MILS VIRTUALIZATION FOR INTEGRATED MODULAR AVIONICS

David Kleidermacher, Green Hills Software, Inc. Santa Barbara, CA
Mike Wolf, Green Hills Software, Inc. Santa Barbara, CA

Abstract

Integrated Modular Avionics (IMA) is driving computer systems to manage and protect increasingly high value and complex information and applications across the aircraft. Virtualization is a promising emerging technology in avionics systems as a mechanism for consolidating disparate operating environments and functions onto a single computer. However, from a security perspective, commercial virtualization solutions add to the already fertile attack surface of general-purpose operating systems. This paper will provide an overview of embedded hypervisor technology and propose a MILS- and IMA-compliant virtualization architecture that assures the safety and security of critical applications while incorporating highly functional general purpose virtualized environments. A practical avionics application of this architecture will be presented.

Introduction

Integrated Modular Avionics (IMA) is an architectural approach to building complex systems for avionics. IMA replaces the historical approach of federated systems, in which each function is assigned to its own computer. In the IMA approach, many applications can use the same computer.

In the federated approach, applications (and computers) pass information to one another using physical wiring and an avionics-grade communication protocol, such as ARINC 429 [1, 2]. ARINC 429 is a multidrop protocol that requires a wired connection from each transmitting processor to a set of receiving processors with a maximum of 20 receivers per transmitter. Data rates are relatively slow, with low speed ARINC 429 at 12.5 Kbps and high-speed at 100 Kbps. This all made ARINC 429 costly and limiting.

Modern avionics systems are increasingly adopting faster Aviation Full DulpeX (AFDX) Ethernet switches, routers, and end points [3] that meet ARINC 664 [4] as the defining standard. ARINC 664 requires connections from each endpoint to a router or switch that then allows communication to all other connected devices. In addition to reduced wiring, AFDX allows communications from 10Mbps to 100Mbps.

In the IMA approach, applications on the same processor can communicate using intraprocessor communications mechanisms. This eliminates wired connections and reduces weight and power.

In addition, all of the applications on a given processor can communicate to the AFDX network over a single set of wires. This results in further savings in wiring, weight, and power.

To participate in an IMA architecture computers use an RTOS that manages their use of memory, processor time, cache, network bandwidth, and other resources according to the ARINC 653 “Avionics Application Software Standard Interface” [5]. The standard includes an application programming interface (API) for the ARINC 653 Application Executive (APEX), a portable operating system abstraction layer that defines required and optional services.

Applications developed for IMA targets must then use the APEX API. This makes them more easily portable to other compliant operating systems. Some newer federated applications also use ARINC 653. This makes it possible to build an application to conform to the ARINC 653 standard and then configure it in either a federated or an IMA architecture.

A key concept of ARINC 653 is partitioning of system resources. Partitions are assigned allocations of memory, processing bandwidth, network bandwidth, and file system storage. The system architects assign resources to a partition and the operating system enforces delivery of those resources.

Legacy Applications

When incorporating a legacy avionics application into a modern system, suppliers have had only two options: run the software as a

978-1-4244-2208-1/08/$25.00 ©2008 IEEE.

federated application using the legacy operating system on dedicated hardware; or port the application software (possibly requiring a complete API overhaul) to an IMA operating system. More complex applications—such as Electronic Flight Bag (EFB) and in-flight entertainment systems—that were developed for non-IMA operating systems will often be prohibitively expensive to port.

MILS- and IMA-compliant virtualization provides another answer to this consolidation challenge and creates new architectural opportunities as well.

Virtualization, in the sense we use it in this paper, refers to the ability to create multiple virtual machines from a single physical machine. MILS, which stands for Multiple Independent Levels of Security, is a security system architecture based on software separation and controlled information flow. A MILS-and IMA-compliant virtualization solution would provide an ARINC 653 APEX API for safety-critical applications, would meet the demanding reliability requirements of DO-178B [6], and would meet the important separation and information flow requirements of the MILS architecture. A technology with all those characteristics would be extremely difficult to develop but would open a wealth of architectural options.

