Malicious Hypervisor Threat – Phase Two: How to Catch the Hypervisor

Research and development by Rubos, Inc. team

(We do independent research on security matters in various domains)

Prepared for DeepSec 2016

Speaker: Mikhail Utin, PhD, CISSP

mikhailutin@hotmail.com

(Questions will be answered after the presentation. Please, submit them to the speaker in writing)

DEEPSEC

Introduction

Malicious Hypervisor (MH) Phase 2 Topics

Phase 1 research (refresher):

- The threat is real and there are likely three instances in use today (two US and one Russian)
- MH provides unlimited control over affected system
- No identification method/software tool

Phase 2 research and development:

- **Technology Vulnerability** – whose responsibility is to fix it and Intel Corporation reaction on two vulnerabilities
- Existing ideas and methods for "rootkit hypervisor" questionable and not practical
- Hypervisor Catcher © software (patent pending) research, ideas and methods
- Testing, statistics and results

Phase 1 – Refresher

Russian Ghost Myth – Case #1 and Following Research (2013 – 2014)

Our research Phase 1 was about three cases:

Case #1 - *Russian research on Malicious BIOS Loaded Hypervisor* (approximately 2007 – 2010); posted at the end of 2011

Case #2 - *US Michigan University Virtual-Machine Based Rootkit* research which was done approximately in 2005 – 2006 years (published 2006)

Case #3 - *US Michigan University IPMI/BMC (Intelligent Platform Management Interface/Baseboard Management Controller) vulnerability* research of approximately 2012 – 2013 (published 2013)

Phase 1 Research Direction

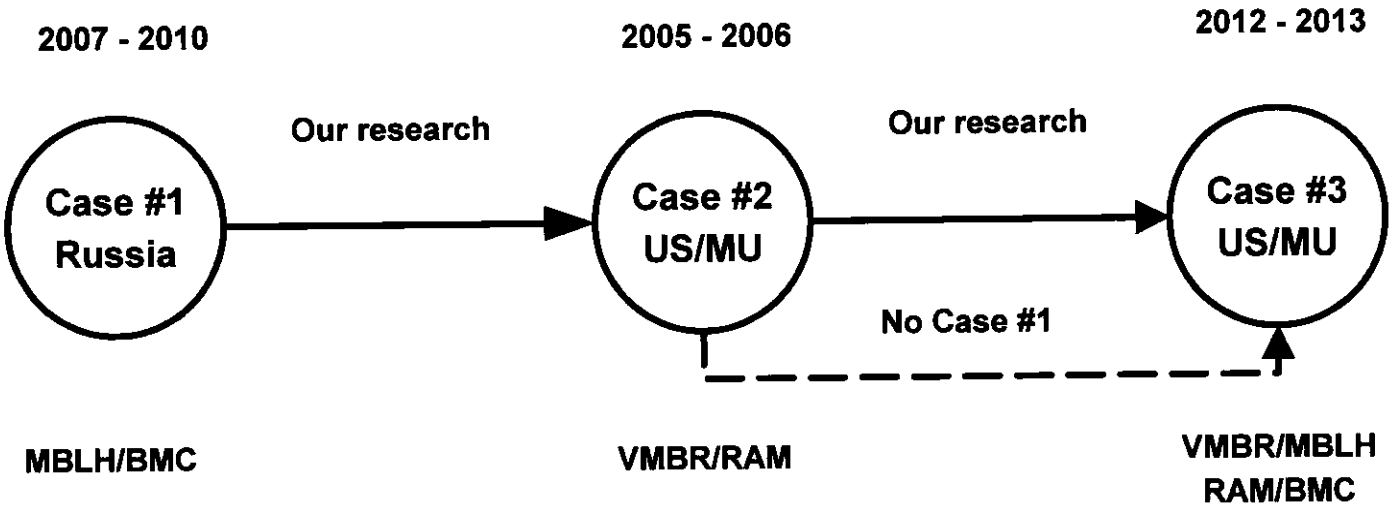


Fig. 1. The process of 2013 - 2014 Malicious Hypervisor research

Case #1 Most Important Conclusions

1. MH has been embedded in BMC management system software. This software is encrypted by Intel secret key and is decrypted when loaded in BMC RAM. Required: the key, BMC software code, software and hardware to work with BMC code and flash memory, the access to logistics system (intercept, change, ship)

   Should a "backdoor" be embedded as well?

