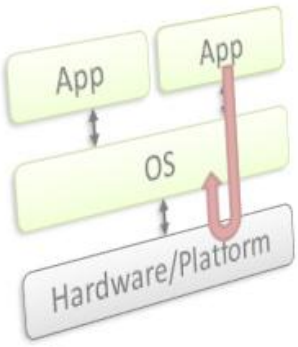
Hardware Involved Software Attacks



Jeff Forristal
jeff.forristal@intel.com

Abstract

Computer security vulnerabilities involving hardware are under-represented within the security industry. With a growing number of attackers, malware, and researchers moving beyond pure software attack scenarios and into scenarios incorporating a hardware element, it is important to start laying a foundation on how to understand, characterize, and defend against these types of hybrid attacks. This paper introduces and details a starting taxonomy of security attacks called hardware involved software attacks, in an effort to further security community awareness of hardware security and its role in upholding the security of the PC platform.

Table of Contents

Preface

The goal of this paper is to start seeding PC platform hardware security concepts and discussions into the broader security community, and to start laying a foundational understanding that can be leveraged if/when hardware security topics start coming to the forefront of public security concern. Many of these concepts are already established and recognized in hardware-centric private forums (within hardware, BIOS, and OS vendors, etc.), so it is a matter of expanding the audience of these concepts. I hope to instigate the security community to start including hardware-specific elements and accommodations into the established security understanding and practices. This will, in turn, naturally lead to a unified view & management of security risk found within a PC platform without distinction for software elements vs. hardware elements.

PC System Stack: Setting the Stage

The basic PC computer system can be represented as an abstracted stack of components (software applications, software operating system, and hardware platform) as depicted in Figure 1. This abstracted stack follows a privilege hierarchy starting with the least privilege at the top (software applications), progressing down to the most privilege at the bottom (hardware platform). The privilege hierarchy generally implies that operational control of any particular layer grants it operational control of all layers above it; thus, a security compromise of a lower layer is also a security compromise of the higher layers.

Community[1] publicized security vulnerability models (e.g. CVSS[i], CWE[ii]) and attack characterizations (e.g. CAPEC[iii]) recognize many "software attacking software" scenarios. The three most commonly recognized scenarios include a software application attacking a peer software application (horizontal attack path illustrated in Figure 2), an unprivileged software application directly attacking higher-privileged OS software elements (vertical attack path illustrated in Figure 2), and a remote party directly attacking any software elements (both unprivileged and privileged) of the system (Figure 3).

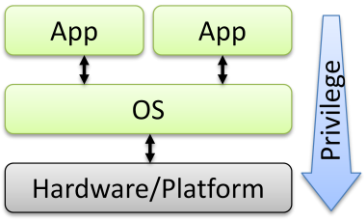


Figure 1 - Generic PC Stack

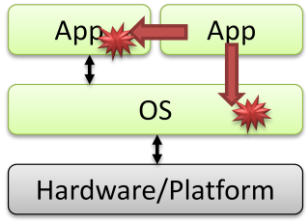


Figure 2 - Local Software Attacks

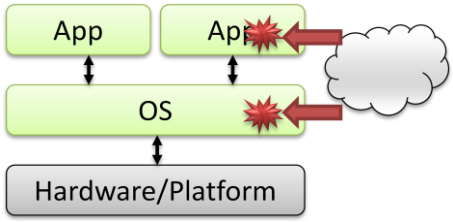


Figure 3 - Remote Attacks

This depiction is missing a notable privileged software layer present on typical PC systems: BIOS[iv]. Early stages of the PC system boot cycle depend upon BIOS software operation; eventually the BIOS hands control over to the OS. The BIOS may also utilize a SMM[v] capability to expose BIOS-privileged services to an OS during OS runtime. Due to the responsibilities BIOS has to bring up and manage the platform hardware, SMM/BIOS has higher privileges than the OS. From a depiction standpoint, SMM/BIOS does

[1] "Community" herein refers to the world security community & security industry

logically sit between the OS and the hardware at times, but at other times the OS can (and does) directly access the hardware. Thus SMM/BIOS is depicted as a partial software layer between the software OS and the hardware (Figure 4). The community does recognize attack scenarios of any lesser-privileged software element conducting direct attacks against SMM/BIOS software (Figure 5). Public examples of BIOS vulnerabilities are few, but they do exist [e.g. CVE-2008-0706, CVE-2008-7096, CVE-2010-0560, etc.][2]. Remote attacks against BIOS/SMM software are generally not considered because BIOS/SMM software (in practice) only exposes associated entry points locally to the system and not in a remotely-accessible manner[3].

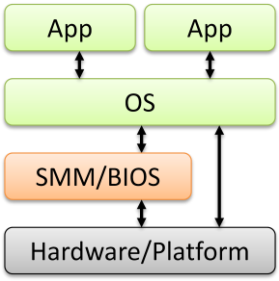


Figure 4 - PC Stack Including BIOS

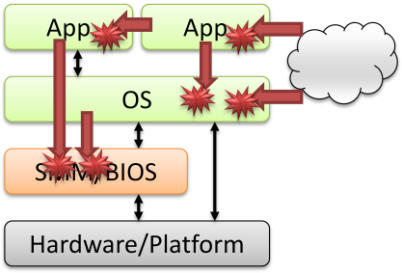


Figure 5 - Elaborated Attack Paths

We can carry forward the depiction of the PC stack with consideration for the additional dynamics of a virtualized environment running with a VMM[vi] implemented as a type-1 hypervisor[4]. Conceptually this just involves adding another software layer of appropriate privilege, and accounting for more peer software applications at the top end of the stack and the boundaries of a virtual machine (Figure 6). Community recognized software attack scenarios introduced in this model include remote attacks against the software hypervisor layer, and "VM escapes" out of the VM into the software hypervisor layer or peer VM software elements (Figure 7).

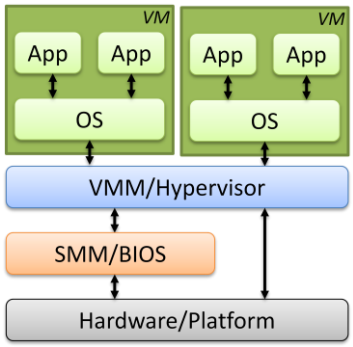


Figure 6 - Virtualized PC Stack

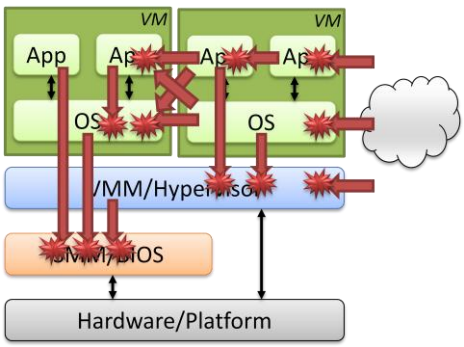


Figure 7 - Elaborated Attack Paths

[2] Descriptions for all referenced CVE identifiers are available in Appendix A

[3] Notwithstanding some infrequent situations, such as PXE boot, where BIOS could implement a TCP/IP stack and thus expose remote entry points

[4] Type-1 hypervisor runs below OS, whereas type-2 runs above OS

We could continue to evolve our depiction of the PC stack as desired, getting into more granular breakouts of certain software layer elements and inclusion of other less-common privilege hierarchy adjustments (e.g. Intel TXT[vii], Xen-style privilege domains). Many of the layers already featured also have granular nuances and sub-layers that are not immediately evident from the depictions. However, the evolution would just continue the already illustrated process of adding more layers, along with adding more software attack paths between those layers. The current level of depiction is foundationally adequate for moving this discussion forward.

