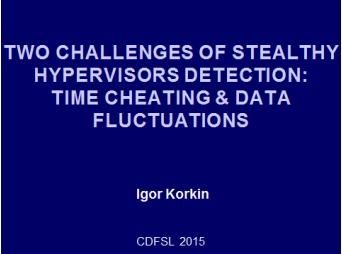
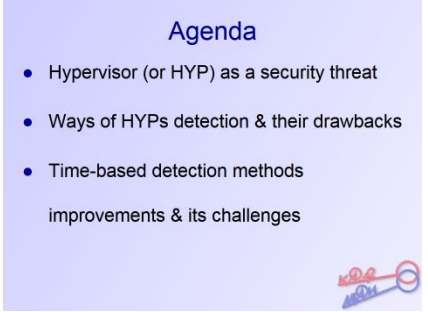


# CONFERENCE SPEECH “TWO CHALLENGES OF STEALTHY HYPERVISORS DETECTION: TIME CHEATING AND DATA FLUCTUATIONS”

Hello! I’m Igor from Moscow. I work as a programmer and teacher at university.  
I’m interested in hidden software research, new security and spyware technologies.  
And today I’ll tell you about one of these technologies.

<p><b>Slide 1 Hello</b></p> <p>Hello!</p> <p>Today I’ll tell you about a new security system which detects hidden software.</p>	
<p><b>Slide 2 “Agenda”</b></p> <p>This talk consists of three parts. First, I’ll describe a hypervisor as a “real cyber threat”.</p> <p>Next I’ll focus on the existing hypervisor detection methods and their drawbacks.</p> <p>Finally I’ll present a new hypervisor detection system, which improved time based detection method.</p>	

### Slide 3 “Any PC can be compared with a big ship”

The idea is that any computer can be compared with a big ship.

Because decks of a ship look like computers modes.

#### Any PC can be compared with a big ship



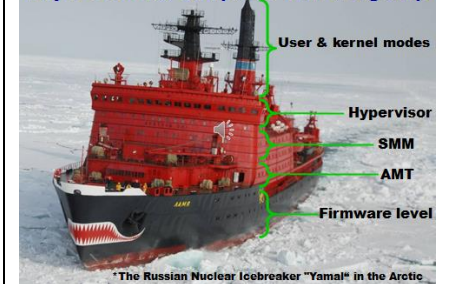
"The Russian Nuclear Icebreaker "Yamal" in the Arctic

### Slide 4 “Any PC can be compared with a big ship”

To protect a ship we need to make sure that there is no spy or intruder all over the ship from the hold to the top of the mast.

In terms of computer the situation is absolutely the same, we need to make sure that there is no spyware in each computer mode.

#### Any PC can be compared with a big ship



"The Russian Nuclear Icebreaker "Yamal" in the Arctic

### Slide 5 “The existing places to plant the backdoor”

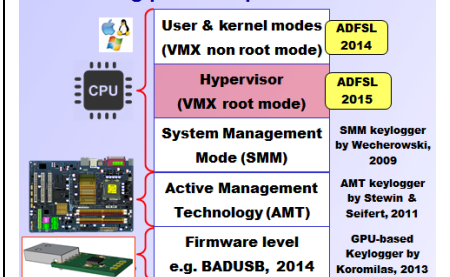
Here you can see the details of different computer modes. Each of these modes can have backdoor.

At the last conference in Virginia one year ago, I discussed kernel mode rootkits; it was the upper deck.

Now we are going deeper.

Today I'll tell you about the detection of new suspicious software... about hypervisors.

#### The existing places to plant the backdoor



### Slide 6 “What & where is a hypervisor?”

All right, we all have laptops and {u-Z-e} use them for various purposes.

Of course, any laptop could be infected.

#### What & where is a hypervisor?



image source: <http://pngimg.com/download/5932>

### Slide 7 “What & where is a hypervisor?”

To remove malware we use an antivirus and we expect that this antivirus works properly.

#### What & where is a hypervisor?



image source: <http://pngimg.com/download/5932>

### Slide 8 “What & where is a hypervisor?”

This is the basic scheme of how computers work.

All these software apps are run by the operating system, which is above {E-B-A-V} the hardware.

#### What & where is a hypervisor?



image source: <http://pngimg.com/download/5932>

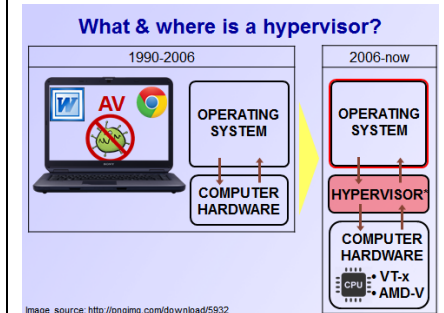
## Slide 9 “What & where is a hypervisor?”

But the situation is changing: the vast majority of computers which have been sold in the last 10 years definitely have a CPU — a processor with “hardware virtualization technology {T-E-K-nologi}”.

For Intel processors this technology is called VT-X, and for AMD processors it’s called AMD-V.

Today we will focus on Intel processors, but all my ideas work well on AMD CPU as well.

Hardware virtualization technology provides a mechanism to put an intermediary between the operating system and the hardware.

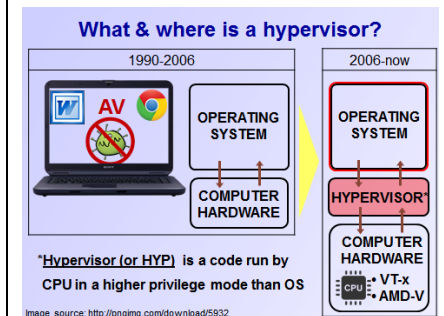


## Slide 10 “What & where is a hypervisor?”

This software is called a hypervisor and it runs directly on the hardware.

A hypervisor is a program that operates at a more privileged level than the operating system; it is just below the operating system. In a nutshell, the hardware virtualization is the CPU feature.

Does any computer system support hardware virtualization?

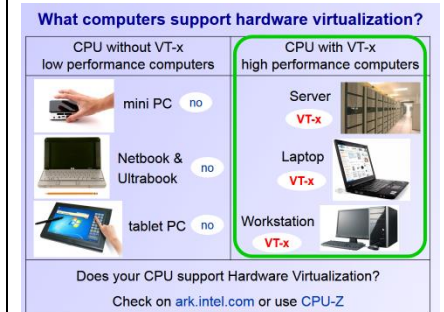


## Slide 11 “Hardware virtualization in our computers“

On the left you can see the computers without hardware virtualization, and on the right you can see the computers that support hardware virtualization technology.

If you want to check whether your computer supports this technology or not go to this website or use tools like {Cee-Pee-Uuu-Z} CPU-Z to receive system information.

Now let’s go through some features of a hypervisor. We will focus on 5 of them.



## Slide 12 “Five features of HYP & the area of its application “

The first one is that a hypervisor can control all interaction between the operating system and the hardware.

A hypervisor has an unrestricted access to any hardware compónents, like hard disks and network cards.

Second, it’s impossible to stop or block a hypervisor by using the operating system, because a hypervisor runs at a more privileged level.

Third, there is no **built-in** tool to detect a hypervisor. Current approaches cannot solve the detection problem in an appropriate way.

Next is that a hypervisor can avoid its detection using a variety of countermeasures, today we will focus on the time cheating.

Five features of HYP & the area of its application	
Features	<ol style="list-style-type: none"><li>1. HYP can <u>control access</u> to memory, HDD etc</li><li>2. Impossible to <u>block or delete</u> HYP from OS</li><li>3. There is <u>no built-in tool</u> for HYP detection</li><li>4. HYP can <u>prevent its detection</u> = stealthy HYP e.g. by using time cheating</li><li>5. HYP <u>installs invisibly</u> for both users &amp; AVs</li></ol>
Areas	

What feels óminous is that a hypervisor can be installed invisibly both {BO-UTH} for users and anti-viruses.

### Slide 13 “Five features of HYP & the area of its application “

By using the first two advantages, we can create an advanced {ADVA-ANSD} and powerful anti-virus, which enhances the level of security, or for example a tool to {í-S-o} isolate internet banking from malware. It sounds practical! Obviously, this is a plus of “hardware virtualization”.

Of course, we can use all these advantages, but what will happen if hackers use all of them?

Surely, on the dark side, hackers can create the most powerful “backdoor”.

Any doubts {DA-U-T-S} that this can happen?




