

Next Generation Secure Computing Base

Preface

These notes are based upon attendance at the Microsoft WinHEC (Windows Hardware Engineering Conference) conference in New Orleans on May 6-8, 2003. Microsoft formally introduced NGSCB at this conference. There were 18 hours dedicated to NGSCB (Next Generation Secure Computing Base—pronounced *ing-scub*) presentations. Many of those hours were highly repetitive. This repetitiveness did have the desirable side effect of drilling in the core concepts.

Using prototype hardware and software Microsoft provided several live demonstrations of NGSCB enabled applications showing resistance to attack. The specific attack vector shown was a Trojan which allowed keyboard-sniffing, and memory snooping.

I intend to make little reference of any social, political or other authoritarian implications of NGSCB—a good source of information here is Ross Anderson's web page. I will state that Microsoft's approach seems quite logical. The ramifications of technical enforcement of legal policy are quite interesting (if NGSCB actually works and can provide such strong enforcement). The OS under consideration in this paper is some future version of Windows, which is currently named Longhorn.

Overview

Microsoft's goal for the NGSCB is to raise the bar on PC system security. The direct aim is to stop nearly all software, and some limited hardware assisted, threats to a Windows based computer. Microsoft says their vision is "to meet customer's requirements for security, privacy, and data protection". It is not clear that NGSCB is a direct result of customer requirements—however it seems clear that Microsoft knows that they must keep making substantive improvements to Windows system security.

Microsoft also hopes to "revitalize the PC ecosystem by enabling a new generation of hardware and software products". NGSCB will require new hardware (new CPU chips, new motherboards, and new specialized chips to provide the crypto/security functions). More importantly, applications that want take advantage of a higher security level will have to be modified, perhaps even redesigned and rewritten.

Importantly, the new processor architecture will be backwards compatible. The NGSCB hardware will have no effect on existing operating systems and applications. The user and the operating system have to cause the NGSCB environment to be invoked. There is no direct threat to Linux and FreeBSD. And Microsoft says that software that illegally plays protected media will safely continue to play the same protected media

Design Goals – The four fundamentals

- Strong Process Isolation

The protected operating environment isolates a secure area of memory that is used to process data with higher security requirements.

Windows currently provides separation by using ring 0 and ring 3 processor mode isolation. Windows, and other operating systems, also make use of virtual memory segmentation to separate processes from each other and from the operating system. Virtual memory used to be considered hard process isolation but Microsoft makes a remarkable admission that “Using the current memory scheme, only virtual memory protection is achievable, and it is relatively easy for an attacker to add malicious programs to both the operating system and user space memory”.¹

An interesting thought to explore here is that under NGSCB even if Windows is completely compromised the NGSCB components can still provide privacy and integrity.

- Sealed Storage

This storage mechanism uses encryption to help ensure the privacy of NGSCB data that persists on the hard disk of NGSCB-capable computers.

Sealed storage is secure information storage on the local hard disk. Sealed storage is provided through the use of a security support component (SSC)—implemented directly in hardware. This provides the notion of a long-lived confidential binding between an application and its data.

A primary threat addressed is that even if another operating system is booted or if the hard drive is moved to another computer the data should stay confidential.

- Attestation

This occurs when a piece of code digitally signs and attests to a piece of data, helping to confirm to the recipient that the data was constructed by a cryptographically identifiable software stack.

This idea is new and different. Attestation may allow an application to both identify a remote partner application and satisfy a requirement that the remote application has integrity. The integrity is provided by a “cryptographically identifiable software stack”.

Microsoft introduces the idea of an APN—an application private network. Considering an OSI model, current VPN mechanisms, such as IPSEC, protect data up through layer 4. With an APN data is protected layer 7 to layer 7.

An analogue is receiving a phone call from an old and trusted friend. If you believe that your phone is not tapped and you recognize the patterns, intonations, and mannerism of the voice at the other end then you may be assured that your conversation is private and that you are indeed just speaking to your old friend.

