исследование Windows Vista/Server Longhorn

real changes

что нового
http://en.wikipedia.org/wiki/Features_new_to_Windows_Vista

что было исключено по ср. с XP (там же)
http://en.wikipedia.org/wiki/Features_new_to_Windows_Vista

что не было реализовано
WinFS
http://en.wikipedia.org/wiki/WinFS

NGSCB (Next-Generation Secure Computing Base)
http://en.wikipedia.org/wiki/Next-Generation_Secure_Computing_Base

история висты
http://en.wikipedia.org/wiki/Development_of_Windows_Vista

http://www.bluebytesoftware.com/blog/PermaLink,guid,17433c64-f45e-40f7-8772-dedb69ab2190.aspx
* Improved memory manager and processes scheduler. Rewritten many kernel data structures and algorithms. Lookup algorithms now run in constant time, instead of linear time as with previous versions.

NT security is done with DAC: Discretionary Access Control, using ACLs: Access Control Lists. (Vista supports M(andatory)AC too)
→ искать MAC

http://www.securiteam.com/windowsntfocus/5TP0B2KC0K.html
nhttp://www.tinykrnl.org/recon2k6.pdf

Memory Manager

http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/kernel-en.doc
http://go.microsoft.com/fwlink/?LinkId=67468 ; открывает Server 2003 SP1
Significant enhancements were made throughout the memory manager in Windows Vista and Windows Server Longhorn. Changes include:

- /* vista */ Improvements to dynamic system address space, including on-demand allocation of system virtual address space and kernel page table pages, and support for very large registries. Instead of at boot time based registry & configuration information. Region sizes bounded only by VA limitations: Applies to non-paged, paged, session space, mapped views, etc. Kernel page tables allocated on demand, No longer preallocated at system boot, saves (1.5MB on x86 systems, 3MB on PAE systems, 16MB to 2.5GB on 64-bit machines)

- /* Server 2003 SP1, vista enhancement */ Enhanced support for nonuniform memory architecture (NUMA) systems and systems with large pages, including additional device

driver and Microsoft Win32® NUMA application program interfaces (APIs). VirtualAllocExNuma, CreateFileMappingExNuma, MapViewOfFileExNuma. Example: Hewlett-Packard Superdome 1TB gaps between chunks of physical memory

  - http://en.wikipedia.org/wiki/Non-Uniform_Memory_Access
    Non-Uniform Memory Access and Non-Uniform Memory Architecture (NUMA) is a computer memory design used in multiprocessors, where the memory access time depends on the memory location relative to a processor. Under NUMA, a processor can access its own local memory faster than non-local memory, that is, memory local to another processor or memory shared between processors.

- /* vista */ Advanced video model support through a new mapping type called rotate virtual address descriptors (VADs). Dramatically different video architecture in Windows Vista More fully exploits modern GPUs & virtual memory MM provides new mapping type Rotate virtual address descriptors (VADs) Allow video drivers to quickly switch user views from regular application memory into Cached, non-cached, write combined AGP or video RAM mappings Allows video architecture to use GPU to rotate unneeded clients in and out on demand First time Windows-based OS has supported fully pageable mappings w/ arbitrary cache attributes
- /* vista */ I/O and section access improvements, including pervasive prefetch-style clustering for all types of page faults and system cache read-ahead. Significant changes in pagefile writing, Larger clusters up to 4GB, Align near neighbors, Sort by virtual address (VA), Reduced fragmentation, Improved reads, Cache manager read ahead size limitations in thread structure removed, Improved synchronization between cache manager and memory manager data flushing to maximize filesystem/disk throughput and efficiency. Mapped file writing and file flushing performance increases, Support for writes of any size up to 4GB instead of previous 64k limit per write, Multiple asynchronous flushes can be issued, both internally and by the caller, to satisfy a single call, Pagefile fragmentation improvements, On dirty bit faults, we use interlocked queuing operation to free the pagefile space of the corresponding page, Avoids PFN lock acquisitions, Reduces needless pagefile fragmentation, Elimination of pagefile writes and potential subsequent re-reads of completely zero pages,Check pages at trim time to see if they are all zero,Optimization used to make this nearly free,User virtual address used to check for the first and last ULONG_PTR being zero; if they both are, then,After the page is trimmed, and TLB invalidated, a kernel mapping used to make the final check of the entire page,Avoids needless scans & TLB flushes,We've measured over 90% success rate with this algorithm, Access to large section performance increases,A subsection is the name of the data structure used to describe on-disk file spans for sections ,The subsection structure was converted,From a singly linked (i.e., linear walk required),To a balanced AVL tree,Enables huge performance gain for sections mapping large files,User mappings & flushes, system cache mappings, flushes & purges, section-based backups, etc,Mapped page writer does flushing based on a sweep hand,Data is written out much sooner than the prior 5 minute "flush everything" model,
- /* vista */ I/O Reduced data loss in the face of crashes, Flush all modified data to its backing store (local & remote) if we are going to bugcheck due to a failed inpage,Only failed inpages of kernel and/or drivers are fatal,Failed inpages of user process code/data merely results in an inpage exception being handed to the application
- General performance improvements, including translation buffer optimizations and improvements to internal data structures and algorithmic performance.
- /* vista */ Microsoft Terminal Services improvements, including new Terminal Services session objects.
- /* vista */ Robustness and diagnosability improvements, including reduced data loss from failed in-page operations.

Developers who write NUMA-aware applications or drivers that run on NUMA systems may want to modify their code to take advantage of new and expanded NUMA-related APIs**.** Other memory management changes should be transparent to both applications and drivers.

Heap Manager

Performance. The heap manager takes advantage of recent changes in the hardware architectures and the growing popularity of 64-bit and symmetric multiprocessing (SMP) systems to provide better performance and scalability on these systems.

In particular, previous versions of Windows support the low-fragmentation heap (LFH) that provides significant scalability improvements on multiprocessor systems, but applications could not take advantage of these improvements in a transparent manner. The heap manager in Windows Vista introduces the concept of automatic tuning, which detects allocation patterns at run time, automatically chooses the right mode, and provides lazy initialization whenever applicable to improve overall system and application performance.

Other significant improvements include rearchitecture of internal algorithms and data structures that handle segment management. As a result, the heap manager provides better fragmentation management, better scalability, and lower overhead for large heaps, especially for 64-bit server applications. In addition, the efficiency of lookup algorithms has been improved from O(n) to O(1).

Security and reliability. Security has been a priority for the heap manager in earlier versions of Windows, but the attack landscape has changed considerably. The heap manager in Windows Vista has significant improvements not only to mitigate against current exploits and attack vectors, but also to proactively fix potential target areas for future attacks.

Some of the changes in this area include block metadata encoding, integrity checks on block headers, and random heap rebasing. In addition, the heap manager provides improved and early detection of heap corruptions and the ability for applications to terminate when heap corruption occurs, thereby deterring "brute force" attacks that exploit a vulnerability.

Registry

Changes to the Windows registry include several significant new innovations:

- **Transaction support for registry operations**. Transaction support allows the registry to provide “all or none” semantics for a group of registry operations. The registry can also collaborate with other resource managers in the system such as transactional NTFS (TxF) to enable transactions that span file system and registry operations.
- **Optimizations to reduce the possibility of registry corruption**. One such mitigation mechanism protects memory pages from accidental corruption by other components by marking pages as read only. When the registry must write data to one of these pages, its access mode is changed from read-only to read/write. After the write operation is completed, the page's access mode reverts to read-only.
- **Registry filtering that is consistent with the file system filtering model**. The registry filtering model was enhanced to be consistent with the file system filtering model. The new model adds support for redirecting calls and modifying parameters in addition to the existing support for monitoring and blocking calls.

