VMRay Technology Whitepaper

www.vmray.com

VMRay Technology Whitepaper

Hypervisor-based monitoring for malware analysis and threat detection

Executive summary

Behavior-based malware analysis has been an established approach for analyzing and detecting threats for over a decade, and is the core function of security solutions defined as “network sandboxing”. The established methods use either emulation or hooking and are showing their age. Each method has serious shortcomings that malware authors are adept at leveraging. While emulating an operating system is slow, complex and doesn’t scale, hooking requires instrumentation of the analysis environment – the resulting modifications can be easily detected by malware.

In this whitepaper we look at the technology behind VMRay – hypervisor-based system monitoring – and explore why this new approach overcomes the performance and detection issues associated with hooking and emulation. By moving monitoring to the hypervisor level, the target machines used for analysis can run completely unmodified – there is nothing for malware to detect. By combining this with VMRay’s innovative monitoring method, this approach delivers substantial performance and scalability advantages over other methods. Equally importantly it provides full visibility into threat behavior.

Network sandboxing – a primer

Sandboxing is a broadly used term that is often applied to desktop or browser sandboxes – secure execution spaces for safely running apps, opening docs or visiting websites. However, in this context we are concerned with network sandboxing – dedicated environments for controlled execution and analysis of files. Network sandboxes use automated behavior-based analysis. That is, the file is dynamically analyzed as it executes in something approximating a native desktop or server environment where the file would be expected to run. This is also referred to as ‘detonation’.

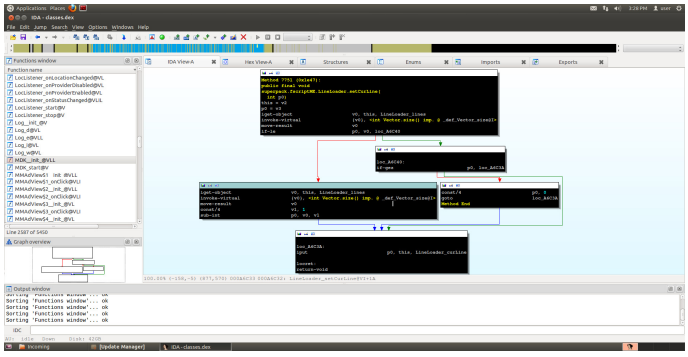

Figure 1 - Static analysis tool screenshot

By contrast, static analysis is where a suspect file is parsed and the file properties as well as individual segments and strings are examined. For automated detection purposes, such as comparing the hash of a suspect file against whitelists and blacklists, static analysis is highly scalable. However, malware authors easily evade this by using encryption, polymorphism, and code obfuscation. Statically dissecting such files can’t be automated and manual analysis is extremely time-consuming.

While most network sandboxes also contain some mix of static analysis components, their power mostly stems from the dynamic behavior-based analysis. Only dynamic analysis is capable of automatically handling encrypted and obfuscated zero-day malware that is used in targeted attacks including APTs.

During dynamic analysis, a suspicious file is executed in the analysis environment and the system monitors its activity and builds a report. Usually some post-processing steps are performed afterwards on the reports, e.g., filtering may be necessary to reduce 'noise' from the normal activity of concurrently running benign applications and highlight suspect behavior indicative of malware. Furthermore, some kind of scoring is applied against the results to autonomously decide whether the file was malicious or not.

The goal is to get an accurate profile of what the malware would be doing if it were executing on its intended victim machines 'in the wild'. Effective dynamic analysis needs to be:

- Scalable – No matter the approach, executing each sample in different environments can take time, typically a couple minutes for hooking, or substantially longer for emulation. The ability to quickly spin up multiple environments and analyzing in parallel is key.
- Detection and evasion resistant – Anti-analysis techniques are pervasive in malware. If no malicious activity is shown during the analysis, threats can be missed.
- Flexible – targeted attacks will look for particular combinations of OS, applications, localization and geo location before executing maliciously. Otherwise they will often simply go to sleep, negating any effective analysis.