- Secure Paths to the User

By encrypting input and output, the system creates a secure path from the keyboard and mouse to trusted applications and from those applications to a region of the computer screen. These secure paths ensure that valuable information remains private and unaltered.

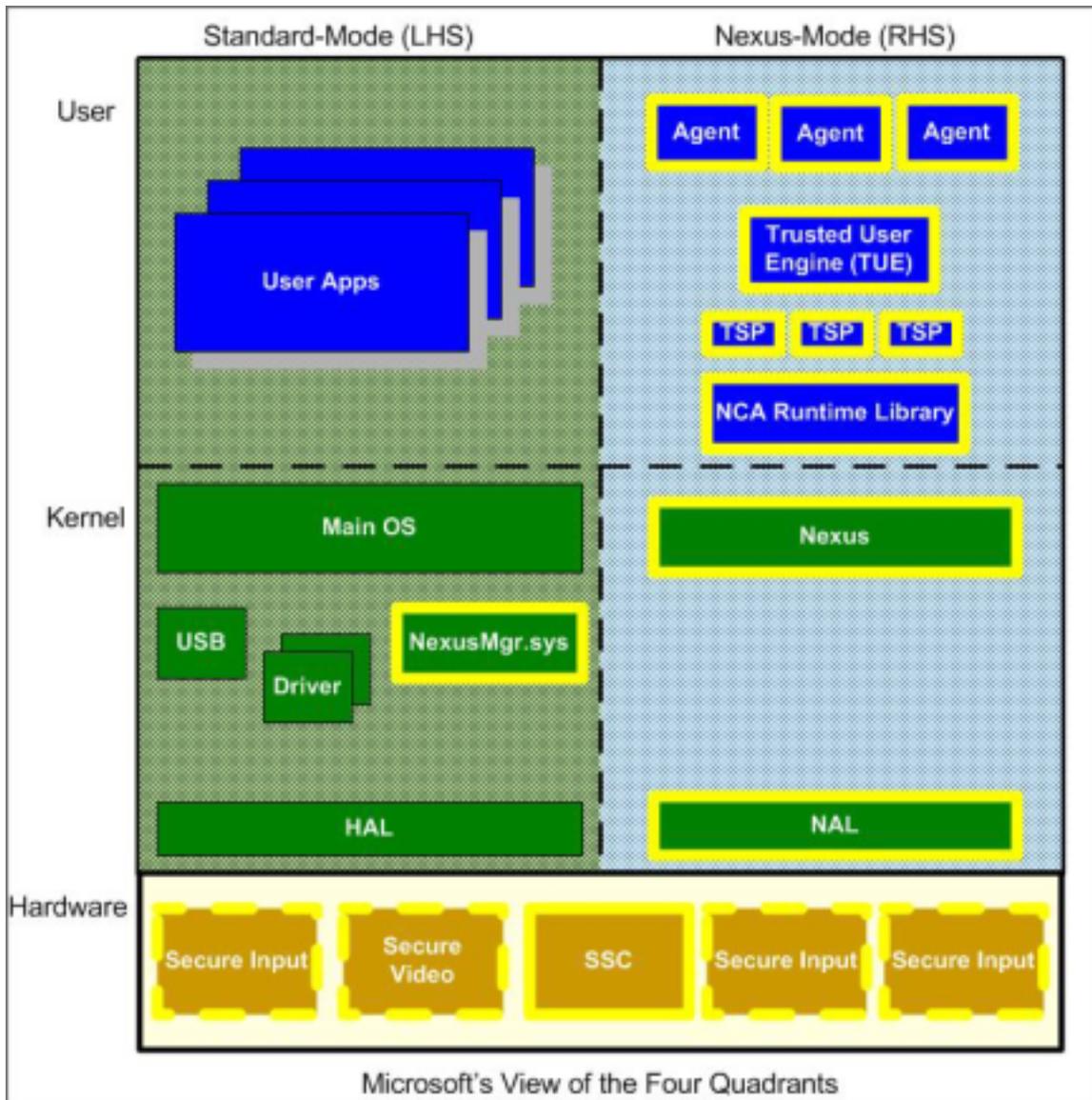
This is not a new idea (see Background below), but this idea has never been widely implemented in a general purpose computer/OS environment. NGSCB secure path specifically means that keyboard input is protected from key press until consumed by a secure mode application (other input devices such as smart card readers and biometric units may be added later). The secure path also means secure output—secure output being a modified graphics adapter—where the adapter card protects against screen scraping and provides a mechanism to denote which area(s) of the screen are secure.