  Similar to the file system filtering model, the new registry filtering model allows filters to register at specific positions on the filtering stack by introducing the concept of altitudes for filter registrations. Developers of registry filter drivers for Windows Vista should register their minifilter altitudes with Microsoft at the location listed in "Windows Registry Allocation" at the end of this section.
- **Registry virtualization support**. Registry virtualization support enables legacy applications to run in non-administrator accounts. Registry virtualization isolates write operations that have a global impact to a per-user location. This redirection of writes is transparent to applications and is applicable only to operations on keys in the software hive (**HKLM\Software)**. All other keys are unaffected by virtualization.
- Minifilter Altitude Allocation http://go.microsoft.com/fwlink/?LinkId=66682

Services Model

- **Delayed start type for services**. To address the problem of the growing number of auto-start services and their negative impact on boot performance, there is a new start type for services that do not need to start early in boot: *delayed start*. This new start type improves boot performance by starting services shortly after boot and yet retains the "unattended start" behavior.
- **Platform support for sandboxing and protection from user-mode attacks**. In an effort to reduce the attack surface exposed by services, the service control manager (SCM) provides platform support for sandboxing services and running them with least privilege by using the Windows service hardening infrastructure. In addition, services are also protected from user interface attacks by running them in Session 0 and isolating this session from other user sessions.
- **Manageability improvements**. The service model provides a new notification API to track service state changes. These notifications work both locally and remotely, and remove the requirement for services or clients to use polling loops to monitor state changes. In addition, support for failure actions has been extended to provide recovery actions for non-crash failures.
- http://go.microsoft.com/fwlink/?LinkId=66644
- http://go.microsoft.com/fwlink/?LinkId=67470
- http://go.microsoft.com/fwlink/?LinkId=67471

Windows Hardware Error Architecture

According to Microsoft's Online Crash Analysis (OCA) data, approximately 7 to 10 percent of all reported crashes are due to some type of hardware error, such as processor, memory, or cache errors. Operating systems earlier than Windows Vista and Windows Server Longhorn do not expose sufficient error information to determine the root cause of these types of crashes from OCA data. Issues include the lack of a common error record format, disparate sources of errors, disparate signaling and reporting mechanisms, the fact that existing error management implementations are proprietary, and other factors.

Windows Server Longhorn implements WHEA, a common operating system/hardware error-handling infrastructure that builds on PCI Express Advanced Error Reporting (AER). WHEA provides rich hardware error data and helps reduce mean time to recovery through the following mechanisms:

- A generic mechanism for error source discovery.
- A common error-record format for operating system and hardware errors.
- A common error-handling flow for operating system and hardware errors.
- A persistence mechanism for operating system error records.
- A common hardware error-eventing model based on Event Tracing for Windows (ETW) for management applications.
- http://download.microsoft.com/download/9/8/f/98f3fe47-dfc3-4e74-92a3-088782200fe7/TWAR05009_WinHEC05.ppt
- http://download.microsoft.com/download/9/8/f/98f3fe47-dfc3-4e74-92a3-088782200fe7/TWAR05008_WinHEC05.ppt

http://en.wikipedia.org/wiki/Features_new_to_Windows_Vista

* Windows Vista features prioritized I/O which will allow developers to set application I/O priorities for read/write disk operations, similar to how currently application processes/threads can be assigned CPU priorities. [48] I/O has been enhanced with I/O asynchronous cancellation and I/O scheduling based on thread priority. Background applications running in low priority I/O do not disturb foreground

applications. Applications like Windows Defender, Automatic Disk Defragmenter and Windows Desktop Search (during indexing) already use this feature. Windows Media Player 11 also supports this technology to offer glitch-free multimedia playback.

https://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/synchronization_functions.asp
* **Reader/writer locks**. The kernel32 function InitializeSRWLock takes a pointer to a SRWLOCK structure, just like InitializeCriticalSection, and initializes it. AcquireSRWLockExclusive and AcquireSRWLockShared acquire the lock in the specific mode and ReleaseSRWLockXXX releases the lock. This is a "slim" RW lock, meaning it's actually comprised of a pointer-sized value, and is ultra-fast and lightweight, much like existing Win32 CRITICAL_SECTIONs. It should be about the cost of a single interlocked operation to acquire. E.g.

```
SRWLOCK rwLock;
InitializeSRWLock(&rwLock);
AcquireSRWLockShared(&rwLock);
// ... shared operations ...
ReleaseSRWLockShared(&rwLock);
```

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/condition_variables.asp
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/using_condition_variables.asp
* **Condition variables**. These integrate with RW locks and critical sections, enabling you to do essentially what you can already do with Monitor.Wait/Pulse/PulseAll. InitializeConditionVariable takes a pointer to a CONDITION_VARIABLE and initializes it. SleepConditionVariableCS and SleepConditionVariableSRW release the specified lock (either CRITICAL_SECTION or SRWLOCK) and wait on the condition variable as an atomic action. When the thread wakes up again, it immediately attempts to acquire the lock it released during the wait. WakeConditionVariable wakes a single waiter for the target condition and WakeAllConditionVariable wakes all waiters, much like Pulse and PulseAll. E.g.

```
Buffer * pBuffer = ...;
PCRITICAL_SECTION pCsBufferLock = ...;
PCONDITION_VARIABLE pCvBufferHasItem = ...;

// Producer code:
EnterCriticalSection(pCsBufferLock);
while (pBuffer->Count == 0) {
   SleepConditionVariableCS(pCvBufferHasItem, pCsBufferLock, INFINITE);
}
// process item...
LeaveCriticalSection(pCsBufferLock);

// Consumer code:
EnterCriticalSection(pCsBufferLock);
pBuffer->Put(NewItem());
LeaveCriticalSection(pCsBufferLock);
WakeAllConditionVariable(pCvBufferHasItem);
```

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/one-time_initialization.asp
* Lazy/one-time initialization. This allows you to write lazy allocation without fully understanding memory models and that sort of nonsense. The new APIs in kernel32, InitXXX, support both synchronous and asynchronous initialization. These have some amount of overhead for the initialization case due to the use of a callback, but in general this will be fast enough for most lazy initialization and much less error prone. Herb Sutter has proposed a similar construct for the VC++ language, and to be honest I wish we had this built-in to C# too

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/thread_pools.asp
* **An overhauled thread pool API**. The Windows kernel team has actually rewritten the thread pool from the ground up for this release. Their APIs now support creating multiple pools per process, waiting for queues to drain or a specific work item to complete, cancellation of work, cancellation of

IO, and new cleanup functionality, including automatically releasing locks. It also has substantial performance improvements due to a large portion of the code residing in user-mode instead of kernel-mode.

 # http://msdn.microsoft.com/msdnmag/issues/06/04/Deadlocks/default.aspx
*** Application deadlock detection.** This is separate from the existing Driver Verifier ability to diagnose deadlocks in drivers. This capability integrates with all synchronization mechanisms, from CRITICAL_SECTION to SRWLOCK to Mutex, and keys off of any calls to XXXWaitForYYYObjectZZZ. Unfortunately, I think this is new to the latest Vista SDK, and thus there isn't a lot of information available publicly. This could probably make a good future blog post if there's interest.

