WHITE PAPER

FireEye

THINKING OUTSIDE THE SANDBOX:

Harnessing Automated Dynamic Analysis to Carry Out the Australian Signals Directorate's Newest Mitigation Strategy

CONTENTS

The news program Four Corners stunned Australian viewers in May 2013 when it exposed widespread data breaches of government agencies and major firms.[1]

As shocking as the report may have seemed to the public, it only confirmed what Australia's security experts have long known. Cyber attacks are growing more frequent. They are growing more effective. And they are growing more serious.

In a September 2013 interview, Australia's Defence Signals Directorate said the number of serious cyber attacks against government rose 39 percent from the year before and was up 205 percent from 2011.[2] Australian government agencies faced more than 1,300 cyber security incidents between January and August 2013 — or 5.4 events per day.[3]

Many of these incidents involve advanced attacks. Sponsored by foreign governments and well-organized cybercriminals, these attacks are easily slipping past standard security tools. Anti-virus (AV) software, traditional and next-generation firewalls, intrusion-prevention systems (IPS), and other tools are useless against them.

A new approach

Responding to the growing threat, the Defence Signals Directorate (also known as the Australian Signals Directorate, or ASD), for the first time is endorsing automated dynamic analysis as a defense tactic. The February 2014 edition of *Strategies to Mitigate Targeted Cyber Intrusions — Mitigation Details* introduces "automated dynamic analysis of email and web content run in a sandbox" as Mitigation Strategy No. 6.[4]

Instead of relying on signatures, automated dynamic analysis systems observe malware behavior using virtual machines (VMs). These walled-off, simulated computer environments allow files to execute without doing any real damage.

By watching the files in these virtual "sandbox" environments, automated analysis systems can flag telltale behavior, such as changes to the operating system or calls to the attacker's command-and-control (CnC) servers.

Sandboxing was the only addition to ASD's top 35 strategies. It is also the only control in the top six that the ASD deemed low cost and well tolerated by users. The ASD says sandboxing has grown more effective and available since its last set of recommendations.

Head in the sandbox: not all technology is created equal

In the wake of the new ASD guidelines, security vendors are scrambling to add sandbox tools to their portfolio. Everyone, even incumbent vendors who have long defended their aging legacy tools, seems to have embraced the concept. With so many choices — and confusingly similar marketing claims — choosing the right dynamic analysis tool can be daunting.

To cut through the hype, agencies must understand that sandboxes are only a tool, not a silver bullet. By themselves, sandboxes can only monitor and report file activity. Analyzing it effectively is more difficult — and critical.

The paper explains how sandboxing works, the failings of most sandbox-based approaches, and what agencies should look for in VM-based analysis.

[1] Andrew Fowler and Peter Cronau (Four Corners). "Hacked!" May 2013.
[2] Christopher Joye (Financial Review). "Spy agency reveals big increases in cyber attacks." Sept. 2013.
[3] Ibid.
[4] Cyber Security Operations Centre. "Strategies to Mitigate Targeted Cyber Intrusions – Mitigation Details." February 2014.

Why Traditional Defenses Are Failing

Security professionals widely agree that standard signature-based security tools are futile against today's sophisticated attacks.[5] Attackers use a range of techniques to outwit signature-based tools, including ever-changing binaries, multi-stage attacks, and zero-day exploits.

Malware binaries outpacing signatures

Attackers have many anti-detection methods at their disposal:

- Binary packing
- Compression
- Encryption
- Compiler variation
- Polymorphism

Using these techniques, attackers generate a large volume of unique binary samples from the same malware family — each with a unique hash value.

Signature matching is limited to specific hash samples. So as the volume of newly introduced unique samples grows, timely detection becomes less and less possible.

Multi-stage attacks

Advanced attacks comprise a number of distinct, coordinated stages, often using multiple attack vectors. They can be delivered through websites, email, files shares, and mobile devices. Malware campaigns often blend attacks vectors. For example, email-based attacks can contain malicious URLs.