2. Motherboards causing problems (having MH) has been labeled "Assembled China" and the clean one - "Assembled Canada". Neither country had motherboard assembly facility in 2007 - 2008. Intel has and had only one facility in Vancouver, Canada – Flash Memory Group (FMG), which is the only one in Intel dealing with flash memory. Labels "Assembled … " were to mislead.
3. Time correlation between Case #1 and Case #2 – MH appeared approximately one year after MH Case #2 publication. It was not able to support nested virtualization either.
4. Case #1 - MH development project by unknown entity- was definitely complex and long term – tune-up, manufacturing, delivery

How Many MH Instances Are There?

Our search did not reveal any other public documents describing anything close to MH in Michigan University Case #2.

We believe that there are three instances:

- The first one is the original Michigan University Case #2 hypervisor, which can be used on systems without hardware assisted virtualization
- The second instance is described in Russian Case #1 - an improved version of the first one: it has been found in "Assembled in China" motherboards and working with hardware assisted virtualization implementing nested hypervisors
- The third instance is likely the result of two discussions of MH threats and advantages with Russian FSB (former KGB) specialists in computer security; this instance could have appeared in the wild around 2011 – 2012 years.

Michigan University Case #3 – The Threat of Easy Distribution of MH

- MH installation requires administrator/root level of local system access and local MH copy or downloading from remote source
- Both tasks get simpler if IPMI/BMC vulnerability is used (Illuminating the Security Issues Surrounding Lights-Out Server Management by Anthony J. Bonkovski et al. 2013). This Michigan University research (Case #3) identified close to one hundred thousand servers ready to be compromised. MH can be installed in main system if exists in BMC memory
- IPMI and BMC neither by design nor implementation have any real security controls – Linux bare-bone OS and server firmware is rarely updated.
- US CERT Alert : TA13-207A “Risks of Using the Intelligent Platform Management Interface (IPMI) July 26, 2013” which describes the risk as (quote) “Attackers can use IPMI to essentially gain physical-level access to the server…”.

Combined Threat

- Therefore we have a combination of two threats – MH and IPMI/BMC which makes possible massive delivery and installation of extremely dangerous malicious software. Such software could be developed within one – two years by qualified group of three people and distributed in millions of computers.
- Are we dealing with software vulnerabilities yet to be fixed?

Phase 2 Research and Development Technology vulnerability

Not Software but Technology Vulnerabilities - 1
We Are Talking NOT about Software Bugs

Definitions from Wikipedia:

General - "**Vulnerability** refers to the *inability* (of a system or a unit) to *withstand* the effects of a *hostile environment*." Simply– it is inability to resist a threat.

In computing - "In computer security, a **vulnerability** is a *weakness* which allows an attacker to *reduce* a system's *information assurance*. Vulnerability is the intersection of three elements: a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw."

MH exploits modern operating systems *inability to resist involuntary virtualization after OS is installed and functioning*. Operating systems do not have protecting mechanisms against malicious virtualization.

Not Software but Technology Vulnerabilities - 2

IPMI & BMC technology vulnerabilities are very similar:

- There is IPMI technology vulnerability of seamless and *unprotected* system level *access* to computer *resources* which is usually utilized by computer management system

- BMC technology vulnerability is *unprotected implementation* of system management software embedded in BMC.

What about fixing technology vulnerabilities? In particular, when such technologies has been in use for years and are likely to be used in near future.

Vendors should take care of such vulnerabilities, and in particular when technology has underlying standard like IPMI.

Microsoft and Intel were two of three of sponsors of Michigan University Case #2 MH research. It means that *discovered vulnerabilities of Windows OS and IPMI & BMC* and underlying computing technology *were known back in 2006*.