Five features of HYP & the area of its application	
Features	<ol style="list-style-type: none"> <li>1. HYP can <u>control access</u> to memory, HDD etc</li> <li>2. Impossible to <u>block or delete</u> HYP from OS</li> <li>3. There is <u>no built-in tool</u> for HYP detection</li> <li>4. HYP can <u>prevent its detection</u> = stealthy HYP e.g. by using time cheating</li> <li>5. HYP <u>installs invisibly</u> for both users &amp; AVs</li> </ol>
Areas	<p>1 + 2 = for security</p> <p>1 + 2 + 3 + 4 + 5 = for backdoor</p>

### Slide 14 “Backdoor hypervisor can“

Before answering this question let’s look at what kind of backdoor a hypervisor can provide.

This backdoor can record keystrokes, steal all your data or even block your computers.

Also a hypervisor can be installed by a driver or it can sit in the BIOS.

Overview of a backdoor HYP facilities	
BackdoorHYP can	Ways to plant a HYP
<ul style="list-style-type: none"> <li>• record keystrokes</li> </ul> 	<ul style="list-style-type: none"> <li>• using OS vulnerabilities to load a driver-based HYP</li> </ul>
<ul style="list-style-type: none"> <li>• steal all data</li> </ul> 	<ul style="list-style-type: none"> <li>• using BIOS-based approach to infect motherboard chip</li> </ul>
<ul style="list-style-type: none"> <li>• block PC</li> </ul> 	

## Slide 15 “Stealthy hypervisor & well-known examples”

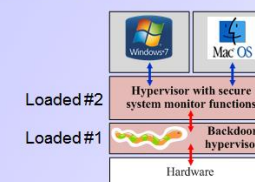
Moreover, the problem with a stealthy hypervisor detection becomes even worse when “several nested hypervisors” sit on your computer.

This is an example of two nested hypervisors which are running at the same time.

In this case, a legitimate hypervisor allows {ELAUS} us to run two operating systems at the same time, like Windows and Mac operating systems simultaneously. It sounds great!

At the same time, the malicious hypervisor is loaded first and therefore it is located between the trusted hypervisor and the hardware. That is why it can compromise the trusted one and steal all our data.

Backdoor HYP & well-known examples



## Slide 16 “Stealthy hypervisor & well-known examples”

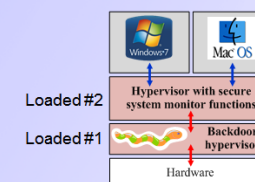
This situation is entirely factual.

Look at the first two lines. Here you can see two examples of hypervisors, which are loaded by drivers.

And also at the last DeepSec conference, the investigation about undocumented suspicious hypervisor was presented. Look at the last line. This ghost {GO-UST} hypervisor uses BIOS to hide its start-up.

All these examples prove that an illegal hypervisor is a “real cyber threat” to our computers, ... to all our data, ... to our business.

Backdoor HYP & well-known examples



HYP example	Author	HYP is loaded by	CPU
Blue Pill	Invisible Things Lab	Windows driver	AMD
Vitriol	Matasano Security	MAC OS driver	Intel
Russian Ghost	M.Utin by DeepSec14	BIOS	Intel



We have to meet this challenge; we have to detect a backdoor hypervisor.

Of course, researchers are trying to solve the hypervisor detection problem.

### Slide 17 “Hypervisor detection tools”

Look at this comparison table with well-known hypervisor detection tools.

The first column includes the tool titles. The next one holds the implemented detection methods.

And the last two columns show if the tools are resilient and portable.

#### Analysis of hypervisor detection tools

	Tool	Detection method	Resilient?	Easy to distribute?
Hardware	Copilot 2004	Signature based	+	—
	Deep Watch 2008			
Software	Symantec EndPoint Protection 2012	Based on the trusted HYP	—	+
	McAfee Deep Defender 2012			
	Actaeon 2013	Signature based		
	Proof of Concepts 2008 - 2015	Behavior based & Time based		
	New proposal tool	Time based	+	+

### Slide 18 “Hypervisor detection tools”

Look at the first two lines. You can see that only hardware tools are resilient to hypervisor countermeasures, but they are not portable so they are good only for laboratories.

That is why further we will not consider the hardware tools.

#### Analysis of hypervisor detection tools

	Tool	Detection method	Resilient?	Easy to distribute?
Hardware	Copilot 2004	Signature based	+	—
	Deep Watch 2008			
Software	Symantec EndPoint Protection 2012	Based on the trusted HYP	—	+
	McAfee Deep Defender 2012			
	Actaeon 2013	Signature based		
	Proof of Concepts 2008 - 2015	Behavior based & Time based		
	New proposal tool	Time based	+	+

### Slide 19 “Hypervisor detection tools”

In terms of portability {porta-BÍ-LITY}, the software tools are better, but they are unable to detect stealthy hypervisors, ... hypervisors which are applying countermeasures.

So far, you can see that there is no tool, which is both {BO-UTH} portable and resilient.

#### Analysis of hypervisor detection tools

	Tool	Detection method	Resilient?	Easy to distribute?
Hardware	Copilot 2004	Signature based	+	—
	Deep Watch 2008			
Software	Symantec EndPoint Protection 2012	Based on the trusted HYP	—	+
	McAfee Deep Defender 2012			
	Actaeon 2013	Signature based		
	Proof of Concepts 2008 - 2015	Behavior based & Time based		
	New proposal tool	Time based	+	+



## Slide 20 “Hypervisor detection tools”

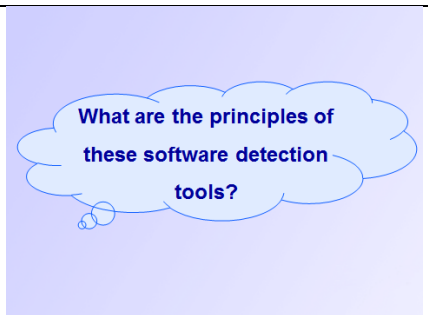
But today I’ll present you a detection tool which is both {BO-UTH}  
portable and resilient.

Analysis of hypervisor detection tools				
	Tool	Detection method	Resilient?	Easy to distribute?
Hardware	Copilot 2004	Signature based	+	—
	Deep Watch 2008			
Software	Symantec EndPoint Protection 2012	Based on the trusted HYP	—	+
	McAfee Deep Defender 2012			
	Actaeon 2013	Signature based		
	Proof of Concepts 2008 - 2015	Behavior based & Time based		
New proposal tool		Time based	+	+

## Slide 21 “What are the principles of these detection methods”

Now let’s focus on the principles of these software detection tools.

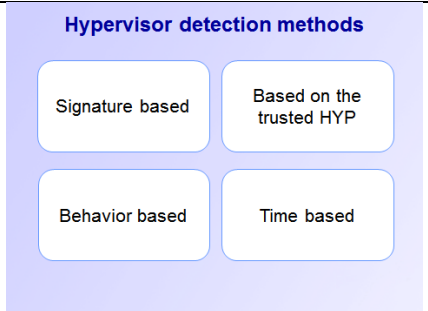
How do they work and why are they vulnerable?



## Slide 22 “Hypervisor Detection Methods”

There are four hypervisor detection methods.

Let’s start with signature-based detection.



## Slide 23 “Hypervisor Detection Methods”

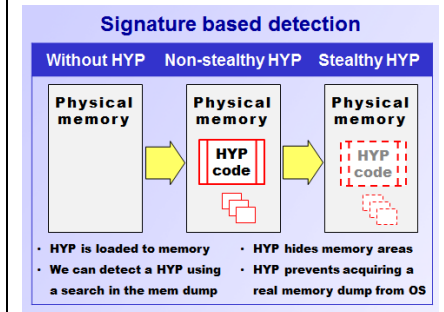
How does it work? After the installation of a hypervisor, its dispatcher or binary code and corresponding data structures are loaded in the memory. Look at the red rectangles.

We can reliably detect a hypervisor by finding its memory fragments.

But a hypervisor can hide its memory pages and prevent its detection.

For example, a hypervisor can use Extended Page Tables for Intel CPU.

You can see that this detection method is vulnerable to hypervisor countermeasures.

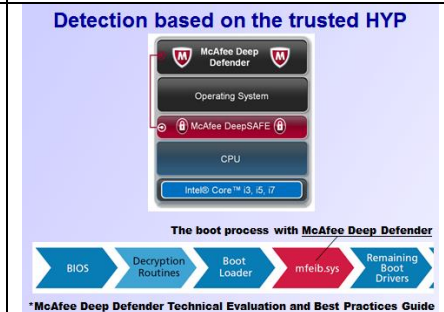


## Slide 24 “Hypervisor Detection Methods”

Another method is a detection based on a trusted hypervisor.

In this case, a trusted hypervisor is loaded first and it's close to hardware.