Focus on the Hardware Layer

In all of the depictions thus far, it should be noticed that none of the illustrated attack paths involve the hardware layer. That's because *software vulnerabilities* are the current topic target, i.e. the final vulnerability/weakness needs to occur in a software element, and the implications of the vulnerability becoming realized would impact the operation of that software layer. If we were depicting attacks against the hardware layer, they would generally be represented as shown in Figure 8. The Pentium F00F[viii] [CVE-1999-1476], Cyrix coma[ix] [CVE-1999-0403], and AMD K6 code segment escape crash [CVE-1999-1442] bugs could be considered instances of hardware attacks since unprivileged software applications could trigger one of those bugs and cause the system to hang—a functional effect that is not meant to be had by unprivileged software. That results in a denial of service, which is an availability concern[x].

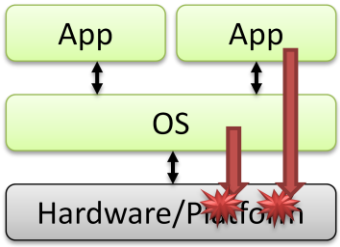


Figure 8 - Attacks Against Hardware

It is fair to say that the security community does recognize the potential for attacks against hardware; however there are a couple of practical nuances that complicate the depth of that recognition. First is the overall lack of (detailed) publicized instances of hardware vulnerabilities/attacks. Out of 48,000 entries in CVE, my research uncovered 86 entries that reasonably represented hardware attacks in a PC system (see Appendix A for the list and what "reasonably represented" means). It is hard to develop practical terminology and classification taxonomies when instances of the items to classify are exceptionally rare. That rarity also makes the investment to formally document, characterize, or address those instances, in an industry-wide manner, a questionable value proposition—why spend resources to address something on such a broad scale that has (historically) had an extremely low rate of occurrence?

The second nuance relates to how asset valuation of the PC stack is viewed in an operational security paradigm. I often ask various industry colleagues "once you have root[5], where would you go from there?" I have never received an answer that even remotely resembled "go deeper in the stack" to target lower layers of higher privilege. I believe that is because, on a practical operational value scale, full control of the OS (or hypervisor, in the case of virtualized environments) typically provides sufficient privilege to control the highest valued assets the system is perceived to offer. This is a fair perception, because that is often the case—there are few valuable assets (with widespread appeal) wholly contained in hardware that are not already accessible to the lower software layers. There is no need to go lower, because you already have control of everything worth controlling.

Forced Migration Down the Stack

There is heavy investment, represented by the security community & computing industry, into public awareness of software security issues and the need for software security. Terms like "anti-virus" and "firewall" have recognition even in non-technical communities. Numerous hacking and software security book titles can be found in the computer/technology section of your nearby book store. Security risk management and risk reduction best practices are being incorporated into industry processes and legal governance. Organizations like MITRE and NIST have taken charge to catalogue and represent the historical record of security weaknesses and vulnerabilities.

All of this investment is making an impact. Albeit it may be hard to see the impact, because the increase in security posture of key software areas may be drown out by the sheer volume of new insecure apps flooding the market elsewhere. But make no mistake: key security-sensitive software areas, such as operating systems, network services, web browsers, and virtualization hypervisors, are typically at a much better security posture now than they were in years past[xi]. It takes more investment on behalf of an attacker to find and realize a security exploit against select targets compared to the lesser level of investment in previous generations to achieve the same net effect[xii].

This means low-investment security vulnerabilities (i.e. the "low-hanging fruit") are quickly disappearing (or have already disappeared) in the key software areas frequented by attackers. This will require attackers to increase their investment and start seeking the medium-investment security vulnerabilities; in turn the supply of those will eventually dwindle, requiring an increase in investment by attackers if they still wish to continue.

Alternatively, attackers can turn their attention to alternate, less-frequented areas where low-investment security vulnerability opportunities may still be plentiful. Given that all the layers at and above the OS are slowly increasing in security posture and are heavily frequented by attackers, this foreshadows a forced migration into lower layers. In other words, as the upper layers run out of security vulnerabilities and/or become more secure, we can expect a natural progression to look for

[5] A Unix reference to denote the highest user privilege available on the system; equivalent to Administrator on Windows

security vulnerability opportunities at the lower layers of the PC stack[6], which are recognized to be under-represented and under-explored.

Hardware Background

As previously stated, the security community has strong recognition and investment into the characterization, prevention, and overall understanding of software vulnerabilities and "software attacking software" attack models. As researchers[xiii] and attackers[xiv] are turning their attention to attacks including hardware elements, the realization is that hardware-involved attack scenarios are under-represented in the community's current knowledge pool—particularly in terminology, attack characterization, and defense/mitigation guidance.

Let's start with some hardware specific concept definitions. Generally speaking, the base platform is all the hardware that composes the stack, along with the necessary management software (i.e. BIOS/SMM) that is tightly coupled to that specific hardware in order to achieve fundamental system operation.

Hardware is itself an overly generalized term; it can be used to refer to the CPU, memory, peripherals, or other elements of the platform. The "hardness" of hardware has also been subject to softening over the years; many hardware components feature firmware capabilities that bring new attack vectors and implications to the PC ecosystem. Since firmware is software running in a highly specific and constrained hardware execution environment, it's possible the firmware is subject to some of the traditional software security weaknesses[7].

Overall the hardware layer can be depicted as shown in Figure 9.

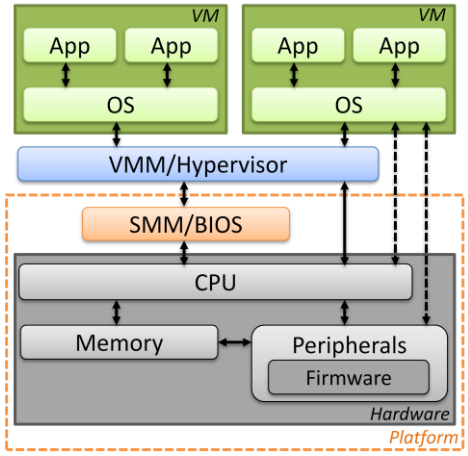


Figure 9 - Platform & Hardware Layer

[6] One colleague proposed attackers are turning their attention to operational infrastructure components instead (e.g. DNS providers, SSL cert authorities); however, I characterize that as staying within the mid-level software layers of the PC stack while just switching to a different PC. Since it's still software, it's still subject to the slowly increasing security posture of software at large.

[7] Attacks against firmware can be rolled into attacks against hardware, which is not the scope of this paper

How Hardware Facilitates Security Attacks

The entire PC platform operation relies on the orchestration of multiple hardware elements in order to achieve the platform operational goals. Each piece of hardware brings something different to the party, and the security relevance of that piece of hardware depends upon its overall native role in the PC platform. For example:

- The hardware may have direct capabilities to affect a critical system resource (e.g. DMA[xv] to system/software memory)
- The hardware may have indirect sideband access to a resource (e.g. PCI cards typically have access to an SMBus[xvi] segment)
- The hardware may store arbitrary software executable code that can be automatically invoked (e.g. HDD or USB drive; PCI device option ROM)
- The hardware may proxy data from an untrusted external source (e.g. NICs[xvii], Wifi radios)

Obtaining Hardware Access

Since we are looking at hardware-involved security attacks on software, we need to characterize how an attacker will first obtain necessary access to the hardware to achieve the intermediary step of the attack.

Hardware access can be realized in a number of ways:

- **Mistakenly passed through by a higher privilege software layer**.
  A higher-privileged software layer may attempt to provide a controlled or limited access to hardware, but wind up being overly-permissive; or the hardware it allowed access to has additional, unrecognized functionality. In other situations, the hardware access functionality provided by the privileged software layer may be a remnant of non-production debugging needs, etc.