Vendors , Vulnerabilities and (Ir)Responsibility

However, we've never heard any word from Microsoft addressing virtualization vulnerability. Neither Intel Corporation reported of working on IPMI & BMC vulnerabilities, which create the gateway to the exploitation of combined vulnerability. Intel Corporation:

- Has 90% of CPU market
- Makes chip sets for motherboards
- Makes motherboards
- * Developed and supports CPU Hardware Assisted Virtualization
- * Developed and supportsComputer System Management Software (CSMS)
- Utilizes * Linux CSMS OS
- Provides production support, including bugs fixing, etc.

* - Are related to our case and may be vulnerable

We requested Intel comments on vulnerabilities

Intel Production Security Team Response

- We'd like to thank you for raising your concerns to Intel regarding Intelligent Platform Management Interface (IPMI) and Virtualization Technology (VT). We take the security of our products and infrastructure seriously and work continuously in the security of both. We have carefully reviewed all the information you have provided as well as the resources you have directed us to.
- Intel published the first IPMI specification in 1998. Since that time it has worked with many other companies to extend the specification. As you know, security depends on how the specification is implemented and deployed by the system owner. As you've pointed out there are risks if vulnerabilities exist in the implementation or if the system is not deployed properly. *At this point we have not received any new information from you that Intel implementations of IPMI have a vulnerability.*
- Many parties in the industry, including Intel, provide detailed guidance regarding the proper use of this technology in order to help ensure systems follow good security practices. *At this point we have not received any new information that VT has a vulnerability. If you are aware of one please do let us know.* Additionally, if you'd like us to facilitate a discussion between you and a systems provider whom you've identified a vulnerability we'd be happy to make the connection.
- Regards,

  Intel PSIRT

Technology Vulnerability Is NOT a Vulnerability?

There is nothing new in such position – we have seen on FullDisclosure list - "it is not vulnerability – it is our system feature". If you found bugs in Intel CSMS or embedded OS, they can be discussed. If you are talking about technology vulnerability – no. Technology vulnerability is not recognized as a vulnerability.

It means that we are on our own to address the threat of three vulnerabilities in question.

Why IPMI/BMC is so vulnerable? First, as we mentioned above, simple Linux OS does not provide usual Linux security (firewall, SELinux, auto-update). Second, there is human factor. Each new fix in OS and CSMS must be installed, and that is traditionally ignored by server administrators.

What does Intel do to secure its CSMS and embedded Linux OS? We did brief search and found that one older motherboard had rare updates while the new one was updated frequently. It is more likely that old motherboards (thus servers) are more vulnerable.

System administration ignorance and 0-days – main contributors.

Phase 2 Research and Development
Existing Ideas and Methods

DEEPSEC

First Step Towards Protection Is Detection - Status

- Root-Kit Hypervisor (RH) traditional identification methods - software runs within installed OS while trying to catch hypervisor operations below it in virtualization level by, for instance, signatures of operations.
- Main problem – RH/MH can intercept identification software activity and block it or alter results. Proposed method of Instruction Execution Time IET) in “Two Challenges of Stealthy Hypervisor Detection: Time Cheating and Data Fluctuations, Igor Korkin, CDFSL 2015” is the most advanced but seems to have the same weakness – it is executed in Secure Hypervisor environment (SH), which is installed above MH, and thus MH could alter testing results or block SH
- IET method requires significant time for data collection and statistic filtering – the paper describes 10 days of collection.

Phase 2 Research and Development Hypervisor Catcher – Research, Ideas and Methods

DEEPSEC

Ideas and Methods – Practicality and Simplicity - 1

We have done the research of how to identify a hypervisor installed in a computer system and following development and testing of the software during 2015 – 2016.

Methods and software are US patent pending

The following, however, is not complete description of what we have done and what exactly was implemented. We would like to set up a discussion of ways and means of MH identification

Our main goal is encouraging the audience to continue the research and find possibly better methods following *the simplicity and practicality requirement. Our basic ideas:*

- MH should consume CPU resources around 1% utilization, therefore increasing execution time; we can then use Time Difference Identification (TDI)
- MH is emulation software and may react differently on some operations increasing deviation - Deviation Difference Identification (DDI) method

Ideas and Methods – Practicality and Simplicity - 2

- How to identify 1% difference in tests? We decided to use 100% CPU utilization. If our identification software execution time without hypervisor (clean system) is Tc then with a hypervisor – Th - we will get hopefully $1.05Tc < Th < 1.01Tc$
- What to run as basic identification software? We opted for software performing various intensive calculations in computer memory and utilizing very basic CPU commands only. The idea behind that is to masquerade identification activity as well. MH thus will be unable to identify whether it is running general purpose software or MH identification software.