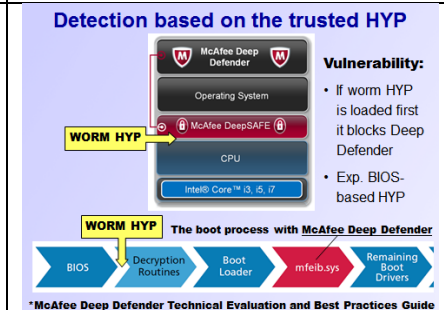
That is why it can prevent any installation of a suspicious one. It's a trusted hypervisor.



## Slide 25 “Hypervisor Detection Methods”

Everything looks okay, but what will happen if a suspicious, ... a worm hypervisor installs before the trusted one? What will happen if the worm gains the control first?

Of course, in this situation, the worm hypervisor can prevent the installation of the trusted one; it can



compromise the trusted one. As a result our computers will be unsafe, will be in danger.

Unfortunately, this method is vulnerable to typical man in the middle attack.

## Slide 26 “Hypervisor Detection Methods”

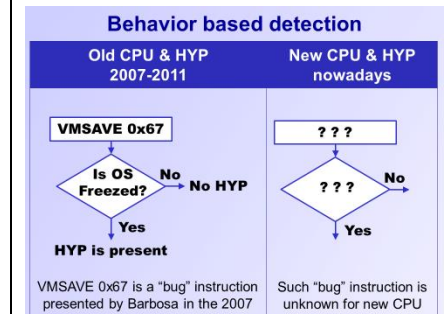
The next method is named behavior-based detection.

This method relies on the fact that the result of specific instructions execution depends on whether a hypervisor is present or not. For example if a hypervisor is running by some old AMD processor, the execution of instruction VMSAVE with this parameter causes the system to freeze. Or it stops the system.

But in the same computer without a hypervisor, such execution does not freeze the system.

You see the difference in the behavior.

Unfortunately, this method works only for old CPUs and there are no such “freezing” instructions for new hardware. This is a serious drawback for the detection.

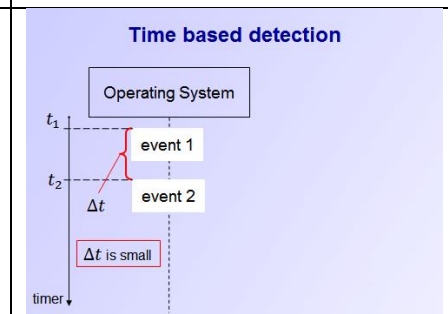


## Slide 27 “Time based detection”

The last method for today is time-based detection.

Now I’m going to demonstrate an interaction between an operating system and a hypervisor.

Without a hypervisor all events occur in the operating system without long delays.



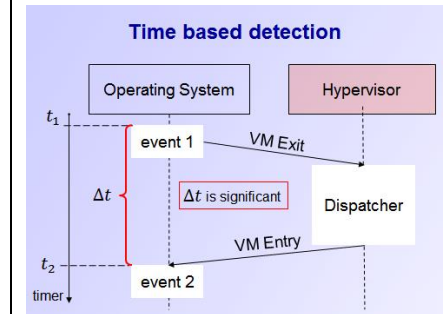
## Slide 28 “Time based detection”

But if a hypervisor is present, it intercepts this event.

As a result the hypervisor code starts to run and it needs some time, ... a delay to process this event.

Therefore, the duration of this event becomes longer, in comparison with no hypervisor case.

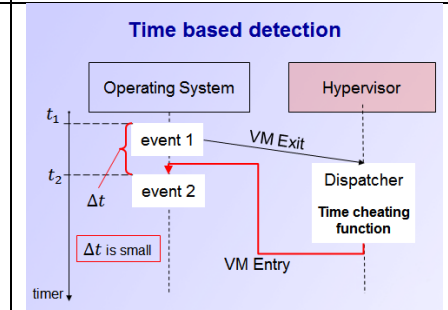
We can precisely detect a hypervisor by measuring this time delay.



## Slide 29 “Time based detection”

Everything looks ok, but a hypervisor can compensate this time overhead, ... this delta time by using time cheating function. As a result, this delta time is decreased and becomes the same as in no hypervisor case.

We can see that this method is vulnerable to time-cheating.



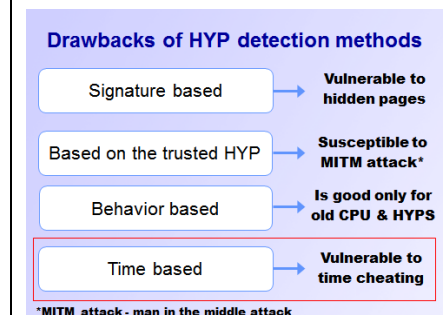
## Slide 30 “Conclusion about detection method”

So far we can see that all software detection methods are vulnerable to hypervisor countermeasures.

For my further analysis, I chose the time-based detection. Why?

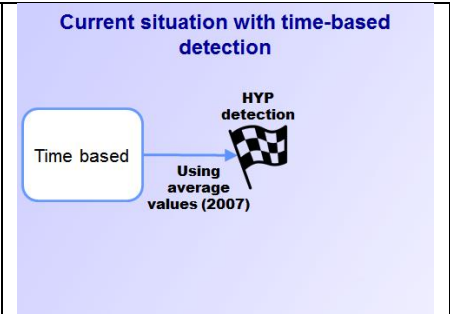
I focused on the fact that a hypervisor always spends time to process an event and this time needs to be hidden. It's impossible to process an event without spending time. That was my idea.

Let's focus on the road map of current situation with time-based detection and mark my contribution on it.



### Slide 31 “Current situation with time-based detection”

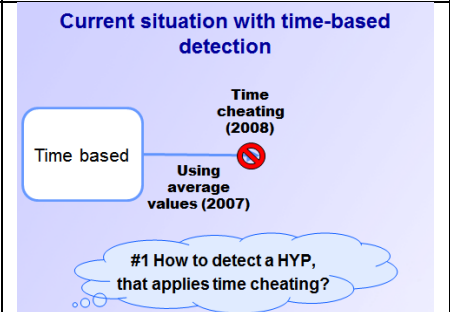
We know that by using the time values or average time values we can precisely detect only a non-hidden hypervisor.



### Slide 32 “Current situation with time-based detection”

But if a hypervisor is using time-cheating it's impossible to detect it using average values, because they are the same. So, a stealthy hypervisor avoids time-based detection.

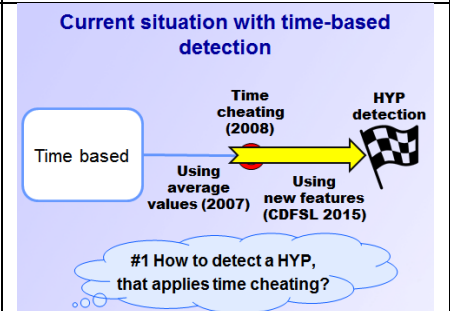
For this situation there are no appropriate time-based detection methods.



### Slide 33 “Current situation with time-based detection”

And today, I'll describe how to improve it. Look at the yellow arrow.

This new method works well even if a hypervisor applies time-cheating.

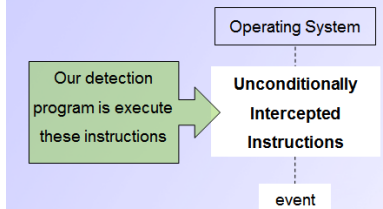


### Slide 34 “Let's focus on the time-based detection”

To detect a hypervisor as “an event” I chose unconditionally intercepted instructions.

Let's focus on them.

#### Let's focus on the time-based detection by unconditionally intercepted instructions

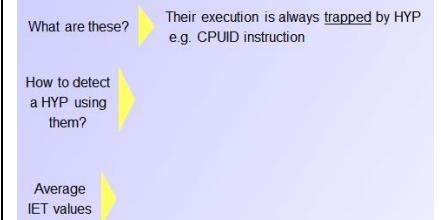


### Slide 35 “Time-based Detection by Unconditionally Intercepted Instructions”

First of all, what instructions do we call unconditionally intercepted?

Execution of each of these instructions is always intercepted by any hypervisor. Or this event causes a hypervisor dispatcher to start. An example of such instructions is “Cee Pee Uu I Dee”.

#### Time based detection by Unconditionally Intercepted Instructions



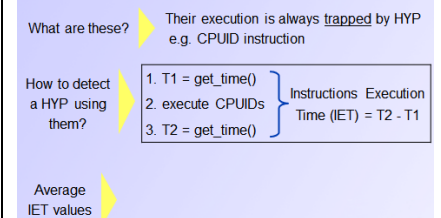
### Slide 36 “Time-based Detection by Unconditionally Intercepted Instructions”

How to use them for detection? There are three steps:

Read a time counter, execute a block of these instructions and read the time counter repeatedly.

