

Anti-Sandbox Evasion

Why Defeating Anti-Sandbox Evasion Checks is Critical for Successful Sandbox Automation



Table of Contents

Introduction	5
Circumventing Enterprise AV Security Controls	5
The Different Types of Sandbox Technology	6
Circumventing Sandbox Detection – Anti-Sandbox Evasion	8
Common Detection Challenges	8
Not All Sandboxes Are Created Equal	9
Anti-Sandbox Evasion Checks and How to Resist Them	9
Exploiting Sandbox “Detection” Checks	9
Detecting Virtualization/Hypervisor	10
Detecting Sandbox Artifacts	10
Using Vendor-Specific Knowledge	10
Old Vs. New Sandboxing Technologies for Detection	11
Defeating Sandbox “Detection” Checks	12
Attacking Sandbox “Technology” Weaknesses	12
Blinding the Monitor	13
File Size Bloating	14
Reserved Characters in Filename	14
Defeating Sandbox “Technology” Weaknesses	14
Exploiting “Context-Aware” Evasion Techniques	15
Time Bombs	15
System Events	15
User Interaction	16
Fake Installers	16
Office Documents with Malicious Embedded Content	16
Detect a Specific Target System	16
Simple Checks Including String Checks	16
Defeating “Context-Aware” Evasion Techniques	16
Conclusion: The Last Line of Defense	17
About VMRay	18
Ready for the next step?	18
Portfolio	18

Introduction

Over the last ten years, malware sandboxes have become critical in the defense of Enterprise networks by identifying zero-day malware and phishing threats that bypass other security controls. Perimeter and desktop Anti-Virus solutions typically use a combination of reputational, static, and heuristic analysis which malware authors have learned over time to disable and bypass.

Considered the last line of defense to stop new malware threats, malware sandbox technology has evolved from an obscure research tool to become a critical part of the Enterprise security stack. To counter the threat that sandboxes pose in stopping the proliferation of malware in its tracks, the authors of advanced modern malware families have engineered pre-payload deployment checks to assess if the malware is being detonated in a monitored sandbox environment or not.

Known as Anti-Sandbox evasion checks, malware implementing these techniques can stop or stall payload deployment in kernel-mode and hooking-based sandboxes because the malware can identify specific indicators within the detonation environment. With hundreds of different environmental checks, advanced modern malware can evade a sandbox if a predefined list of checks are not met. This makes the submitted file or URL appear suspicious or benign, requiring manual triage by a Tier 3 Security Operations Center (SOC) Analyst to validate, often taking hours or days to investigate.

Circumventing Enterprise AV Security Controls

A major cause of corporate IT disruption stems from advanced unknown malware, phishing, and custom crafted attacks. Advanced modern malware is decidedly different from those detected by traditional signature-based anti-malware solutions as the malware is engineered to avoid detection using sophisticated evasion techniques such as obfuscation using packing to alter the file hash, or string encryption within the binary files themselves. Once past the detection engine, the malware can “unhook” the APIs used by the AV engine and effectively disable it. Even with the improvements in heuristic analysis and phishing detection, malware and phishing attacks continue to be the most successful attack vector to infiltrate the enterprise.

Static detection signatures filter out known threats and heuristic engines go some way in flagging known malicious artifacts in previously unknown malware. However, this detection process is not foolproof even with hourly updates, as bad actors continually change malware to make it unidentifiable to existing detection methodologies.

Malware is becoming more evasive, avoiding detection by perimeter and endpoint security controls.



Sandboxes expose the inner workings of malware, and identify behavior that can mitigate the threat