- **Explicitly passed through by a higher privilege software layer**.
  Many hardware devices exist with intent they be accessible and utilized by the local user, who typically is running in the application privilege layer. Graphics is a great example: GPGPU workloads, DirectX shaders, and OpenCL kernels originate at the app layers and are passed through to the graphics hardware for GPGPU interpretation & execution. Elsewhere, user-mode driver frameworks, particular in the USB device arena, allow flexibility by the OS to offload select arbitrary device handling to user-mode applications. In all of these situations, the OS layer is allowing access to a select portion of hardware to facilitate the management & use of that hardware by the application layer.

- **Explicitly provided by hardware architectural intent**.
  Hardware assisted virtualization technologies like Intel VT-x, AMD-V, EPT[xviii], VT-d[xix]/AMD-Vi/IOMMU[xx], and SR-IOV[xxi] facilitate direct hardware access (e.g. access by VM guests in a virtualized environment, to the benefit of the hypervisor). These technologies get leveraged in

VMM product features such as Xen PCI passthrough[xxii], Xen VGA passthrough[xxiii], and VMWare VMDirectPath I/O[xxiv]. Similarly but on a more conceptual level, application (ring 3[xxv]) software is allowed to directly execute many instructions on the CPU without OS (ring 0) involvement. Overall a VMM or OS may utilize hardware architecture to provide bounded access to less-privileged software in a controlled manner; the assumption is that it is safe to do so.

- **The attacker is already deemed to have access.**
  Particular to SMM/BIOS software layer, SMM is simply not in a strong architectural position to gate all hardware access by the software layer above it; therefore it must always operate with consideration that system resources are shared with a potentially untrusted or compromised OS/hypervisor layer. Hardware assisted virtualization technologies also provide hypervisors with the ability to only intercept a subset of hardware access and CPU instructions originating from a VM guest; certain VM guest hardware operations simply do not have a corresponding VMM trap/exit available for the hypervisor to leverage[8].

- **The attacker is physically proximate to the system**.
  Physical possession or access to a PC system allows for various hardware tampering attacks (e.g. Evil Maid[xxvi], cold boot[xxvii], hardware keylogger) and use of externally exposed hardware capabilities (e.g. Firewire DMA). Physical proximity is sufficient for attacks using radio hardware (e.g. Wifi, LTE/Wimax, Bluetooth, GSM/cellular) as the entry point into the system. Looking beyond PCs for a moment, the community already has seen many instances of embedded system "jailbreaking[xxviii]" and game console hacks where physical access to the device was leveraged to achieve a software advantage of some sort.

Taxonomy of Hardware Involved Software Attacks

The driving purpose of this paper is to present a starting taxonomy of attacks dubbed *hardware-involved software attacks*. An attack fitting into the hardware-involved software attack class will generally:

- Originate in a lower-privileged software/layer or be remote/physically proximate
- Leverage or depend upon an operation of hardware
- Achieve a vulnerability in a higher-privileged software/layer or a peer in current software/layer

A locally originated hardware-involved software attack is depicted in Figure 10, while a remote or physically proximate attack is depicted in Figure 11.

[8] Hypervisors can address this via other means (as seen in hypervisors that inspect every guest instruction), but those generally start moving outside the capability realm of what the hardware assisted virtualization technologies are aiming to provide.

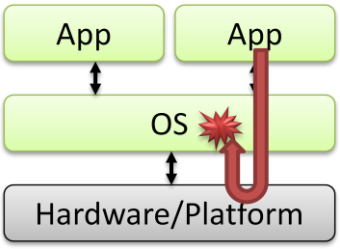


Figure 10 - Local Hardware Involved Software Attack

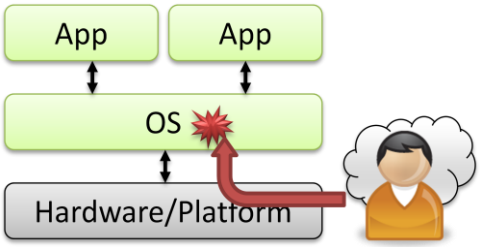


Figure 11 – Remote & Physically Proximate Hardware Involved Software Attack

The following sections detail different hardware-involved attack scenarios, along with publicly documented examples that illustrate the attack.

Inappropriate General Access to Hardware

This attack scenario is where a higher-privileged software layer (i.e. OS or hypervisor) incorrectly grants or proxies general hardware access to a lower-privileged software layer. This allows the lower-privileged software to utilize general hardware access methods to cause a security vulnerability. This can also be characterized as a hardware access enforcement failure by the OS/hypervisor; sometimes this is referred to as a "confused deputy" attack[xxix]. General hardware access mechanisms include MMIO, port IO, MSR, and PCI configuration space access. In certain situations access to hardware-consumed data structures in normal memory may be relevant—but the extent of that memory access, and its relevance to hardware involvement, may need to be qualified/reviewed since it is already recognized that giving lower-privileged software access to arbitrary system memory is an immediate vulnerability that doesn't require any hardware involvement.

Examples [CVE-2005-0204], [CVE-2007-5633], [CVE-2007-5761], [CVE-2010-1592] involve instances where privileged OS elements (kernel, drivers) exposed general-purpose hardware access (e.g. MSR, port IO, etc.) to unprivileged applications. Reconfiguration of the MSRs/IO by the application layer could be done to realize a compromise of the OS software layer. For example, writing a new value to the SYSENTER_EIP MSR would effectively allow the application to provide an arbitrary memory address that gets invoked with ring 0 privileges upon the next SYSENTER instruction execution.

In certain cases, the OS/hypervisor layer may allow general purpose hardware access subject to a whitelist or blacklist access control mechanism. IO permission bit maps are a perfect example. In such cases the OS/hypervisor layer is not offering unfettered hardware access; yet a failure to correctly define the blacklist/whitelist may still result in this attack scenario.

There are other miscellaneous examples such as [CVE-2001-1578].

Unexpected Consequences of Specific Hardware Function

The attack preconditions start with a higher-privileged software layer (i.e. OS or hypervisor) granting or proxying specific, limited hardware access to hardware for a lower-privileged software layer data. This is done under the presumption that only particular hardware functionality is exposed, and access to that specific hardware functionality is not believed to harbor any security concerns. The attack then leverages an unknown or unrecognized consequence to that hardware access that was not originally considered when making the decision to allow access.

Example [CVE-2011-1898] involves the use of DMA to generate MSI interrupts, which was leveraged from within a VM to compromise the Xen hypervisor software layer. Examples [CVE-2007-3532], [CVE-2011-1016] involve instances where OS drivers gave applications access to specific device hardware functions (such as a graphics card), and that device hardware had capabilities that could be leveraged to compromise the OS software layer.

Generally speaking, this attack includes all situations where a privileged software layer provides access to a DMA-capable hardware device (i.e. any PCI peripheral), in such a way the unprivileged software layer can influence the DMA operations of that device. Use of IOMMU technologies, on platforms where it is available, may mitigate the attack.

The widespread public community discussion over WebGL[xxx] also fits into this attack profile. Microsoft Security Research & Defense published a position[xxxi] that indicated the severe risks of taking remotely-originated graphics content and providing it to the graphics pipeline:

> "As WebGL vulnerabilities are uncovered, they will not always manifest in the WebGL API itself. The problems may exist in the various OEM and system components delivered by IHV's." (Microsoft, 2011)

In particular, their perception is that both graphics software *and hardware* do not have a sufficient security posture to afford them being directly exposed to low-privilege remote third parties.

While Microsoft was speaking in general terms, there are WebGL vulnerabilities already publicly known that serve as an example. [CVE-2011-2367] is a vulnerability where access to graphics GPU hardware by a least-privileged remote party results in the exposure of GPU memory contents from previous workloads; that same vulnerability can also lead to an availability issue (crash).

