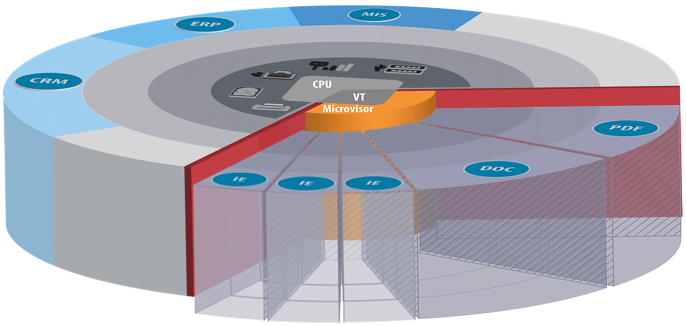
Bromium®

Micro-virtualization vs Software Sandboxing

A Comparison

This document briefly compares Bromium micro-virtualization with other techniques used in the security industry, in particular software sandboxes. It concludes that software sandboxes will always be vulnerable to attack, and makes the case that the Bromium architecture offers a quantum leap forward both in protection and detection, forensics and live analysis of ongoing attacks.

Micro-virtualization

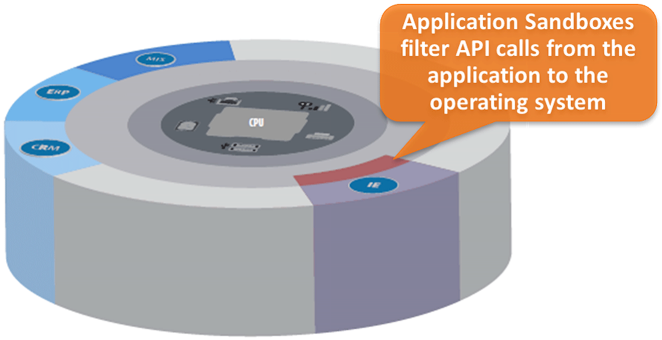


Bromium vSentry relies on **hardware isolation for tasks** that are untrustworthy – those that involve code or data from some external system that is being processed locally, such as browser tabs, documents and media. These hardware-isolated tasks are called micro-VMs and they protect the operating system and other tasks from compromise by malware that executes in a micro-VM. Valuable data, networks and devices are not available in a micro-VM, and no matter what malware does to the micro-VM, including arbitrary escalation of privileges and manipulation of the OS and system state, no changes are reflected back on the operating system or its file system, preventing attack persistence. Upon the termination of the task, the micro-VM and its contents (and changes) is simply discarded, including all malware. This represents a major step forward in securing computer systems because unlike any software based isolation, the protection in vSentry is afforded by the CPU itself. This document contrasts the different approaches used by security vendors to prevent attacks, highlighting the differences between micro-virtualization and the state of-the-art in the industry today.

Methods for Protection

There are three ways to protect an endpoint under attack:

1. **Detect and then block the attack** within the OS, by foiling its attempts to access system resources, the moment the attack is detected. This is the approach taken by the current state-of-the art of endpoint security software. It presumes that the detection of malware will be successful, which is problematic in the context of today's advanced polymorphic attacks – for which detection rates are extremely low. It also necessitates post-attack remediation: even if the attack is blocked, the system must be cleaned of malware – perhaps even to the extent of re-imaging the system.
2. **Sandbox the attack** using an application-specific or more general purpose software sandbox. (This link offers a full description of sandboxing, including its limitations). In summary, sandboxes attempt to limit the ability of an untrusted user space process to enter the OS kernel using a software replacement for all system calls and other access methods for the kernel. Sandboxes typically have a very broad interface to protect – for example Microsoft® Windows offers more than 2,000 system calls as well as many OS service interfaces and interprocess communication methods, all of which must be intercepted. As a result a sandbox must necessarily rely on millions of lines of additional (potentially vulnerable) code. For example the Google® Chrome sandbox counts approximately 1.5 million lines of code, and has been developed over a period of 12 years. Application sandboxes are extraordinarily complicated, and frequently exploited – for example the recently documented flaws in the Java® 7 virtual machine.