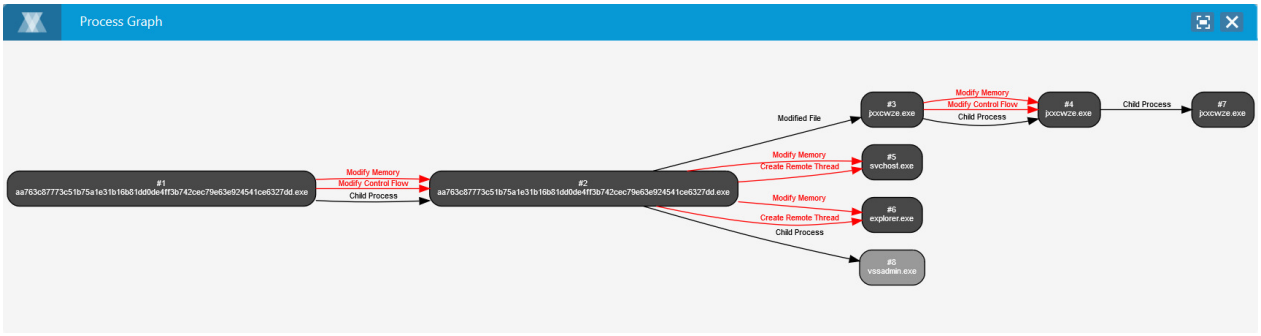

Figure 2 - Behavior-based analysis generates information about what happened during execution of the malware, such as this process graph

Virtually all existing network sandboxes take one of two approaches to dynamic analysis:

Emulation-based analysis

Available commercially for over a decade, products using this approach either emulate the operating system (OS emulation), or the underlying hardware (system emulation). OS emulation is trivially simple to evade and is usually not used as a standalone solution anymore. In the case of system emulation, the complete hardware system is emulated that the OS needs to run on – primarily of course the CPU and memory. The biggest disadvantage of this approach is its level of computing complexity, which results in very poor runtime performance. In order to speed up the analysis, practical real-world systems have to take a lot of shortcuts and simplifications. This leads to reduced visibility into the malware's behavior and still results in systems that are far slower than other non-emulated ones. Furthermore, with the complexity of today's CPUs, there are inevitably semantic and timing differences between the emulated and the native system. These imperfections often lead to detectability by malware.

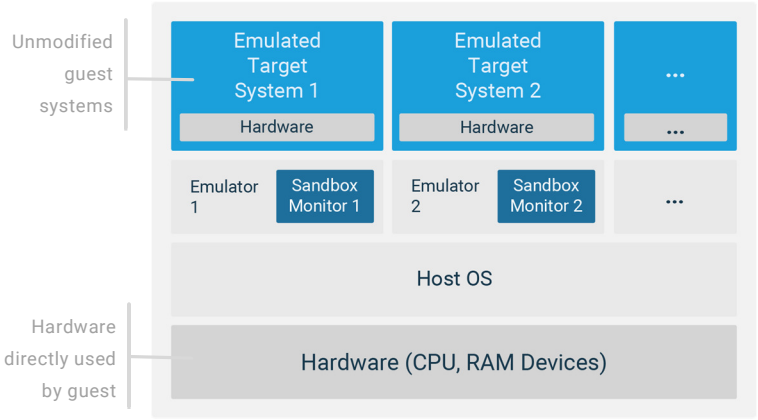

Figure 3 - In full system emulation, the CPU instruction set and memory of one CPU type can be emulated while residing in a native OS on another type of (physical) CPU

Hooking-based analysis

To overcome the performance issues of emulation a different approach for sandboxing has been developed that is prevalent today. The idea is not to emulate a system, but instead use a real one to execute the suspicious file inside it. In order to monitor its activity and changes made, the underlying systems need to be instrumented. The most common method of such instrumentation is called hooking and it works by intercepting certain function calls through a hook function. Usually it isn't the complete system that is hooked, but only a small subset of the security related APIs and system