I calculate Instructions Execution Time — “I-Ee-Tee” or just the duration, ... the delta time, ... by subtracting the first value from the second one.

#### Time based detection by Unconditionally Intercepted Instructions



### Slide 37 “Time-based Detection by Unconditionally Intercepted Instructions”

To make it clear I’ll show how this scheme works. For example, I measured ten (10) “Cee-Pee-Uu-I-Dee” instructions with the help of Time Stamp Counter in two situations: without a hypervisor and with it.

You can see a ten times difference between “I-Ee-Tee” averages without a hypervisor and with a non-stealthy hypervisor.

In other words, there is no problem to detect a non-stealthy hypervisor.

**Time based detection by Unconditionally Intercepted Instructions**

What are these? → Their execution is always trapped by HYP  
e.g. CPUID instruction

How to detect a HYP using them? →

1. T1 = get\_time()  
2. execute CPUIDs  
3. T2 = get\_time()

Instructions Execution  
Time (IET) = T2 - T1

Average IET values →

	Non Stealthy
Without HYP	~2,000
With HYP	~20,000

\* Lifebook E752 Core i5, Windows Live CD XP DDD

### Slide 38 “Time-based Detection by Unconditionally Intercepted Instructions”

However, if a hypervisor uses time cheating these average values are the same, ... are absolutely the same.

Their values are often not the same but really close to each other and that is why we cannot be sure about whether a hypervisor is present or not.

This is the most complicated situation, that hasn’t been taken into account before. What should we do?

**Time based detection by Unconditionally Intercepted Instructions**

What are these? → Their execution is always trapped by HYP  
e.g. CPUID instruction

How to detect a HYP using them? →

1. T1 = get\_time()  
2. execute CPUIDs  
3. T2 = get\_time()

Instructions Execution  
Time (IET) = T2 - T1

Average IET values →

	Non Stealthy	Stealthy HYP
Without HYP	~2,000	~2,000
With HYP	~20,000	~2,000

\* Lifebook E752 Core i5, Windows Live CD XP DDD

### Slide 39 “How do we want to detect a HYP?”

Let’s think.

What steps can be used to detect a hypervisor?

**How do we want to detect a HYP?**

What will be the steps to detect HYP using a time-based method?



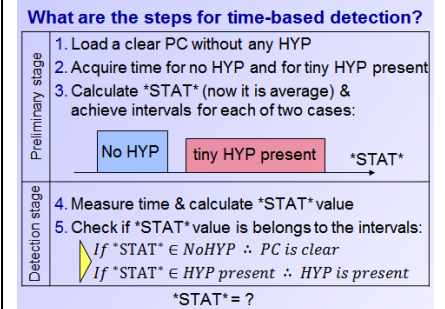
## Slide 40 “What are the steps for time-based detection?”

First of all we need a computer without any hypervisor.

We will measure the time in the following cases: without a hypervisor and with it.

Next we calculate some statistic value and by comparing their values we can find threshold values.

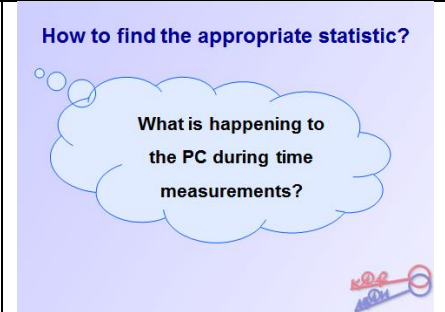
It’s similar to the previous situation. But you know that because of time-cheating attack the average values are not applicable. Now we just have to invent the appropriate statistic to replace an average.



## Slide 41 “How to find the appropriate statistic?”

To do this to invent the statistic value let’s move on to the question.

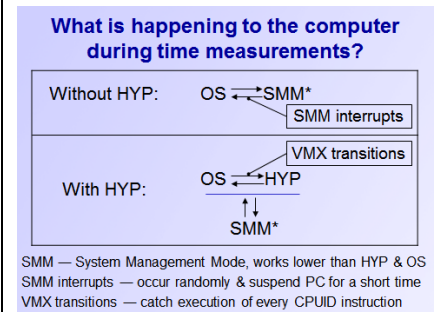
What is happening to the computer during time measurements?



## Slide 42 “What is happening to the computer during time measurements?”

Look at the interaction between the operating system and other modes during the measurements of “I-Ee-Tee”.

Without a hypervisor, the operating system interacts only with System Management Mode — S-M-M, which is the most privileged mode and its interruptions occur randomly. S-M-M suspends the work of the



operating system only for a short time.

If a hypervisor is loaded, such interruptions to S-M-M suspend both {BO-UTH} of them, ... the operating system and the hypervisor.

In a nutshell, the S-M-M is the most privileged mode and its interruptions occur randomly.

### Slide 43 “Different switching between modes”

Let’s specify the process of modes switching during “I Ee Tee” measurements.

Without a hypervisor, instructions are executed in the operating system.

This process is indicated by a small blue loop.

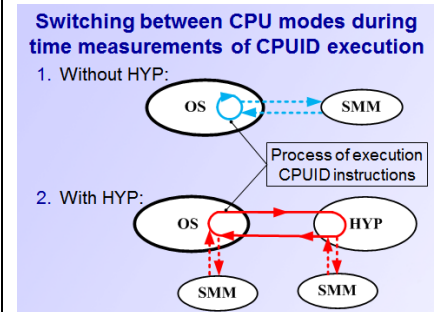
Randomly generated {jenerated} interruptions to the S-M-M are indicated as dashed blue arrows {ARO-UZ}.

A scheme with a hypervisor is shown below. The hypervisor catches {CAT-CHIZ} the execution of every “Cee Pee Uu I Dee” instruction from the operating system. This execution is indicated by a red oval loop.

The S-M-M is duplicated {DUPLICATID} for better clarity. Of course, there is only one mode.

S-M-M interruptions occur either from the operating system or from the hypervisor in a random way.

Now let’s compare these cases and think about “I Ee Tee” peculiarities here.



## Slide 44 “Theoretical analysis of switches between modes”

First due to the random nature of the interruptions to the S-M-M, the “I Ee Tee” is a random variable.

Another point is that “I Ee Tee” is randomly distributed in several fixed layers.

This layered nature is also caused by the interruptions to the System Management Mode.

Let’s focus on the following “I Ee Tee” characteristics: average value and variability indexes {INDISIZ}, such as the number of layers, variance and fourth order moment.

After a hypervisor is loaded all these statistics are increased in comparison with a no-hypervisor case.

### Theoretic analysis of switches between modes

- CPU works as a stochastic system ➡ IET is a random variable
- SMM interrupts both OS & HYP ➡ IET has a layered structure
- IET indexes are increased after HYP is loaded:

Average
Number of layers
Variance & 4 <sup>th</sup> order moment

## Slide 45 “Theoretical analysis of switches between modes”

We remember that a hypervisor can reduce the average value by time cheating.

In this situation, we have to apply only variability indexes as a possible way for detection.

Let’s check these three ideas about random nature with layers and variability indexes.

### Theoretic analysis of switches between modes

- CPU works as a stochastic system ➡ IET is a random variable
- SMM interrupts both OS & HYP ➡ IET has a layered structure
- IET indexes are increased after HYP is loaded:

Average	Time-cheating by HYP
Number of layers	➡ Both are possible for stealth HYP detection
Variance & 4 <sup>th</sup> order moment	

→ let’s check these three ideas by experiment

## Slide 46 “Let’s check these three ideas by experiment”

The experiment was done in the following way.

Let’s check these three ideas by  
experiment



## Slide 47 “Scheme of the present experiment”

First, I use a tiny {TA-INY} hypervisor with minimum functionality. It has basic functions and time cheating. This is the most complicated case for detection.

Next, you can see the source code fragment for measuring “I Ee Tee”.

The inner loop runs one thousand (1000) times and measures time, after that I suspended this driver for a two-second delay to make observations {ABZERVATIONZ} independent.

The outer loop runs ten (10) times. As a result, I got a matrix with one thousand (1000) by ten (10).

**Scheme of the experiment**

1. Run a tiny HYP with time cheating
2. Measure IET by the own driver:

```
for ( 10 ) /*< outer loop */
{
    for ( 1000 ) /*< inner loop */
    {
        read_tsc(T1)
        CPUID // #1
        ...
        CPUID // #10
        read_tsc(T2)
        save_one_IET_value(T2-T1)
    }
    Sleep( 2 sec )
}
```

→ matrix 1000 x 10

## Slide 48 “Experimental matrix of IET values”

Here you can see the example of this matrix for no hypervisor case.