*** A new file-based disk image format called Microsoft Windows Imaging Format (WIM), which can be mounted as a partition, or booted from. An associated tool called ImageX provides facilities to create and maintain these image files.**

* Process creation overhead is reduced by improvements to DLL address-resolving schemes.

* Windows Vista introduces a **Protected Process,** which differ from usual processes in the sense that other processes cannot manipulate the state of such processes, nor can threads from other processes be introduced in these. Such processes have enhanced access to DRM-functions of Windows Vista. However, currently, only the applications using Protected Video Path can create such processes.

* **Data Redirection:** Also known as data virtualization, this virtualizes the registry and certain parts of the file system for applications running in the protected user context. Reads and writes in the HKLM\Software section of the Registry by user-mode applications while running as a standard user, as well as to folders such as "Program Files", are "redirected" to the user's profile. The process of reading and writing on the profile data and not on the application-intended location is completely transparent to the application

* The new Kernel Transaction Manager enables atomic transaction operations across different types of objects, most significantly file system and registry operations.

* The NTLDR boot loader has been replaced by a more flexible system, with NTLDR's functionality split between two new components: winload.exe and Windows Boot Manage

Windows Vista automatically tunes up the heap layout for improved fragmentation management.
Lazy initialization of heap initializes only when required, to improve performance.
The Windows Vista memory manager does not have a 64 kb read-ahead cache limitation unlike previous versions of Windows and can thus improve file system performance dramatically.

* A new user-mode driver model called the User Mode Driver Framework, which is part of Microsoft's new driver model, Windows Driver Foundation. User-Mode Drivers in Windows Vista are not able to directly access the kernel but use it through a dedicated API. If an error occurs the new framework allows for an immediate restart of the driver and does not impact the system. A user-mode driver would typically be used for devices which plug into a USB or Firewire bus, such as digital cameras, PDAs and mass storage devices, as well as "non-hardware" drivers, such as filter drivers.
The User Mode Driver Framework is a device-driver development platform first introduced with Microsoft's Windows Vista operating system, and is also available for **Windows XP**. It facilitates the creation of drivers for certain classes of devices, where the driver operates entirely in user-land

Cycle Time Counter
Prior to Vista, the kernel accounted for CPU time based on the interval clock timer which had a resolution of between 10-15ms. This timing interval was not always fair or accurate since threads where charged for interrupts that occurred while they were running and a thread might not get a turn to execute or could get up to three turns to execute.
Vista changes this timing mechanism by reading the Time Stamp Counter (TSC) at each context switch. This allows the CPU to charge the thread with the actual CPU cycles consumed and does not

charge the thread for interrupt time. This allows for a more accurate time accounting model and means that threads get at least one turn and can get at most one turn plus one tick.
This new time accounting is handled by the following API functions:
QueryThreadCycleTime
QueryProcessCycleTime
QueryIdleProcessorCycleTime
Multimedia Class Scheduler
The multimedia class scheduler is a new service that boost the thread priorities of multimedia applications to support glitch-free audio and video streaming. This service runs in a Svchost process and is implemented in the Mmcss.dll used by Media Player 11.
In order to make use of this service, threads must declare themselves as multimedia with the following API functions:
AvSetMmThreadCharacteristics
AvSetMmThreadPriority
Multimedia threads are boosted into real-time for 80% of a task's clock rate (default is 1ms). If they consume all of that time, they are lowered so other threads can run. The percentage can be reconfigured through the following registry key:
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Multimedia\SystemProfile

New Synchronization APIs
Vista also introduces some new kernel synchronization methods such as:
Condition variables (*ConditionVariable*)
Reader/writer locks (*SRWLock*)
One time initialization (Initonce*)
Extended version of Create object APIs to allow specification of desired access
Event, Mutex, Semaphore, Waitable Timer
Protected Processes
Protected processes prevent unauthorized access to media content and can only be created through the new Protected Media Path APIs that are part of Media Foundation. This is used to enforce a secure path to the output devices and allows only signed images to be mapped into a secure process. Images must be signed by Microsoft and third-party codes must be signed with a Windows Media DRM certificate. Standard processes (even with debugging rights) have limited access to protected processes.

http://blogs.msdn.com/windowsvistasecurity/archive/2006/08/11/695993.aspx
J>>Один мой знакомый думает, что Windows Vista это тема оформления для XP, просто потому, что видел тему с таким названием. Он не верет мне, что это ОС.
A>+1 не верю, что это ОС
прально, это скорее ПЧЕЛ..

Народ, гуру, откликнитесь...
Ну скажите мне, почему у меня на моем AMD Athlon(tm) 64 X2 Dual Core Processor 3800+ (!) под вистой наблюдаются тормоза?
Открываю окно (тот-же проводник) и задержка прямо чувствуется. Не то чтобы прямо тормозит неподецки, а всетаки медленнее икспы.
ФФ и Опера тоже заметно медленнее пускаются.
Другое управление памятью? по другому взимодействие?
Или дотнет всетаки накладывает свои лапы на камень?

обработка структурных исключений, теперь они не в стеке, а в секции .pdata

http://blogs.msdn.com/windowsvistasecurity/archive/2006/08/11/695993.aspx

Key Driver Concepts
http://www.microsoft.com/whdc/driver/kernel/default.mspx

Kernel Patch Protection

Kernel patch protection was first implemented in Windows Server 2003 SP1 x64–based systems and Windows XP for x64–based systems. No significant changes have been made to kernel patch protection in Windows Vista and Windows Server Longhorn. However, because this security feature is still relatively new, it is discussed in this paper to increase awareness of the feature and its potential impact on driver and application developers.

Protected Processes

The Windows Vista operating system introduces a new type of process, called a *protected process*, that enhances support for digital rights management functionality in Windows Vista and Windows Longhorn Server. These protected processes exist alongside typical processes in Windows Vista.

Which Applications Can Create a Protected Process. Currently only the Windows Protected Media Path can create protected processes.

Process Security and Access Rights http://go.microsoft.com/fwlink/?LinkId=67480
Protected Processes http://go.microsoft.com/fwlink/?LinkId=69749

Dynamic-Link Library Loader

The user-mode dynamic-link library (DLL) loader plays a fundamental role in process and thread creation. Every time a process or thread is created, the loader is invoked to complete the following tasks that are related to DLLs and that are required by the process or thread in question:

- Determine appropriate DLL load order and resolve dependencies
- Load all required DLLs
- Unload DLLs when they are no longer needed

The Windows Vista DLL loader has undergone a number of improvements, including:

- **Improved process creation time performance**. A significant portion of process creation time is spent resolving DLL dependencies and processing DLL imports.

  Improvements to import processing algorithms have reduced processing time by an impressive factor of 1000x in some scenarios (from ~800,000 cycles to ~800 cycles). End-to-end process creation times were reduced by as much as 10 percent for some of the more DLL-heavy applications.

- **Fewer reboots resulting from system DLL servicing**. In earlier versions of Windows operating systems, before Windows Vista, whenever a system or core DLL was updated by using a service pack, security patch, or similar mechanism, a reboot was required. The reboot was required because no mechanism could identify which system services had loaded the DLL in question. Therefore, the entire system had to be rebooted to ensure that the earlier version of the DLL was unloaded and the updated DLL was loaded instead.

