TECH NOTE 224

HYPERVISOR CODE INTEGRITY (HVCI) OVERVIEW

The GO-Global Host does not support systems that have Hypervisor Code Integrity (HVCI) enabled. When the GO-Global Host is installed on a system where HVCI is enabled, one of the following will occur:

- The system will crash on startup. When this occurs, GO-Global will detect that the system failed to start, and it will disable itself so the system can start on subsequent attempts. When the system starts and GO-Global has disabled itself, it will display an *access denied* message to users when they try to connect to the computer, and it will notify administrators that GO-Global has disabled itself when they start the Admin Console.

- Alternatively, the GO-Global System Extensions Driver will fail to load. When this occurs, GO-Global will display an *access denied* message to users when they try to connect to the computer, and the Application Publishing Service (APS) log will contain the message, *The GO-Global System Extensions Driver failed to load.*

When either of these occur, you should determine if HVCI is enabled by running System Information (msinfo). If System Information reports that Virtualization-based security or Device Guard virtualization-based security is *Running*, HVCI is enabled.

If HVCI is enabled, you will need to disable HVCI in the computer's BIOS to use GO-Global.

To disable HVCI in the BIOS

1. Restart your computer and press the appropriate key to enter the BIOS. This key is usually **F2**, **F10**, or **Del**.
2. In the BIOS, look for an option to disable virtualization-based security or virtualization technology. This option may be called **VT-x**, **AMD-V**, or similar.
3. After locating the option, disable it and save your changes.

When the computer restarts, run System Information (msinfo) and verify that virtualization-based security is no longer *Running*. Then run the Admin Console. If it reports that GO-Global is disabled, select the option to re-enable GO-Global and restart the computer again.

Background

GO-Global does not block malicious software from attacking kernel memory like HVCI does, but its multi-session architecture mitigates the risk of malicious software inserting hooks into kernel modules by moving the most common and highest risk targets to unique and unpredictable kernel memory locations.

In a standard Windows system, Windows reserves a session-private region of virtual memory in the kernel for each session. This region of memory is called the Session Space. Windows loads an instance of the Win32 subsystem along with its associated modules and drivers into the Session Space. It loads the Win32 subsystem at the same virtual address in each session. This makes it easy to locate the addresses of functions though which sensitive data may pass, such as functions that output text, draw images, process keystrokes, etc.

While the addresses of functions in the Win32 subsystem are easy to locate in a standard Windows system, it is not easy to hook these functions and capture sensitive data. Since the mid-2000s, Kernel Patch Protection has prevented software from modifying sensitive system modules such as Ntoskrnl and the Win32 subsystem. Kernel Patch Protection provides strong protection, but it has been circumvented. Microsoft has released patches that block known circumventions, but there is always the possibility that someone will find a new way to circumvent these protections. HVCI is designed to prevent these circumventions. It provides another layer of very strong, hardware-enforced protections that are very difficult to circumvent.

GO-Global does not support the added layer of protections that HVCI provides, but it does support Kernel Patch Protection. On GO-Global Hosts, Kernel Patch Protection prevents modifications to the Win32 subsystem in standard Windows sessions and prevents modification of the single instance of Ntoskrnl that GO-Global sessions share with the system. Kernel Patch Protection does not protect the memory of the Win32 subsystem from being modified in GO-Global sessions, but GO-Global's architecture mitigates the risk of malware hooking the Win32 subsystem by loading it and the modules that depend on it into a session-private sandbox in the kernel.

When a GO-Global session starts, the GO-Global System Extensions Driver:

- Reserves a virtual address range in the kernel using the GO-Global Heap Driver. The location of the virtual address range is determined by Windows and is different for every session.
- Maps an instance of Win32 subsystem, its associated modules, and the GO-Global Display Driver into the reserved address range.
- Initializes the modules' import address tables, so the modules within the sandbox can call each other. It does this without using the Windows loader.

In this way, the System Extensions Driver creates an isolated, container-like sandbox for the session's kernel modules. Then, whenever a process starts in a session, the System Extensions Driver connects the process to its session's sandbox by loading a modified instance of win32u.dll into the process that routes the process's Win32 system calls to the session's instance of the Win32 subsystem. In addition, the System Extensions Driver installs a custom system call gate in the process's instance of ntdll.dll that redirects select system calls to functions that provide session-private kernel objects and perform other session-specific functions.

These are major modifications to the operating system, made at runtime. Many of these modifications are not possible when HVCI is enabled. As a result, HVCI cannot be enabled on GO-Global Hosts. But while malware that is designed to hook the Win32 subsystem on a typical Windows system would not be blocked from modifying the memory of the Win32 subsystem on a GO-Global host, the malware would not know *when* to install its hooks or *where* to install its hooks. It would not know when to install its hooks because no Windows sessions are created by GO-Global (all processes run in Windows session 0), and it would not know where to install its hooks because GO-Global loads each session's instance of the Win32 subsystem at a unique and unpredictable address.

Furthermore, if the malware attempted to install its hooks in user mode in ntdll.dll or win32u.dll, its hooks would likely be incompatible with GO-Global's system call gates and cause a system crash or prevent sessions from starting. And while system crashes and session start failures are not desirable events, they are preferrable to having data compromised.

Malware could, of course, be designed to explicitly target GO-Global systems. GraphOn mitigates this risk by not releasing debug symbols. This makes it much more difficult to reverse engineer GO-Global than it is to reverse engineer Windows.

In summary, GO-Global does not replace or provide the same protections that HVCI provides, but its architecture greatly reduces the risk that malware that would otherwise be blocked by HVCI will run successfully on GO-Global hosts and allow data passing through the Win32 subsystem to be compromised.

GraphOn Corporation
189 North Main Street, Suite 102 • Concord, NH 03301 USA • sales@graphon.com

[2023.08.22]