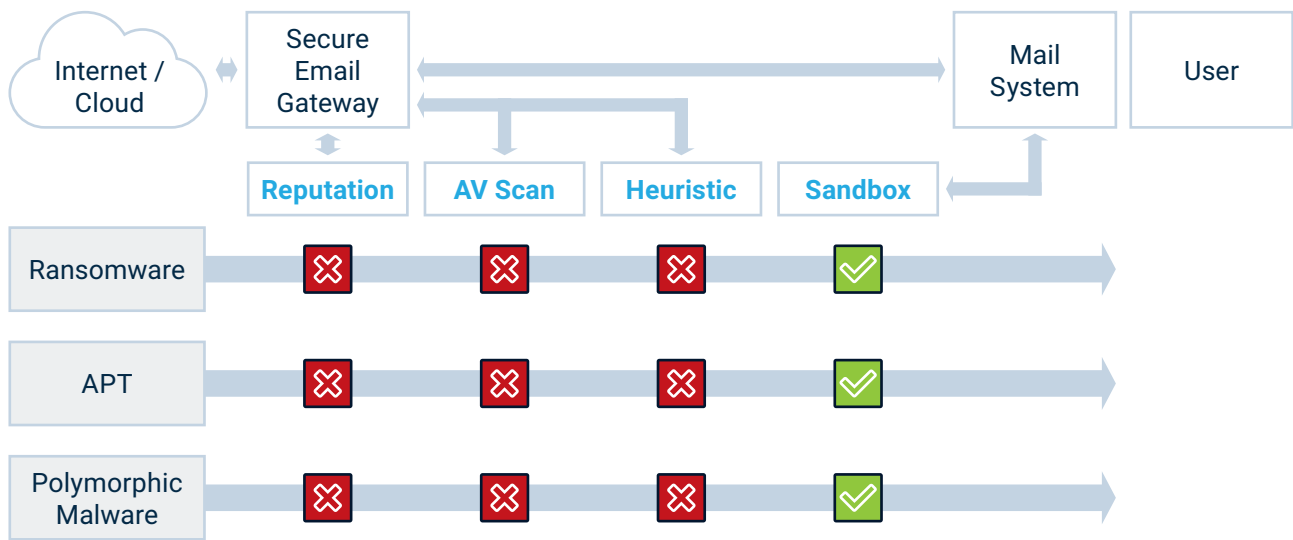


Figure 1. Evasive malware and phishing attacks can only be detected using sandbox technology.

Currently, the only way for an organization to accurately identify unknown malware is to detonate the payload in a safe, controlled environment like a malware Sandbox, analyze the results, and extract the indicators of compromise (IOCs) and other artifacts. The extracted IOCs can then be used to create signatures for future detection and update mitigating controls such as firewalls and intrusion detection systems. While AV and Anti-Malware evasion is slightly different from Anti-Sandbox evasion, the goal remains the same – to avoid detection.

The Different Types of Sandbox Technology

There are significant differences in the architecture and implementation of monitoring technology which make some sandboxes better than others. The architecture can affect the speed of analysis, requirements for scalability, performance for full automation, and importantly, the many ways modern malware families can evade sandbox detection. All of these factors should play a critical role in the decision-making process and must be considered by the SOC when implementing a sandbox technology-based solution. The three main approaches used in sandbox architectures today include outdated emulation, older hooking or “kernel-mode” analysis, and today’s superior hypervisor-based analysis

Traditional sandbox techniques such as emulation or hooking-based analysis do not work well when detecting evasive threats. This is due to tools or variables within the detonation environment that are detectable by the malware. To counter this, the only way to ensure malware detonation is to provide an environment free of any indicators that would inform the malware that it is potentially being monitored.

Hooking and kernel-mode sandboxes have **detectable instrumentation** within the monitoring environment.



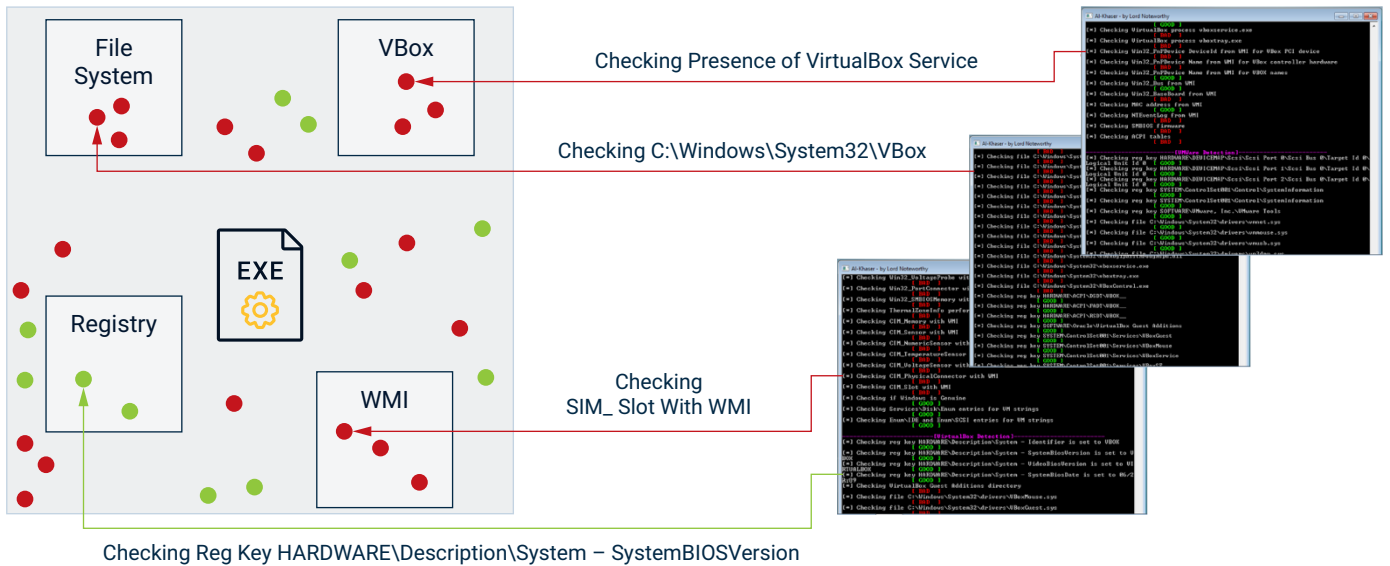


Figure 2. Hooking and Kernel-mode sandboxes have detectable artifacts, causing malware to "sleep".

