Virtualized rootkits - Part 1

Federico Biancuzzi,

There has been a lot of buzz around the topic of virtualized rootkits. Joanna Rutkowska has been working on a new version of Blue-Pill, her proof of concept invisible rootkit, while a team made by three prominent security experts (Thomas Ptacek, Nate Lawson, Peter Ferrie) challenged her that there is not an "invisible" rootkit, and that they were going to present at BlackHat conference various techniques to detect Blue-Pill. Federico Biancuzzi interviewed both sides to learn more. Part 1 of 2

PART 1: Joanna Rutkowska.

Could you introduce yourself?

Joanna Rutkowska: I'm a security researcher focusing on operating system security research. I'm mostly interested in stealth technology as used by rootkits and covert channels, OS isolation mechanisms and virtualization technology. A few months ago I started my own consulting company, Invisible Things Lab.

Did you see the talk given at Blackhat by Peter Ferrie, Nate Lawson, Thomas Ptacek? Your reaction?

Joanna Rutkowska: Of course I saw it! This presentation promised to definitely solve the problem of virtualization based malware. The authors went so far to announcing actually that "the virtualized rootkit is dead". No wonder that it was one of the most expected talks at Black Hat this year...

Unfortunately authors failed to prove their claims and all they presented was just a bunch of hacks of how to detect virtualization, but not virtualization based malware.

As hardware virtualization technology gets more and more widespread, many machines will be running with virtualization mode enabled, no matter whether "bluepilled" or not. In that case blue pill-like malware doesn't need to cheat that virtualization is not enabled, as it's actually expected that virtualization is being used for some legitimate purposes. In that case using a "blue pill detector", that in fact is just a generic virtualization detector is pointless.

During our presentation (I co-presented with Alex Tereshkin who works with me and who wrote most of the New Blue Pill's code) we have also showed that even the virtualization detection methods they (and some other researchers) presented were not reliable and needed some improvements. I discussed the problems with TLB profiling, that was one of the key methods used by "blue pill dead announcers". I explained how we need to take extra care (e.g. of avoiding collisions in data L1 cache). None of the researchers discussing TLB profiling methods before, touched this subject, which suggested that they never actually tested their methods on AMD processors.

Needless to say we have also published an improved version of TLB profiler, that was reliable and could be used for SVM mode detection. But again, this is still not a good blue pill detection approach, as it only tells you about virtualization mode being enabled, not about the actual malware. So, in any case that you're using some legitimate hardware virtualization application already, this detector is useless.

What make you believe that we can build a 100% invisible rootkit?

Joanna Rutkowska: The "100% undetectability" phrase applies to practical detection, as even last year we knew some methods that could be used for unexpected hypervisors detection. It passed a year and we still don't have any good method for virtualization malware detection and I don't believe we could have any without the help from hardware.

Let's say that building a 100% invisible system is possible. This means that anyone could use it to do legal (hidden auditing, honeypots, hids) or illegal things without being detected. If we are sure it's completely undetectable, would giving it away on the internet be ethically acceptable? That would mean giving away a sort of "invincible weapon", no?

Joanna Rutkowska: That would depend on the scenario. It will probably be possible to detect the presence of a hypervisor (using various tricks), so in case we would expect that there should not be a hypervisor present, like in most cases today, but much fewer situations in the future, then we would be able to detect that something is not ok.

Consider a honeypot for example. Today, if we built a honeypot using hardware virtualization, then, taking into account that very few servers in the wild are virtualized using VT-x or SVM technology, it might be possible for an attacker to find out that the server is suspicious. However, in the coming months we expect more and more servers to be virtualized using VT-x/SVM, in which case, both the normal servers and honeypots would be virtualized. The fact that an attacker detects the server is under the control of a hypervisor would not be useful, as most servers would be expected to be virtualized anyway.

Another trend is to use virtualization technology to increase desktop security, by isolating application from each other (e.g. user might have a dedicated VM for "unsafe" browsing, yet another machine for "IM-ing", etc). Again, in such a scenario, it's expected that the virtualization mode is enabled, so blue pill-like malware can simply sit above all those application (thanks to nested hypervisor support), control the OS, but doesn't need to bother to cheat about the virtualization being not used, as it's actually expected that the virtualization is being used. Again, detecting the presence of a virtualization mode, is pretty useless.

I guess we can say that virtualization is always detectable, so should the new focus be how we can distinguish a legitimate hypervisor from Blue Pill?

Joanna Rutkowska: Virtualization is always detectable, but we don't have any good, robust documented method to do that -- all we have (and will have) is a bunch of more or less complex hacks.

For example, I showed that the methods presented by Ptacek, Ferrie and Lawson during thier Black Hat talk, were either easy to defeat or were immature and unstable.

Distinguishing legitimate virtualization from malicious virtualization sounds logically as being the next step, but I somewhat don't see any good methods we could use to do that effectively.

I don't believe we can solve this problem without the help from hardware. But I still don't know how we should do it really.

From a security standpoint, do you see any difference between virtualization features included in AMD and Intel cpus?

Joanna Rutkowska: The AMD SVM technology is a bit richer then current Intel VT-x implementation, which also allows malware authors for a bit more freedom (e.g. they do not have to intercept CPUID instruction obligatory on SVM).