  Enhanced bookkeeping techniques in the Windows Vista loader allow the system to maintain a complete list of services that have loaded a particular DLL. When that DLL is updated, the system can identify and restart only the specific services that are using the DLL to be updated, instead of rebooting the entire system. This change helps to decrease downtime, thus increasing the availability of the system even while it is being upgraded or serviced.

  Note that some system DLLs, including NTDLL.dll and kernel32.dll, still require a reboot when they are being serviced**.**

Best Practices for Creating DLLs
http://go.microsoft.com/fwlink/?LinkId=69750

Boot Environment Enhancements

Microsoft has completely reengineered the boot environment for Windows Vista to address the increasing complexity and diversity of modern hardware and firmware. A key aspect of this reengineering is a new firmware-independent data store called boot configuration data (BCD).

BCD is designed to handle boot environment data for any type of system. It provides access to the information and applications that Windows Vista and later versions of Windows use to load the operating system or run boot applications such as memory diagnostics. Some key characteristics include:

- BCD provides clean and intuitive structured storage for boot settings.
- BCD abstracts the underlying data store, making BCD independent of the underlying firmware or processor architecture. BCD currently handles both PC/AT and EFI systems, and can be readily extended to accommodate future improvements in hardware and firmware.
- BCD is available at run time as well as during the boot process.
- BCD provides improved security over boot.ini. It allows secure lockdown of the storage format and flexible assignment of the rights that are required for changing boot settings. BCD is stored in a binary format and cannot be edited directly.
- BCD is designed to handle systems with multiple versions and configurations of Windows, including versions earlier than Windows Vista.
- BCD is the only boot data store that is required for Windows Vista and later versions of Windows. BCD supports the loading of earlier versions of Windows, but they are ultimately loaded by Ntldr.exe and must still store their boot options in a boot.ini file.

Instead, users can interact with BCD through several tools. Developers can programmatically manipulate a BCD store through the BCD Windows Management Instrumentation (WMI) provider that can be used for both local and remote management of BCD stores independent of the underlying firmware.

Dynamic Hardware Partitioning

Dynamic hardware partitioning allows systems to be partitioned at a hardware level to run multiple operating systems instances on a single server. In addition, on dynamic hardware partitioning–capable systems, I/O devices, memory, and processors can be dynamically added and removed from, to, and between partitions without powering down the system.

Windows Server 2003 already supports hot add of memory on x86-based, x64-based, and Itanium-based systems.

For compliant hardware platforms, Windows Server Longhorn supports hot add of processors and memory plus hot replace of processors and memory on x64-based and Itanium-based systems. Hot add of I/O host bridges is also planned.

Power Management

Windows Vista and Windows Server Longhorn continue to advance and adopt power management technologies with a rich set of new features and capabilities:

- Simplified power policies that make it easier for users to match usage patterns to the appropriate power policy.
- A new hybrid sleep state, which combines aspects of standby (suspend to RAM) and hibernate (suspend to disk) to protect the user’s data while in standby.

- Windows Direct Experience, which provides support for starting a PC directly to a specific media experience (for example, to launch directly into a media player when a "media" button is pushed).
- A new "away mode" that supports media-oriented PC usage models.
- Reliable power management transitions.
- Extensive diagnostic tracing and error reporting infrastructure.
- Comprehensive group policy control for power management settings.
- Improved battery meter that provides simple, quick access to basic power management settings.
- Significant improvements to the overall power management user interface and user experience.

ACPI

Windows Vista and Windows Server Longhorn continue to advance support for the ACPI specification, including selected features from ACPI 3.0:

- Complete support for ACPI 2.0 processor performance objects.
- ACPI 3.0 processor domain dependency and throttling objects.
- Operating System Capabilities (_OSC) method invocation to facilitate transitions to native support of PCI-E features.
- ACPI General Purpose Event (GPE) block device support.
- New firmware tables API.

ACPI / Power Management – Architecture and Driver Support
http://go.microsoft.com/fwlink/?linkid=67487

PCI and PCI Express

Windows Vista and Windows Server Longhorn support a significant number of features in the PCI Express 1.1 specification, including:

- Extended configuration space.
- Segments.
- PCI Express registers, including capability registers, save and restore of configuration settings, and BIOS configurations.
- Device serial numbers.
- PCI Express hierarchy.
- Message-signaled interrupts (MSI and MSI-X).
- Hot plug.
- Native power management events.
- Active-state power management.
- Advanced Error Reporting (AER).

System manufacturers, hardware vendors, and driver developers should be familiar with these features as described in the PCI Express 1.1 specification and should design their products to comply with that specification.

new api

http://www.rsdn.ru/Forum/?mid=2108722
Т.е. как я понимаю NT native апликуху вы успешно собрали и прописали ее на запуск в процессе загрузки и она успешно грузится? Теперь нужно из natvive апликухи запустить другую native апликуху. Ну вы попали . Нормального CreateProcess'a (как в kernel32) в ntdll.dll нету, Для запуска другого native апликейшна вам в теории необходимо будет:
1) открыть файл той апликухи с SYNCHRONIZE|FILE_EXECUTE правами
2) создать на него секцию с помощью ZwCreateSection(... SEC_IMAGE)
3) создать процесс (именно _процесс_): ZwCreateProcess
4) через RtlCreateProcessParameters создать блок параметров инициализации для дочернего процесса
5) выделить в дочернем процессе память, записать туда блок параметров и проставить в Peb->ProcessParameters дочернего процесса адрес на блок параметров
6) создать стек и контекст для первого потока дочернего процесса
7) создать сам поток (ZwCreateThread)
8) далее для win32 процесса надо регистрировать его в csrss, но для native на этапе загрузке этого делать не надо
9) одеть цаку и радоваться

PS сие повествование основано на общеизвестных исходниках
PPS в Vista появился сисколл NtCreateUserProcess. Похоже им надоел этот геморрой в юзерспейсе и они перепихнули его kernel-team'y
PPPS возможно проще будет банально загрузить необходимый native exe'шник себе в адресное пространство через LdrLoadDll, настроить ему импорты и отправить на исполнение EntryPoint.

обработка структурных исключений

http://msdn.microsoft.com/msdnmag/issues/06/05/x64/default.aspx
In contrast to the Win32 exception handling, Win64 (both x64 and Itanium versions) uses table-based exception handling. No linked list of try data blocks is built on the stack. Instead, each Win64 executable contains a runtime function table. Each function table entry contains both the starting and ending address for the function, as well as the location of a rich set of data about exception-handling code in the function and the function's stack frame layout. See the IMAGE_RUNTIME_FUNCTION_ENTRY structure in WINNT.H and in the x64 SDK for the nitty-gritty on these structures.

http://blogs.msdn.com/windowsvistasecurity/archive/2006/08/11/695993.aspx

DMR

http://www.rsdn.ru/Forum/?mid=1843064
От: Maxim S. Shatskih mvp
Дата: 11.04.06 21:38
Оценка: 4 (1)
K>Интересно узнать отношение народа к подобной политике Мелкософтовцев ...

Отвратительное отношение практически у всех DDK MVP, но вопрос тут очень высокого уровня (возможно, выше, чем уровень руководства девелопмента в MS), и речь не о борьбе с малварью, а скорее о DRM.

Некоторые американские DDK MVP использовали свои связи в MS, чтобы на это повлиять, и получили ответы в духе "это политика очень высокого начальства", скорее всего, связанная с DRM. Вопросы борьбы с малварью на такой уровень обычно не поднимаются.