The hypervisor is the underlying computing platform that creates, runs, and manages virtual machines on bare-metal hardware. VMRay solutions run their monitoring technology in the hypervisor, providing visibility from outside the workload by using Virtual Machine Introspection (VMI). By running in the hypervisor and not the detonation environment, malware is unable to identify any indicators that would signal a monitoring environment, fooling the malware into payload detonation.

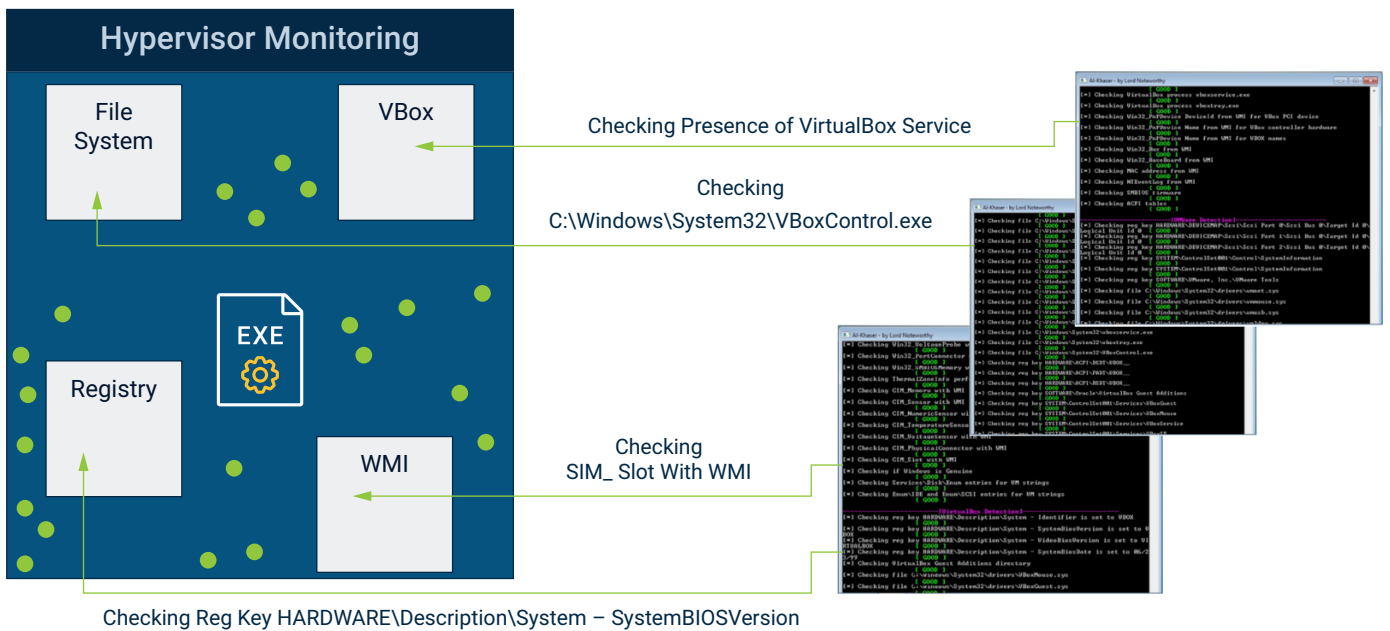


Figure 3. Hypervisor-based sandboxes monitor outside the detonation environment, with no detectable artifacts.

VMRay's Intelligent Monitoring provides untainted visibility into the malware or phishing sample's payload actions during and after detonation. The collected observations of sample behavior are then passed to the automated analysis process for extensive evaluation using 30+ different analysis technologies.

Circumventing Sandbox Detection – Anti-Sandbox Evasion





Unfortunately for malware authors, sandboxes reveal the behavior and inner workings of malware and phishing attacks. The knowledge of how malware works can ultimately be used to defeat it. Obviously, malware authors want their code to propagate far and wide and infect as many systems as possible, compromising sensitive data or encrypting disks to ransom their victims. Malware sandboxes are key to identifying the Indicators of Compromise (IOCs) needed to block propagation and shorten the malware lifecycle considerably.

Malware authors began to write environment checks into their code to identify some sandboxes prior to payload detonation, making their malware “sandbox aware”. Raccoon v2, Emotet, TrickBot, and BumbleBee are just four malware families engineered to run Anti-Sandbox evasion checks to determine whether the malware is being run (or detonated) in a monitored environment. In most cases, commercial malware sandboxes are run as virtual machine. Malware authors have access to over 200+ environmental checks they can use to determine if a system environment is either a sandbox or a virtual machine. Primarily, Anti-Sandbox evasion can be broken down into three basic categories: Detection, Attack, and Context-Aware.