Hardware Reflected Injection

A hardware reflected injection attack is when the attacking application originates particular malicious data, and that data traverses (without modification) through the higher-privileged software stack layers and into the hardware for storage; later, a privileged software layer (such as driver in the OS) receives/retrieves the malicious data from the hardware, and immediately operates upon, interprets, or otherwise uses the malicious data in an insecure manner leading to a security vulnerability. Hardware reflected injection attacks have more opportunities for success against a software target that implicitly trusts data coming from the hardware. Potential hardware storage areas include CPU MSRs, CPU

registers, BIOS data structures in memory, platform hardware volatile and non-volatile storage areas, and peripheral hardware registers (port IO, MMIO).

A (fictitious yet conceptually demonstrative) example would be manipulating BIOS configuration values saved in platform RTC/CMOS storage area, to cause a security vulnerability in BIOS when BIOS uses/interprets those values upon next system (re)start.

There are three characterized variants to hardware reflected injection attacks.

Variant #1: This attack variant resembles a traditional software second-order injection style attack[xxxii]. Namely the value retrieved from the hardware is used in an integer operation (resulting in an integer overflow or signed integer issue), a memory buffer operation (resulting in a buffer overflow), etc. The key difference from a traditional software second-order injection is the storage source of the malicious data is explicitly provided by the hardware.

Alexandre Gazet's 2011 presentation[xxxiii] at Recon security conference discloses a scenario where the firmware of a KBC can be updated, which in turn will feed a malicious value back to SMM and cause a buffer overflow. The example is a bit more complex than a simple value, but ultimately it showcases the variant. Another related public example, [CVE-2010-4530], details an integer handling issue in a privileged software driver, triggered by a value retrieved from a hardware peripheral (in this case, a smartcard reader). In a situation where local software on the host can provide/set/configure (in the smartcard hardware) the particular value responsible for the vulnerability, it would lead to an exact embodiment of this attack variant.

Variant #2: The next attack variant focuses on situations where there is a unique interpretation of the hardware value in some security-sensitive operation, leading to a vulnerability. These attacks generally don't result in a typically recognized software weakness such as a buffer overflow or integer handling issue; instead, they are contextual to a particular hardware circumstance and may relate more to improper software logic design than to programmatic implementation errors.

One public example of this is [CVE-2009-4419], which is a disclosed attack involving untrusted software manipulating the MCHBAR hardware register; the register value is then utilized by trusted software and acted upon in a manner that prevents VT-d protection from properly being applied, leaving the system at risk.

Other examples include [CVE-2007-5906].

Variant #3: This attack variant addresses specific Hardware Reflected Injection attack instances that involve updating blobs of data stored in hardware, where those blobs are later treated as executable software code and executed with increased privilege.

A public example is the Mebromi malware[xxxiv], which included a malicious OptionROM executable code blob that it installed into BIOS platform flash (hardware) storage. Upon system (re)start, BIOS will execute, with BIOS privileges, all hardware peripheral OptionROM code blobs—including the malicious

blob installed by Mebromi. Mebromi used the OptionROM to achieve pre-OS code execution and persist the malware infection.

Another general example would be writing malicious code to a hard drive MBR through direct hardware means. Software security threat models already include a characterization of an application trying to update critical system areas of a file system in order to affect a privilege escalation; this approach is typically thwarted by file system-based logical access control semantics. For a hardware reflected injection attack involving an MBR update, the attacker would need to use more direct hardware access mechanisms to cause the MBR update, rather than utilizing the typical OS file system driver stack. Access routes could include direct access to ACHI[xxxv] hardware control registers, or utilizing ATA hardware command pass-through capabilities offered by the OS[xxxvi].

Interference with Hardware Privilege Access Enforcement

This attack is typically targeted to the hypervisor layer (or emulation software), and involves scenarios where a less-privileged software layer can cause the hypervisor to perform a hardware operation (e.g. instruction execution) or access that would normally be unavailable to that less-privileged software layer due to hardware privilege enforcement, but the hardware will now allow it since it is done with hypervisor privileges.

Examples [CVE-2009-1542], [CVE-2010-0298] involve instances where a hypervisor passed through a privileged instruction from a VM guest application (ring 3) layer without recognizing it was a privileged instruction that should only be executable by the VM guest OS (ring 0) layer; this effectively creates a privilege escalation vulnerability within the VM.

Access By a Parallel Executing Entity

The PC platform has many components that are executing operations in parallel/independent manner. Some components, like the CPU, further feature parallel execution capabilities represented by having multiple CPU cores and additional hardware-assisted threads (e.g. Hyper-Threading[xxxvii]).

The PC platform also features many shared resources—memory in particular is a good example. If a trusted software security agent needs to operate on memory, it may be necessary to ensure all other parallel executing entities (both software & hardware) are temporarily blocked, shut down, or otherwise prevented from interfering with the security sensitive operations occurring on that shared resource—i.e. quiesce[xxxviii] all executing entities. This behavior can be witnessed in many places, such as CPU-supported rendezvous of all CPU threads and cores for certain SMM operations, or the use of PCI bridge device standard BME bit[xxxix] to control peripheral access to system memory.

An attack is possible if the performer of a security-sensitive operation doesn't account for a parallel executing entity having the ability to monitor or interfere with that operation. Example [CVE-2010-0306] involves some tricks by the less-privileged software layer (in an SMP environment) to utilize a

parallel executing thread to switch instructions of the other thread in-flight from an unprivileged instruction to a privileged instruction without the hypervisor noticing. Vulnerability [CVE-2005-0109] involves using one CPU thread to monitor memory cache misses of another thread in order to recover information such as cryptographic keys.

External Control of a Hardware Device

Maliciously behaving hardware devices may be able to affect the security of the system; the key is when and how they become malicious. We've been focusing on attacks originating from software on the system that is the target for the attack—however this attack scenario recognizes that it may be less-privileged attacks originating external to the system.

One public example is [CVE-2011-3215], which involves the use of DMA over a Firewire port, allowing any device connected to Firewire to read and write system memory. Generally this device would begin in a malicious state and be connected to the system to explicitly perform an attack; however we want to recognize that an already connected, innocuous Firewire device may offer firmware re-programmability which, when reprogrammed by a malicious local software agent, would put the device into a malicious state without any external involvement and bring this attack to fruition.

The same goes for less-privilege software reprogramming of keyboard firmware ([CVE-2009-2834]) or reprogramming other devices to look like keyboards ([CVE-2011-0640], [CVE-2011-0639], [CVE-2011-0638]). A reprogrammed keyboard could be leveraged across a system restart to emulate user interaction by hitting keystrokes that affect BIOS configuration changes.

Attacks over radio hardware ([CVE-2010-3832], [CVE-2009-0282], [CVE-2006-5972]) can also resemble this attack scenario. Other miscellaneous examples include [CVE-2010-4530].

Incorrect Hardware Use

This is a bit of a catch-all attack scenario to address usage and configuration errors by privileged software layers to utilize hardware in a manner that will create an appropriate secure & privileged enforced environment.

One example is [CVE-2006-1056], where AMD CPUs don't implement FXSAVE/FXRSTOR in the same way Intel CPUs do; Linux kernels didn't recognize the discrepancy (despite it being documented), and that lead to a leak of floating point data between processes.

Other miscellaneous examples of this attack include [CVE-2006-0744], [CVE-2005-1764], [CVE-2004-0812], [CVE-2010-2938], [CVE-2006-7215].

Where to Go From Here

Hardware involved software attacks are not a fancy speculation; as seen by the indicated examples, they are already being publicly witnessed in the community at increasing rate. Yet despite publicly seeing attack examples, the attacks are not characterized and described with a consistent terminology, nor is there any publicly available guidance for how to operationally reduce or pro-actively address these types of attacks. There are public knowledge gaps in everything from secure software developer guidance, hardware implementation security best practices, operational system monitoring capabilities for attacks (IDS/IPS/AV), and system forensics support for investigation into software attacks involving hardware.