MILS Systems

The MILS security architecture is gaining wide acceptance in high robustness environments. High robustness is one of three robustness levels defined by the United States National Security Agency (NSA). The other two are basic and medium robustness. NSA has published guidance documents for establishing consistent requirements for basic robustness [7] and medium robustness [8]. It has not published guidance documents for high robustness, but a paper [9] by Nguyen, Levin, and Irvine of the Naval Postgraduate School describes requirements for high robustness.

Advocates of MILS describe the architecture's principal properties with the acronym NEAT. A MILS system is

- **N**on-bypassable, meaning that there is no way to cause information to flow through the system other than through the defined flow paths;
- **E**valuatable, meaning that components can be independently assessed to a high level of assurance that they are complete, consistent, and correct. For high robustness evaluation, experts agree that a component must be small, on the order of 5,000 lines of code;
- **A**lways-on, meaning that every operation that requires permissions is checked every time the operation is carried out by the appropriate "reference monitor" and;
- **T**amperproof meaning the ability to modify the system's security properties is under control of the system. The requirement to be tamperproof also levies requirements on the development and delivery process as well as the system's operating characteristics.

A complete system based on a MILS architecture would include an operating system, middleware, and application code. The operating system kernel is at the heart of a MILS-architected solution, and a MILS system relies on the properties of the kernel. A MILS kernel, sometimes called a "Separation Kernel", must be designed so that the other components of the solution are isolated. Separation is necessary to ensure that the evaluation of one component is largely independent of other components. Many of the NEAT characteristics of other components can be derived by reference to the properties of the separation kernel.

The INTEGRITY operating system family from Green Hills Software was designed ten years ago with these principles in mind. Engineers have used it successfully in applications ranging from consumer products, to networking, to avionics. INTEGRITY-178B, the high-assurance implementation of the architecture, is in final stages of certification based on a kernel specification called the "Protection Profile for Separation Kernels in Systems Requiring High Robustness" or SKPP. [10]

Protection Profiles

A Protection Profile, such as SKPP, is an implementation-independent requirements specification for a class of products meeting a particular need. Unlike the typical requirement specification, a Protection Profile is written according to a rigorous methodology established in an international standard, the Common Criteria for Information Technology Security Evaluation, (IEC/ISO standard 15408)[11,12,13]. The Protection Profile itself must also be evaluated using another, related international standard, the Common Methodology for Information Technology Security Evaluation (ISO/IEC 18045) [14].

Every Protection Profile contains two sets of requirements: functional requirements and assurance requirements. Functional requirements answer the question "What does the product do?" Assurance requirements answer the question "What information can you provide that will demonstrate that it does what you say?"

SKPP's Protection Profile requirements include separation and information flow control. Meeting the separation requirement ensures that MILS components are isolated from each other, except for policy-defined information flows. These properties go a long way towards ensuring that any MILS component satisfies the ***N***, ***A***, and ***T*** properties of a NEAT system. Evaluation is still required, but the Separation Kernel ensures that independent components can be evaluated independently, which vastly simplifies the process.

Evaluation

Evaluation is the process of testing and verifying claims for a Protection Profile or for an implementation of a product matching a Protection Profile. Evaluation is performed by third parties, not the product's developers, and provides independent validation that the product meets its requirements and claims. The Common Criteria defines Evaluation Assurance Levels (EAL) numbered EAL 1 through EAL 7 with rigor increasing at each level. Products like INTEGRITY that are evaluated against SKPP must meet requirements that substantially exceed EAL 6 and include many important requirements from EAL 7 including the use of formal methods (mathematical proofs of the security properties of the kernel.) The protection profile also includes requirements that go beyond what is required for EAL 7. An evaluation to SKPP represents the world's highest Common Criteria security level for a commercial operating system.

A governmental body oversees each evaluation, and that body is responsible for approving third party evaluators that operate within its borders. Certifications at and below EAL 4 are recognized by all countries who are signatories to the Common Criteria Recognition Agreement (CCRA).

For certifications above EAL 5, the governmental body is more heavily involved and certifications are not automatically recognized by other nations. For a high assurance evaluation in the United States, the National Security Agency (NSA) is responsible for oversight and penetration testing.