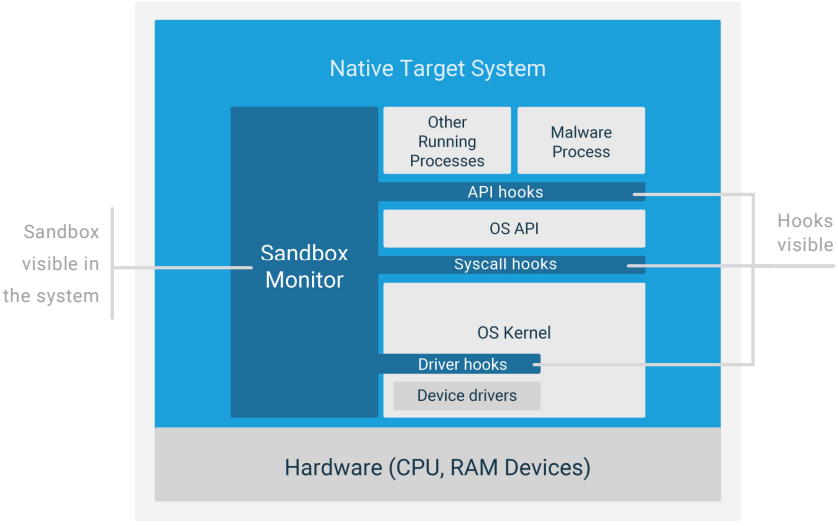

Figure 4 - Possible hooking locations in Windows kernel

calls. The reason for this limitation is that the more function calls are intercepted, the slower the system gets.

With hooking-based analysis, detection is trivial in most cases as the modifications have to be placed in memory that is (directly or indirectly) accessible by the executed malware.

Avoiding detection and evasion by malware becomes a race to the bottom but within the OS there is only so low you can go. No matter the steps taken to obfuscate, the target environment needs to be modified to gain visibility and a footprint is left that can be detected.

Hooking can be implemented on bare metal systems but for ease of use, scale and isolation virtually all commercial solutions implement hooking in virtualized target machines (VMs).

Down to the hypervisor

Emulation fails to scale. Hooking is inevitably detectable. In the quest to have scalable, automated dynamic analysis of malware in a way that can't be detected or evaded, we're left with one logical place to go – the hypervisor.

The hypervisor is the underlying computing platform that creates, runs, and manages virtual machines on the underlying hardware. Most sandboxing solutions use a hypervisor as a launch pad for either the emulators or virtual machines that are hooked and monitored.

At VMRay we took a different approach – monitor the activity of the target machine entirely from the outside – by using Virtual Machine Introspection (VMI). VMRay combines virtualization extensions with an innovative monitoring concept called Intermodular Transition Monitoring (ITM) to deliver agentless monitoring of VMs running native OS without emulation or hooking.

VMRay runs as part of the hypervisor on top of the host OS which in turn is running on bare metal.

Each component is independent – so theoretically VMRay can run on different hypervisor and hardware combinations.

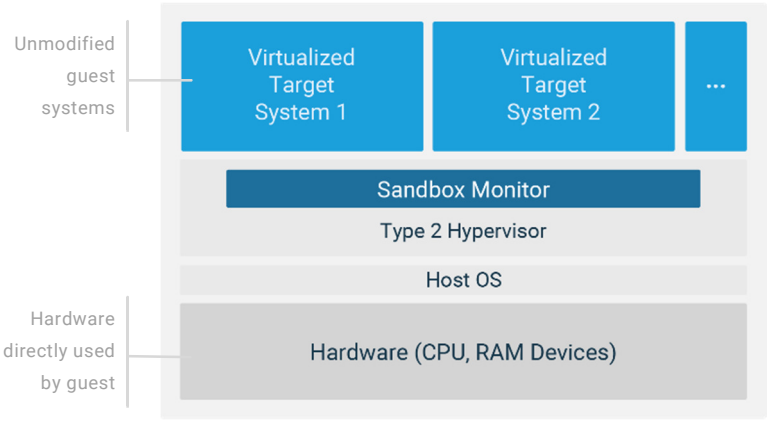

Figure 5 - VMRay runs on a type 2 hypervisor - hypervisor + OS + hardware kernel

See everything – touch nothing

Most CPUs today support MMU (Memory Management Unit) virtualization in hardware and provide Second Level Address Translation (SLAT) to greatly speed up memory access in the guest.