How to Identify?

TDI method - we need to find a difference in execution time between the same system with and without a hypervisor using two phases testing:

- **Clean system – the same model right after production or disabling virtualization support via system BIOS (if supported)**
- **Statistical "clean system" variant – assuming that "clean" systems are more common and MH infected are unusual, we can collect a database of multiple systems testing and identify clean**
- **Disabling virtualization support in BIOS, however, may be intercepted by MH and considered as the identification attempt. MH can then block changes and alter output**
- **DDI testing in general also requires two phases as we are going to use execution time difference. However, we were able to discover specific effect of virtualization which made possible to test only once**
- ***Statistical filtering in both methods as we expect high deviation***

Phase 2 Research and Development Testing, Statistics and Results

Development and Testing

We used the most advanced and reputable hypervisor for testing – VMware ESXi 5.5.0 with VMKernel Build 2068190

The development and testing environment:

- Two high end Lenovo notebook computers supporting Intel hardware virtualization
- Desktop computer running VMware vSphere Client 5.5 to run our software
- Bootable OS CD with HyperCatcher auto-starting executable

Initial tests utilizing CentOS Live CD v.6.x had high deviation level of execution time caused by various OS services (GUI, update, security, etc.).

Further testing used CentOS 7.0 Minimal Installation and finally Ubuntu 14.04

Statistical Filtering

Three steps testing procedure for both TDI and DDI - “run”:

- First step is to execute basic identification software a few thousand times to decrease the deviation and then calculate average value for all CPU cores; this is a “cycle”
- Then do several cycles in a trial, find the average execution time value for this trial and the deviation
- Final step in statistical filtration by executing several trials in one run. The average time value is to be used to calculate the time increase of TDI testing. The deviation proves that the execution time random value is inside three standard deviation intervals
- If the deviation is still high, there is the possibility for fourth step of statistical filtering by executing a few runs in this test. However, as of today we use three steps filtration.

Deviation Difference Identification Testing

Execution of some operations changes hypervisor behavior:

- Increase of time difference (TDI) – approximately twice
- Increase of time deviation – from 2 to 50 times randomly

Thus, our idea of forcing the hypervisor to change its behavior was correct and such testing is named Deviation Difference Identification (DDI).

Table 1 shows results without hypervisor; we did only ten runs to accumulate statistics without hypervisor as deviation is really low; thus we do not need to collect more statistics. Tables 2 and 3 represent the total of 40 runs with hypervisor. We did 40 runs to show the behavior of time execution deviation if hypervisor is present.

Testing results without hypervisor

Run#	Exec. Time Tci	Deviation Dci
1	5.5013	0.00081
2	5.4992	0.00062
3	5.5017	0.00093
4	5.4987	0.00033
5	5.5024	0.00060
6	5.5035	0.00135
7	5.5006	0.0017
8	5.5002	0.00082
9	5.4981	0.00057
10	5.4998	0.00088
	AvTc=5.5005	AvDc=0.0008

Table 1 – Testing phase without hypervisor

Testing with hypervisor - 1

Run #	Test #1		Test #2	
	Exec. Time Thi	Deviation Dhi	Exec. TimeThi	Deviation Dhi
1	5.5553	0.0069	**5.6109**	**0.0328**
2	**5.6209**	**0.0312**	5.5591	0.0034
3	5.5507	0.0023	5.5641	0.0070
4	5.5492	0.0082	**5.5918**	**0.0544**
5	5.5651	0.0058	**5.6125**	**0.0255**
6	5.5584	0.0076	**5.5774**	**0.0224**
7	**5.7345**	**0.0190**	5.5662	0.0037
8	5.6233	0.0018	**5.5771**	**0.0277**
9	**5.6185**	**0.0337**	**5.7033**	**0.0471**
10	**5.6997**	**0.0423**	**5.5723**	**0.0151**
	AvTh1=5.6076	AvDh1=0.0159	AvTh2=5.5935	AvDh2=0.0239