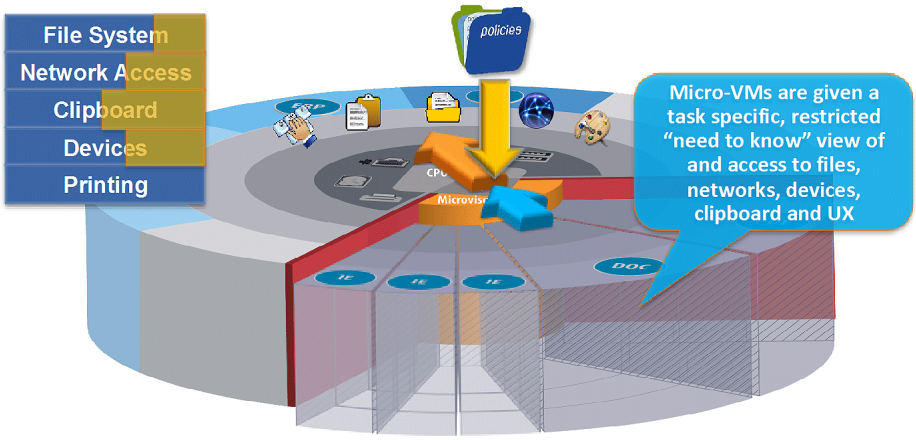


Application sandboxes are usually application specific or require substantial porting of an application in order to function correctly. For example Sandboxie offers fairly general purpose characteristics, but the applications must be re-installed in the sandbox, and the user experience is modified. Moreover they conflict with enterprise application delivery technologies such as Microsoft® App-V, VMware® Thinstall and Citrix Application Virtualization. Ultimately one should expect key applications and application vendors to adopt app virtualization to simplify delivery, making general purpose application virtualization increasingly difficult to manage.

Application sandboxes are typically very poorly instrumented – focused on interception, and not delivering insight to security teams about how tasks are being attacked, by whom, and in what ways. Ultimately the information that can be gained within a sandbox is no better than could be gained using AV software. Application sandboxes do not permit introspection, nor do they offer insights into the origins, targets and methods of an attack.

Both (1) and (2) above are vulnerable to human frailties, with catastrophic consequences: When the detector fails to detect, or when malware finds a way to escape from the sandbox and enters the kernel, perhaps through a zero day, the system will be completely compromised.

3. The third, and only viable approach for defeating advanced malware, is **micro-virtualization**. The Microvisor isolates an entire task within a micro-VM, using Intel VT to hardware isolate the execution of the task, protecting the OS, the network infrastructure, and all valuable data from malware. This of course raises the question **"What is a task in the Bromium architecture, and why is it a superior isolation construct?"** We address this in more detail below.



Micro-virtualization and Definition of a Tas

vSentry defines a task to be the most granular unit of computation, initiated on behalf of a user, that can completely and successfully execute with the least possible resource access. Resources here mean files, network services, the clipboard, interaction with the user, or any devices or network shares. In vSentry, user tasks are typically:

- **Document centric:** The task involves all processing related to the document, both user space and kernel. In general for documents, communication with other systems is not permitted unless specific policies override this (for example a spreadsheet might need data from an Intranet based Sharepoint site, but a PDF doc doesn't need access to an external site).
- **Web centric:** The TLD of the URL uniquely identifies the task. If a micro-VM for the TLD already exists, then additional processing for all URLs within that TLD will be executed within the existing micro-VM, otherwise a new micro-VM is created. (There is an exception: DOMs with a Meta NOFRAMES tag are injected into a new micro-VM, for security reasons.)

A micro-VM isolates all computation for a single task – **both kernel and user space**. All execution within a micro-VM is copy-on-write against the IT-provisioned system image – both the in-memory image of Windows and the "golden file system" containing windows and applications provisioned for the user.