On the other hand AMD SVM contains several interesting features that might be used for better system protection (e.g. External Access Protection (EAP) and SKINIT instruction). But I know that similar technologies are to be introduced in the upcoming Intel processors as well.

All in all - both technologies seem to be similar from the security point of view (at least at the design level) and both are vulnerable to similar threats, like virtualization based malware.

You talked about Vista and x86 hardware. What about MacOS X, Linux or (Open)Solaris? Can we blue-pill them?

Joanna Rutkowska: I have always stressed that Blue Pill is OS-independent and Vista is just an example. Any OS working on AMD processors that support hardware virtualization is vulnerable to this type of rootkits.

What is your opinion on the security of software virtualization solutions (VMWare, Xen, VirtualPC, ...)?

Joanna Rutkowska: Obviously the most intriguing question here is whether it is possible to escape from the VM and "get into" the hypervisors/VMM... So far nobody found an exploitable bug inside anyy VMM engine as far as I know. There were only some bugs in some additional mechanisms (e.g. network NAT modules).

However, I believe, that in the coming years we will see some exploits in this area, as virtualization will be getting more and more popular.

I would still risk saying that probably the isolation capabilities provided even by today's software based VMMs, are much stronger then those provided by current general purposes operating systems like Linux, Windows or BSD.

What is the so called Blue Chicken strategy?

Joanna Rutkowska: It's a funny feature that allows Blue Pill to defeat timing-based virtualization detectors, so they can't find out that they're inside a VM. Obviously we do *not* need Blue Chicken in case there is Virtual PC in the system or any other application that makes use of hardware virtualization already.

Do you plan to constantly update the Blue Chicken strategy to catch all the different timing attempts?

Joanna Rutkowska: I don't think we're planning to work further on "virtualization cheating". We most likely would be working further on supporting various nested virtualization scenarios, where there is no need for "virtualization cheating" and where all the virtualization detectors are pointless by definition. Our goal here is to convince people that generic virtualization detection approach is not the right way to detect virtualization based malware.

Why do you need to support nested virtualization?

Joanna Rutkowska: Nested virtualization is needed in case we have some other applications in the target system that also want to make use of virtualization (e.g. Virtual PC 2007) or we have a system with built-in hypervisor. In both cases Blue Pill must run those applications and/or OS' own hypervisor as nested ones.

Please note the distinction between "virtualization applications" and "OS's own hypervisor". While it's believed that the latter will be effective in preventing Blue Pill attacks (by blocking other hypervisor installations), the former do not prevent Blue Pill from loading. It's expected that we will have many virtualization based applications in the future on our desktops, but I don't think that Microsoft will come up with their own global hypervisor for desktop systems anytime soon (next couple of years).

If Blue Pill is more invisible by hiding in the kernel (using Blue Chicken), why do you need to use virtualization?

Joanna Rutkowska: Blue Chicken forces Blue Pill into a "sleep mode", so, even though for a few tens of milliseconds Blue Pill does live in the kernel, it's there in a "sleep mode", which means it doesn't hook anything. It just sleeps there (possibly encrypted) waiting for a DPC callback to wake it up and bring back into ring -1.

VirtualPC does not allow nested virtualization so if Blue Pill is the hypervisor and Virtual PC (or any guest OS) finds nesting is allowed, isn't this clear evidence of Blue Pill?

Joanna Rutkowska: "Nesting" is not a processor feature that you can enable or disable - "nesting" is the way we support running other hypervisors underneath our own - think of it as of an algorithm to support nested hypervisors.

Virtual PC's guest can't find out that "nesting is allowed", just because it doesn't see anything beyond its own hypervisors (the Virtual PC hypervisor).

It's theoretically possible that Virtual PC hypervisor could find out that "nesting" is taking place... to find out that it's running as a guest and some people might argue that all those techniques that were presented to detect virtualization (e.g. those presented by Tom Ptacek &co.) could be used here. This is not true however, as all those techniques try to detect that somebody is cheating that virtualization is not used, while it actually is being used. In case of nested Virtual PC hypervisor, we do not need to (we should not actually) cheat that virtualization is not used, as it expected that it's actually being in use. Thus we don't need to intercept EFER accesses nor we have to intercept CPUID, so all those virtualization detection methods do not work.

What might work here however, is if we could build some instructions into the Virtual PC hypervisor, that would be behaving differently, just because the hypervisor would be running as somebody's guest. The GIF setting/clearing problem we discussed during our presentation is a good example of this.

One might ask then, that maybe there will always be situations like this, which would mean that we would never be able to have 100% nested hypervisor support. Most likely this is true, but, unfortunately, this is not a proper way to fight the Blue Pill threat, as it requires building some tricky hacks into hypervisors code. But the principle rule for building hypervisors is... simplicity (e.g. for security reasons).

Also, we could expect that those tricky hacks built into hypervisors would have to be updated from time to time to support new processor models. So would we allow our beloved A/V programs to insert modules into hypervisors? That would be insane - we already witnessed many bugs in A/V kernel components, should we now allow them to do that with hypervisors?