Microsoft emphasizes the difference between the terms secure channel and secure path. Secure channel is a protected channel between two computer applications. Secure path is a protected path between the computer and the user.

Design Implementation – Living in a bi-polar world

The Left Hand Side (LHS) represents PC hardware and architecture as we currently know it. The LHS also represents the future computing model for non-NGSCB enabled OS's. There is NO change if NGSCB is not enabled. The Right Hand Side (RHS) is where all the new security features are implemented. If NGSCB is widely adopted then the term Left Hand Side will immediately connote *insecure* and Right Hand Side will connote *secure*.

If the NGSCB services are desired then there is an addition of one device driver, the NexusMgr.sys, on the LHS. The NexusMgr.sys is responsible for loading the Nexus, and for calling the Nexus for NGSCB provided services. The NexusMgr also provides service and coordination for the Nexus and RHS applications (the RHS has to go left to communicate with the outside world). The services the NexusMgr provides for the RHS are: device driver (I/O), file system (the RHS/Nexus does not directly support file system access), memory management, windows management coordination, and (gulp!) user debugging (that's right, there is RHS code debugging running in the LHS—my notes are that running a debugger breaks security—obviously attestation—I am not sure about sealed storage).



Support of the RHS requires several hardware changes. The CPU must support a mode flag and context switching between LHS and RHS. The CPU must ensure that memory that is marked as trusted can only be accessed from the RHS. The CPU must support nexus initialization. The overall NGSCB chipset must prevent bus mastering devices, including DMA, from accessing trusted memory. The NGSCB motherboard must contain an SSC (System Security Component). This device is also known as the TPM—Trusted Platform Module (a future Trusted Computing Group specification level will formally define an NGSCB capable TPM). The SSC provides functions such as RSA public key operations, AES encryption and decryption, and SHA-1 hash computation. The SSC also stores at least one unique RSA private key and one AES key. The new Intel architecture to support NGSCB is called Le Grande after a town in Oregon (as Intel says they are developing Le Grande to support secure computing, and it would be nice if SW companies would make the small changes needed to support it). The AMD initiative, for now, is called Secure Execution Mode (SEM) and is “2 or 3 years away”. As mentioned

earlier, keyboards and graphics cards must be modified to encrypt data from key press until display upon the video screen to support secure paths.

The security kernel, a mini-OS that runs in the RHS, is known as the nexus. The nexus is supposed to be sized between 100K to 300K lines of code to foster review and provability. This sounds reasonable to me. At one time I was an expert on IBM's Virtual Machine (VM) operating system. The Control Program (CP-which had little user interface, no file system, etc.) of that OS at one time was about 300K lines. I found 300K lines to be understandable and knowable—I am not sure how much higher this number can go while maintaining intuitiveness. The nexus contains no device drivers. The Nexus does not support a file system. All I/O and file operations must be passed to the LHS. The nexus does include services such as memory, process, and thread management. Paging is not supported in the RHS. The nexus is interruptible. When interrupted the nexus must save state and pass the interrupt back to the LHS where the NexusMgr will replay it. To support multiple CPU and SSC-TPM vendors there is a NAL—Nexus Abstraction Layer, similar to the Hardware Abstraction Layer (HAL).