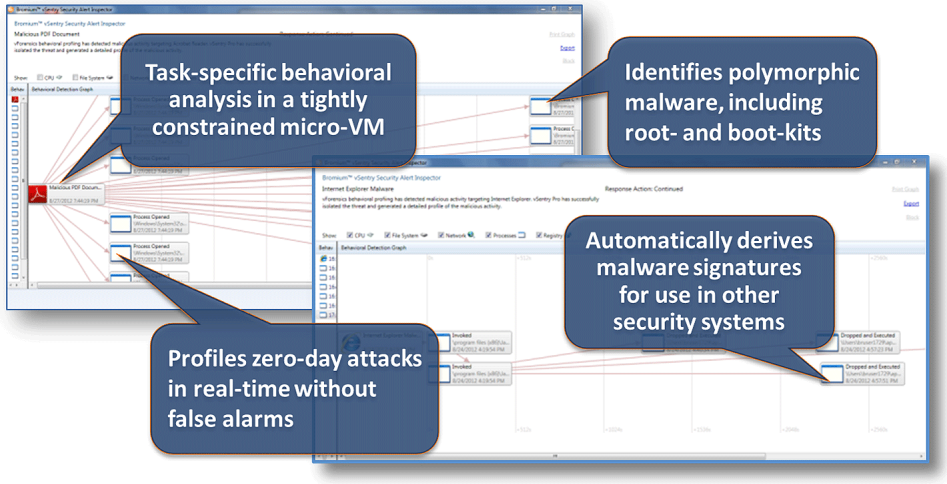
Since our goal is to always protect the system – by design, and then also to deliver complete live forensics in real-time for ongoing attacks, it is crucial that the architecture isolate all computation for the task – both user and kernel space activity. This allows vSentry to protect the system from any changes made by malware – whether or not the malware is detected. It also allows us to ensure that no access to privileged data, networks or resources is possible either in user space or in the kernel, for an untrustworthy task. So, when malware escapes from an application sandbox and then compromises the kernel, vSentry will both protect the system and deliver real time forensics. vSentry's ability to provide live attack visualization and analysis depends on an ability to introspect not just user space execution, but also to fully observe all kernel activity for each task. For example

- A DNS query by a PDF document seeking to resolve the address of a remote botnet C&C, or even the attempted transmission of a datagram that is directly addressed to some Internet site.
- Every process spawned by the task
- Every file opened, saved or read by the task and any attempt to access raw storage devices
- Every attempt to access the clipboard
- Every system call and process creation, including those blocked by Microsoft patchguard, which invalidates many of the hooks used by "in-OS" detection schemes, including all application sandboxes.
- Every registry access

Moreover, since the micro-VM executes CoW, if it modifies kernel memory or overwrites a file in the golden image, these changes are specifically and separately saved withi the micro-VM. In short, every change made to any system state from the moment a task starts until it ends, is recorded, with a timeline of changes.

Micro-VM Introspection vs In-OS Inspection

Introspection is only possible from a hypervisor, and the Bromium microvisor is a special purpose hypervisor. Introspection of a legacy OS-VM is nearly impossible – there are too many concurrent activities in the VM, making it impossible to detect which process made which changes. But in the narrow confines of a micro-VM that contains only a single running task, introspection afford profound insights that are simply not available to in-OS detection tools, including sandboxes.



Finally, the protection afforded by a micro-VM is so substantial that it requires malware to break the CPU in order to compromise the system. The entire code base of the microvisor and all code that could be exploited by malware in an attempt to escape the micro-VM containment, is O(100KLOC). And even if this code is compromised, the system is designed to fail safe – untrustworthy tasks may not execute, but the user will still have full access to their IT provisioned LOB applications, and will have the full protection of traditional AV. By contrast, any failure to detect, on the part of AV, or any break out from the sandbox will cause complete system compromise. The Bromium architecture is designed assuming compromise.

Conclusion

Micro-virtualization and three additional capabilities of the Bromium solution allow it to offer protection that is tens of thousands of times better than any existing protection mechanism – essentially making it too expensive for an attacker:

- **Hardware isolation:** drastically reduces the code base required for isolation. To break the isolation container, malware needs to break Intel VT.
- **Task isolation in micro-VMs:** Protects the kernel and user space execution resulting from any initiated user activity, guaranteeing that even if malware gets into the kernel, it will certainly be defeated.
- **Micro-VM Introspection:** affords insights that are not available to in-OS detection methods, by taking advantage of the hypervisor's privileged role in the system. This permits live attack visualization and analysis without false positives, and provides a full kill-chain for forensic analysis, including signature generation for malware payloads.

Bromium HQ
20813 Stevens Creek Blvd, Suite 150
Cupertino, CA 95014
info@bromium.com
+1.408.598.3623

Bromium UK Ltd
Lockton House
2nd Floor, Clarendon Road
Cambridge CB2 8FH
+44 1223 314914

For more information refer to www.bromium.com, contact sales@bromium.com or call at 1-800-518-0845