Что более всего неприятно в этой заварухе:

- требования сертификатов от _одного лишь Verisign_, нет выбора CA, причем сертификат обязан быть корпоративным (и, как я понимаю, американской корпорации).
- отсутствие возможности отключить это на production машине (даже через зад, с потерей функционала или с какими-нить вылезалками) с целью установки своего драйвера для какого-то лабораторного железа. Скажем, проверку WinQual можно отключить. Это - нет.
- помимо подписи, там еще и какие-то ограничения на device classes упоминались. Это уже совсем мерзко. Это уже не требование подписывать свой софт, а прямое указание - вот такой софт вы имеете право писать, а такой - нет.

Digital Signatures

download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/kernel-en.doc

::related to Server 2003 x64, XP Professional x64, Vista
This protection has been for the first time implemented in Vista Beta 2 build 5384

…. ever since the release of Windows 98 Digital signatures allow an administrator or an end user who is installing Windows-based software to know whether a legitimate publisher has provided the software package.
Windows Vista relies on digital signatures on kernel-mode code to increase the safety and stability of the Microsoft Windows platform and enable new customer experiences with next generation premium content:

- Drivers must be signed for devices that stream protected content. This includes audio drivers that use Protected User Mode Audio (PUMA) and Protected Audio Path (PAP), and video device drivers that handle protected video path-output protection management (PVP-OPM) commands.
- Unsigned kernel-mode software does not load and does not run on x64-based systems.

  Even users with administrator privileges cannot load unsigned kernel-mode code on x64-based systems. This applies for any software module that loads in kernel mode, including device drivers, filter drivers, and kernel services.

The mandatory kernel-mode code signing policy applies to all kernel-mode software on x64-based systems that are running Windows Vista. However, Microsoft encourages publishers to digitally sign all software, including device drivers for both 32-bit and 64-bit platforms. Windows Vista performs kernel-mode signature verification on x86-based systems to support protected media content. Kernel-mode driver signatures are not mandatory for 32-bit systems.

Digital Signatures aims

- защита от малвари (прикрытие);
- digital rights management (DRM);
- контроль над сторонними разработчиками;

The PMP and Output Protection

Protected Media Path (PMP), Protected User-Mode Audio (PUMA), Protected Video Path (PVP), protected video path-output protection management (PVP_OPM), Secure Audio Path (SAP)

This section provides a brief summary of key PMP concepts. For further information, see the white paper titled *Output Content Protection*.

The PMP consists of four primary components, MIG, PVP-OPM, PVP-UAB, and PUMA:

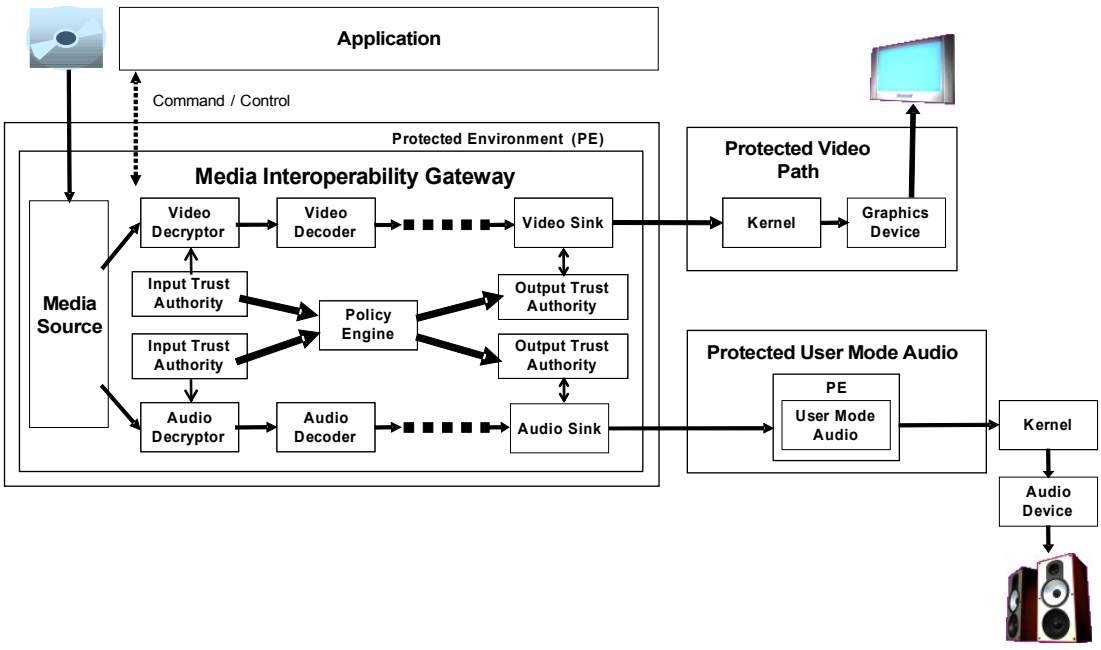
- MIG provides content protection for Media Foundation applications. It is an extensible platform for sourcing, sinking, and manipulating protected media content. MIG governs policy usage and runs media in a separate process to ensure that media content is used only in a way that is consistent with the intent of the content provider.
- PVP-OPM ensures that a PC's integrated graphics adapter outputs have the protection that is required under license agreement with content owners. It provides reliable control

of output protection schemes such as high-bandwidth digital content protection (HDCP), Macrovision, and Copy Generation Management System-Analog (CGMS-A).

- PVP-UAB encrypts premium content as it passes over the PCI Express (PCIe) bus to a discrete graphics adapter. This encryption is required when a content owner's policy regards the PCIe bus as a user-accessible bus.
- PUMA provides a safer environment for audio playback, as well as checking that the enabled outputs are consistent with what the premium content provider allows. PUMA includes the same level of audio output protection management that SAP provided in Windows XP, but it is handled in a completely different way and takes advantage of the PE.

Manufacturers of graphics adapters must implement the required protection mechanisms on card outputs and must ensure that the associated drivers have robust control of those outputs. Manufacturers must sign a PVP-OPM or PVP-UAB license agreement to receive a PVP certificate, which must be embedded in their drivers. Without the embedded PVP certificate, Windows Vista is not allowed to pass premium content to the driver.