The user will choose which nexus to run. But the nexus will be unconditionally trusted. “Because of the importance of nexus source code, it will be made available for inspection, and so will the procedures designed to assure that the hash of the nexus can be verified against the source code producing it.”²

The Windows OS kernel (LHS) will require less than 200 lines of code to support NGSCB. The NexusMgr provides required services for both LHS and RHS—again the NexusMgr, and all LHS services, are not trusted. The idea is that the LHS has the unmitigated power for a DOS. However the LHS will not be able to affect the confidentiality and integrity of the RHS. The nexus will be able to be loaded and unloaded at any time without affecting the stability of the Windows OS. The LHS will cause the nexus to be loaded. Several new hardware instructions are required. Most importantly there is a function for the hardware to perform an atomic cryptographic hash of the nexus image.

RHS applications are known as nexus computing agents (NCAs). Only “good” applications should be allowed to run in the RHS. Instead of just directly using a hash of a program an XML document known as a *manifest* will represent an NCA. The manifest may directly include a hash of the program or may allow the program to be identified by a public key. The manifest identifies modules to be loaded as a part of the NCA, supplies names to be associated with the NCA, and provides version numbers. The manifest also includes a debug flag—meaning that the NCA supports a debugger. I don't have an understanding of how (or even if) secure debugging could possibly happen.

The Trusted User Interface engine (TUE) supports a limited set of XML based dialogue management; it was also called a mini-MFC (Microsoft Foundation Class). The Trusted Service Providers (TSPs) provide common library function. My notes say they are somehow different from DLLs though.

Background – What’s old is new again

James P. Anderson’s “Computer Security Technology Planning Study”³ in 1972 may be the first paper to describe modern requirements for secure computing. As a measure of this paper’s importance it was later used extensively in the creation of the DOD’s “Trusted Computer System Evaluation Criteria”, the *Orange Book*.⁴ The TCSEC describes the familiar divisions D through C1, C2, B1, B2, B3, to A1 levels of system protection. Windows NT was developed to address the security concerns as specified in level C2. Highlights of C2 are: Discretionary Access Control (DAC from level C1), control of object reuse, and accountability through identification, authentication, and audit.

Anderson’s cornerstone of security is a *reference monitor*:

It is hypothesized that a system secure against internal malicious threat from a programmer can result from employing a reference monitor to validate all references to programs or data according to the access authority of the user on whose behalf the program is executing. In concept, the reference monitor meditates each reference made by each program in execution by checking the proposed access against a list of accesses authorized by that user. The reference monitor concept is implemented as a reference validation mechanism. Accompanying this concept are the operating principles that:

- a. the reference validation mechanism must be tamper proof.
- b. the reference validation mechanism must always be involved.
- c. the reference validation mechanism must be small enough to be tested (exhaustively if necessary).

These principles, vigorously applied, can result in integrating all of the system security controls for a system into one hopefully small portion of the operating system code. If this portion is then implemented correctly (i.e., without any programming flaws), and cannot be altered by any other part/function of the system the security concern of how the rest of the system or any user program is implemented is focused on the access authorizations(s) permitted to the user(s) or programs.

Anderson’s principles are sometimes referenced by the acronym *NEAT*: Non-Bypassable, Evaluate-able, Always Invoked, and Tamper-Proof. In TCSEC, those portions of a system that are relevant to security are referred to as the Trusted Computing Base (TCB). The TCB is now more generally defined as all components (HW, SW, and procedural) that are required for enforcement of security policy.

Most modern operating systems, including Windows NT and above, are architected, or refitted, in accordance with TCSEC and Anderson. The problem lies in the fact that the OS’s have grown big and large segments of code share the same Ring 0 access level. Of particular concern is that thousands of vendor’s device drivers, freely downloadable and installable, share this Ring 0 mode. Therefore there is little effective isolation of the reference monitor.

It is apparent that the RHS and the nexus represent an attempt at a new level of compliance to the reference monitor concept (hardware mode isolation, curtained storage, and a small nexus). Poorly written or directly hostile device drivers will not be able to affect the privacy or integrity comported by the RHS. Unfortunately no additional strengths will be afforded the operating system or current generation applications. Only applications that are modified to utilize NGSCB functionality are improved. See limitations (next section).