Many of these advanced attacks are also multi-flow. Attackers do not send a single malicious file to a targeted system, where it might trigger a malware alert. Instead, they send several files or objects that appear harmless by themselves. When combined, these files and objects reveal their true nature.

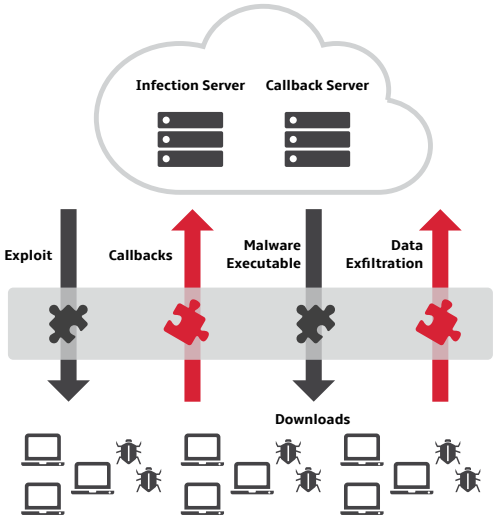

Figure 1: The multi-stage, multi-flow, and multi-vector nature of today's attacks

For instance, many Web-based attacks comprise multiple downloaded files or objects. These objects often stem from multiple HTTP request and responses, including redirects, and multiple TCP sessions.

One object might be used for a heap spray. Another object might include an overflow or un-sanitized input to exploit. Another object might defeat OS defenses such as address space layout randomization (ASLR) and data execution prevention (DEP). And finally, another downloaded binary might be an image with hidden malicious code, which executes only when extracted by another seemingly benign file.

Zero-day exploits

Zero-day vulnerabilities are software flaws that leave users exposed to cyber attacks before a patch or workaround is available. Sometimes, a zero-day vulnerability is unknown to anyone but a cyber attacker (or a supplier who sells zero-day discoveries on the black market). In other cases,

[5] Gartner. "Best Practices for Mitigating Advanced Persistent Threats." January 2012.

the software vendor knows about the flaw but has not yet issued a fix.

By definition, signature-based defenses work only for threats that have been discovered and recorded.

Even when vulnerabilities are known and have an available fix, agencies cannot always patch the software quickly. The patch might create compatibility issues or break a custom application.

Using automated dynamic analysis with other ASD controls

The ASD's top four mitigation guidelines prevent many client-side attacks. But they are not a cure-all.

Take ASD strategy No. 1, application whitelisting, for example. Whitelisting prevents a system from executing arbitrary code, which is delivered in the dropper stage of an advanced attack or as part of a social-engineering attack. Most advanced attacks, however, begin well before a malicious binary enters the targeted system.

During the first stage of an advanced attack, exploit code delivers shellcode that executes within the context of an exploited process, such as Internet Explorer, Java, or Adobe Reader. Whitelisting does not restrict all types of executable code. Several interpreted languages, including the following, run under a whitelisted host process:

- VBScript
- Jscript
- Batch files,
- Java applications,
- PowerShell
- Python
- Visual Basic for Applications and Office macros

Although no binary has yet been written to disk at the first stage, the attacker is already running code on the compromised system.

ASD strategies No. 2 and No. 3 advise patching operating systems and applications, respectively. These actions also stop some client-side attacks. But many attackers exploit zero-day vulnerabilities.

That's where dynamic analysis comes in. Sandboxing is designed to cover the gap left open by earlier ASD controls.