Evaluation is a time consuming process with a high benefit to consumers concerned with security. It is possible for anyone to make a security claim, but hard for a buyer to know what it true and what is hype. Continued progress through the evaluation process increases certainty that claims made are reliable. Completion raises certainty still further. INTEGRITY is the first operating system to be accepted for evaluation by the responsible U.S. Government agencies for a high assurance SKPP evaluation. [15]

Virtualization

Virtualization is a broadly used term for the abstraction of computer resources. Some applications of the virtualization concept-—virtual memory for example—are so pervasive that we do not even think of them as virtualization anymore. Today when people think about virtualization they usually refer to the use of one physical computer to create one or more logical environments, each of which acts like a real computer. The real machine is called the host machine. The machines in the created environments are called virtual machines. In the late 1990's VMware proved the commercial practicality of full system virtualization on commodity PC hardware, hosting unmodified, general purpose, "guest" operating systems such as Windows and Linux.

This sort of computer virtualization is not new; IBM pioneered the concept in its mainframes of the 60s and 70s. Computer scientists have long understood many of the applications of virtualization, including the ability to run distinct and legacy operating systems on a single hardware platform, sandboxing untrusted software, server provisioning and consolidation, and enhanced portability of legacy software.

Modern data centers are filled with x86 class servers fitted with virtualization software that can host several x86 class virtual machines. This has made virtual IT into a big business with dozens of players, with VMware the largest and best known. Virtual IT, powered by virtual machines, lets multiple operating environments (e.g. Windows, Linux) run on a single hardware platform with the goal of improving the flexibility and availability of IT resources.

With high-powered server computers, multiple server functions running on common server operating systems, such as Windows Server and UNIX, can be consolidated onto a single platform using virtual machine technology that turns the server operating system into a guest of the underlying virtual machine software (Figure 1). Failures in a server may be handled by restarting or migrating a guest instance to another computer with minimal impact to downtime.

Figure 1. Server Consolidation

Outside of server applications, virtual IT provides consumers with increasingly popular options for flexibility. One example is the use of the Parallels virtual machine technology that runs Windows along side of the native Mac OSX environment on Intel-based Apple desktops and laptops. This lets Apple fans use (when necessary) the Windows environment without requiring a separate PC or a reboot.

Hardware-Assisted Virtualization

On contemporary PC platforms, another use of the virtualization moniker may add to the confusion: Intel's Virtualization Technology (VT). Given the name, one may erroneously assume that PCs containing Intel VT provide built-in virtual machines of the VMware ilk. That is not the case. VT (which includes constituent technologies VT-x and VT-d for virtualized execution and virtualized directed I/O) is a set of hardware acceleration capabilities added to Intel Architecture chips and chipsets. VT makes it easier for virtual machine software—such as Green Hills Software's INTEGRITY Padded Cell, VMware, and Parallels—to provide a fully virtualized PC platform in which one or more unmodified guest operating systems, such as Windows, Linux, and Solaris, can execute with very good performance. Although the concept of hardware-assisted virtualization is not unique to Intel (IBM and Freescale have hardware assisted virtualization support in Power Architecture-based computers), the ubiquitous availability of this technology on standard PCs made possible by Intel (and AMD with its similar Pacifica technology) is helping the world to realize a much wider range of virtualization applications.

With the advent of hardware-assisted virtualization, the virtual machine software has become simpler. The term hypervisor is often used to refer to trimmed down virtualization applications.

Hypervisor Attacks

Some people tout virtualization as another technique in a "layered defense" for system security. The theory proposes that since only the Guest Operating System is exposed to external

threats, an attacker who penetrates the Guest will be unable to subvert the rest of the system including the other Guests. Obviously, if a hacker can subvert the hypervisor itself, then a "guest escape" may be possible, placing all of the guests at risk. An IMA and/or MILS-compliant virtualization solution must ensure that inter-VM intrusions, whether malicious or accidental, cannot occur even if the system is attacked by the most determined, well-funded attackers.

In fact, such escapes are possible with standard commercial virtualization solutions. Commercial virtualization solutions have not met high robustness security requirements and were never designed or intended to meet these levels. A number of studies of virtualization security and successful subversions of hypervisors have been published, including [16] and [17] and more are being published as researchers focus more attention on virtualization. The risk of an "escape" from the virtual machine layer, exposing all the guests is very real and unacceptable for IMA. As one analyst has said, "Virtualization is essentially a new operating system …, and it enables an intimate interaction between underlying hardware and the environment. The potential for messing things up is significant." [18]

At the 2008 Black Hat conference, security researcher Joanna Rutkowska and her team presented three different ways to exploit vulnerabilities in Xen (a widely-used commercial virtualization solution) that could allow the entire computer to be commandeered. [19-22] One of these attacks took advantage of a buffer overflow defect in Xen's (optional) Flask layer. Flask is intended to increase security, not decrease it. This further underscores an important general principle: the more code, the more vulnerability. Software that has not been evaluated to high levels of assurance must be assumed to be hackable by determined and well-resourced entities.