Common Detection Challenges

The most dangerous malware is not only based on the maliciousness of the payload, but the intelligence it uses to hide and evade detection. Modern malware families are designed to bypass known security controls within email systems, secure email gateways, content inspection on firewalls, and IDS/IPS detection. Advanced malware requires certain conditions prior to activation and payload deployment. If these conditions are not met, the malware lays dormant until all the predefined conditions are in place.

The detection of a sandbox environment by the malware will suspend any activity, denying the sandbox the ability to identify malicious code, including:

 Delays the connection to Command & Control Servers	 Delays the download of additional code
 Delays the modification or injection of code	 Delays any network-based propagation actions

In addition, some evasive malware will also require additional conditions prior to activation, such as user interaction, hibernation, geolocation, and/or username. Advanced sandboxes – such as VMRay’s – can simulate human behavior using AI/ML, which is highly effective in deceiving malware to detonate where other sandbox architectures would fail.

On detecting a monitored environment, the malware lays dormant and miscategorized as benign.



On average, modern malware families use 5-6 evasion checks to identify sandbox or VM environments.



Not All Sandboxes Are Created Equal

Malware that is “sandbox aware” can lead to stalled analysis, partial detonations, and inconsistent threat scores in some kernel-mode or hooking-based sandboxes. These failures are often due to multiple detectable exposures in the sandbox monitoring environment where the malware decides not to detonate – making a malicious file appear benign. The lack of detonation leaves SOC Analysts with a false sense of security and puts an organization at risk of an outbreak.

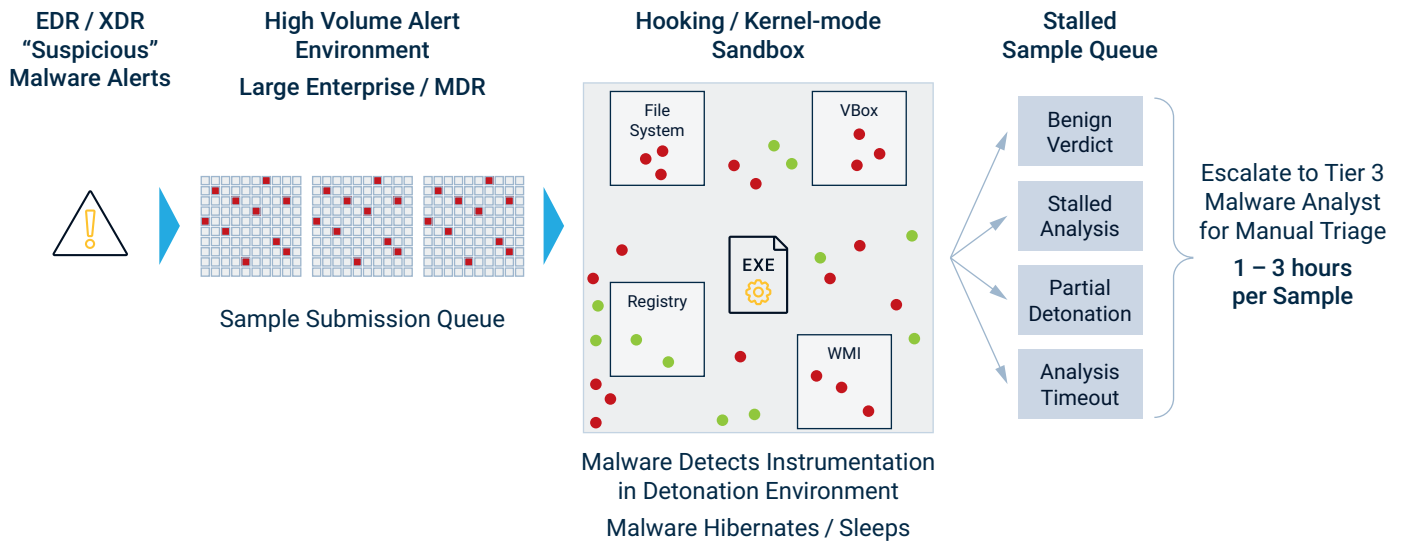


Figure 4. Hooking and kernel-mode sandboxes can stall sample submission queues and require more manual triage.

When evaluating a sandbox technology for full automation in the SOC, Anti-Sandbox evasion resistance is perhaps one of most important aspects to consider. By minimizing the chances of evasion check failures by monitoring from the hypervisor, there are no submission queue stalls, no malicious samples miscategorized as benign, and a significantly reduced need for Tier 3 manual triage if the sandbox is ever evaded. Clearly, this is not the case when utilizing hooking or kernel-mode sandboxes.

