New hyper-evasive threats are killing sandboxing as we know it

by Sigurdur Stefnisson

[+] (IN)SECUREMagazine

Issue 51, 09/2016

HACKING IS THE NEW ESPIONAGE

SECURING YOUR SPOT AT THE TOP: HOW TO COLLABORATE AND WHEN TO COMPETE

WHY NEW HYPER-EVASIVE THREATS ARE KILLING SANDBOXING AS WE KNOW IT

WHAT CAN MICROSOFT PATCH TUESDAY TELL US ABOUT SECURITY TRENDS IN 2016?

Originally published in (IN)SECURE Magazine #51, September 2016.

www.insecuremag.com

New hyper-evasive threats are killing sandboxing as we know it

Sigurdur Stefnisson

Like military generals, IT security professionals frequently fight the last war using weaponry and tactics that worked in the past, while the enemy has studied our countermeasures and is already applying new methods.

Today we face a new generation of malware which we might term “hyper-evasive” – threats that have never been seen before, in volumes never seen before, and which have been designed to evade traditional malware defenses, and specifically, current sandboxing technology. These threats require a rethinking of our battlefield strategy.

The evolution of threats

To understand today’s threat landscape, we should consider the evolutionary history of cyber threats, while keeping in mind that each new class of threats emerged from the criminals’ detailed understanding of the limitations of then-current protection technologies. While a rough chronological retelling is possible, my purpose here is to give a quick sense of classification, and emphasize that techniques are frequently cumulative – nothing ever really “goes away.” We might consider the following “tiers” of malware sophistication:

Tier 1: Basic malware

These compromise a computer in order to gain access to its resources and – on occasion – its data. Attacks of this type are slow-moving and frequently noisy and obvious, and can be blocked by analyzing a virus to capture a hash fingerprint or signature, and then using this hash to identify and block subsequent copies of the virus.

Tier 2: Polymorphic malware

Although they are often basic viruses under the hood, polymorphic viruses first emerged in the early 1990s: malicious code that mutates and changes its appearance each time it infects a new object in order to avoid pattern recognition by antivirus software.

The emergence of polymorphic malware triggered the arrival of behavioral heuristics as a countering technology, whereby the behavior of the code execution during an emulation is observed. The development of application sandboxing in the 1990s was a key response to polymorphic malware.

The advent of server-side polymorphism has taken malware to the next level, with back-end web services hiding the mutation engine where defenders cannot inspect it. Sophisticated algorithms ensure that each time a download occurs from a URL you receive a different file, and attack methods frequently involve encryption, droppers and packers.

Tier 3: Hyper-evasive malware

Cyren has noted an emerging trend of threats incorporating many known evasion techniques within a single piece of malware. Attackers have evolved their techniques to the point where malware rarely contains obviously suspicious code and originates from "unknown" sources or from code lodged in compromised, trusted sites. As the use of sandboxing for malware defenses has increased, malware that is "sandbox aware" has become more prevalent.

Usually assumed to be the purview of large enterprises, a recent study commissioned by Cyren and conducted by Osterman Research (July 2016) found that over 50 percent of small and mid-sized companies (100 to 3,000 employees) have also deployed an appliance-based sandboxing capability.

But such "hyper-evasive" threats are increasingly difficult for traditional, appliance-based sandboxing to detect, as the malware coders use sophisticated evasive tactics to exploit limitations in the architecture of appliance-based sandboxing.

Limitations of traditional application sandboxing

Traditional application sandboxing has become a critical last layer for corporate information security to attempt to stop infiltration. It pushes suspicious objects to a simulated end-user computing environment running on an appliance in a corporate data center.

While the overall volume and sophistication of unknown objects was relatively low, and turnaround time was consequently fast, this approach was deemed sufficient. However, the broad deployment and very success of appliance-based sandboxes has led to not-surprising innovation by criminal enterprises, as per the usual historical pattern.

It is worth restating that the variety and depth of testing that is possible within first-generation sandboxes is limited. Among the realities of traditional sandboxing that are being exploited today are:

1. The fixed amount of physical resources (i.e. memory and processing power) available in a sandbox appliance, which limits the scalability of the solution in terms of total analysis object load and depth of analysis performed.
2. The reliance on virtualized environments, the presence of which can be detected by malware.
3. The lack of diversity in the scope and origination of the tests employed, with the variety and nature of tests limited to those devised by the specific sandbox vendor.
4. The fact that any specific sandbox is best at one kind of analysis, e.g. OS or registry or network behavior analysis.