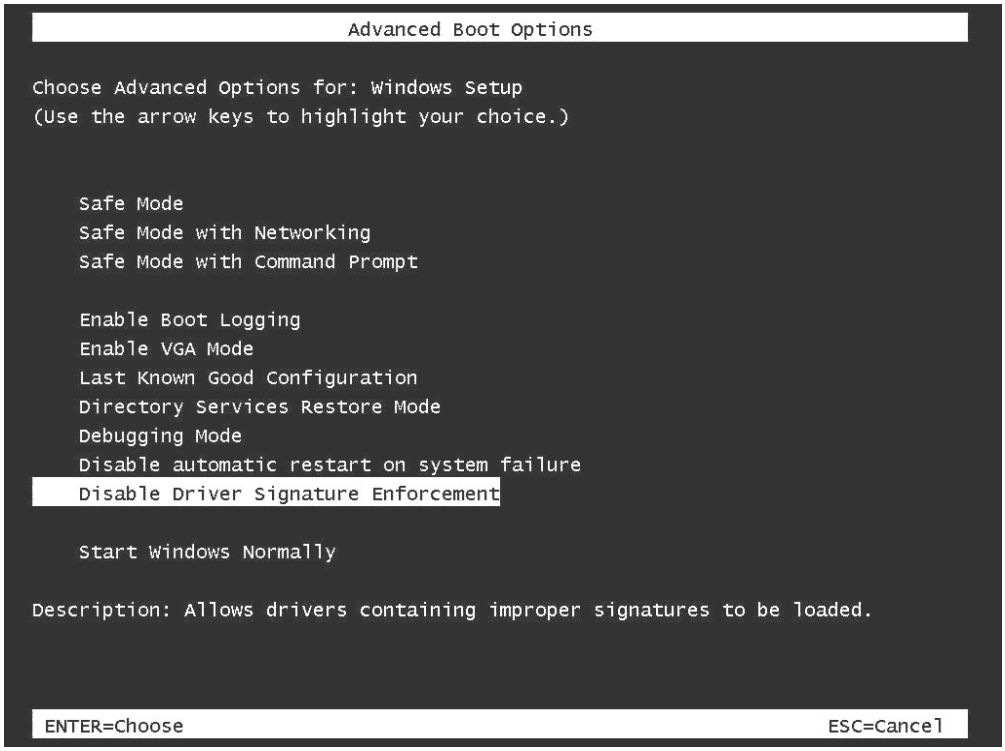
The following figure provides a quick summary of how components that are discussed in this paper interact in Windows Vista.



How to Disable Signature Enforcement during Development

During the early stages of development, developers can disable enforcement in Windows so that driver signing is not necessary. The following options are available for developers to disable kernel mode code signing enforcement temporarily so that Windows Vista will load an unsigned driver.

- **Attaching a kernel debugger.** Attaching an active kernel debugger to the target computer disables the enforcement of kernel mode signatures in Windows Vista and allows the driver to load.
- **Using the F8 option.** An F8 Advanced Boot Option introduced with Windows Vista —"Disable Driver Signature Enforcement"—is available to disable the kernel-signing enforcement only for the current boot session. This setting does not persist across boot sessions.



- **Setting the boot configuration.** A boot configuration setting is available in the Windows Vista Beta2 release that disables the enforcement of kernel mode signatures to be persisted across boot sessions.

  Windows Vista includes a command-line tool, BCDedit, which can be used to set the option in Windows Vista Beta2 to disable signature checks. To use BCDedit, the user must be a member of the Administrators group on the system and run the command from an elevated command prompt. An elevated command prompt can be launched by creating a desktop shortcut to cmd.exe, and then using right-click and “Run as administrator”.

  The following shows an example of running BDCedit at the command prompt:

```
// Disable enforcement - no signing checks
Bcdedit.exe -set nointegritychecks ON

// Enable enforcement - signing checks apply
Bcdedit.exe -set nointegritychecks OFF

// Disabling integrity check on an alternate OS
// specified by a GUID for the system ID
Bcdedit.exe -set {4518fd64-05f1-11da-b13e-00306e386aee} nointegritychecks
ON
```

Note: The Bcdedit option to disable integrity checks is only available for loading unsigned drivers on the Windows Vista Beta2 release. For more information, see the BCD Editor FAQ on the MSDN Web site.

How to use Test Signing

Test signing provides additional options to development organizations for incorporating kernel mode code signing for pre-release software that is not ready for publication. Test signing allows the use of “test” code signing certificates to sign drivers that will load on Windows Vista when the Windows Vista boot configuration setting allows test signatures.

Test signing may be appropriate to use in the following scenarios:

- Development teams need to test pre-release versions of a driver on test systems where it is not practical to attach a kernel debugger.
- Automated testing of kernel mode software make it impractical to use the F8 Advanced Boot Option to temporarily disable driver signature enforcement on every machine boot cycle.

Test signing allows developers to sign pre-release versions of kernel mode binaries in such a way that Windows Vista can verify and load the signed driver. Test signing involves the following differences from normal production or release signing:

- Certificates used for test signing can be generated using the Makecert.exe code signing tool, or issued by an Enterprise CA, instead of using a SPC issued by a commercial CA.
- The Windows Vista boot configuration option to enable Test Signing must be enabled on the Windows Vista system that will load the test signed driver.

Development organizations can setup an enterprise PKI and issue their own test code signing certificates to use for test signing. When Windows Vista enables Test Signing, verification of the digital signature on the driver binary will accept certificates issued by *any* CA or Issuing authority. Test signing verifies the driver image is signed, but certificate path validation performed in kernel mode does not require the issuer to be configured as a trusted root authority. This allows organizations to use individual signatures on test binaries, based on the credentials issued for code signing within the organization. Microsoft recommends this form of deployment for test signing within the Kernel Mode Code Signing.

Using certificates generated by the makecert.exe tool is also acceptable for test signing. However, certificates generated by makecert often do not provide useful identity information and there is no way to track which individual developer created a test signed version of the pre-release binary.

Note: The Windows Vista Beta2 release only accepts test certificates generated by the makecert tool. Test code signing certificates issued by an enterprise CA for test signing is not available in Windows Vista Beta2.

The Signtool instructions in this document work the same way whether you are using a SPC, or a certificate generated by makecert utility, or using a certificate issued by an enterprise CA. The only difference will typically be the issuer and subject name in the certificate.

The WHQL Test Signature program is also supported for test signing. Participants in the program can submit driver packages for WHQL Test Signatures. The signature on the test signed catalogs are generated by a certificate issued under the Microsoft Test Root Authority. The Microsoft Test Root Authority is accepted by default on Windows Vista Beta2 as part of the beta program. In the final release of Windows Vista, the Microsoft Test Root Authority is accepted when the Windows Vista boot configuration setting enables Test Signing.

Test signed kernel mode binaries will not load on Windows Vista systems by default. The digital signatures on test signed binaries are not valid on Windows Vista systems by default because the kernel mode code signing policy does not accept and does not trust test signing certificates.

MakeCert

MakeCert generates digital certificates that can be used for test signing. They can be either self signed or issued and signed by the Root Agent key. The test certificate can be placed in a file, a system certificate store, or both. If the test environment requires authentication, use only self-signed certificates. The Windows Vista Beta 2 release accepts only test certificates that are generated by Makecert.

Enabling Test Signing

Use the Bcdedit command-line tool to enable test signing. To use BCDedit, the user must be a member of the Administrators group on the system and run the command from an elevated

command prompt. An elevated command prompt can be launched by creating a desktop shortcut to cmd.exe, and then using right-click and “Run as administrator”.

The following shows an example of running BDCedit at the command prompt:

```
// Accept test signed kernel mode signatures
Bcdedit.exe -set TESTSIGNING ON

// Do not accept test signed kernel mode signatures
Bcdedit.exe -set TESTSIGNING OFF
```

The TESTSIGNING boot configuration option determines whether Windows Vista accepts test signed kernel mode binaries. The option is not defined by default which means digital signatures on test signed kernel mode drivers will not verify and will not load. When Windows Vista accepts test signed kernel mode binaries, some premium content that is protected may not be accessible on the system.

Patching Policy for x64-Based Systems

http://www.microsoft.com/whdc/driver/kernel/64bitpatching.mspx

::related to Server 2003 x64, XP Professional x64, Vista

Kernel patch protection was first implemented in Windows Server 2003 SP1 x64–based systems and Windows XP for x64–based systems. No significant changes have been made to kernel patch protection in Windows Vista and Windows Server Longhorn. However, because this security feature is still relatively new, it is discussed in this paper to increase awareness of the feature and its potential impact on driver and application developers.

