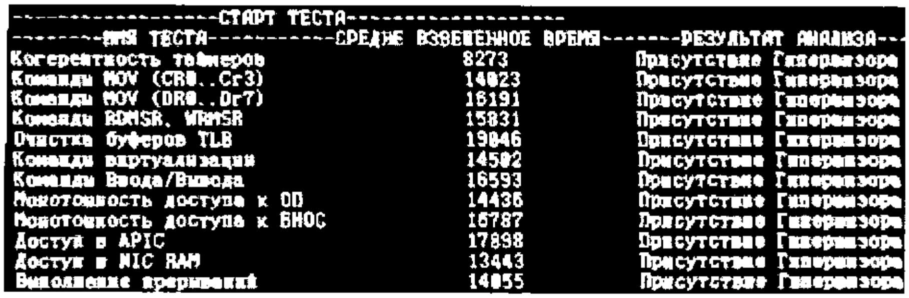
I was still not sure if I'm right, and needed additional experiments. I had to invent my own methods for detection of hardware hypervisor (comment: "root host" in his terms). Later, however, it turned out that I had reinvented the wheel. My method allows the identification of system commands’ execution time, and such execution requires mandatory emulation in the root host hypervisor. As a timer I used the cyclical frames counter in USB-controller hardware, and I wrote a program working in real mode to minimize side effect and uncontrolled interrupts that masked the true time of executing of system commands. I first tested "clean" system using sample motherboards from Canada.



```
------------------СТАРТ ТЕСТА-------------------
--------ИМЯ ТЕСТА-----------СРЕДНЕ ВЗВЕШЕННОЕ ВРЕМЯ-------РЕЗУЛЬТАТ АНАЛИЗА---
Когерентность таймеров              299            Гипервизор не обнаружен
Команды MOV (CR0..Cr3)              235            Гипервизор не обнаружен
Команды MOV (DR0..Dr7)              237            Гипервизор не обнаружен
Команды RDMSR, WRMSR                235            Гипервизор не обнаружен
Очистка буферов TLB                 234            Гипервизор не обнаружен
Команды виртуализации               233            Гипервизор не обнаружен
Команды Ввода/Вывода                233            Гипервизор не обнаружен
Монотонность доступа к ОП           259            Гипервизор не обнаружен
Монотонность доступа к БНОС         236            Гипервизор не обнаружен
Доступ в APIC                       235            Гипервизор не обнаружен
Доступ в NIC RAM                    234            Гипервизор не обнаружен
Выполнение прерываний               236            Гипервизор не обнаружен
-----------------Конец ТЕСТА--------------------
```

Pic.2. Testing motherboards from Canada

Runtime, see the photo above - approximately corresponds to the number of CPU cycles (comment: the third column says "Hypervisor is not detected"). Then I ran the same test on the serial motherboard (from China) and found out that my paranoiac assumptions were correct - instructions' execution time has significantly increased (see below) (comment: the third column says "Hypervisor is detected").



```
------------------СТАРТ ТЕСТА-------------------
--------ИМЯ ТЕСТА-----------СРЕДНЕ ВЗВЕШЕННОЕ ВРЕМЯ-------РЕЗУЛЬТАТ АНАЛИЗА---
Когерентность таймеров              8273           Присутствие Гипервизора
Команды MOV (CR0..Cr3)              14023          Присутствие Гипервизора
Команды MOV (DR0..Dr7)              16191          Присутствие Гипервизора
Команды RDMSR, WRMSR                15831          Присутствие Гипервизора
Очистка буферов TLB                 19046          Присутствие Гипервизора
Команды виртуализации               14502          Присутствие Гипервизора
Команды Ввода/Вывода                16593          Присутствие Гипервизора
Монотонность доступа к ОП           14436          Присутствие Гипервизора
Монотонность доступа к БНОС         16787          Присутствие Гипервизора
Доступ в APIC                       17898          Присутствие Гипервизора
Доступ в NIC RAM                    13443          Присутствие Гипервизора
Выполнение прерываний               14055          Присутствие Гипервизора
```

Pic.3. Testing the motherboards “Assembled in China”

It means that in the BMC flash memory of the server boards from China, produced under the label Intel, there is undeclared software module that acts as a host hypervisor, and which is embedded

during manufacturing/production of the boards. It remained yet to convince the others. I went to Intel representative in Russia first. It was not difficult, as the staff of the Russian office frequently appeared in Kraftway.

I've explained and demonstrated everything, but was not sure that the technical specialist understood. These, so-called technical experts, on the level of competence differ just a little from those of managers. However, he promised to report everything to the management. I do not know whether he did it, but there was no response from Intel, and all gone as water in the sand. Work in Kraftway by that time was over, and I started a new project in a company related to information security. The head of this company, with whom I shared my "discoveries", took my words seriously. In this regard, it was decided to talk to the leadership of FSB (comment: in past known as KGB) Center for Protection of Information and Special Communications (comment: meaning secure government communication network). This organizational structure is inside of the FSB, and is engaged in providing information security in the country and regulates the activity of government and commercial organizations that are related to information protection. It also regulates the measures for the protection of information for government agencies and businesses that handle sensitive and confidential information. Firm in which I was then working, maintained official contacts with the Center to certify and license their commercial projects, so to arrange a meeting at the professional level was quite easy. It was assumed that the experts of the Center will report their opinion to the leadership, and if they decide that it is necessary to listen to us, the next step would be a meeting at a higher level.