Secure Path, another NGSCB fundamental, is also old (if not widely implemented). TCSEC describes a trusted path (consider TCB ~ RHS):

The TCB shall support a trusted communication path between itself and users for use when a positive TCB-to-user connection is required (e.g., login, change subject security level). Communications via this trusted path shall be activated exclusively by a user of the TCB and shall be logically isolated and unmistakable distinguishable from other paths.

Limitations – CI + AA not CIA + AAA

The CIA Triad (Confidentiality, Integrity, Availability) and the Triple-A (Authentication, Authorization, Auditing) are primary goals of computer security. It is important to realize the limitations of the NGSCB architecture, even if it meets its lofty goals. Of CIA, availability may be the hardest objective to meet. Fortunately, in most scenarios, availability is of lesser importance than confidentiality and integrity. With NGSCB, system resource access (CPU scheduling and I/O) are at the discretion of the LHS. So viruses, Trojans, and DOS attacks that affect the LHS (all of the DOS weapons of today) will affect the RHS (again, RHS availability, not confidentiality and integrity). NGSCB will not solve availability/DOS problems.

A goal for system audit—besides providing rich reporting—is to have the audit log repository be immutable (the data can not be modified, appended to, deleted, or renamed). Verifiable integrity of audit logs would be of great importance (and is not generally available currently). The sealed storage feature of NGSCB could guaranty integrity (the data is not changed, appended to), but NGSCB can not guaranty that the audit data is not deleted and can not even guaranty that the data was ever committed to the storage device. In fact, because I/O and all device drivers are on the LHS there is no guaranty that any data on disk is safe (from an availability perspective).

Another limitation of NGSCB will be with rich dialogue. The presentation stack on the RHS will be trim and extremely limited. Graphics and rich GUI will be quite cramped. User dialogue boxes may be Spartan (akin to plain text)—only small portions of an application will move to the right. It appears that in may be quite a while (if ever) until a secure browser can run in the RHS.

Remaining Attack Vectors

DOS is acknowledged. But Microsoft claims that NGSCB is not BOBE (Break Once, Break Everywhere). It is acknowledged that, however difficult, it will be possible to pry the private keys out of the NGSCB hardware. Microsoft says in this event only the machine that has thus been brute forced will be compromised. In other words, high expense and much effort for one machine only.

However, if NGSCB hardware, including the SSC, could be virtualized then an attacker could inspect and modify memory at critical times—with virtualization there is no notion of atomic operations. The nexus could be compromised. This may still appear to be non-BOBE. But what if the virtualization could be portable and possibly replayable? Perhaps full virtualization isn't necessary. If one machine's private keys are available perhaps a spoofed application could act as part of an APN—all in the LHS only. Obviously the key would be revoked. But, the strength and practicality of the revocation process is not yet known. In addition, resistance to APN/attestation man-in-the-middle attacks depends on the strength and practicality of the certificate chaining process—also not yet known.

¹ “Security Model for the Next-Generation Secure Computing Base” Microsoft Whitepaper (May 15, 2003) at <http://www.microsoft.com/ngscb>.

² “NGSCB: Trusted Computing Base and Software Authentication” Microsoft Whitepaper (May 15, 2003) at <http://www.microsoft.com/ngscb>.

³ Anderson, J. P., *Computer Security Technology Planning Study*, ESD-TR-73-51, ESD/AFSC, Hanscom AFB, Bedford, MA (Oct. 1972) (May 27, 2003) at <http://seclab.cs.ucdavis.edu/projects/history/seminal.html>.

⁴ Department of Defense Computer Security Evaluation Center; *Trusted Computer System Evaluation Criteria (Orange Book)*; (1983, 1985) (May 27, 2003) at <http://seclab.cs.ucdavis.edu/projects/history/seminal.html>.