Anti-Sandbox Evasion Checks and How to Resist Them

Malware authors are always looking for new, innovative ways to evade sandbox detection by concealing the real behavior of the malware. Other than detecting a sandbox environment, malware can also attack or exploit shortcomings in the sandboxes attempts to automate the analysis.

Exploiting Sandbox “Detection” Checks

As previously mentioned, some sandbox environments such as hooking- or kernel-mode sandboxes come with indicators in their detonation environment of that are un-maskable, such as memory or system file artifacts. Going more in-depth from a technical perspective, the following checks are just some examples of how malware can evade sandbox analysis to appear benign.

VMRay uses best-in-class Hypervisor-based sandbox technology, hardened against Anti-Sandbox evasion techniques.



Detecting Virtualization/Hypervisor

This is one of the oldest evasion techniques. However, it is less relevant today as many production environments (workstations and servers) are virtualized anyway and virtual machines (VMs) are no longer only used just by researchers and malware analysts. The earliest approach detected technical artifacts that existed due to the lack of full hardware support for virtualization (Paravirtualization).

These techniques include:

- ♦ Detecting artifacts of popular VM hypervisors
- ♦ Detecting generic hypervisor artifacts

These artifact detection techniques are not very effective today. With hardware virtualization support, there are very few visible artifacts (if any) inside the VM since most hardware aspects are now virtualized and handled by the CPU itself. Therefore, they do not have to be simulated by the hypervisor.

One approach that is still relevant today is detecting the implementation artifacts of the hypervisor. For example, VMWare ("port 0x5658") or VirtualBox via a backdoor ("invalid opcode"). Another approach is to detect the presence of a VM by looking at registry values. In one example, the malware queried the registry key "HKEY_LOCAL_MACHINE\SOFTWARE\" to look for values associated with common VM implementations like VMWare.

Detecting Sandbox Artifacts

In this approach, it is not the hypervisor that the malware is trying to detect, but the sandbox itself. This can be done either by two of the following techniques:

Using Vendor-Specific Knowledge

Common VM Products: For example, the existence of certain files, processes, drivers, file system structure, Windows ID, or username.

The Ecosystem: For example, mechanisms to revert the analysis environment back to a clean state after infection (Deepfreeze, Reborn Cards, etc.). In addition, performing communication with the sandbox controller by adding additional listening ports and detecting the specific network environment.

With VMRay, no more stalled analysis, partial detonations, or analysis timeouts that inhibit full automation.



Old Vs. New Sandboxing Technologies for Detection

Most older sandboxes use hooks by injecting or modifying code and data within the analysis system. The 'hook' is essentially a shim layer capturing the communication between processes, drivers, and the OS. A hook can be implemented in many ways such as inline hooks, IAT, EAT, proxy DLL, or filter drivers. This makes them detectable by explicitly inspecting certain instructions or pointers or verifying the integrity of the system such as hash signatures of relevant system files.

Other malware sandboxes use emulation, which comes with side-effects and small differences compared to a native system. This includes different instruction semantics and cache-based attacks. Emulation gaps can be detected by invoking an obscure CPU instruction that was not included in the emulation. When the call fails, malware will know it is running in an emulated environment.

An example of vendor-specific detection is where the malware looks for the presence of the module "SbieDll.dll" – an indicator that it would be running in under Sandboxie, a common sandboxing environment.

Detecting An Artificial Environment

Sandboxes are usually not production systems but specifically set up for malware analysis. Hence, they are not identical to real computer systems and these differences can be detected by malware.

Differences may include:

- Hardware / Software / System / User properties
- Unusually small screen resolution, no USB 3.0 drivers, lack of 3D rendering capabilities, only one (V)CPU, small HDD size and memory size
- Atypical software stack, with no IM client or mail client
- System uptime ("system was restarted 10 seconds ago"), network traffic ("system uptime is days, but only a few MB have been transmitted over the network"), no printer or only default printers installed
- Clean desktop, clean filesystem, no cookies, no recent files, no user files

For example, in addition to checking for VM presence, some malware will also look for the presence of Wine, a software emulator. The malware does this by executing a query, GET_PROC_ADDRESS and attempts to determine from the returned result something that is expected in a Wine environment.

Timing Based Detection

Monitoring the behavior of an application comes with a timing penalty, which can be measured by malware to detect the presence of a sandbox by checking for RDTSC, the time-stamp counter. Sandboxes try to prevent this by faking the time. However, some malware can bypass this by incorporating external time sources such as NTP.

Faster sample detonation aids automation with detailed reporting to speed investigations.



Defeating Sandbox “Detection” Checks

In order to evade these types of detection by malware, an analysis environment should:

- Not rely on modifying the target environment
- Show the presence of a hook. It is virtually impossible to completely hide the presence of a hook.
- Either implement full system emulation perfectly or not at all
- Just as all software has bugs, it’s a near certainty that any given emulation environment will have flaws that can be detected and profiled

Use a Target Analysis Environment That is ‘Real’

If the malware sandbox can run an image copied from actual production endpoints, then the risk of detection falls dramatically. As we wrote earlier in this post, coupling that with randomization of the environment helps to ensure that there are no tell-tale signs for malware to identify the target environment as ‘fake’.