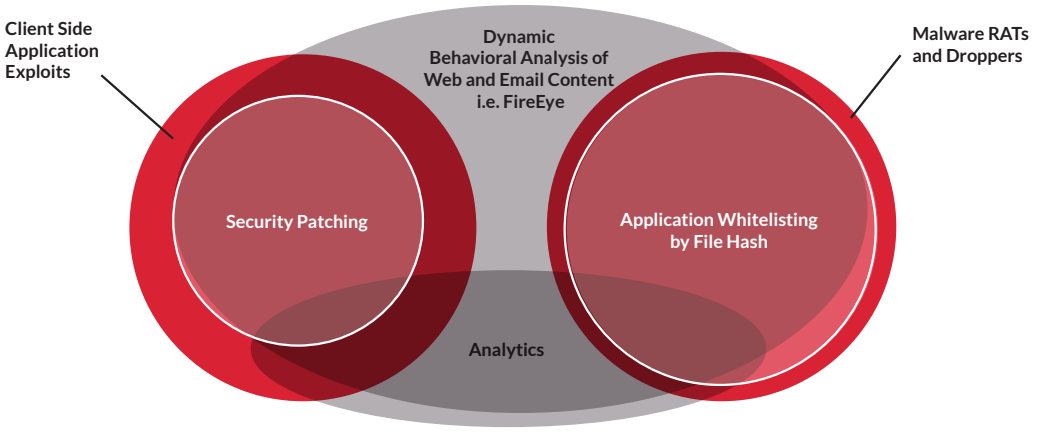

Figure 2: How automated dynamic analysis (sandboxing) is designed to complement existing ASD controls

Why Most Sandboxes Fall Short

Many sandboxes have fundamental flaws that leave agencies vulnerable. Many are easily detected and evaded. Some analyze files in isolation rather than part of a coordinated whole. Some myopically focus on a single threat vector. Some fail to emulate complete systems or emulate only a single "golden" image. Some measure only the beginning and end states of a virtual system — missing everything that happens in between.

Many sandboxes have fundamental flaws that leave agencies vulnerable. Many are easily detected and evaded.

Many sandboxes easily detected and evaded

Mindful that their code may execute in a sandbox before it reaches its target, malware authors are creating VM-aware code. This code hides any telltale behavior until it has reached "live" prey. Observing no suspicious actions in the sandbox, the security analysis deems the code harmless.

These sandbox-evasion techniques include the following:[6]

- **Human interaction**. Some malware waits for mouse clicks and interaction with dialog boxes before executing. Because sandboxes do not mimic user actions, they can miss malware that requires human interaction to run.
- **Sleep calls and time triggers**. Most sandboxes execute files for a limited time. Sandbox-aware malware can lie dormant for a set time or until reaching a preset trigger that falls outside of the execution window.
- **Hiding processes**. Using undocumented internal pointers for Windows' *PsCreateProcessNotifyRoutine* function, malware can cancel registered callbacks to prevent sandboxes from detecting malicious processes.
- **VMware-specific checks**. Some sandbox security tools use off-the-shelf virtual machines such as VMware. VMware-based system images have telltale features that are easily detected — and sidestepped — by advanced malware.

The ASD guidelines advise agencies to avoid analysis tools that are fooled by these techniques.[7]

Many sandboxes analyze files in isolation

Most sandboxes analyze suspicious files and objects one at a time. But as explained earlier, today's advanced attacks use multiple components working in tandem.

A typical attack follows this cycle:

1. Exploit
2. Callback
3. Malware download
4. Data exfiltration

The individual parts of an attack may seem harmless when analyzed in isolation, so most sandboxes see nothing amiss. But once the components make their way into a targeted system, they combine to devastating effect.

Detecting the initial exploit is especially critical because later phases are often encrypted or hidden. For example, recent attacks have included remote access tools that send stolen data over encrypted command-and-control (CnC) channels to avoid detection.

[6] FireEye. "Hot Knives Through Butter: Evading File-based Sandboxes." August 2013.
[7] Cyber Security Operations Centre. "Strategies to Mitigate Targeted Cyber Intrusions – Mitigation Details." February 2014.