IMA-Compliant Virtualization

An article in the Intel Technology Journal [23] describes three different virtualization architectures. The first and best-known architecture is "OS hosted" (Figure 2). This is the architecture used by VMware's Workstation product. An OS-hosted architecture adds a large body of virtual machine management code to the privileged region of the host operating system kernel--Windows or Linux, for example. Within each virtual machine an instance of the same or different operating system can run. An OS-hosted virtualization solution will not only be exposed to the safety and security vulnerabilities of the underlying host operating system, but also to new vulnerabilities and exploits against. the virtualization code.

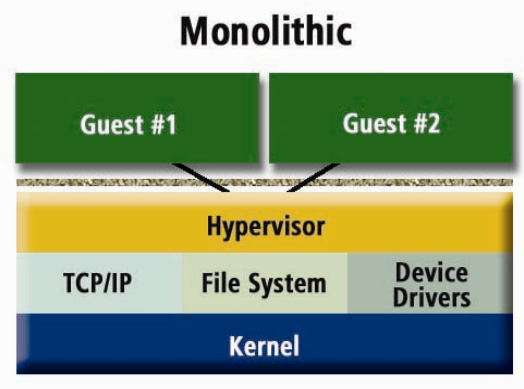

Figure 2. Large Computing Base for Typical Hypervisor

The second approach is a stand-alone or "bare-metal" hypervisor. VMware ESX server is an example of this technology. Instead of an operating system running in privileged space, the hypervisor itself controls the hardware. Some stand-alone hypervisors result in a smaller trusted computing base than OS-hosted systems but still usually consist of millions of bytes of privileged code. The Xen hypervisor is another well-known example of the bare-metal hypervisor. A special, privileged guest, referred to as dom0 or domain 0, handles guest I/O and administrative tasks on behalf of the hypervisor. This adds a lot more code to the trusted computing base and yet more safety and security vulnerabilities (Figure 3).

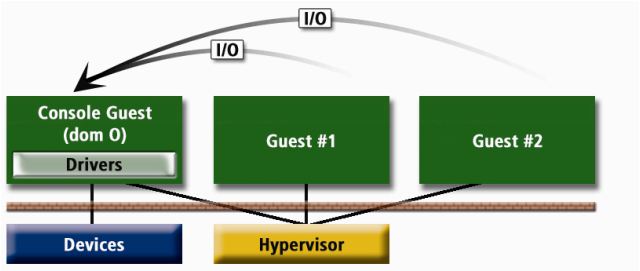

Figure 3. Bare-Metal Hypervisor with Administrative Guest

Enter a third architecture (Figure 4): one that uses a microkernel-based real-time operating system, designed for high assurance applications and capable of meeting IMA and MILS requirements and adapted to virtualization.

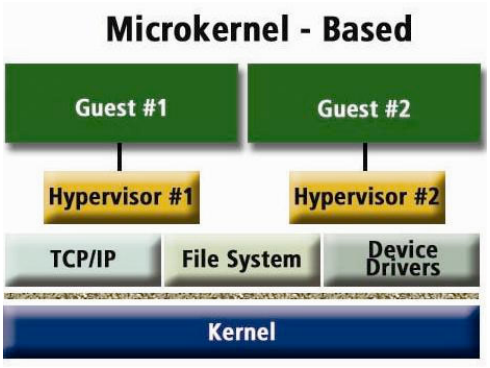

Figure 4. Microkernel-Based Hypervisor

This is the approach taken with Green Hills Software's INTEGRITY PC technology. The microkernel provides an IMA-compliant environment that can host multiple, partitioned ARINC 653 applications as well as virtual machines that execute completely within their own partitions. The Guest OS and its applications can run unmodified in the partition, referred to as a "Padded Cell". The Padded Cell ensures that nothing that happens in the guest environment can escape its confines and affect, much less bring down, a critical partition. The microkernel-based approach enables powerful hybrid systems that combine the most critical applications with general purpose guest environments (Figure 5).