VMRay’s technology ensures that there is a minimal attack surface for malware to detect it is running in a sandbox. By not modifying the target environment, not relying on emulation, and allowing real-world images to run as the target environment, VMRay gives nothing for malware to flag it as a sandbox environment.

Attacking Sandbox “Technology” Weaknesses

Malware authors often seek to bypass sandbox analysis by directly attacking and exploiting weaknesses in the underlying technology or in the surrounding ecosystem. For example, we have seen in the past a large volume of malware samples using Microsoft COM internally because most sandboxes cannot correctly analyze these files.

Other malware will use obscure file formats that cannot be handled by the sandbox, or they exploit the sandbox’s inability to process files that exceed a certain size. A good example of this is OneNote file format from Microsoft that many sandboxes do not currently support. After Microsoft disabled macros by default in Word and Excel Office documents, threat actors began turning to the embedded OneNote file format to bypass AV detection as well as sandbox analysis.

In some cases, malware can be seen ‘blinding the monitor’ by performing illegitimate API usage. API-hammering is just one technique used where the file being analyzed makes thousands of Windows API calls, putting the sandbox under extreme duress to degrade the analysis performance in a form of extended “sleep”. This can be an effective method to hide from malware sandboxes that rely on a hook or driver injected into the target machine. However, since VMRay does not use hooking, these evasion attempts are easily detected.

Explicitly searching for the existence of a sandbox can be detected as a suspicious activity during analysis. A more advanced approach for malware, therefore, exploits weaknesses in the sandbox technology itself to perform operations without being detected. By exploiting these sandbox weaknesses, malware does not have to worry about being detected even if it is being executed in a sandbox system.

Modern malware families can **identify sandbox and virtual environments** and adapt their behavior to look benign.



Some of the techniques include:

Blinding the Monitor

Most sandboxes do in-guest-monitoring, (i.e., they place code, processes, and/or hooks) inside the analysis environments. If these modifications are undone or circumvented, the sandbox is blinded – in other words, visibility into the analyzed environment is lost. This blinding can take the following forms:

Hook Removal

Hooks can be removed by restoring the original instruction or data.

Hook Circumvention

Hooks can be circumvented by using direct system calls instead of API calls, such as calling private functions (which are not hooked) or performing unaligned function calls (skipping the “hook code”). While hooks could solve this problem for these particular internal functions, there are many of these present in the operating system and they vary with each Windows version. Furthermore, the problem of unaligned function calls cannot be adequately solved by hooking.

System File Replacement

Hooks usually reside in the system files that are mapped into memory. Some malware families will un-map those files and reload them. The newly loaded file version is then “unhooked”.

Kernel Code

Many sandboxes are not capable of monitoring kernel code or the boot process of a system.

Obscure file formats

Many sandboxes do not support all file formats. OneNote, PowerShell, .hta, and .dzip are some examples of file formats that may slip by and simply fail to execute in a sandbox environment.

Many Sandboxes Do Not Support All Technologies

While the initial infection vector (for example, a Word document with a macro) may open and the macro run in the sandbox, the macro will download and run a payload that uses an obscure technology hidden from the analysis. COM, Ruby, ActiveX, JAVA are just some examples.

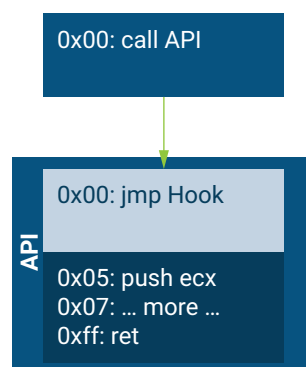
Operating System Reboots

Many sandboxes cannot survive a reboot. Some systems try to emulate a reboot by re-logging in the user. This can be detected by the malware when not all triggers of a reboot are executed.

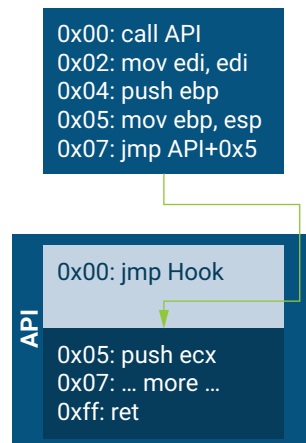
Blinding the Ecosystem

By simply overwhelming the target analysis environment, malware can also avoid analysis with this crude but sometimes effective approach. For example, some sandboxes only support files up to a certain size, for example, 10 MB and others don't support multiple compression layers.

Hooked Call



Unaligned Call



File Size Bloating

Many commercial sandboxes have file size limitations on sample submissions (some free public-facing sandboxes have 100 Mb maximum file size). To circumvent sandbox submission, malware authors are bloating the size of the file by padding the file with zeros.