The meeting took place. I explained and showed everything that we discovered, and then demonstrated the presence of illegal software module comparing motherboards from Canada and China. When the conversation turned to discussing BMC, misunderstanding appeared in the eyes of the colleagues from the Center. Thus, I needed to explain well known things. In the process, it became clear that they were not even aware of the existence of a special chip (comment: not actually a chip but hardware function) in the South Bridge with access to the network adapter, and the presence of the cryptographic module in the BMC that actually violate s Russian law. Finally, we, all of sudden, heard that this threat model has been studied, and mitigating measures are in place, and that generally they are not afraid of such "add-on", because their systems do not have Internet access.

Our further inquiries have resulted in nothing; everything resulted in secrecy statements like: we are smart and know all about that, and you are not allowed to know what we know. However, I strongly doubt their technical competence, as they simply did not understand much of what I said and showed. We parted on the conclusion that they will report to their superiors who will decide on further actions.

Later I learned about this "secret method" of BIOS add-ons detecting of "root". I learned that by quite an accident during discussions with the firm specialists (the Center is the licensee authorized to check the BIOS for "add-on" illegal software). The technicians of the company conducting BIIOS code analysis said that the search for BIOS software modules that use hardware virtualization should be done by looking for the signatures of virtualization commands. Indeed, CPU commands for hardware virtualization contain three or four bytes in the code, but who said that they will find the code in unencrypted part of the flesh memory? Or whether they will be able to get the code by scanning RAM, if the memory is hardware secured from viewing at?

In general, the outcome of the first meeting was really disappointing. I was in bad mood waiting for what happens next. After a month and a half, we were invited to the FSB Center itself to demonstrate the "add-on" that we had found. This time people listening to us were not just ordinary employees, but managers and leading specialists (at least, so they've been introduced). The meeting turned into a lecture. They listened for almost three hours. However, it was clear that it is the first time they hear that. I have explained this vulnerability of x86 platforms, demonstrated the "add-on", told how to detect it, and finally answered many questions. At the end of the meeting they thanked us, and said that the issue should be investigated in the framework of special research, and then we parted.

My euphoria vanished when unofficial information reached us that they did not want to believe in that issue. However, this did not dampen my desire to prove my case. As it had seemed to me, the decision was on the surface - I should write such a "add-on" myself. I would not be able to put the "add-on" in the flash memory of the BMC, but can load the code in the main BIOS. I decided to add security features to my "add-on hypervisor" module for masking it in the memory and on the flash memory chip, and also blocked writing to the flash chip where the "add-on" will be placed. It would make it impossible to remove my code by reprogramming the motherboard, and the only one option would be removing the flash memory from the board and reprogramming it in special hardware.

It remained only to decide on the "evil" features that the hypervisor should perform. I remembered the statement of one of the experts of the FSB Center that they are not afraid of such "add-on" because their systems are disconnected from the global network. But information from the outside must somehow get into these protected local networks, at least through writable optical discs (CD-ROM). Thus, I came to the obvious conclusion and decided to analyze incoming information in the "add-on" software utilizing hyper-driver means. My Last Judgment Day weapon will utilize the "add-on" to kill the computer system by an external command, and which will be masked in incoming flow by steganography.

Which would be the entry point for my destruction command? It was obvious - the FIFO (buffer) of disk and network adapter input/output (I/O) system. The scanning of I / O buffers is trivial task for hardware virtualization. This module of approximately 20KB in size has been loaded in the motherboard BIOS together with anti-detection protection. My software actually perform single function task of wiping out BIOS flash chip when receiving a command to destroy the system. For easy implementation, the command itself had been written hidden into a DOS text format file debugging tags.

When all was ready, the management of my company proposed to FSB again to see the work of our own "add-on" demonstrating that virtualization technology is a real threat.

However, no one wanted to look at our demo. I do not know who ordered, but it was a command to stop communicating with us. The main fighters for the security did not want to listen to us. Then, having almost no hope, we tried to convey our findings to users. We got in touch with "Gazprom" (comment: state enterprise developing oil and gas reserves in Russia) to inform their specialists on modern information security threats applicable to distributed process control systems. We managed to arrange a meeting with the department managing security systems of the corporation, and prepared demo version of our "add-on" with simplified command interface. In this version the "add-on" is activated after downloading our text file to the testing computer, the contents of which included as well two words - "Gazprom" and "stop" (comment: not clear description of the demo; it sounds like the text file had a command to start the "add-on" which displayed "Gazprom stop" and wiping out the computer BIOS after a delay - remarks). The computer died after a delay of five minutes. Of course, we could make any delay. Then the staff of "Gazprom" complained on their low level of information security, and said that it is actually not their business to handle such issues, as they are guided by the requirements and standards that set the FSB. So, it was a dead-end, and we cannot break this "monolithic" system of irresponsible information security.

For three and a half years that have passed since then, I have never heard anyone talking about the hardware virtualization as an instrument of penetration into a computer system. Is there a paradox? I do not think so. The specificity of the topic is that we know only about the failed technology (comment: meaning identified and investigated security incidents). Successfully penetrating technologies were not discovered and their authors are unknown.

We should understand that secret placement of "add-on" in to a BIOS is possible only during manufacturing process. And, because of the following utilization of such exploit, it should be done to certain models of motherboards. Such limitations mean that hackers are not interested in developing such methods, and rather are interested in general use exploits. I am sure that there are entities which are interested in precise control of computer systems, and virtualization is the best method permitting efficient cover-up. It happened that we almost got them by a chance, but nowadays that is likely impossible. And, by our experience, nobody is going to hunt authors down.