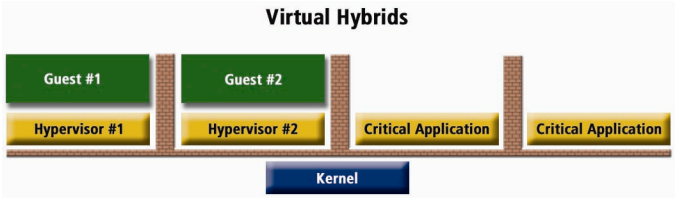

Figure 5. Hybrid Virtualization Environment

MILS-Compliant Virtualization

Building a highly secure system is far harder than building a highly reliable system. To build a reliable system it's necessary to show that the system will do, in every case, exactly what it is supposed to do, despite erroneous data, I/O errors and other unpredictable but not unexpected problems. A secure system must perform correctly in the face of those same problems and in the face of malevolent actions during operation, during deployment, during delivery, and even during development. To build a secure system it's not only necessary to show that not only will the system do what it is supposed to do in all these cases, it's also necessary to show that it will do nothing else.

Example

Electronic Flight Bags (EFBs) are computer-based replacements for the carry-on flight bags traditionally used by pilots. Flight bags (electronic or otherwise) contain reference material such as aeronautical maps and charts, operations and aircraft manuals, forms for fault reporting, minimum equipment lists, and logbooks. Using data in the flight bag, pilots and crew can perform calculations and carry out other activities needed for takeoff and flight. [24]

The idea of an EFB first took shape in the 1990s when some pilots used personal computer and general-purpose software to help them with these activities. Since then, EFBs have become more common and some airlines have entirely replaced the unwieldy paper-based flight bags that sometimes as much as 40 lbs, with EFBs that weigh less and do more. Pilots still use general-purpose software as well as special-purpose software to carry out some activities.

Three hardware classes and three application types are now defined for EFBs. Class 1 devices are standard COTS laptops and handheld computers. Class 2 devices range from modified COTS to special-purpose devices and can be connected to onboard power and data sources. Class 3 devices are considered "installed equipment" and are subject to airworthiness requirements. In some cases, DO-178B requirements are levied on the devices.

Type A applications include those that display documents, replacing heavy paper manuals. Type A can also include software that carries out performance calculations. Type B software contains interactive map software, including real-time weather maps. Type C applications have broader scope and require a Class 3 EFB.

Virtualized EFBs

Virtualization creates an opportunity to add new functionality to EFBs without compromising safety or imposing certification costs on non-critical components. Using an IMA- and MILS-compliant virtualization architecture, general purpose non-critical applications can run under an operating system such as Windows that does not need certification while applications that are more critical can run in safe and secure partitions on the same platform. The architecture can also support a safe, controlled information flow across partitions to make some jobs easier.

For example, pre-flight calculations, carried out using general-purpose software such as a spreadsheet, are transferred manually to flight equipment. A virtualized, IMA-compliant EFB can drastically improve this. First, a pilot would carry out the calculations using commercial grade software on a general-purpose operating system as in the current generation of EFB applications. The software and operating system would run in a Padded Cell partition that protects the rest of the EFB from any problems in these components. Next, the certified IMA operating system transfers the data from that environment to a second partition. A small, safety-certified software component in that partition displays the data and lets the pilot or crewmember confirm that the data displayed matches the data that was calculated. The partition only allows these data values to pass; no malicious content can. Finally, the system moves the data to another safety-certified partition that safely and securely transfers the appropriate information to the on-board avionics.

With this kind of architecture, the only partition (and application) that requires Level A safety certification is the one that communicates with the aircraft systems. The data display partition needs medium to high robustness, but it is a very simple application so this will not be difficult or costly. The general purpose Windows environment need not be certified at all. Yet its powerful software environment is fully utilized by the pilots. The pilots could even send e-mail over the Internet without adding flight risk.

Conclusions

As aircraft systems grow in complexity, IMA must manage and protect increasingly high value and complex information and applications across the aircraft while maintaining system reliability and system safety.