Microsoft Windows Server 2003 Service Pack 1 x64 and Windows XP x64 support kernel patch protection only for x64 hardware platforms. ("x64" refers to the 64-bit architecture that is used in AMD64 and Intel Extended Memory 64 Technology systems.) Kernel patch protection, only for x64-based platforms, is planned for future versions of Windows, including Microsoft Windows Vista and Microsoft Windows Server code name "Longhorn."

Kernel patch protection is not currently supported for Intel Itanium hardware platforms or for 32-bit hardware platforms, but support on the platforms could change in the future.

Kernel patch protection in the x64-based versions of Microsoft Windows Server 2003 SP1, Microsoft Windows XP, and later versions of Microsoft Windows for x64-based systems protects code and critical structures in the Windows kernel from modification by unknown code or data. This paper answers frequently asked questions about kernel patch protection in Windows.

This information applies for the Microsoft Windows XP Professional 64-bit Edition operating system.

x64-based systems do not allow the kernel to be patched except through authorized Microsoft-originated hot patches. (In this article, "x64" refers to the 64-bit architecture that is used in AMD64 and Intel Extended Memory 64 Technology systems.) Kernel-mode drivers that extend or replace kernel services through undocumented means (such as hooking the system service tables) can interfere with other software and affect the stability of the operating system. For x86-based systems, Microsoft discourages such practices but does not prevent them programmatically because doing so would break compatibility for a significant amount of released software. A similar base of released software does not yet exist for x64-based systems, so it is possible to add this level of protection to the kernel with less impact on compatibility.
Many system structures are protected on x64-based systems, including the system service dispatch tables, the interrupt descriptor table (IDT), and the global descriptor table (GDT). The operating system also does not allow third-party software to allocate memory "on the side" and use it as a kernel stack. If the operating system detects one of these modifications or any other unauthorized patch, it will generate a bug check and shut down the system.

For compatibility with Windows for x64-based systems, drivers must avoid the following practices:

Modifying system service tables, for example, by hooking KeServiceDescriptorTable
Modifying the interrupt descriptor table (IDT)
Modifying the global descriptor table (GDT)
Using kernel stacks that are not allocated by the kernel
Patching any part of the kernel (detected only on AMD64-based systems)

+ and certain critical processor MSRs (Bypassing PatchGuard on Windows x64)

- SSDT (System Service Descriptor Table)
- GDT (Global Descriptor Table)
- IDT (Interrupt Descriptor Table)
- System images (ntoskrnl.exe, ndis.sys, hal.dll)
- Processor MSRs (syscall)

At a high-level, PatchGuard is implemented in the form of a set of routines that cache known-good copies and/or checksums of structures which are then validated at certain random time intervals (roughly every 5 - 10 minutes). The reason PatchGuard is implemented in a polling fashion rather than in an event-driven or hardware-backed fashion is because there is no native hardware level support for the things that PatchGuard is attempting to accomplish. For that reason, a number of the tricks that PatchGuard resorted to were done so out of necessity.

обход через hot-патчи.
http://www.microsoft.com/technet/security/topics/patchmanagement/patchmanagement.mspx

Drivers for other platforms should avoid these practices, to help ensure stability and reliability of the operating system and a better experience for customers.

If your driver must perform a task that you think cannot be accomplished without patching the kernel, contact Microsoft Product Support Services or your Microsoft representative to help determine if a documented alternative exists. If no documented alternative exists for the functionality you want to implement, then the functionality will not be supported on any Windows operating system that includes patch protection support.

Patching Police Aims

Vista kernel limits have security vendors on edge
http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci1210002,00.html

- борьба с малварью (просто как прикрытие);
- позиционирование висты как платформы для проигрывания защищенного контента *;
- устранение разработчиков av и firewall'ов и проч.

* digital rights management (DRM)

http://www.microsoft.com/whdc/driver/kernel/64bitpatch_FAQ.mspx
The Windows kernel is tested extensively before any release of the operating system to ensure a high level of quality. Because patching replaces kernel code with unknown, untested code, there is no way to assess the quality or impact of the third-party code. Furthermore, kernel code is by its nature complex and critical to system stability, so bugs in unknown code can have a significant negative impact on system stability. An examination of Online Crash Analysis (OCA) data at Microsoft shows that system crashes commonly result from both malicious and non-malicious software that patches the kernel.

Address Space Layout Randomization (ASLR)

to prevent Return-to-libc buffer overflow attacks.
Windows Vista uses Address Space Layout Randomization (ASLR) to load system files at random addresses in memory[15]. By default, all system files are loaded randomly at any of the possible 256 locations. Other executables have to specifically set a bit in the header of the PE file, which is the file format for Windows executables to use ASLR. For such executables, the stack and heap allocated is randomly decided. By loading system files at random addresses, it becomes harder for malicious code

to know where privileged system functions are located, thereby making it unlikely for them to predictably use them. This helps prevent most remote execution attacks by preventing Return-to-libc attacks.

The Portable Executable format has been updated to support embedding of exception handler address in the header. Whenever an exception is thrown, the address of the handler is verified with the one stored in the executable header. If they match, the exception is handled, otherwise it indicates that the run-time stack has been compromised, and hence the process is terminated.

Function pointers are obfuscated by XOR-ing with a random number, so that the actual address pointed to is hard to retrieve. So would be to manually change a pointer, as the obfuscation key used for the pointer would be very hard to retrieve. Thus, it is made hard for any unauthorized user of the function pointer to be able to actually use it. Also metadata for heap blocks are XOR-ed with random numbers. In addition, check-sums for heap blocks are maintained, which is used to detect unauthorized changes and heap corruption. Whenever a heap corruption is detected, the application is killed to prevent successful completion of the exploit.

Session 0 Isolation

http://en.wikipedia.org/wiki/Shatter_attack
: Previous versions of Windows ran System services in the same login session as the locally logged-in user (Session 0). In Windows Vista, Session 0 is now reserved for these services, and all interactive logins are done in other sessions. This is intended to help mitigate a class of exploits of the Windows message-passing system, known as " Shatter attacks"
Shatter attacks became a topic of intense conversation in the security community in August 2002 after the publication of Chris Paget's paper titled, "Exploiting design flaws in the Win32 API for privilege escalation". The paper, which coined the term "shatter attack", explained the process by which an application could execute arbitrary code in another application. This could occur because Windows allows unprivileged applications to send messages to message loops of higher-privileged application - and some messages can have address of callback function in application's address space as its parameter. If an attacker manages to put his own string into the memory of the higher-privileged application (say by pasting shellcode to an edit box) at a known location, they could then send WM_TIMER messages with callback function parameters set to point to the attacker's string.
A few weeks after the publication of this paper, Microsoft responded, noting that: "The paper is correct that this situation exists, and it does correctly describe its effect. ... Where the paper errs is in claiming that this is a flaw in Windows. In reality, the flaw lies in the specific, highly privileged service. By design, all services within the interactive desktop are peers, and can levy requests upon each other. As a result, all services in the interactive desktop effectively have privileges commensurate with the most highly privileged service there."
Solutions
In December 2002, Microsoft issued a patch for Windows NT 4.0, Windows 2000, and Windows XP that closed off some avenues of exploitation. This was only a partial solution, however, as the fix was limited to services included with Windows that could be exploited using this technique; the underlying design flaw still existed and could still be used to target other applications or third-party services. With Windows Vista, Microsoft aimed to solve the problem in two ways: First, local users no longer log in to Session 0, thus separating the message loop of a logged-in user's session from high-privilege system services, which are only ever loaded into Session 0. Second, a new feature called "User Interface Process Isolation" (UIPI) was introduced, whereby processes can be further protected against shatter attacks by assigning a "privilege level" to each process. Attempts to send messages to (or interact with in any way via the Windows API) a process with a higher privilege level will fail, even if both processes are owned by the same user. Internet Explorer 7+, for example, utilizes this feature to prevent its rendering components from interacting with the rest of the system.
The introduction of UIPI in Windows Vista could be interpreted as an implicit acknowledgement that Microsoft's earlier viewpoint, that the possibility of shatter attacks was "not a flaw in Windows", was mistaken.