Many sandboxes are focused on a single vector

Most sandboxes also focus on a single threat vector such as Web or email. But many advanced attacks unfold over multiple vectors. For instance, email spear-phishing campaigns often start with a malicious attachment with the exploit. But later stages of the attack include, say, downloading a malicious payload over the Web.

If the sandbox cannot connect activity across vectors, it cannot detect the true nature of the individual components.

Many sandboxes fail to emulate complete systems

Many sandboxes are designed to analyze only executable files. While executables are indeed important, today's advanced attacks also utilize documents and other content. These weaponized files exploit vulnerabilities in client software such as Adobe Reader, Microsoft Office and the Java Runtime Environment.

Targeted spear phishing attacks, for example, commonly use weaponized PDF or Office documents. Drive-by and watering hole attacks hide their exploits in Web page content.

To analyze exploits and document-based content, sandboxes need a matching client-side application that can open them.

Many sandboxes emulate only the "golden image"

For sandboxes that do emulate complete systems, many use a single image, often a company's "golden" base installation. This approach is a mistake.

Many advanced attacks targeted specific combinations of operating systems and client software. If that combination is not present in the sandbox, the malware lies dormant and passes undetected. Once it reaches a system with the targeted software combo, it executes.

End-user systems are rarely identical, even in well-managed IT environments. Some users may have updated their Web browser. Others may have failed to apply an Adobe reader patch. If the sandbox's golden image varies even slightly from those of end users, it can miss the attack.

To detect advanced attacks, dynamic analysis must be able to test suspect files and objects in a variety of settings.

Many sandboxes use delta analysis, not runtime analysis

Pacing the right content into a sandbox with the right applications is only part of the story. Sandboxes' ability to detect advanced attacks hinges on the way monitor and react as the content executes.

Sandboxes typically take one of two approaches:

- **Delta analysis**: comparing the sandbox image before and after execution
- **Runtime analysis**: embedding instrumentation so that execution can be observed as it happens

Many sandboxes take the delta approach. Delta analysis record only changes that appear after execution. It cannot "see inside the box" during runtime.

Delta analysis cannot monitor operations that run in memory, for example. It also cannot detect files that are written and then deleted. So it misses stealthy malware that covers its tracks. And it cannot react to sandbox-evasion techniques (such as sleep timers and requesting human input) outlined earlier in this paper.

How FireEye Is Different

Built from the ground up to combat a new generation of threats, the FireEye Multi-Vector Virtual Execution (MVX) engine transcends file- and object-based sandboxes. The MVX engine captures and confirms zero-day and targeted threats by detonating suspicious files, Web objects, and email attachments within multiple instrumented virtual-machine environments.

Rather than analyzing a subset of files in isolation, the MVX engine correlates activity across multiple threat vectors and attack stages. In other words, it connects the dots to analyze the full context of an attack.

Proprietary hardened hypervisor

Unlike some sandboxes, the MVX engine is not an off-the-shelf virtual machine that is easily detected and foiled. Instead, the MVX engine features a hardened hypervisor built for one purpose: analyzing malware. This patented technology allows several virtual machines to run on a single appliance and leverages parallel micro-tasks within each virtual machine to speed up execution.

The MVX engine manages real-world high-speed traffic streams. It is also equipped with an evolving set of countermeasures to guard against malware, as advised by the ASD guidelines.

Analysis in context

Stateful attack analysis is critical when analyzing the entire attack life cycle from initial exploit to data exfiltration. Point products that focus on single objects (such as malware executables, DLLs, or PDF files) miss most attacks. They are blind to the full attack life cycle.

The MVX engine supports many parallel execution environments. That means it can analyze attacks the way they really occur — in multiple flows, over multiple stages, across multiple threat vectors.