Table 2. First two tests (each has ten runs) with hypervisor

Testing with hypervisor - 2

Run #	Test #3		Test #4	
	Exec. Time Thi	Deviation Dhi	Exec. Time Thi	Deviation Dhi
1	**5.6352**	**0.0283**	5.5614	0.0042
2	**5.6642**	**0.0649**	**5.5652**	**0.0200**
3	5.5637	0.0039	5.5527	0.0019
4	5.5438	0.0082	5.5607	0.0060
5	5.5660	0.0019	**5.5862**	**0.0424**
6	**5.5593**	**0.0189**	**5.5969**	**0.0288**
7	5.5572	0.0052	**5.6439**	**0.0420**
8	**5.7767**	**0.0153**	**5.7073**	**0.0459**
9	**5.5868**	**0.0399**	**5.6006**	**0.0255**
10	5.5516	0.0086	5.5452	0.0010
	AvTh3=5.6005	AvDh3=0.0195	AvTh4=5.5920	AvDh=0.0217

Table 3. Additional two test (each of ten runs) with hypervisor

Hypervisor Identification by Time Difference (TDI)

We highlighted in bold the runs when the deviation is >= 0.01. The deviation values fall in two big groups – less than 0.01 – 18 and more than 0.01 – 22. Statistically it is close to 50% each.

Therefore in "without hypervisor" test (Table 1) we have average time AvTc=5,5005 and average deviation AvDc=0.0008 (0.001)

Average execution time by four tests with hypervisor (Tables 2 and 3) – AvTh=5.5984

Time difference AvTh – AvTc=5.5984-5.5005=0.0979

Average deviation by four tests with hypervisor – AvDh=0.0202

The increase of execution time is 0.0979 and which is 1.8%

The maximum sum of average deviations is AvDc + AvDh = 0.0212

It is possible to identify the hypervisor by time difference method because the time difference 0.0979 is more than four times of maximum sum of average standard deviations: 0.0979 > 4x 0.0212

Hypervisor Identification by Deviation Increase (DDI)

Each of four tests show deviation values more than 0.01, and that is ten times higher than the deviation without hypervisor AvDc=0.001

- Test 1: there are 4 high deviation values (0.0312, 0.0190, 0.0337, 0.0423) and first high value is in the second run
- Test 2: there are 7 high deviation values (0.0328, 0.0544, 0.0255. 0.0224, 0.0277, 0.0471, 0.0151) and first high value is in first run
- Test 3: there are 5 high deviation values (0.0283, 0.0649. 0.0189, 0.0153, 0.0399) and the first high value is in the first run
- Test 4: there are 5 high deviation values (0.0283, 0.0649, 0.0189, 0.0153, 0.0399) and first high value is in the second run.

More likely that high deviation value will happen within 1 – 5 runs. The probability that next run will have high deviation value is 50%.

The probability that MH will be found in at least one run in 10-runs test is about 99.9%

Is There a Hypervisor – DDI testing Reporting

While we are not dismissing the value of TDI testing, *we currently consider DDI method to be superior, primarily because it does not require having "clean" system results to compare with. The test can be stopped when the first high time deviation is identified.*

We can consider 0.01 as a borderline value and any higher value means that a hypervisor is present:

- Values lower than 0.002 should be considered as no hypervisor
- Between 0.001 and 0.005 - there is some likelihood that hypervisor exists
- Between 0.005 and 0.01 that a hypervisor is possible

Fig. 1 below shows sample deviation distribution for DDI testing of 8 runs.

The following Fig. 2 and Fig. 3 screenshots are results of two tests running HyperCatcher v.1.0. Each test consisted of five runs. Yellow crosses show runs' execution time deviation values and indicate whether the hypervisor is present