Reserved Characters in Filename

Reserved characters cannot be used when creating folders or files due to them being part of system functions. Malware authors are using these illegal characters in the file name in the hope that the sandbox will reject the sample.

VMRay will recognize the use of reserved characters and automatically rename the file prior to submission

Defeating Sandbox “Technology” Weaknesses

In order to ensure malware cannot evade analysis by these methods a sandbox analysis environment should:

Not Rely on Modifying the Target Environment: A common approach for sandbox analysis is hooking. That presence of a hook (the injected user-mode or kernel-level driver that monitors and intercepts API calls and other malware activity) gives malware the opportunity to disable analysis.

Run Gold Images as Target Analysis Environments: For efficiency and convenience, many sandboxes have a ‘one size fits all’ approach. A single type of target environment is used for all analyses. A better approach is to use the actual gold images (that is, the standard and server OS and application configurations that your enterprise uses) as the target environment. That way, you can be assured that any malware that is targeting your enterprise and could run on your desktops or servers will also run in the analysis environment.

Monitor all malware-related activity, regardless of application or format:

Some malware sandboxes, particularly those using a hooking-based approach, take shortcuts and compromises for the sake of efficiency in determining what activity is monitored. This can leave blind spots.

VMRay’s technology accommodates all these scenarios. When used in conjunction with using real-world VM images as the target analysis machines, VMRay gives full visibility into malware activity, regardless of attempts by the malware to obfuscate its intentions.

Confidently automate response actions to incidents with accurate analysis and definitive verdicts.



Exploiting “Context-Aware” Evasion Techniques

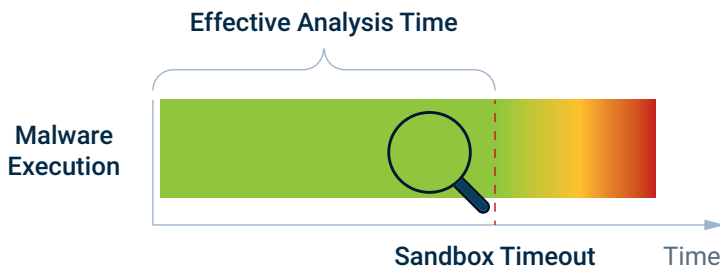


Figure 5. On detecting a check, malware can hibernate for a random amount of time, making it appear benign.

Some malware does not try to detect or attack the sandbox at all. Instead, it exploits the natural shortcomings of automated analysis or research systems. Because of the high volumes of unique malware seen in many environments – especially where automation is needed, sandbox analysis systems usually only spend a few minutes on each file. By delaying the execution of a malicious payload by a random amount of time, malware can remain undetected. Besides time-triggers, malware can also use other events that usually do not occur in a sandbox, for example, initiating a system reboot or detecting user interaction. Additionally, the malware may be looking for specific triggers present on the intended target machine, such as a specific application or localization setting.

Triggers can be grouped into seven categories:

Time Bombs

One of the most common techniques is to delay execution for a certain amount of time since sandboxes usually run samples only for a few minutes. As with many other evasion techniques, the utilization of time bombs is an ongoing cat and mouse game; the malware goes asleep, the sandbox tries to detect sleep and shorten the time, malware detects shortened time, the sandbox tries to hide time forward by also updating system timers and so on.

Time bomb techniques include:

- Simple to very complex sleeps; concurrent threads that watch each other or are dependent on each other.
- Executing only at a certain time or on a specific date
- Slowing down execution significantly. An example of this is injecting millions of arbitrary system calls that have no effect except to slow down execution (API-Hammering), especially when being executed in a monitored or emulated environment.

System Events

The malware only becomes active only on shutdown, after reboot, or when someone logs on or off. This is where a second-stage payload is pulled down only after a reboot. An executable is installed by the malware (the initial payload) that will run automatically on startup after reboot. It's the startup executable process that fetches the second payload.

Time triggers allow malware to go undetected by simply **delaying its detonation**.



User Interaction

Waiting for mouse movements or keyboard input. Interacting with certain applications, such as the browser, email, Slack, or an online banking application.

Fake Installers

Malware only becomes active after a user has clicked multiple buttons and checked various checkboxes.

Office Documents with Malicious Embedded Content

The malicious content only becomes active when the user scrolls down to view it or clicks on it.

Detect a Specific Target System

Sophisticated targeted malware only works on the intended target system. The identification is usually based on the current username, time zone, keyboard layout, IP address, or some other system artifacts. The check itself can be done in various ways, ranging from simple to very complex methods.

Simple Checks Including String Checks

Complex checks are nearly unbreakable if the expected target environment is not known. The malware will only proceed to the second stage to download the main payload if it determines it is in the expected target environment.