I hope this paper serves as a call-to-arms to initiate efforts to address these knowledge gaps. I presented a starting, high-level taxonomy of hardware-involved software attacks, which can be built upon and used to create and align defensive capabilities to address the indicated attacks, and grow the community knowledge base regarding hardware security.

Appendix A – Publicized Hardware Vulnerabilities

To support this paper, I conducted research regarding known (public) hardware security vulnerabilities. I chose to use the Common Vulnerability Enumeration[xl] (CVE) dictionary of publicly known information security vulnerabilities and exposures. It represents the most comprehensive collection of known security problems—over 48,000 at the time of this writing.

Doing a manual review of all 48,000 entries would represent an effort beyond what I could afford, so I opted to first filter candidate entries based on keyword searches for common and related hardware security terms the CVE writers would typically include, and then manually reviewed the filtered results for final applicability. Overall this resulted in my manual review of a few thousand entries, many of which were redundant between keywords.

Since one discussion topic of this paper is to identify the lack of established terminology to characterize hardware security attacks, it's reasonable to wonder whether a keyword search would be fruitful at all given a lack of consistent terms to search for. Hardware bugs may be mis-characterized as software bugs, or vaguely described in non-hardware terms. In order to compensate for this, I opted to cast a wide net and also include keyword searches for specific vendor/product/technology names that have strong hardware relationships. I purposefully tried to search for and review every vulnerability involving a hardware driver, OS kernel, or VMM since those components traditionally have high amounts of interaction with hardware.

The keywords I used for preliminary filtering of entries, in no particular order:

> *mmio, iopl, vmm, registers, pci, dma, instruction, mchbar, vtd, iommu, msr, hypervisor, smm, bios, processor, uefi, amd, .sys, ahci, webgl, webcl, opencl, gpu, shader, hyper-v, nvidia, radeon, insyde, phoenix, xen, hardware, usb, aware, cmos, motherboard, driver, i2c, cpu, peripheral, intel, smbus, physical memory, microcontroller, firewire, chipset, keyboard, acpi, cpuid*

After searching for the above keywords, I then manually reviewed the result set to further determine applicability.

Overall, 86 entries were identified. The full list of identified entries is included below. In some cases, the CVE entry description directly indicates hardware relevance; in other cases, I had to review supporting resources/advisory details, software patch diffs, etc. to confirm the hardware involvement. There were many suspicious entries of "unknown attack vectors" leading to "unknown impact" mentioned in components that normally facilitate hardware access (e.g. kernels, drivers); I tried to make reasonable determinations on whether or not to include the entry based on where the vulnerability was manifested, but overall some entries were discarded simply because there just wasn't enough information available to make even an educated guess. The list below may not be perfect, and certainly there may be items missing from the list—but the challenge to produce a conclusive list supports this paper's position that better recognition and vulnerability characterization are needed before we have the opportunity to produce an accurate list.

Additional CVE entry meta data, including official vendor advisories and any known mitigation/fix information (where applicable), is available by reviewing the full CVE entry data via CVE search sites such as http://web.nvd.nist.gov/; only the description of the CVE entry is included in the list below for quick reference.

CVE List of Hardware Involved Software Vulnerabilities

CVE-1999-0728	A Windows NT user can disable the keyboard or mouse by directly calling the IOCTLs which control them.
CVE-1999-1442	Bug in AMD K6 processor on Linux 2.0.x and 2.1.x kernels allows local users to cause a denial of service (crash) via a particular sequence of instructions, possibly related to accessing addresses outside of segments.
CVE-1999-1482	SVGAlib zgv 3.0-7 and earlier allows local users to gain root access via a privilege leak of the iopl(3) privileges to child processes.
CVE-2000-0946	Compaq Easy Access Keyboard software 1.3 does not properly disable access to custom buttons when the screen is locked, which could allow an attacker to gain privileges or execute programs without authorization.
CVE-2001-0659	Buffer overflow in IrDA driver providing infrared data exchange on Windows 2000 allows attackers who are physically close to the machine to cause a denial of service (reboot) via a malformed IrDA packet.
CVE-2001-1273	The "mxcsr P4" vulnerability in the Linux kernel before 2.2.17-14, when running on certain Intel CPUs, allows local users to cause a denial of service (system halt).
CVE-2001-1347	Windows 2000 allows local users to cause a denial of service and possibly gain privileges by setting a hardware breakpoint that is handled using global debug registers, which could cause other processes to terminate due to an exception, and allow hijacking of resources such as named pipes.
CVE-2001-1578	Unknown vulnerability in SCO OpenServer 5.0.6 and earlier allows local users to modify critical information such as certain CPU registers and segment descriptors.
CVE-2002-1125	FreeBSD port programs that use libkvm for FreeBSD 4.6.2-RELEASE and earlier, including (1) asmon, (2) ascpu, (3) bubblemon, (4) wmmon, and (5) wmnet2, leave open file descriptors for /dev/mem and /dev/kmem, which allows local users to read kernel memory.
CVE-2002-1722	Logitech iTouch keyboards allows attackers with physical access to the system to bypass the screen locking function and execute user-defined commands that have been assigned to a button.
CVE-2002-2127	Integrity Protection Driver (IPD) 1.2 and earlier blocks access to \Device\PhysicalMemory by its name, which could allow local privileged processes to overwrite kernel memory by accessing the device through a symlink.
CVE-2003-0248	The mxcsr code in Linux kernel 2.4 allows attackers to modify CPU state registers via a malformed address.
CVE-2003-1233	Pedestal Software Integrity Protection Driver (IPD) 1.3 and earlier allows privileged attackers, such as rootkits, to bypass file access restrictions to the Windows kernel by using the NtCreateSymbolicLinkObject