Such matrices were measured for both {BO-UTH} situations without a hypervisor and with a hidden one.

To make it clear let me show you fragments of columns with the help of a “scatter plot”.

Instruction Execution Time in CPU ticks\*

		Number of outer loop interactions				
		1	2	3	...	10
Number of inner loop interactions	1	2004	2008	2048	...	2044
	2	2000	2008	2048	...	2048
	3	2012	2004	2048	...	2044
	4	2008	2000	2048	...	2048
	5	2008	2004	2044	...	2040
	...	...	...	...	...	...
	1000	2008	2000	2040	...	2036

\* without HYP, Lifebook E752 Core i5, Windows Live CD XP DDD

## Slide 49 “Experimental results”

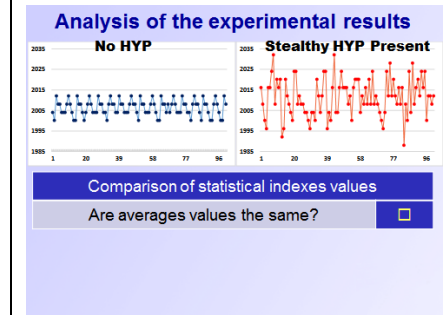
Here you can see the scatter plots for two mentioned cases.

This is the plot for the case with no hypervisor, and here is the plot with one hidden hypervisor.

Each point on the plot represents a value from the column. Numbers of experiments are on the x-axis.

The y-axis shows the execution {EX-SECUTION} time in CPU ticks or “I Ee Tee” values.

Let’s compare the average values of these two plots.

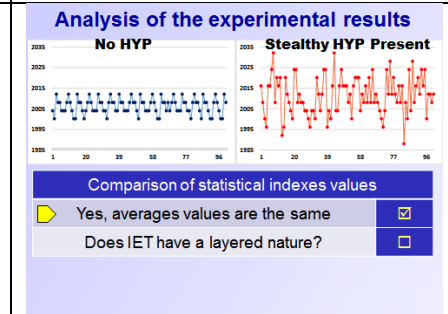


## Slide 50 “Experimental results” {magnify average values}

We can see that due to time cheating the average values of these two cases are the same.

The average is about two thousand and seven (2007).

As a result, existing time based detection method does not work.

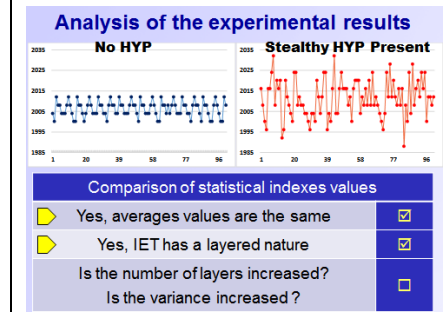


## Slide 51 “Experimental results” {magnify on the layers}

However, something is different. Let’s focus on the structures of these plots.

Without a hypervisor, a set of points consists of few horizontal layers. There are just 4 layers and no outliers. With one hypervisor present, a set of points consists of several layers with a lot of outliers.

All these plots in both {BO-UTH} cases have a layered structure.



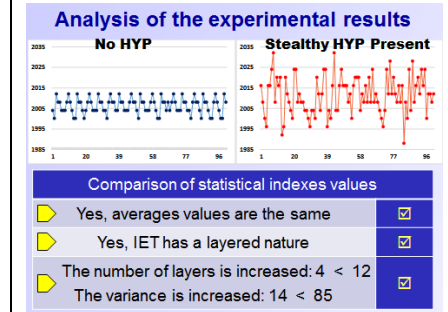
## Slide 52 “Experimental results”

Moreover, we can see that after a hypervisor has been loaded, the number of layers has grown three-fold.

Also, a variance has increased {INCRI-ST} five-fold.

We see that all variability indexes are increased {INCRI-ST} so they can be used for detection.

Perfect!



## Slide 53 “Perfect! We’ve done it!”

We can conclude that all my theoretical {THIO-retical} principles have been confirmed by experiments.

In particular, “I Ee Tee” is a random variable, whose spread increases after the loading of a hypervisor.

Now we are focusing on three types of statistics: number of layers, variance and fourth (4<sup>th</sup>-) order moment.

To detect a hypervisor we need to compare samples in both {BO-UTH} cases without and with a hypervisor. But it isn’t so easy! Let’s look at the reasons.

**Yeah! We’ve done it!**

We’ve found the following “resilient” statistics:

- number of horizontal layers
- variance  $V = \frac{\sum (x_i - \bar{X})^2}{n}$
- 4<sup>th</sup> order moment  $\bar{M}_4 = \frac{\sum (x_i - \bar{X})^4}{n}$

Let’s use statistical tests to complete samples

#### Slide 54 “But also IET has the following anomalies:”

My experiments {I-KSPERiments} show that, apart from the layered {LAYYED} nature “I Ee Tee” samples also include partial deviation and other anómalies, like noise which significantly change variability indexes, so that they cease {SEASE} to be useful for detection.

In addition, our analysis shows that “I Ee Tee” is not normally distributed.

Let’s think which of the statistical tests are applicable in our situation.

##### But also IET has the following anomalies:

- IET samples include noise
- IET samples statistics fluctuate daily
- IET random variable is not normally distributed

What statistical tests are appropriate to compare these samples?

#### Slide 55 “Possible ways to compare samples”

First, we cannot use classical statistical or parametric tests, for example, Student’s t-tests, because all of them require normal distribution.

Non-parametric tests are possible. But, our experiments show that these tests, for example a Wilcoxon test, are not precise enough for our experimental data.

##### Possible ways to compare the samples

Classical parametric tests	Non-parametric tests
<ul style="list-style-type: none"><li>• Student's t-test</li><li>• ANOVA &amp; ANCOVA</li></ul>	<ul style="list-style-type: none"><li>• Wilcoxon test</li></ul>
Require normal distribution	Give bad approximation

#### Slide 56 “Possible ways to compare samples”

To compare these samples I use another non-classical method, which was introduced by Mark Kornfeld in Russian and later by Johann {Yo-HANN} Strellen in English. Using this method the confidence interval is calculated {CAL-KJU-LATED} as the interval between the minimum and maximum values.

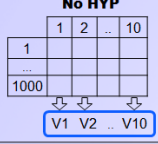
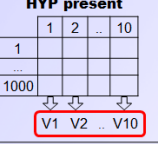
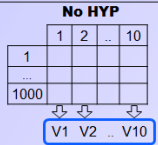
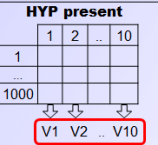
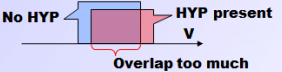
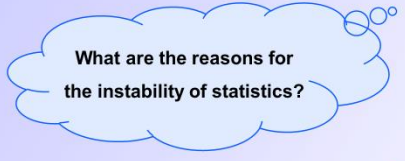
##### Possible ways to compare the samples

Classical parametric tests	Non-parametric tests
<ul style="list-style-type: none"><li>• Student's t-test</li><li>• ANOVA &amp; ANCOVA</li></ul>	<ul style="list-style-type: none"><li>• Wilcoxon test</li></ul>
Require normal distribution	Give bad approximation

Kornfeld (USSR'65) or Strellen (FRG'01) method:

Let  $T_1, T_2, \dots, T_n$  is a sample, therefore  
▶ confidence interval:  $(T_{MIN}, T_{MAX})$   
▶ confidence level:  $P = 1 - 0,5^{n-1}$



<p>Here is the corresponding <u>confidence level</u>. Let's use this method.</p>	
<p><b>Slide 57 “Calculate statistics &amp; variation intervals”</b></p> <p>First, I measured “I Ee Tee” for two situations: without a hypervisor and with it.</p> <p>Then I calculated statistic value, for example, variance according to the matrixes columns {CALOMS}.</p> <p>After that, I compared these variation intervals with each other: blue and red intervals.</p>	<p><b>Calculate statistics &amp; variation intervals</b></p> <p>1. Calculate variances for each matrixes of IET values:</p> <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p><b>No HYP</b></p>  </div> <div style="text-align: center;"> <p><b>HYP present</b></p>  </div> </div> <p>2. The result:</p>
<p><b>Slide 58 “Calculate statistics &amp; variation intervals”</b></p> <p>As a result, I <u>found out</u> that the variation intervals overlap so much, that I could not get the threshold value to detect a hypervisor.</p>	<p><b>Calculate statistics &amp; variation intervals</b></p> <p>1. Calculate variances for each matrixes of IET values:</p> <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p><b>No HYP</b></p>  </div> <div style="text-align: center;"> <p><b>HYP present</b></p>  </div> </div> <p>2. The result: instability of statistics values</p> <div style="text-align: center;">  <p>Overlap too much</p> </div>
<p><b>Slide 59 “Data fluctuation: instability of statistics”</b></p> <p>What are the reasons for this data fluctuation?</p> <p>We need to understand why the calculated statistics are so different.</p>	<p><b>Data fluctuation: instability of statistics</b></p> <div style="text-align: center;">  <p>What are the reasons for the instability of statistics?</p> </div>

## Slide 60 “Reasons of data instability or data fluctuations are outliers & jumps”

To answer this question I repeatedly made scatter plots and now show you typical anomalies.