Sample DDI Testing Distribution

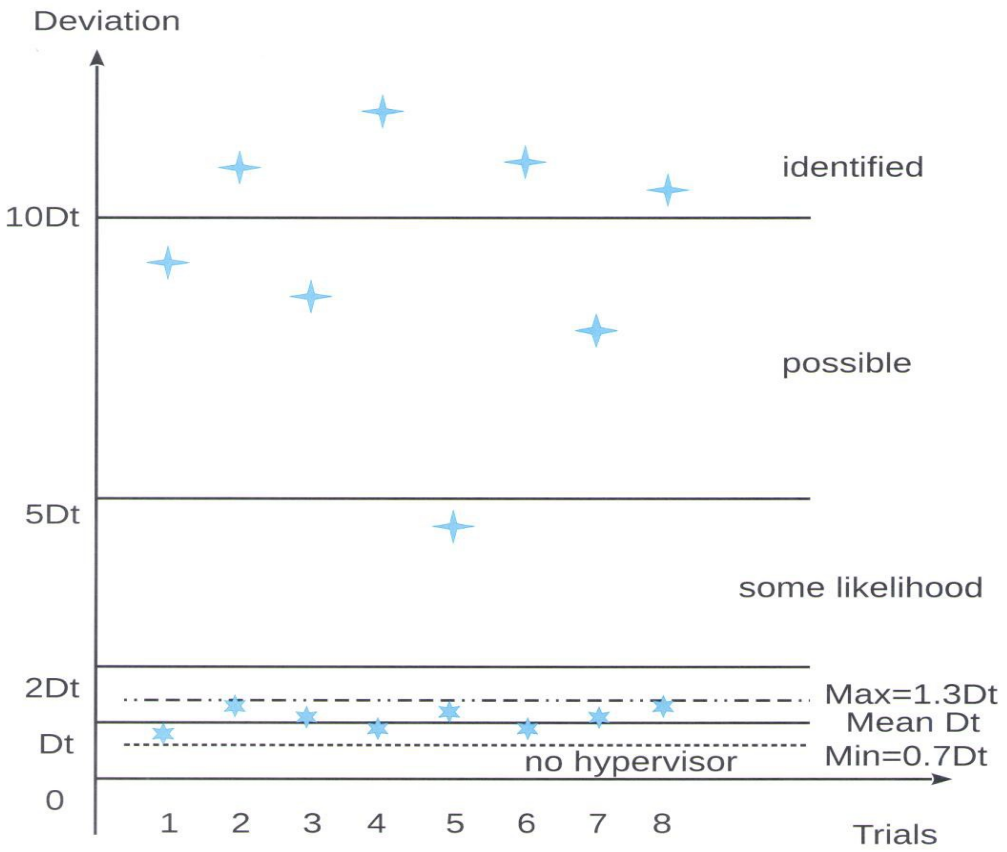