	function to create a symbolic link to (1) \Device\PhysicalMemory or (2) to a drive letter using the subst command.
CVE-2004-0812	Unknown vulnerability in the Linux kernel before 2.4.23, on the AMD AMD64 and Intel EM64T architectures, associated with "setting up TSS limits," allows local users to cause a denial of service (crash) and possibly execute arbitrary code.
CVE-2004-1017	Multiple "overflows" in the io_edgeport driver for Linux kernel 2.4.x have unknown impact and unknown attack vectors.
CVE-2004-1038	A design error in the IEEE1394 specification allows attackers with physical access to a device to read and write to sensitive memory using a modified FireWire/IEEE 1394 client, thus bypassing intended restrictions that would normally require greater degrees of physical access to exploit. NOTE: this was reported in 2008 to affect Windows Vista, but some Linux-based operating systems have protection mechanisms against this attack.
CVE-2004-1056	Direct Rendering Manager (DRM) driver in Linux kernel 2.6 does not properly check the DMA lock, which could allow remote attackers or local users to cause a denial of service (X Server crash) and possibly modify the video output.
CVE-2005-0109	Hyper-Threading technology, as used in FreeBSD and other operating systems that are run on Intel Pentium and other processors, allows local users to use a malicious thread to create covert channels, monitor the execution of other threads, and obtain sensitive information such as cryptographic keys, via a timing attack on memory cache misses.
CVE-2005-0204	Linux kernel before 2.6.9, when running on the AMD64 and Intel EM64T architectures, allows local users to write to privileged IO ports via the OUTS instruction.
CVE-2005-1036	FreeBSD 5.x to 5.4 on AMD64 does not properly initialize the IO permission bitmap used to allow user access to certain hardware, which allows local users to bypass intended access restrictions to cause a denial of service, obtain sensitive information, and possibly gain privileges.
CVE-2005-1399	FreeBSD 4.6 to 4.11 and 5.x to 5.4 uses insecure default permissions for the /dev/iir device, which allows local users to execute restricted ioctl calls to read or modify data on hardware that is controlled by the iir driver.
CVE-2005-1764	Linux 2.6.11 on 64-bit x86 (x86_64) platforms does not use a guard page for the 47-bit address page to protect against an AMD K8 bug, which allows local users to cause a denial of service.
CVE-2005-2388	Buffer overflow in a certain USB driver, as used on Microsoft Windows, allows attackers to execute arbitrary code.
CVE-2005-2890	SecureOL VE2 1.05.1008 does not properly restrict public access to physical memory, which allows local users to bypass intended restrictions and gain access to the secured environment via direct access to the PhysicalMemory device.
CVE-2006-0744	Linux kernel before 2.6.16.5 does not properly handle uncanonical return addresses on Intel EM64T CPUs, which reports an exception in the SYSRET instead of the next instruction, which causes the kernel exception handler to run on the user stack with the wrong GS.
CVE-2006-1056	The Linux kernel before 2.6.16.9 and the FreeBSD kernel, when running on AMD64 and other 7th and 8th generation AuthenticAMD processors, only save/restore the FOP, FIP, and FDP x87 registers in FXSAVE/FXRSTOR when an exception is pending, which allows one process to determine portions of the state of floating point instructions of other processes, which can be leveraged to obtain sensitive information such as cryptographic keys. NOTE: this is the documented behavior of AMD64 processors, but it is inconsistent with Intel processers in a security-relevant fashion that was not addressed by the kernels.
CVE-2006-1368	Buffer overflow in the USB Gadget RNDIS implementation in the Linux kernel before 2.6.16 allows remote attackers to cause a denial of service (kmalloc'd memory corruption) via a remote NDIS response to OID_GEN_SUPPORTED_LIST, which causes memory to be allocated for the reply data but not the reply structure.
CVE-2006-2147	resmgrd in resmgr for SUSE Linux and other distributions does not properly handle when access to a USB device is granted by using "usb:<bus>,<dev>" notation, which grants access to all USB devices and allows local users to bypass intended restrictions. NOTE: this is a different vulnerability than CVE-2005-4788.
CVE-2006-2935	The dvd_read_bca function in the DVD handling code in drivers/cdrom/cdrom.c in Linux kernel 2.2.16, and later versions, assigns the wrong value to a length variable, which allows local users to execute arbitrary code via a crafted USB Storage device that triggers a buffer overflow.
CVE-2006-2936	The ftdi_sio driver (usb/serial/ftdi_sio.c) in Linux kernel 2.6.x up to 2.6.17, and possibly later versions, allows local users to cause a denial of service (memory consumption) by writing more data to the serial port than the hardware can handle, which causes the data to be queued.
CVE-2006-3146	The TOSRFBD.SYS driver for Toshiba Bluetooth Stack 4.00.29 and earlier on Windows allows remote attackers to cause a denial of service (reboot) via a L2CAP echo request that triggers an out-of-bounds

	memory access, similar to "Ping o' Death" and as demonstrated by BlueSmack. NOTE: this issue was originally reported for 4.00.23.
CVE-2006-3507	Multiple stack-based buffer overflows in the AirPort wireless driver on Apple Mac OS X 10.3.9 and 10.4.7 allow physically proximate attackers to execute arbitrary code by injecting crafted frames into a wireless network.
CVE-2006-3508	Heap-based buffer overflow in the AirPort wireless driver on Apple Mac OS X 10.4.7 allows physically proximate attackers to cause a denial of service (crash), gain privileges, and execute arbitrary code via a crafted frame that is not properly handled during scan cache updates.
CVE-2006-5405	Unspecified vulnerability in Toshiba Bluetooth wireless device driver 3.x and 4 through 4.00.35, as used in multiple products, allows physically proximate attackers to cause a denial of service (crash), corrupt memory, and possibly execute arbitrary code via crafted Bluetooth packets.
CVE-2006-5710	The Airport driver for certain Orinoco based Airport cards in Darwin kernel 8.8.0 in Apple Mac OS X 10.4.8, and possibly other versions, allows remote attackers to execute arbitrary code via an 802.11 probe response frame without any valid information element (IE) fields after the header, which triggers a heap-based buffer overflow.
CVE-2006-5972	Stack-based buffer overflow in WG111v2.SYS in NetGear WG111v2 wireless adapter (USB) allows remote attackers to execute arbitrary code via a long 802.11 beacon request.
CVE-2006-6059	Buffer overflow in MA521nd5.SYS driver 5.148.724.2003 for NetGear MA521 PCMCIA adapter allows remote attackers to execute arbitrary code via (1) beacon or (2) probe 802.11 frame responses with an long supported rates information element. NOTE: this issue was reported as a "memory corruption" error, but the associated exploit code suggests that it is a buffer overflow.
CVE-2006-6106	Multiple buffer overflows in the cmtp_recv_interopmsg function in the Bluetooth driver (net/bluetooth/cmtp/capi.c) in the Linux kernel 2.4.22 up to 2.4.33.4 and 2.6.2 before 2.6.18.6, and 2.6.19.x, allow remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via CAPI messages with a large value for the length of the (1) manu (manufacturer) or (2) serial (serial number) field.
CVE-2006-6125	Heap-based buffer overflow in the wireless driver (WG311ND5.SYS) 2.3.1.10 for NetGear WG311v1 wireless adapter allows remote attackers to execute arbitrary code via an 802.11 management frame with a long SSID.
CVE-2006-6651	Race condition in W29N51.SYS in the Intel 2200BG wireless driver 9.0.3.9 allows remote attackers to cause memory corruption and execute arbitrary code via a series of crafted beacon frames. NOTE: some details are obtained solely from third party information.
CVE-2006-6730	OpenBSD and NetBSD permit usermode code to kill the display server and write to the X.Org /dev/xf86 device, which allows local users with root privileges to reduce securelevel by replacing the System Management Mode (SMM) handler via a write to an SMRAM address within /dev/xf86 (aka the video card memory-mapped I/O range), and then launching the new handler via a System Management Interrupt (SMI), as demonstrated by a write to Programmed I/O port 0xB2.
CVE-2006-7215	The Intel Core 2 Extreme processor X6800 and Core 2 Duo desktop processor E6000 and E4000 incorrectly set the memory page Access (A) bit for a page in certain circumstances involving proximity of the code segment limit to the end of a code page, which has unknown impact and attack vectors on certain operating systems other than OpenBSD, aka AI90.
CVE-2007-0933	Buffer overflow in the wireless driver 6.0.0.18 for D-Link DWL-G650+ (Rev. A1) on Windows XP allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via a beacon frame with a long TIM Information Element.
CVE-2007-1876	VMware Workstation before 5.5.4, when running a 64-bit Windows guest on a 64-bit host, allows local users to "corrupt the virtual machine's register context" by debugging a local program and stepping into a "syscall instruction."
CVE-2007-2455	Parallels allows local users to cause a denial of service (virtual machine abort) via (1) certain INT instructions, as demonstrated by INT 0xAA; (2) an IRET instruction when an invalid address is at the top of the stack; (3) a malformed MOVNTI instruction, as demonstrated by using a register as a destination; or a write operation to (4) SEGR6 or (5) SEGR7.
CVE-2007-2927	Unspecified vulnerability in Atheros 802.11 a/b/g wireless adapter drivers before 5.3.0.35, and 6.x before 6.0.3.67, on Windows allows remote attackers to cause a denial of service via a crafted 802.11 management frame.
CVE-2007-3532	NVIDIA drivers (nvidia-drivers) before 1.0.7185, 1.0.9639, and 100.14.11, as used in Gentoo Linux and possibly other distributions, creates /dev/nvidia* device files with insecure permissions, which allows local users to modify video card settings, cause a denial of service (crash or physical video card damage), and obtain sensitive information.