{*magnify up-left*} We see that this scatter plot includes a high outlier {OUTLAE} apart from basic layers.

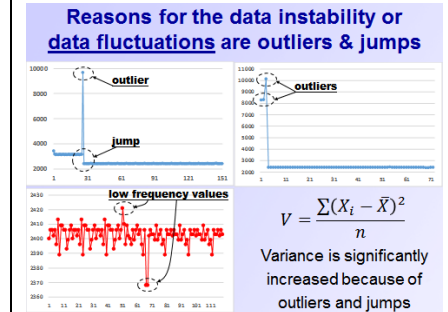
Also, the plot includes one jump.

{*magnify up-right*} This is an example of high outliers.

{*magnify left-down*} And this is an example of low frequency values.

All these anomalies considerably influence the variance and the other statistics.

As a result, I cannot calculate the appropriate {APRO-UPRIAT} threshold values.



-up left

-up-right

-left-down

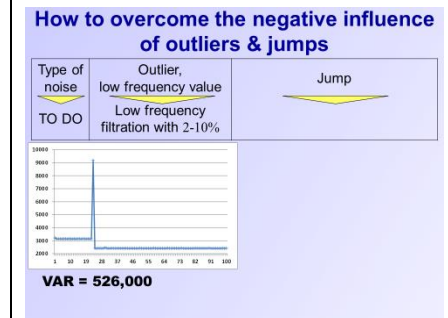
## Slide 61 “How to overcome negative influence of these jumps & outliers”

Let’s look at how to reduce the influence of these anomalies.

To remove outliers and low frequency values I applied filtration.

As a result, all values whose frequency is lower than 2, 5 or 10% are deleted.

Here is the original plot with one outlier and jump. The corresponding variance is about half a million.

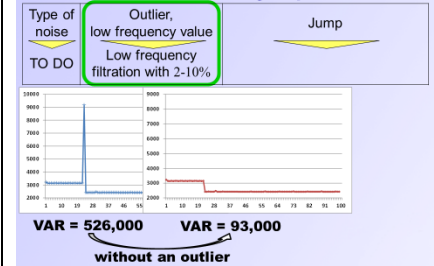


## Slide 62 “How to overcome negative influence of these jumps & outliers”

After the filtration, one outlier {OUTLI-YA} was deleted, and variance became about one hundred thousand. You see the difference.

Let's move on to the jump.

### How to overcome the negative influence of outliers & jumps



## Slide 63 “How to overcome negative influence of these jumps & outliers”

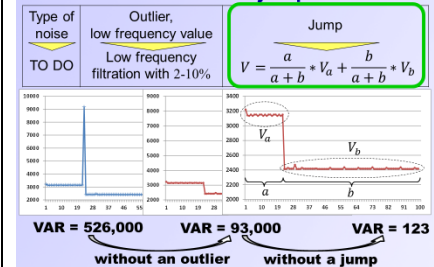
To correct each jump I calculated the statistic values, for example variance, before and after the jump.

Next, I added these values using their section lengths. In other words, I averaged out the statistic before and after the jump.

Finally, the variance is a bit more than a hundred.

These measures significantly helped me to stabilize statistics and their variation intervals.

### How to overcome the negative influence of outliers & jumps



## Slide 64 “Obtain different statistical values on different days”

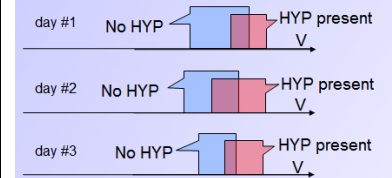
I decided to test these ideas every day and do more experiments.

I decided to test these ideas &  
try to detect a HYP every day

### Slide 65 “Obtain different statistical values on different days”

Here you can see variation intervals for both cases: without and with a hypervisor, which were calculated during three days. Day first, second and third.

#### Obtain different statistical values on different days

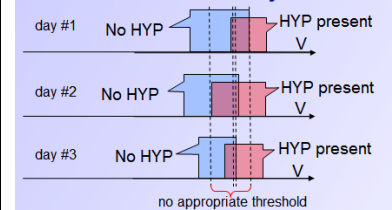


### Slide 66 “Obtain different statistical values on different days”

The point is that I received different variation intervals on different days.

So it seemed impossible to get the appropriate threshold values to detect a hypervisor.

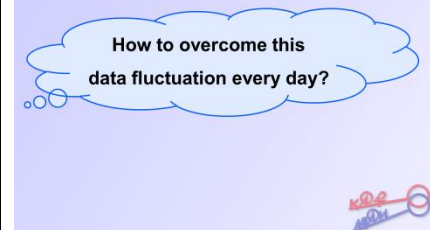
#### Obtain different statistical values on different days



### Slide 67 “Data fluctuation: lack of repeatability”

Now let’s move on to the question – how to overcome this lack of repeatability? {RE’PETA’BELETE}

#### Data fluctuation: lack of repeatability



## Slide 68 “How to overcome lack of repeatability?”

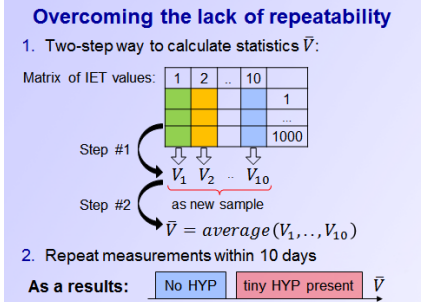
I tackled this issue by using a two-step way to calculate new statistics. First, I calculated, for example, variance, for each column in the matrix and got a set of values: Vee-one (V1), Vee-two (V2) and so on Vee-ten (V10), which I considered {CONSIDED} as a “new sample”.

Second step was to calculate the statistics of this new sample. Here “Vee-bar” was the average of this new sample. As a result overlapping of new statistics intervals was seriously reduced.

Another idea was to repeat measurements within 10 days.

This period was enough to stabilize variation intervals.

These two approaches were sufficient to get appropriate threshold values. It looked like a success.

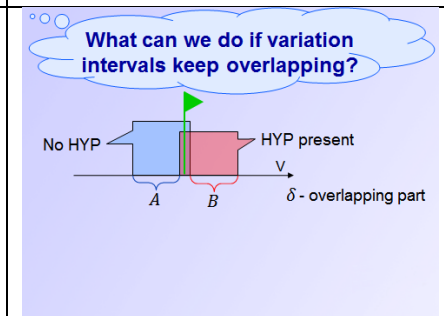


## Slide 69 “What can we do if variation intervals keep overlapping?”

But, what can we do if variation intervals keep overlapping? Look at the green flag.

If the calculated value is in the overlapping part it is impossible to detect a hypervisor precisely.

What should we do in this situation?



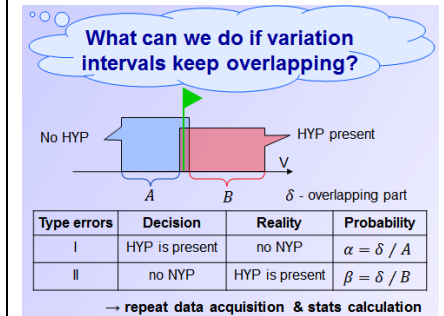
## Slide 70 “What can we do if variation intervals keep overlapping?”

The first idea was to repeat “I Eee Tee” measurements and recalculate statistics.

Here you can see the probabilities of errors of the first and second kinds {KA-INDS}.

The thing was that because this overlapping part is really small, the probability of repeating this negative situation is quite low. It is highly unlikely that the calculated value occurs in the overlapping part again because of the multiplication rule for independent events.

Let’s join all the previous steps to calculate the threshold values.

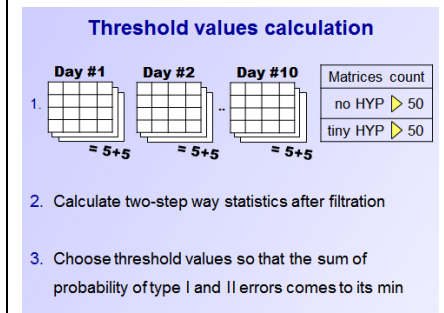


## Slide 71 “Threshold values calculation”

The first step was to obtain the data for ten days. I got five matrixes every day for each case: without a hypervisor and with my tiny {TA-INY} hypervisor. All in all I obtained fifty matrixes for each case.