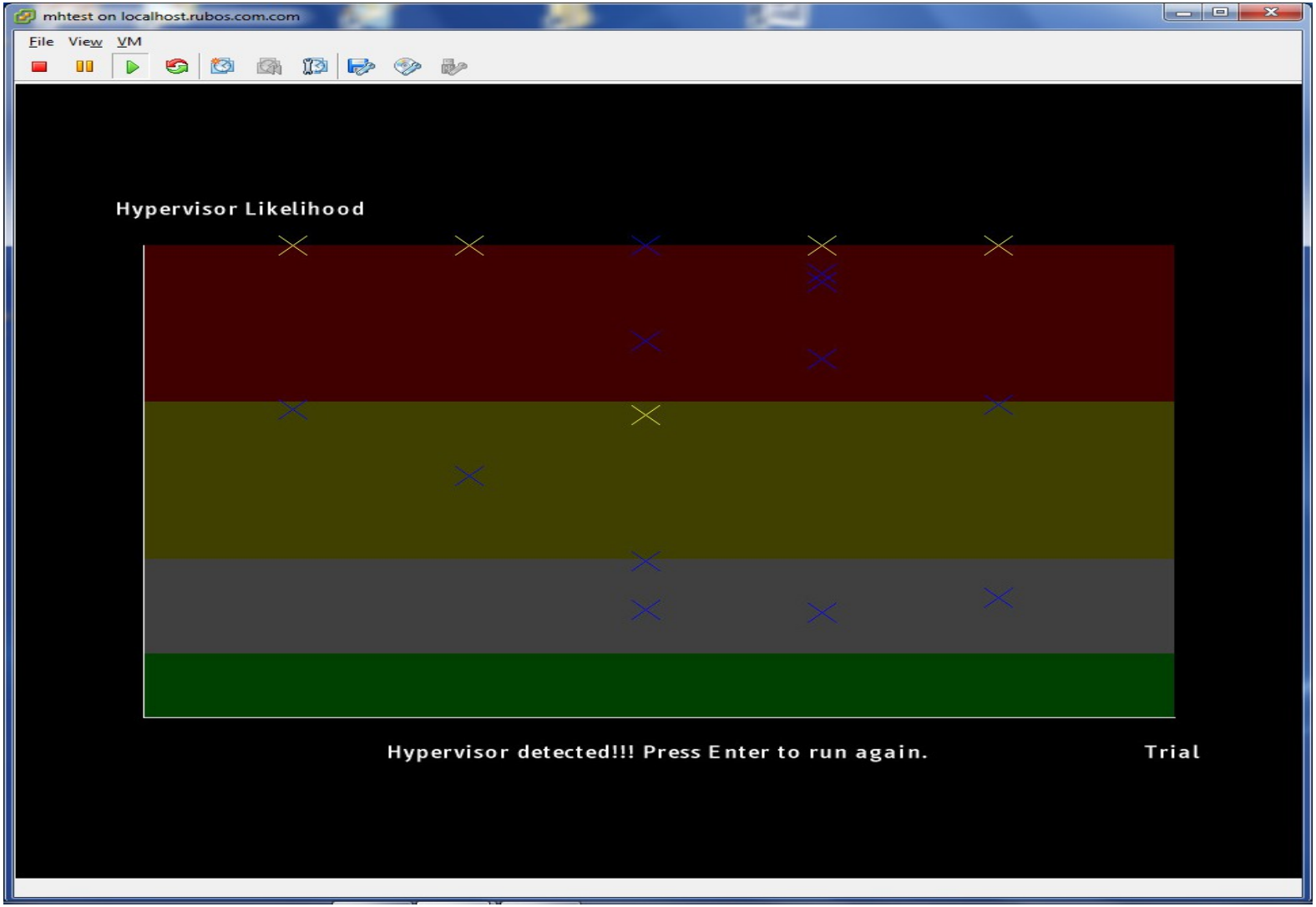
FIG. 1 Deviation distribution in 8 – trials test