I/O

Booting Sequence

The sequence of booting Windows Vista is slightly different to any previous version of windows that uses the NT kernel. First, when the computer is switched on, either the BIOS or the EFI is loaded. In the case of a BIOS system, the MBR of the boot disk (This can be a hard drive or external media) is accessed, which in turn loads the boot sector of the relevant hard disk partition. This boot sector then loads the Windows Boot Manager (Filename:winload.exe) which accesses the Boot Configuration Data store and uses the information to load the final stage, the Operating System
[edit]

Windows Boot Manager

The Windows Boot Manager reads the Boot Configuration Data and "displays an operating system selection menu"[1], and is thus, in some respects, equivalent to the boot selection menu functionality of NTLDR in prior versions of Windows NT.

To maintain a consistent boot experience, on Extensible Firmware Interface systems, which also have a boot manager of their own, the Windows Boot Manager, and hence all of the installed Windows operating systems that can be booted using it, appear as a single entry on the EFI boot manager menu. (On EFI systems, the Windows Boot Manager is an EFI application stored on the EFI System Partition.) Microsoft only adds multiple entries to the Windows Boot Manager menu itself, and sets the timeout of the EFI boot manager to 2 seconds.
[edit]

winload.exe

winload.exe is the operating system boot loader. It is invoked by the Windows Boot Manager in order to load the operating system kernel (ntoskrnl.exe) and (boot-class) device drivers[1], and is in that respect functionally equivalent to (the operating system loader functionality of) NTLDR in prior versions of Windows NT.
[edit]

Other programs by the same name

Confusingly, winload.exe is also the name of a spyware program, PC Tattletale, that has nothing to do with the Windows NT Startup Process.[2]
[edit]

Boot Configuration Data

Boot Configuration Data (BCD) is a database for boot-time configuration data. It replaces the boot.ini that was used by NTLDR, and is used by Microsoft's new Windows Boot Manager.[3][4]

Boot Configuration Data is stored in a data file (formatted in the same way as a Windows registry hive) that is located either on the EFI System Partition (on machines that use Extensible Firmware Interface firmware) or on the system volume (on machines that use IBM PC compatible firmware).

Boot Configuration Data may be altered using a command-line tool (bcdedit.exe) or by using Windows Management Instrumentation.

Boot Configuration Data contain the menu entries that are presented by the Windows Boot Manager, just as boot.ini contained the menu entries that were presented by NTLDR. These menu entries can include:
Options to boot Windows Vista by invoking winload.exe.
Options to resume Windows Vista from hibernation by invoking winresume.exe.
Options to boot a prior version of Windows NT by invoking its NTLDR.

Options to load and to execute a Volume Boot Record.

Boot Configuration Data allows for third party integration so anyone can implement tools like diagnostics or recovery options.

BitLocker Drive Encryption.

Formerly known as "Secure Startup", this software utilizes a Trusted Platform Module (compliant with the 1.2 version of the TCG specifications) to improve PC security. It ensures that the PC running Windows Vista starts in a known-good state, and it also protects data from unauthorized access through full volume encryption[13]. Data on the volume is encrypted with a Full Volume Encryption Key (FVEK), which is further encrypted with a Volume Master Key(VMK) and stored on the disk itself. The VMK is then stored on the TPM chip.

ReadyDrive

is a feature of Windows Vista that enables Windows Vista PCs equipped with a hybrid drive to boot up faster, resume from hibernation in less time, and preserve battery power. Hybrid hard drives are a new type of hard disk that integrates non-volatile flash memory with a traditional hard drive.
As at June 2006, there has been some concern expressed in the computer press[1] that ReadyDrive will sacrifice data-integrity for speed and battery savings.
The drive-side functionality will be standardized in ATA-8.

SuperFetch

is a new technology that speeds up the loading of commonly-used files and programs by pre-loading them into memory.
The intent is to improve performance in situations where running an anti-virus scan or back-up utility would result in otherwise recently-used information being paged out to disk, or disposed from in-memory caches, resulting in lengthy delays when a user comes back to their computer after a period of non-use.
While the necessary files by default are loaded into main memory, Windows Vista has the ability to instead use alternate storage methods, such as USB flash drives, which, though not as fast as RAM, often can be significantly faster than a hard disk drive; thereby freeing up main memory.
SuperFetch also keeps track of what times of day that applications are used, which allows it to intelligently pre-load information that is expected to be used in the near future.

Transactional NTFS

http://msdn.microsoft.com/library/en-us/fileio/fs/portal.asp
http://en.wikipedia.org/wiki/Transactional_NTFS
http://blogs.msdn.com/because_we_can/
Transactional NTFS (abbreviated TxF) brings the concept of atomic transactions to the NTFS file system, allowing Windows application developers to write file output routines that are guaranteed to either completely succeed or completely fail.
Transactional NTFS is implemented on top of the Kernel Transaction Manager (KTM), which is a Windows kernel component first introduced in Windows Vista that provides transactioning of objects in the kernel. The NTFS file system already supports journaling of low-level operations, such as writing a block of data. Transactional NTFS expands on this capability to include:

* Atomic operations on a single file:
A common example of this is saving a file from an application; if the application or machine were to crash while writing the file, then only part of the file could be written, possibly resulting in a corrupted file. This would be a very significant problem if a previous version of the file was being over-written, as data would likely be lost.
* Atomic operations spanning multiple files:

If an application needs to update several files at once with a set of changes, all the necessary file operations can be performed as a single transaction, preventing inconsistent updates in the event of a failure.
* Atomic operations spanning multiple computers:
Performing the same operation on multiple computers is a fairly common administrative task in a corporate network; Transactional NTFS integrates with the Distributed Transaction Coordinator to ensure that the change is successfully applied to all machines.
Transactional NTFS allows for files and directories to be created, renamed, and deleted transactionally.

net-stack

common links

Driver Signing Requirements for Windows
http://www.microsoft.com/whdc/winlogo/drvsign/drvsign.mspx

Key Driver Concepts
http://www.microsoft.com/whdc/driver/kernel/default.mspx

Kernel Patch Protection: Frequently Asked Questions
http://www.microsoft.com/whdc/driver/kernel/64bitpatch_FAQ.mspx

WinHEC 2006 Conference Papers
http://www.microsoft.com/whdc/winhec/papers06.mspx

very useful links

жана
http://www.invisiblethings.org/index.html

Windows Vista Security
http://blogs.msdn.com/windowsvistasecurity/archive/2006/08/11/695993.aspx

Everything Windows Driver Development
http://www.osronline.com/