VMRay leverages this technology to:

1. Transparently monitor all interaction between the monitored malware and any other part of the system
2. Monitor entirely externally to the analysis environment without the need to modify a single bit in the analysis environment
3. Rarely interrupt the malware's execution and far less frequently than existing sandboxing technology approaches –increasing performance and scalability

To get a comprehensive result when monitoring malware execution, the analysis system must capture:

1. All **control flow mechanisms**
   - *All forms of calls to APIs and private functions of the OS; direct system calls; interrupt handler routines, asynchronous and deferred procedure calls and more.*
2. All **calling conventions**
   - *Regular function calls; shellcode execution on the heap or stack; unaligned function calls in which the malware skips the first instructions of an API to circumvent potential hooks; invocation of COM object methods and more.*
3. All **privilege levels**
   - *Ring 3 (userland) and ring 0 (kernelland) code.*

Unlike other approaches, VMRay uses adaptive monitoring to automatically adjust to the optimum monitoring granularity. That means regardless of whether the malware is doing an API call, using special CPU instructions to directly jump into the kernel, or using higher-level concepts such as COM objects, VMRay always intercepts at the highest semantic level possible. No semantic information is lost.

Furthermore, VMRay automatically tracks propagation of analyzed, dropped, or downloaded code, survives reboots, and only monitors those parts of the system that are actually related to the analysis. This make it unnecessary to do filtering on the analysis output as no side-effects of benign applications are ever monitored (for example, Word startup, browser code, background services).

As a consequence, the resulting reports always have the highest possible relevant information density.

Conclusion

Effective dynamic analysis and threat detection depends on successfully addressing:

- **Scalability:** An effective analysis approach is rendered useless in practice if it cannot be automated and scaled up to handle the quantity of unique threats organizations must deal with.

- **Evasion resistance:** Traditional security solutions engage in an endless and ultimately unwinnable arms race with malware authors. As malware adopts one anti-analysis approach to the point of pervasiveness, vendors respond, but then must adapt themselves to the next anti-analysis technique. There's no positive outcome for technologies that modify the target analysis environment and are therefore inherently detectable.

- **Full visibility:** The trade-off for scalability is often incomplete or inaccurate results. Yet threat intelligence is only actionable effectively if the results can be relied upon to be complete and accurate.

- **OS, application and hardware independence:** Each application an enterprise runs, each file type that can execute, represents an additional threat vector. A necessary feature for an effective sandboxing solution is the ability to fully customize the target machines.

- **Automated scoring:** An accurate, complete analysis can overwhelm the most skilled and efficient incident response professional if the system doesn't have the ability to filter out the results of normal, non-malicious activity from the truly malicious. Further, the system needs to provide an interpretation of the degree of risk associated with each identified malicious behavior. This interpretation, or score, is a pre-requisite for turning the analysis into actionable intelligence that can be used for automated blocking and remediation.

Traditional sandboxing technologies (emulation, hooking) all make compromises on some of these criteria in order to address others. There is always a trade-off between comprehensiveness and sophistication of the result and scalability, while sacrificing invisibility. Only VMRay's 3rd generation approach of moving to the hypervisor and embedding all monitoring at that layer can successfully analyze and detect at scale while evading anti-analysis from malware. VMRay Analyzer is successfully deployed in the most demanding analysis environments at some of the largest enterprises and technology vendors globally, proving in the real world the validity of the VMRay approach.

Threat Intel

Actionable Information

Incident Response

Digital Forensics (DFIR)

OEM

Cloud and Appliance
Security Solutions

Protection

Seamless Integration
with EDR, NGFW

About VMRay

VMRay delivers 3rd generation threat analysis and detection using advanced hypervisor-based dynamic analysis. The VMRay analyzer is platform independent and highly scalable. By monitoring at the hypervisor level, it is undetectable by malware running in the target operating system.

Based in Bochum, Germany VMRay works through channel partners and OEMs to deliver our solution to leading enterprises around the world.

VMRAY

3rd Generation Threat Detection