Next was to calculate statistics using a two-step way.

The final step was to calculate threshold values. The results are in the table.



## Slide 72 “Example of threshold values”

This is the table with threshold values.

If the calculated number of layers is “5”, so it is smaller than threshold “7” and it means there is no hypervisor. If the calculated number of layers is bigger than “8”, it means that at least one hypervisor is present.

The complete table with threshold values is in my paper.

### Example of threshold values

Intel Core 2 Duo E6300 + Windows 7 x32

Statistics	Filtration level	Threshold values		Type I error, %	Type II error, %
		No HYP	HYP is present		
Number of layers	0	≤ 7	≥ 8	4	0
Variance	0	≤ 14	≥ 18	2	0
Moment	0.1	≤ 679	≥ 947	2	0

## Slide 73 “How to detect stealthy hypervisors?”

Let’s sum up all previous ideas to formulate how to detect a stealthy hypervisor.

How to detect stealthy hypervisors?  
Step by step method:

## Slide 74 “How to detect a stealthy hypervisor?”

First is to guarantee that there is no hypervisor in BIOS by reflashing it.

Next is to reinstall operating system, which is downloaded from the reliable sources.

The first two steps look like performing a factory data reset in computers.

Next is to calculate threshold values from the no hypervisor case.

After that install supplementary software and monitor messages about a hypervisor presence.

### How to detect stealthy hypervisors?

Stages	Stage description
Preliminary (calculate thresholds)	1. Flash BIOS with a trusted image or firmware
	2. Install OS
	3. Get threshold values in case where no HYP is present
Operational (detect a hypervisor)	4. Check in a loop if a hypervisor is present
	5. Install Office etc
	6. Monitor messages about a hypervisor presence
	7. Go to step 3 to adapt the tool to new legitimate HYP



## Slide 75 “How to detect a stealthy hypervisor?”

You can say, all right Igor, but your head is in the clouds.


The first two steps are good only for crazy computer experts. They are not appropriate for everyday life.

Yes, you are absolutely right. Let’s keep both {BO-UTH} feet on the ground.

How to make it userfriendly?

### How to detect stealthy hypervisors?

Stages	Stage description
Preliminary	1. Flash BIOS with a trusted image or firmware 2. Install OS



\*\* Detached BIOS Chip, wikipedia.org/wiki/BIOS  
\* Eprom programmer, batronix.com/versand/programmiergeraete/BX32P/index.htm

## Slide 76 “How to detect a stealthy hypervisor?”

Now I have the basis to guarantee no hypervisor presence in the running computer just by checking the structure of the scatter plot.

The details are {AR} in the paper.

### How to detect stealthy hypervisors?

Stages	Stage description
Preliminary (calculate thresholds)	1. Guarantee the absence of a HYP by checking a scatter plot (coming soon) 2. Get threshold values in case where no HYP is present
Operational (detection)	3. Check in a loop if a hypervisor is present 4. Install Office etc 5. Monitor messages about a hypervisor presence 6. Go to step 3 to adapt the tool to new legitimate HYP

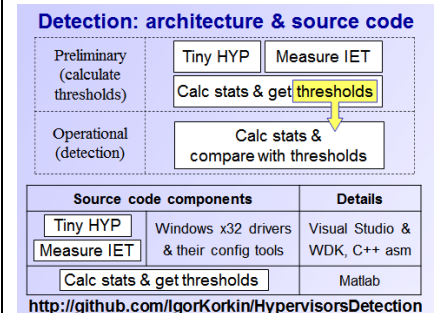
## Slide 77 “Detection: architecture & source code”

Here you can see the architecture of this detection system.

At the preliminary stage, I use “measure IET” block, tiny hypervisor, block for calculating statistics and getting thresholds.

At the operational stage, I calculate statistics and compare their values with thresholds.

This detection system is developed using Cee + + and MATLAB, the source code is free & available here.



## Slide 78 “Positive results on different PCs & HYPs”

Experiments {IkspErimens} proved that this system detects any hidden hypervisor, as well as 2 nested ones. Here you can see cases, which were considered. There are hypervisors based on drivers and BIOS-based ones. Detection system works well on both {BO-UTH} CPUs: on Intel and AMD.

### Positive results on different PCs & HYPs

Is run by	HYP title	HYP authors & details	CPU
Driver	Only 1 tiny HYP	tested on 5 PCs	
	2 nested HYPs= ADD* + tiny HYP	ADD is loaded first, the tiny HYP is above it	
BIOS	TRace Explorer (TREX)	A.Tichonov & A.Avetisyan (ISP RAS)	
	Russian Ghost	A.Lutsenko aka R_T_T	

ADD — Acronis Disk Director for Windows x86

## Slide 79 “List of challenges and how I accepted them”

To conclude, I’d like to highlight all the challenges which I faced and how I dealt with them.

**First**, a stealthy hypervisor uses time cheating to prevent its detection. I have solved {SOLVD} this by using variability indexes. **Second** is the data fluctuation and noise values. I solved them by applying filtration, calculating statistics using two-step way and repeating measurements. **One more thing**, there were no appropriate methods to compare samples. I used the Kornfeld method for solving this task.

### List of challenges

Challenges	How to achieve
Stealthy HYP cheats time	▶ Use variability indexes of IET
Data fluctuation: jumps & outliers	▶ Apply filtration & two-step way statistics
Lack of repeatability	▶ Repeat measurements within 10 days
And also:	
IET is not normally distributed & no HOV	▶ Use Kornfeld method

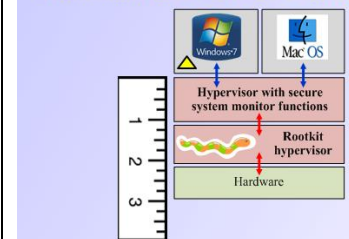
## Slide 80 “Statistical ruler” detects stealthy HYPs

In a nutshell, in this paper I presented a statistical ruler for hypervisors. This is the first software system, which can detect a stealthy hypervisor and calculate several nested ones even under countermeasures.

If you are interested I can show how it works, it will take about twenty minutes.

Thank you.

### “Statistical ruler” detects stealthy HYPs



▲ - the detection tool is running in the background



## ADDITIONAL SLIDES

### Slide 1 “Additional slides”

### Slide 2 “Почему Вы отказались от использования классических методов статистики?”

Слева классический метод Стьюдента, справа метод Корнфельда.

В нашем случае нарушаются условия применения параметрических тестов. Для сравнения выборок с помощью классических параметрических критериев необходимо одновременное выполнение двух условий:

1. Закон распределения обоих сравниваемых выборок должен быть близок к нормальному закону распределения
2. Дисперсии сравниваемых выборок должны быть однородными.

Только при соблюдении этих двух требований возможно использование классической мат статистики. В нашем случае оба условия не выполняются, поэтому применять классические методы мат статистики некорректно.

Был применён непараметрический метод Корнфельда, который не требует выполнения ни

#### Analysis of sample comparison methods

Student t-test (parametric stats)	Kornfeld method (non parametric stats)
1. Requirements to input samples	
Normal distribution ✗ Homogeneity of variances ✗	No requirements ✓
2. Calculating confidence interval for mean	
$(\bar{x} - t * s_{\bar{x}}, \bar{x} + t * s_{\bar{x}})$ , $s_{\bar{x}} = s / \sqrt{n}$ , $s^2$ -sample variance	$(x_{min}, x_{max})$
3. Getting corresponding confidence level	
95% or 99% see t-table	$1 - 0,5^{n-1}$

нормального закона распределения, ни однородности получаемых выборок.

Отмечу два благоприятных фактора для применения этого метода:

1. Возможность получать выборки больших объёмов, порядка 10 тысяч элементов.
2. Возможность повторять такие опыты.

Чего не наблюдается на практике в большинстве случаев параметрической статистики.

Also I got a huge amount of experimental data and repeated measurements as much as I needed, which is not always possible to other statistical cases.

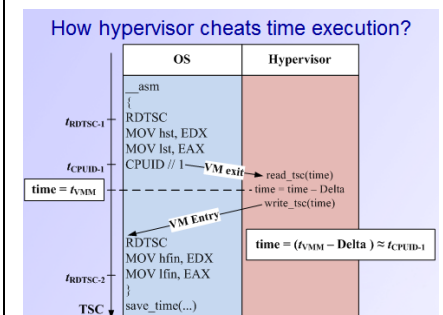
Указанные факторы способствовали получению приемлемых статистических результатов для обнаружения гипервизора.