Related to this is the inverse scenario where the malware detects that the environment is most likely an artificial analysis environment. This can be the result of checks such as:

- If network usage statistics of the system are too low, then don't do anything
- If 'recently used documents' are almost empty, then don't do anything
- If a number of processes are $< x$, then don't do anything.

Defeating "Context-Aware" Evasion Techniques

Of the three categories of sandbox evasion techniques commonly used, context-aware malware is the least sensitive to the underlying malware sandbox technology. As sandbox technology improves and finds ways to circumvent sandbox detection, environmental triggers will become increasingly important to malware authors.

It is critical for security teams to ensure they are using target analysis environments that accurately replicate in every detail the actual desktop and server environments they are protecting. Furthermore, as we wrote previously, it's important to have pseudo-random attributes as part of the target analysis environment.

Generic sandboxes running identical standard target environments are no longer sufficient. Further, the analysis environment needs to be able to detect environment queries and identify hidden code branches. VMRay can randomize analysis environments, including when desktop or server gold images are used as the targets. Additionally, VMRay will flag when malware is making environment queries. Combined, these ensure that security teams get the full picture and know when they are dealing with context-aware malware.

Alert enrichment with threat actor attribution, IOCs, artifacts, and VMRay Threat Indicators.



Conclusion: The Last Line of Defense

To address the threat posed by advanced malware, enterprises are implementing specialized, resolute teams focused on the detection, analysis, and response to unknown cyber threats. Whether Digital Forensics and Incident Response (DFIR), Cyber Threat Incident Analysis (CTIA), SOC Analysts, or Threat Hunters, their goal is the same. That goal is to identify and observe malware indicators and suspicious activity to mitigate current and future intrusions.

Threat intelligence feeds and public data sources do provide valuable information on now-known malware or phishing campaigns, allowing threat analysts to leverage curated data to thwart attacks before detection signatures become readily available. However, in the case of high-value, specific targets such as a Government Agency, or a financial services company, unknown customized attacks may be specifically crafted and unleashed by bad actors, negating the value of threat feed data to some extent. These crafted, sole use attacks leave the targeted enterprise with the complex task of creating their own operational / tactical intelligence. This typically happens using forensic analysis, after the damage has been done and an attack has succeeded.

Surprisingly, 75% of malware threat intelligence gathering by organizations utilize sandbox technology to extract the necessary IOCs and artifacts. Those IOCs and artifacts are then used to strengthen the network and system defenses prior to, during, or after an attack.

If the malware fails to detonate because it has detected a monitored sandbox environment, the time and costs associated with manual triage of every sample that stalls, fails to fully detonate, or returns a verdict of benign when suspect, can be staggering. That doesn't include the recovery costs should a malicious sample of ransomware incorrectly flagged as benign inadvertently becomes unleashed upon the network. This is why Anti-Sandbox evasion resistance in a sandbox is important – especially where full automation and autonomous responses are required.

Anti-Sandbox evasion resistance alone is a strong argument for boosting Return on Investment (ROI) and Total Cost of Ownership (TCO), not to mention improving the economy of SOC services and incident response times to meet organizational or client Service Level Agreements (SLAs). Speed and accuracy of analysis, as well as clutter free reporting all play a part, especially when automated workflows are involved, but malware sandbox evasion could be the one thing that scuttles the dream of full SOC automation.



Increase the success of threat hunting programs with clear IOCs, artifacts, and uncluttered reporting.



About VMRay

At VMRay, our purpose is to liberate the world from undetectable digital threats.

Led by reputable cyber security pioneers, we develop best-of-breed technologies to detect unknown threats that others miss. Thus, we empower organizations to augment and automate security operations by providing the world's best threat detection and analysis platform.

We help organizations build and grow their products, services, operations, and relationships on secure ground that allows them to focus on what matters with ultimate peace of mind. This, for us, is the foundation stone of digital transformation.

Read more about our solutions at vmray.com

Ready for the next step?



VMRay Public Threat Feed

Explore 1M+ analysis reports

<https://threatfeed.vmray.com/>



See DeepResponse in action


Request free trial

<https://www.vmray.com/try-vmray-products/>


Portfolio

Our portfolio of products (DeepResponse, FinalVerdict, and TotalInsight) offers the ultimate solution for organizations looking to overcome their challenges in detecting and responding to malware and phishing threats.

Whether you need to automate alert processing, share industry-specific threat intelligence or build a comprehensive threat repository, our portfolio has you covered.



DeepResponse



<https://www.vmray.com/products/vmray-deepresponse/>



FinalVerdict



<https://www.vmray.com/products/vmray-finalverdict/>



TotalInsight



<https://www.vmray.com/products/vmray-totalinsight/>

VMRay Professional Services



Contact Us

Email: sales@vmray.com
Phone: +1 888 958-5801

VMRay GmbH

Suttner-Nobel-Allee 7
44803 Bochum • Germany

VMRay Inc.

75 State Street, Ste 100
Boston, MA 02109 • USA