- trials without hypervisor

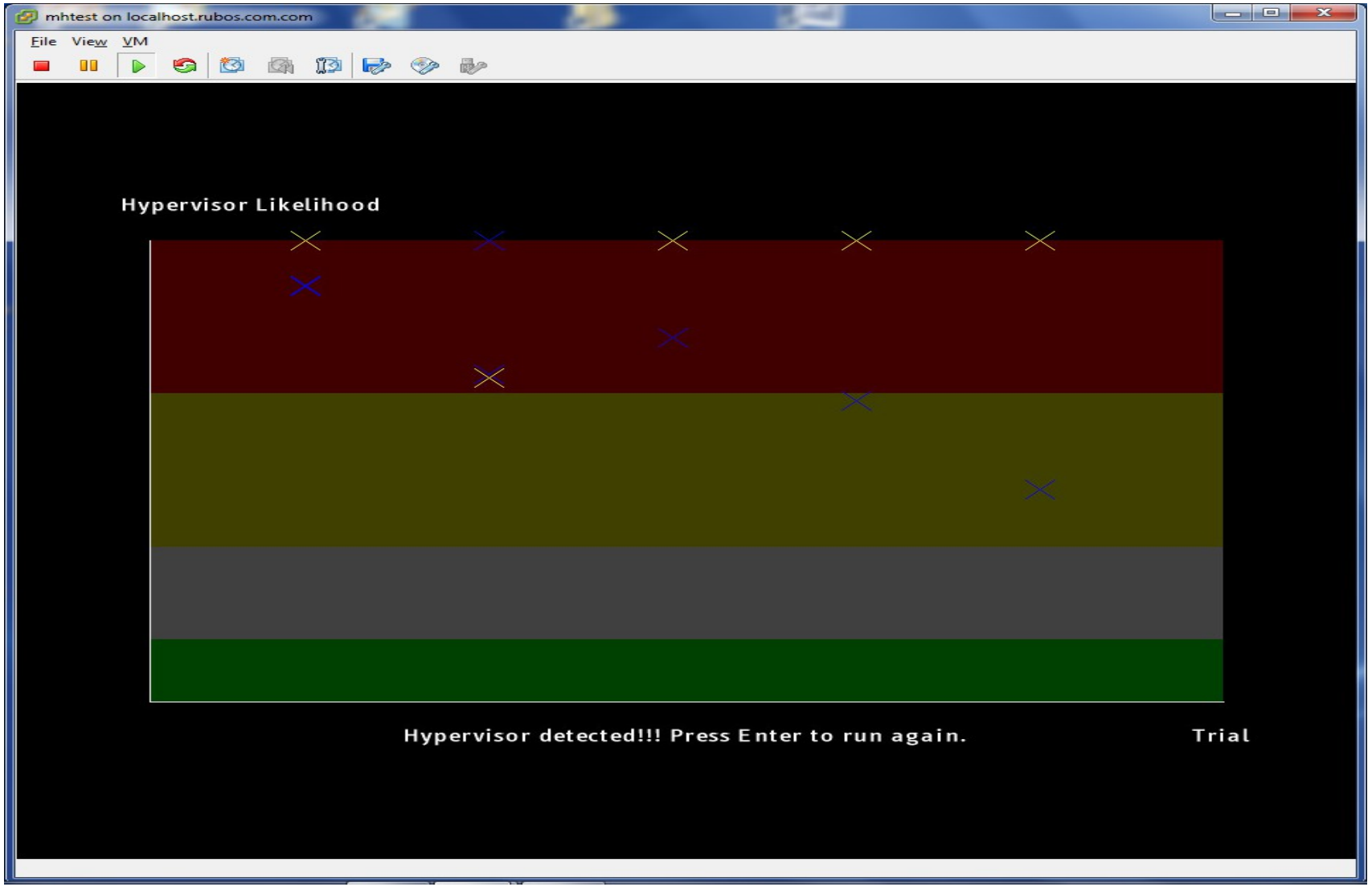
- trials with hypervisor

DEEPSEC

Sample DDI Testing Screenshot #1



Sample DDI Testing Screenshot #2



Phase 2 Research and Development Conclusion

Conclusion

1. The software which will very likely identify any hypervisor is developed. We have now the first layer of protection – detection of MH. It is still not a protection but we are much better now than before.
2. There are two methods of MH detection and each works efficiently and reliably. The HyperCatcher v.1.0 is first production version utilizing DDI method. The software does not require any specific skills to use – it identifies MH automatically.
3. We do not expect that major computing technology vendors will agree with "computing technology vulnerability" cases which we discussed. They may act securing some features but not implementing Security Development Life Cycle for each and all technologies affecting billions of users.
4. We hope to see more security research in detecting and protecting against MH threat. We think that security community may take a lead fixing the vulnerabilities we discussed.

References

1. Chinese Add-ons: True Stories of virtualization, information security and computer spying; post on http://xakep.ru/articles/58104/ 12/26/2011. Translated from Russian, Copyright © DeepSec, GmbH and Rubos, Inc., 2014.
2. SubVirt: Implementing malware with virtual machines. Samuel T. King, Peter M. Chen (University of Michigan); Yi-Min Wang, Chad Verbowski, Helen J. Wang, Jacob D. Lorch (Microsoft Research); IEEE Symposium on Security and Privacy, Berkley/Oakland, CA, USA, 21-24 May, 2006.
3. Illuminating the Security Issues Surrounding Lights-Out Server Management by Anthony J. Bonkoski, Russ Bielawski, J. Alex Halderman; Michigan University. 7Th USENIX Workshop on Offensive Technologies, August 13, 2013, Washington, DC. https://www.usenix.org/conference/woot13/workshop-program/presentation/bonkoski
4. Alert TA13-207A The Risks of Using the Intelligent Platform Management Interface (IPMIO), US CERT, July 26, 2013 https://www.us-cert.gov/ncas/alerts/TA13-207A

Thank you very much!
(I'll answer all face-to-face questions)

Rubos, Inc. Team
mikhailutin@hotmail.com
mutin@rubos.com