This last element is the most critical limitation of all, as it enables malware developers to optimize analysis detection techniques for each sandbox platform, knowing that once they have found a "tell" for the particular sandbox being used, their evasive techniques will get them past what is effectively the organization's last line of defense. No method is provided by traditional sandboxing solutions to harness together sandboxes of different types (or from different vendors) in a collaborative analysis model, to enable a broader and deeper scope of testing with a pooling of analysis results.

How hyper-evasive threats evade detection

Sandboxing solutions deploy two types of analysis:

• Static analysis is performed by the system without executing the suspected code. Examples of static analysis techniques include file fingerprinting, extraction of

- hard-coded strings, file format metadata, emulation, packer detection, and disassembly.
- Dynamic analysis is performed by the system while the suspected code is executed inside a protected (sandbox) environment. Examples of dynamic analysis techniques include analyzing the difference between defined points, and observing run-time behavior.

To combat the growing use of first-generation sandboxes, cybercriminals have developed evasive techniques for use against both types of analysis.

Some of the techniques are sophisticated, while others perform simplistic tests to determine if the malware is in a real or simulated (sandbox) environment.

TO COMBAT THE GROWING USE OF FIRST-GENERATION SANDBOXES, CYBERCRIMINALS HAVE DEVELOPED EVASIVE TECHNIQUES FOR USE AGAINST BOTH TYPES OF ANALYSIS

Common techniques for evading sandbox analysis include:

- Detecting the existence of a virtual environment
- Delayed activation – attempting to "outwait" the sandbox
- Awaiting human interaction like mouse movements that could not result from a simulation
- Making payload execution conditional.

As an example of this last approach, we are seeing recent ransomware downloaders that have added the requirement of an additional parameter for the execution of the downloaded ransomware code. A sandbox may have the download file itself, but it does not have the full script – so it would not detonate in the sandbox because it is missing one component (a parameter).

Consider the impressive list of functions identified by Cyren researchers within a single variant of Cerber ransomware, which uses multiple methods to check for the presence of, and therefore hide from, a sandboxing environment.

Virtual machine check functions:

- Parallels
- QEMU
- Oracle VirtualBox
- VMWare
- an unknown VM.

Debugger process check functions:

- CommView Network Monitor
- WinDump
- WireShark
- DumPCAP
- OllyDbg
- IDA Disassembler
- SysAnalyzer
- SniffHit
- SckTool
- Proc Analyzer
- HookExplorer
- MultiPot.

Sandbox check functions:

- Loaded modules check against
    - sbiedll.dll - Sandboxie
    - dir_watch.dll, api_log.dll - Sunbelt Sandbox
- Volume serial number checks against
    - ThreatExpert
    - Malwr
- Mutex name checks against
    - Deep Freeze - Frz_State
- Other file path checks on modules used in sandbox setups
    - C:\popupkiller.exe
    - C:\stimulator.exe
    - C:\TOOLS\execute.exe
- String checks from memory
    - test_item.exe
    - \sand-box\
    - \cwsandbox\
    - \sandbox\

The inclusion of this variety of functions shows that malware writers are hard at work researching sandbox and debugging technologies that the security industry is most likely to use in solutions, and proves that they can simply embed more anti-sandbox/debugger/VM modules in their malware as they see fit, significantly increasing the evasiveness of the threats.

Conclusion

The appropriate response to the advent of hyper-evasive malware, which can evade any given sandbox and was designed to exploit the relatively limited processing power of appliance-based solutions, is to exponentially improve the analytical capacity of the sandboxing systems. This can be done by subjecting malware to multiple and varied sandboxing environments while applying increasingly sophisticated analysis, something now possible via the elastic processing scale and Big Data analytical capabilities of cloud computing.

The best countermeasure is to automate the strengths of human analysts – in particular their capacity for complex decision-making and even “hunches” – through a cloud-based processing model.

Sigurdur Stefnisson is the Vice President of Threat Research at CYREN (www.cyren.com).