CVE-2007-3850	The eHCA driver in Linux kernel 2.6 before 2.6.22, when running on PowerPC, does not properly map userspace resources, which allows local users to read portions of physical address space.
CVE-2007-3851	The drm/i915 component in the Linux kernel before 2.6.22.2, when used with i965G and later chipsets, allows local users with access to an X11 session and Direct Rendering Manager (DRM) to write to arbitrary memory locations and gain privileges via a crafted batchbuffer.
CVE-2007-4315	The AMD ATI atidsmxx.sys 3.0.502.0 driver on Windows Vista allows local users to bypass the driver signing policy, write to arbitrary kernel memory locations, and thereby gain privileges via unspecified vectors, as demonstrated by "Purple Pill".
CVE-2007-5633	Speedfan.sys in Alfredo Milani Comparetti SpeedFan 4.33, when used on Microsoft Windows Vista x64, allows local users to read or write arbitrary MSRs, and gain privileges and load unsigned drivers, via the (1) IOCTL_RDMSR 0x9C402438 and (2) IOCTL_WRMSR 0x9C40243C IOCTLs to \Device\speedfan, as demonstrated by an IOCTL_WRMSR action on MSR_LSTAR.
CVE-2007-5761	The NantSys device 5.0.0.115 in Motorola netOctopus 5.1.2 build 1011 has weak permissions for the \\.\NantSys device interface (nantsys.sys), which allows local users to gain privileges or cause a denial of service (system crash), as demonstrated by modifying the SYSENTER_EIP_MSR CPU Model Specific Register (MSR) value.
CVE-2007-5906	Xen 3.1.1 allows virtual guest system users to cause a denial of service (hypervisor crash) by using a debug register (DR7) to set certain breakpoints.
CVE-2007-6207	Xen 3.x, possibly before 3.1.2, when running on IA64 systems, does not check the RID value for mov_to_rr, which allows a VTi domain to read memory of other domains.
CVE-2007-6416	The copy_to_user function in the PAL emulation functionality for Xen 3.1.2 and earlier, when running on ia64 systems, allows HVM guest users to access arbitrary physical memory by triggering certain mapping operations.
CVE-2008-0211	Unspecified vulnerability in the BIOS F.04 through F.11 for the HP Compaq Business Notebook PC allows local users to cause a denial of service via unspecified vectors.
CVE-2008-0706	Unspecified vulnerability in the BIOS F.26 and earlier for the HP Compaq Notebook PC allows physically proximate attackers to obtain privileged access via unspecified vectors, possibly involving an authentication bypass of the power-on password.
CVE-2008-4218	Multiple integer overflows in the kernel in Apple Mac OS X before 10.5.6 on Intel platforms allow local users to gain privileges via a crafted call to (1) i386_set_ldt or (2) i386_get_ldt.
CVE-2008-4917	Unspecified vulnerability in VMware Workstation 5.5.8 and earlier, and 6.0.5 and earlier 6.x versions; VMware Player 1.0.8 and earlier, and 2.0.5 and earlier 2.x versions; VMware Server 1.0.9 and earlier; VMware ESXi 3.5; and VMware ESX 3.0.2 through 3.5 allows guest OS users to have an unknown impact by sending the virtual hardware a request that triggers an arbitrary physical-memory write operation, leading to memory corruption.
CVE-2008-4992	The SPARC hypervisor in Sun System Firmware 6.6.3 through 6.6.5 and 7.1.3 through 7.1.3.e on UltraSPARC T1, T2, and T2+ processors allows logical domain users to access memory in other logical domains via unknown vectors.
CVE-2008-7096	Intel Desktop and Intel Mobile Boards with BIOS firmware DQ35JO, DQ35MP, DP35DP, DG33FB, DG33BU, DG33TL, MGM965TW, D945GCPE, and DX38BT allows local administrators with ring 0 privileges to gain additional privileges and modify code that is running in System Management Mode, or access hypervisory memory as demonstrated at Black Hat 2008 by accessing certain remapping registers in Xen 3.3.
CVE-2009-0061	Unspecified vulnerability in the Wireless LAN Controller (WLC) TSEC driver in the Cisco 4400 WLC, Cisco Catalyst 6500 and 7600 Wireless Services Module (WiSM), and Cisco Catalyst 3750 Integrated Wireless LAN Controller with software 4.x before 4.2.176.0 and 5.x before 5.1 allows remote attackers to cause a denial of service (device crash or hang) via unknown IP packets.
CVE-2009-0066	Multiple unspecified vulnerabilities in Intel system software for Trusted Execution Technology (TXT) allow attackers to bypass intended loader integrity protections, as demonstrated by exploitation of tboot. NOTE: as of 20090107, the only disclosure is a vague pre-advisory with no actionable information. However, because it is from a well-known researcher, it is being assigned a CVE identifier for tracking purposes.
CVE-2009-0282	Integer overflow in Ralink Technology USB wireless adapter (RT73) 3.08 for Windows, and other wireless card drivers including rt2400, rt2500, rt2570, and rt61, allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via a Probe Request packet with a long SSID, possibly related to an integer signedness error.
CVE-2009-1385	Integer underflow in the e1000_clean_rx_irq function in drivers/net/e1000/e1000_main.c in the e1000 driver in the Linux kernel before 2.6.30-rc8, the e1000e driver in the Linux kernel, and Intel Wired Ethernet (aka e1000) before 7.5.5 allows remote attackers to cause a denial of service (panic) via a crafted frame size.