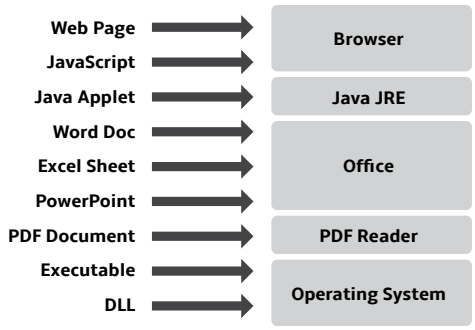

Figure 3: Attacks combine multiple files and objects in a single attack

A wide mix of environments and conditions

The MVX engine's virtual machines execute suspect files and objects in a wide range of operating systems, service packs, and applications. This variety enables the MVX engine to detect highly targeted malware that sidesteps sandboxes that emulate only the OS or a single system image.

Figure 4: FireEye virtual detection model

Conclusions and Recommendations

Sandboxes are no cure-all for advanced attacks — they are only as good as the analysis they enable.

To truly protect IT assets, virtual-machine-based analysis must overcome sandbox-evasion techniques of advanced malware. And when new evasion techniques emerge, vendors must quickly update their tools.

Dynamic analysis must analyze files and objects in context and across multiple threat vectors. And they must offer a wide variety of environments to detect targeted malware.

Virtual-machine-based analysis is even more effective when augmented by dynamic, real-time threat intelligence and a full complement of services. With a complete view of attacks within an enterprise, geography, or industry, security teams can better prevent, detect, contain, and resolve advanced attacks.

What to look for in an automated dynamic analysis tool

When appraising automated dynamic analysis systems, agencies should look for a solution that offers the following features.

A wide range of environments and applications

- Support for the widest range of operating system versions and patches.
- Support for a wide variety of Web browsers and plugins. This variety should include multiple versions of Internet Explorer, Firefox, and Chrome along with plugins such as Java, Flash, Shockwave, and Silverlight.
- Support for desktop applications such as Adobe Reader and Microsoft Office. The tool should analyze suspicious files and objects against all versions of apps used in an agency's network — not just the most common version.
- The ability to automatically and dynamically match browser type and version and plugin versions used in VM-based analysis with those installed on real-world client workstations.

A full view of object and file behavior

- The ability to capture and correlate the full attack life cycle.
- The ability to determine and block the callback channels observed when executing malware in the VM.
- The ability to detect and block the binary dropper stage observed when exploit shellcode executes in the VM.
- The ability to detect multi-vector, multi stage attacks. These attacks include browser-based exploits that comprise multiple files and dependencies on multiple client-side applications.
- The ability to detect and classify polymorphic malware by observing deterministic traits.
- The ability to detect previously unknown software vulnerabilities and zero-day exploits. This capability requires observing techniques such as heap sprays, code injection, misusing Windows APIs, API hooking, and modifying kernel routines.

Virtual-machine-based analysis is even more effective when augmented by dynamic, real-time threat intelligence and a full complement of services.

- Complete forensic details of observed malware behaviors, such as Windows API calls made, memory corruptions and overflows, file and registry locations read or changed, known mutexes, and other malicious indicators.

- "As-it-happens" runtime analysis rather than basic delta comparison.

A security-focused hypervisor

- A dynamic execution environment built from the ground up to analyze malware.

- A proprietary hypervisor that counters sandbox detection and evasion techniques.

- Regular updates from the vendor to address evolving sandbox-evasion techniques.

Dynamic analysis that works the way you do

- A very low rate of false positives and no false negatives.

- The ability to analyze files and objects on premises, not just in the cloud.

- The ability to analyze email before delivering it to users so that users are not exposed to malicious content.

- For Web content that reaches users before it can be identified as malicious, the ability to quickly contain any damage. Steps might include quarantines and blocking infected machines' network access.

To learn more about how the FireEye platform can help your agency fulfill the ASD's Mitigation Strategy No. 6 and other guidelines, email FireEye Australia at australia@fireeye.com.

FireEye, Inc. | 1440 McCarthy Blvd. Milpitas, CA 95035 | 408.321.6300 | 877.FIREEYE (347.3393) | info@fireeye.com | **www.fireeye.com**

FireEye