### Slide 3 “How hypervisor cheats time & duration”

This scheme presents the interaction between the operating system and the hypervisor during measuring the instructions execution time.

We measure time of CPUID execution using RDTSC which reads the value of time stamp counter.

Every time when our driver calls CPUID, the VM exit occurs. Each VM exit causes the hypervisor code



<p>to run. After that, the hypervisor reads the time stamp counter, decreases its value by delta and writes the updated value back to counter.</p> <p>By trial and error we choose an appropriate delta value. As a result the registered duration is increased and is the same with the value without a hypervisor.</p>	
<p><b>Slide 4 “Min-max confidence interval method by Strelen’04 or Kornfeld’65”</b></p> <p>Let <math>X_1, X_2, \dots, X_n</math> be a random sample of variable <math>X</math>.</p> <p>The confidence interval for average <math>X</math> is between <math>X_{\min}</math> and <math>X_{\max}</math>, with the following confidence level.</p> <p>In other words, the probability that <math>X</math> is be between <math>X_{\min}</math> and <math>X_{\max}</math> is the following.</p>	<p>Min-max confidence interval method by Strelen'04 or Kornfeld'65</p> <p><math>x_1, x_2, \dots, x_n</math> - is a random sample</p> <p>The confidence interval for <math>\bar{x}</math> is <math>(x_{\min}, x_{\max})</math></p> <p>where <math>x_{\min} = \min_{1 \leq i \leq n} x_i</math> and <math>x_{\max} = \max_{1 \leq i \leq n} x_i</math></p> <p>with the confidence level</p> $P\{x_{\min} \leq \bar{x} \leq x_{\max}\} = 1 - 0,5^{n-1}$ 
<p><b>Slide 5 “Comparison of the confidence interval methods”</b></p> <p>The confidence intervals method was introduced by Kornfeld in Russian and later by Strelen in English.</p> <p>This is just the comparison of Kornfeld and Strelen ideas.</p> <p>You can see the similar formulas and results: Kornfeld sorts sample values by ascending, that is why <math>X_1</math> is the minimum and <math>X_n</math> is the maximum. Strelen gets straight the maximum and the minimum.</p> <p>Formulas with the confidence intervals are the same.</p>	<p>Comparison the confidence interval methods Kornfeld method (USSR 1965)</p> <p>Обозначим истинную величину через <math>x</math> и расположим полученные при <math>n</math> измерениях числа в порядке возрастания: <math>x_1, x_2, x_3, \dots, x_n</math></p> <p>Тогда, в соответствии со сказанным, вероятность неравенства <math>x &lt; x_1</math>, так же как и вероятность неравенства <math>x &gt; x_n</math>, будет равна <math>(1/2)^n</math>.</p> <p>Отсюда вероятность того, что не удовлетворяется ни одно из этих неравенств, т. е. вероятность неравенства <math>x_1 &lt; x &lt; x_n</math>, есть</p> $p = 1 - 2(1/2)^n \quad (2.1)$ <p>Таким образом, с надежностью <math>p</math> можно утверждать, что <math>x</math> находится в пределах между <math>x_1</math> и <math>x_n</math>.</p> <p><small>—Kornfeld, B. (1965, 1967). Accuracy and Reliability of a Simple Estimator. In: Russian: 85-93, English: 83-942, 835-842. Reprinted in: Science 15, 274, from: http://csl.stanford.edu/~csli/lectures/</small></p> <p>Strelen method (Germany 2001)</p> <p>Now let <math>\tilde{X}_1, \dots, \tilde{X}_n</math> be IID random variables with finite mean <math>\mu</math> and finite median <math>\tilde{\mu}</math>, hence <math>P\{\tilde{X}_i &lt; \tilde{\mu}\} \leq 0.5</math> and <math>P\{\tilde{X}_i &gt; \tilde{\mu}\} \leq 0.5</math>. With <math>\tilde{X}^{\min} = \min_{1 \leq i \leq n} \tilde{X}_i</math> and <math>\tilde{X}^{\max} = \max_{1 \leq i \leq n} \tilde{X}_i</math>, <math>P\{\tilde{X}^{\min} &lt; \tilde{\mu} &lt; \tilde{X}^{\max}\} \leq 1 - 0.5^{n-1}</math> holds.</p> $[\tilde{X}^{\min}, \tilde{X}^{\max}]$ <p>is the median confidence interval for the confidence level <math>1 - \alpha</math> with <math>\alpha = 0.5^{n-1}</math></p> <p><small>© Dr. Stefan Wedler: Confidence Intervals. In: E. J. H. Kerckhoffs and M. Bronck, editors. Modeling and Simulation 2001 - Proc. of the ESM2001, pages 771-776. The ESC Publishing House, Braunschweig, 2001. http://web.informatik.uni-siegen.de/stw/lehre/ModellingSimulation/ESM2001.pdf</small></p>

## Slide 6 “How to debug HYPs”

These are the ways how to debug a hypervisor.

1. To debug a hypervisor driver we can use Vmware with kernel debugger WinDbg.
2. We can also apply hardware emulator Bochs ("box") or AMD SimNow and their own debuggers.
3. It is possible to use classical approach with DbgPrint messages and DbgView to collect it. This approach is appropriate more for demonstration not for debugging.
4. Another approach which is used in BluePill is to send debug messages to COM port.
5. The last method is to use “Debug PCI Card” to see, for example POST BIOS messages.

### The possible ways to debug a hypervisor

- WinDbg and VMWare to debug a HYP driver
- hardware emulators: Bochs, AMD SimNow etc
- debug output by DbgPrint & DbgView

- COM port:



- debug card



## Slide 7 “Details of type I and II errors”

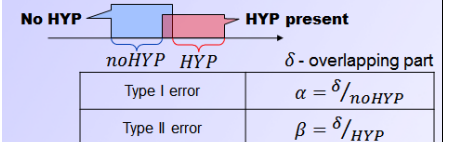
You can see the meaning of the type I and II errors and how to calculate them.

### Details of type I and II errors

1. Definition of type I and II errors:

		Reality situation	
		No HYP	HYP present
Our decision	No HYP	"true"	<i>type II</i>
	HYP present	<i>type I</i>	"true"

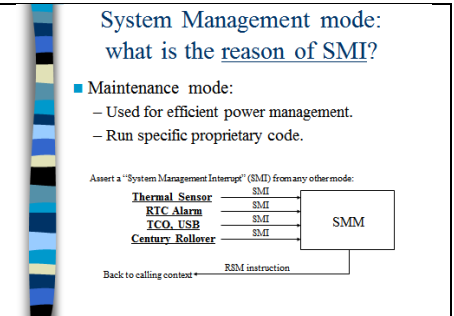
2. How to calculate them?





## Slide 8 “Details of type I and II errors”

Here you can see the reasons of interruption to System Management Mode, there are signals from Thermal Sensor, RTC alarm, using USB devices instead of COM, like COM keyboard and mouse.



## Slide 1 Hello

I’d been researching this topic for 2 years during my Ph.D. And I think my experience might be interesting and helpful.

First of all, I’d like to specify two points of this work. “Time Cheating” means that a hypervisor can prevent its own detection by using different countermeasures.

“Data Fluctuation” ... is the concept of statistics ... means that the data received from experiments are often not applicable for detection, because the data values differ and they often differ significantly.

## TWO CHALLENGES OF STEALTHY HYPERVISORS DETECTION: TIME CHEATING & DATA FLUCTUATIONS

Igor Korkin

CDFSL 2015

## Slide 2 “Agenda”

This talk consists of three parts. During the first part of my talk, I’d like to describe a hypervisor as a “sophisticated cyber threat”. During the second part of my talk, I’d like to look at the existing “hypervisor detecting tools” and “approaches” and their “drawbacks”. The third part covers a



“new hypervisor detecting method” and “challenges”, which I <u>dealt with</u> during my work.	
---	--

## Scatter plots & frequency polygons

I repeatedly make scatter plots and a corresponding frequency polygon or relative frequency chart. Polygon dots correspond to the layers on the scatter plot. We see that scatter plot includes tall outliers, apart from basic layers. Similarly, the frequency polygon has a long tail or has a large skew.

First, it's important to remember that interruptions to S-M-M have a random nature; therefore, the I-E-T duration is a random variable as well. Moreover, we can find another feature of I-E-T duration. It is randomly distributed in several fixed layers. The number of S-M-M interruptions determines the number of layers.

And also, the Instructions Execution Time is a random variable, which is not normally distributed. At the same time non-parametric methods are not precise enough.

And the last case is the intersection of two variation intervals. My idea is repeat the measurement and calculation.

All these anomalies have a significant influence on the calculated statistical values; as a result we cannot use it for detection.