CVE-2009-1389	Buffer overflow in the RTL8169 NIC driver (drivers/net/r8169.c) in the Linux kernel before 2.6.30 allows remote attackers to cause a denial of service (kernel memory corruption and crash) via a long packet.
CVE-2009-1542	The Virtual Machine Monitor (VMM) in Microsoft Virtual PC 2004 SP1, 2007, and 2007 SP1, and Microsoft Virtual Server 2005 R2 SP1, does not enforce CPU privilege-level requirements for all machine instructions, which allows guest OS users to execute arbitrary kernel-mode code and gain privileges within the guest OS via a crafted application, aka "Virtual PC and Virtual Server Privileged Instruction Decoding Vulnerability."
CVE-2009-2715	Sun VirtualBox 2.2 through 3.0.2 r49928 allows guest OS users to cause a denial of service (Linux host OS reboot) via a sysenter instruction.
CVE-2009-2834	IOKit in Apple Mac OS X before 10.6.2 allows local users to modify the firmware of a (1) USB or (2) Bluetooth keyboard via unspecified vectors.
CVE-2009-3638	Integer overflow in the kvm_dev_ioctl_get_supported_cpuid function in arch/x86/kvm/x86.c in the KVM subsystem in the Linux kernel before 2.6.31.4 allows local users to have an unspecified impact via a KVM_GET_SUPPORTED_CPUID request to the kvm_arch_dev_ioctl function.
CVE-2009-4005	The collect_rx_frame function in drivers/isdn/hisax/hfc_usb.c in the Linux kernel before 2.6.32-rc7 allows attackers to have an unspecified impact via a crafted HDLC packet that arrives over ISDN and triggers a buffer under-read.
CVE-2009-4419	Intel Q35, GM45, PM45 Express, Q45, and Q43 Express chipsets in the SINIT Authenticated Code Module (ACM), which allows local users to bypass the Trusted Execution Technology protection mechanism and gain privileges by modifying the MCHBAR register to point to an attacker-controlled region, which prevents the SENTER instruction from properly applying VT-d protection while an MLE is being loaded.
CVE-2010-0298	The x86 emulator in KVM 83 does not use the Current Privilege Level (CPL) and I/O Privilege Level (IOPL) in determining the memory access available to CPL3 code, which allows guest OS users to cause a denial of service (guest OS crash) or gain privileges on the guest OS by leveraging access to a (1) IO port or (2) MMIO region, a related issue to CVE-2010-0306.
CVE-2010-0306	The x86 emulator in KVM 83, when a guest is configured for Symmetric Multiprocessing (SMP), does not use the Current Privilege Level (CPL) and I/O Privilege Level (IOPL) to restrict instruction execution, which allows guest OS users to cause a denial of service (guest OS crash) or gain privileges on the guest OS by leveraging access to a (1) IO port or (2) MMIO region, and replacing an instruction in between emulator entry and instruction fetch, a related issue to CVE-2010-0298.
CVE-2010-0419	The x86 emulator in KVM 83, when a guest is configured for Symmetric Multiprocessing (SMP), does not properly restrict writing of segment selectors to segment registers, which might allow guest OS users to cause a denial of service (guest OS crash) or gain privileges on the guest OS by leveraging access to a (1) IO port or (2) MMIO region, and replacing an instruction in between emulator entry and instruction fetch.
CVE-2010-0560	Unspecified vulnerability in the BIOS in Intel Desktop Board DB, DG, DH, DP, and DQ Series allows local administrators to execute arbitrary code in System Management Mode (SSM) via unknown attack vectors.
CVE-2010-1085	The azx_position_ok function in hda_intel.c in Linux kernel 2.6.33-rc4 and earlier, when running on the AMD780V chip set, allows context-dependent attackers to cause a denial of service (crash) via unknown manipulations that trigger a divide-by-zero error.
CVE-2010-1592	sandra.sys 15.18.1.1 and earlier in the Sandra Device Driver in SiSoftware Sandra 16.10.2010.1 and earlier allows local users to gain privileges or cause a denial of service (system crash) via unspecified vectors involving "Model-Specific Registers."
CVE-2010-2938	arch/x86/hvm/vmx/vmcs.c in the virtual-machine control structure (VMCS) implementation in the Linux kernel 2.6.18 on Red Hat Enterprise Linux (RHEL) 5, when an Intel platform without Extended Page Tables (EPT) functionality is used, accesses VMCS fields without verifying hardware support for these fields, which allows local users to cause a denial of service (host OS crash) by requesting a VMCS dump for a fully virtualized Xen guest.
CVE-2010-2963	drivers/media/video/v4l2-compat-ioctl32.c in the Video4Linux (V4L) implementation in the Linux kernel before 2.6.36 on 64-bit platforms does not validate the destination of a memory copy operation, which allows local users to write to arbitrary kernel memory locations, and consequently gain privileges, via a VIDIOCSTUNER ioctl call on a /dev/video device, followed by a VIDIOCSMICROCODE ioctl call on this device.
CVE-2010-3448	drivers/platform/x86/thinkpad_acpi.c in the Linux kernel before 2.6.34 on ThinkPad devices, when the X.Org X server is used, does not properly restrict access to the video output control state, which allows local users to cause a denial of service (system hang) via a (1) read or (2) write operation.
CVE-2010-3832	Heap-based buffer overflow in the GSM mobility management implementation in Telephony in Apple iOS before 4.2 on the iPhone and iPad allows remote attackers to execute arbitrary code on the baseband processor via a crafted Temporary Mobile Subscriber Identity (TMSI) field.
CVE-2011-1016	The Radeon GPU drivers in the Linux kernel before 2.6.38-rc5 do not properly validate data related to the AA resolve registers, which allows local users to write to arbitrary memory locations associated with (1)

	Video RAM (aka VRAM) or (2) the Graphics Translation Table (GTT) via crafted values.
CVE-2011-1898	Xen 4.1 before 4.1.1 and 4.0 before 4.0.2, when using PCI passthrough on Intel VT-d chipsets that do not have interrupt remapping, allows guest OS users to gain host OS privileges by "using DMA to generate MSI interrupts by writing to the interrupt injection registers."
CVE-2011-2367	The WebGL implementation in Mozilla Firefox 4.x through 4.0.1 does not properly restrict read operations, which allows remote attackers to obtain sensitive information from GPU memory associated with an arbitrary process, or cause a denial of service (application crash), via unspecified vectors.
CVE-2011-3215	The kernel in Apple Mac OS X before 10.7.2 does not properly prevent FireWire DMA in the absence of a login, which allows physically proximate attackers to bypass intended access restrictions and discover a password by making a DMA request in the (1) loginwindow, (2) boot, or (3) shutdown state.

[i] Common Vulnerability Scoring System; http://www.first.org/cvss
[ii] Common Weakness Enumeration; http://cwe.mitre.org/
[iii] Common Attack Pattern Enumeration and Classification; http://capec.mitre.org/index.html
[iv] Basic Input Output System; http://en.wikipedia.org/wiki/BIOS
[v] System Management Mode; http://en.wikipedia.org/wiki/System_Management_Mode
[vi] Virtual Machine Manager; http://en.wikipedia.org/wiki/Hypervisor
[vii] Trusted Execution Technology; http://en.wikipedia.org/wiki/Trusted_Execution_Technology
[viii] http://en.wikipedia.org/wiki/Pentium_F00F_bug
[ix] http://en.wikipedia.org/wiki/Cyrix_coma_bug
[x] CIA: Confidentiality, Availability, Integrity; http://en.wikipedia.org/wiki/Information_security
[xi] http://cansecwest.com/csw11/Showing%20How%20Security%20Has%20Improved%20-%20Kaminski,%20Cecchetti,%20Eddington.pptx
[xii] "Hacking into Macs is so much easier" [than Windows]; http://www.zdnet.com/blog/security/questions-for-pwn2own-hacker-charlie-miller/2941
[xiii] E.g. http://www.invisiblethingslab.com/resources/2011/Software%20Attacks%20on%20Intel%20VT-d.pdf
[xiv] E.g. http://en.wikipedia.org/wiki/Stuxnet
[xv] Direct Memory Access; http://en.wikipedia.org/wiki/Direct_memory_access
[xvi] System Management Bus; http://en.wikipedia.org/wiki/System_Management_Bus
[xvii] Network Interface Card/Controller; http://en.wikipedia.org/wiki/Network_interface_controller
[xviii] http://en.wikipedia.org/wiki/Extended_Page_Table
[xix] http://www.intel.com/technology/itj/2006/v10i3/2-io/5-platform-hardware-support.htm
[xx] http://en.wikipedia.org/wiki/IOMMU
[xxi] http://www.intel.com/content/www/us/en/pci-express/pci-sig-sr-iov-primer-sr-iov-technology-paper.html
[xxii] http://wiki.xen.org/xenwiki/XenPCIpassthrough
[xxiii] http://wiki.xen.org/xenwiki/XenVGAPassthrough
[xxiv] http://kb.vmware.com/kb/1010789
[xxv] Ring 3 & ring 0; http://en.wikipedia.org/wiki/Ring_(computer_security)
[xxvi] http://theinvisiblethings.blogspot.com/2009/01/why-do-i-miss-microsoft-bitlocker.html
[xxvii] http://en.wikipedia.org/wiki/Cold_boot_attack
[xxviii] http://en.wikipedia.org/wiki/Jailbreak
[xxix] http://en.wikipedia.org/wiki/Confused_deputy_problem
[xxx] http://en.wikipedia.org/wiki/WebGL
[xxxi] http://blogs.technet.com/b/srd/archive/2011/06/16/webgl-considered-harmful.aspx
[xxxii] http://www.windowsecurity.com/whitepapers/Advanced-Code-Injection.html
[xxxiii] http://esec-lab.sogeti.com/dotclear/public/publications/11-recon-stickyfingers_slides.pdf
[xxxiv] http://blog.webroot.com/2011/09/13/mebromi-the-first-bios-rootkit-in-the-wild/
[xxxv] http://en.wikipedia.org/wiki/Advanced_Host_Controller_Interface
[xxxvi] http://msdn.microsoft.com/en-us/library/windows/hardware/ff559309(v=vs.85).aspx
[xxxvii] http://en.wikipedia.org/wiki/Hyper-threading
[xxxviii] http://en.wikipedia.org/wiki/Quiesce

[xxxix] Bus Master Enable; http://en.wikipedia.org/wiki/Bus_mastering
[xl] http://cve.mitre.org/