Common commercial virtualization solutions increase the attack surface and cannot be used for safety-rated separation in an IMA system. A new virtualization architecture that is both MILS and IMA-compliant assures the safety and security of critical applications and provides a general-purpose virtualized environment that can support legacy applications or applications designed to use a different operating environment.

References

[1] ARINC Specification 429P1-16, “Mark 33 Digital Information Transfer System (DITS), Part 1, Functional Description, Electrical Interface, Label Assignments and Word Formats” Sept 2001

[2] ARINC Specification 429P2-15, 429P2-15 Mark 33 Digital Information Transfer System (DITS), Part 2 - Discrete Data Standards, March, 1996

[3] ARINC Reports ARINC Report 664P7 Aircraft Data Network, Part 7, Avionics Full Duplex Switched Ethernet (AFDX) Network 06/2005

[4] ARINC Reports ARINC Report 664P1-7 Aircraft Data Network, Parts 1-7, 06/2005.

[5] RTCA, Incorporated, “Airlines Electronic Engineering Committee, Aviation Application Software Standard Interface,” Parts 1 and 2, December 1, 2005, Annapolis, MD, Aeronautical Radio, Inc.

[6] RTCA, Incorporated, document RTCA/DO-178B, “Software Considerations in Airborne Systems and Equipment Certification,” December 1, 1992.

[7] Information Assurance Directorate, Feb 1, 2005, Consistency Instruction Manual For Development of US Government Protection Profiles in Basic Robustness Environments, Release 3.0, http://www.niap-ccevs.org/pp/basic_rob_manual-3.0.pdf

[8] Information Assurance Directorate, Feb 1, 2005, Consistency Instruction Manual For Development of US Government Protection Profiles in Medium Robustness Environments, Release 3.0, http://www.niap-ccevs.org/pp/med_rob_manual-3.0.pdf.

[9] Nguyen, Thuy D., Timothy E. Levin, Cynthia E. Irvine, High Robustness Requirements in a Common Criteria Protection Profile, *Monterey, California, Naval Postgraduate School,* http://cisr.nps.navy.mil/downloads/06paper_highrobust.pdf

[10] Information Assurance Directorate, "U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness," Version 1.03, 29 June 2007,

[11] The Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model Revision 3.1, September 2007, http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R1.pdf

[12] The Common Criteria for Information Technology Security Evaluation Part 2: Introduction and general model Revision 3.1, September 2007, http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R2.pdf

[13] The Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Revision 3.1, September 2007, http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R3.pdf

[14] The Common Criteria for Information Technology Security Evaluation, Evaluation methodology, Version 3.1 Revision 2, September 2007, http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R2.pdf

[15] I*NTEGRITY-178B - NIAP products in evaluation*, http://www.niap-ccevs.org/cc-scheme/in_evaluation/

[16] Samuel King, et al., "SubVirt: Implementing malware with virtual machines", http://www.eecs.umich.edu/virtual/papers/king06.pdf, 2006

[17] Tavis Ormandy, "An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments," http://taviso.decsystem.org/virtsec.pdf, 2007

[18] Rich Ptak quoted by Denise Dubie, "Security concerns cloud virtualization deployments," http://www.networkworld.com/news/2007/112107-security-virtualization.html, *Network World*, November 21, 2007

[19] Rutkowska, Joanna, "Our Xen 0wning Trilogy Highlights," invisiblethings blog, August 08, 2008, http://theinvisiblethings.blogspot.com/2008/08/our-xen-0wning-trilogy-highlights.html

[20] Wojtczuk, Rafał, "Subverting the Xen Hypervisor", Black Hat USA 2008, August 7th, Las Vegas, NV, http://invisiblethingslab.com/bh08/part1.pdf

[21] Rutkowska, Joanna Rutkowska and Wojtczu Rafał, "Preventing and Detecting Xen Hypervisor Subversions" Black Hat USA 2008, August 7th, Las Vegas, NV, http://invisiblethingslab.com/bh08/part2.pdf

[22] Rutkowska, Joanna and Tereshkin, Alexander, "Bluepilling the Xen Hypervisor," Black Hat USA 2008, August 7th, Las Vegas, NV, http://invisiblethingslab.com/bh08/part3.pdf

[23] Abamson, Darren, et al, "Intel® Virtualization Technology for Directed I/O", Intel Technology Journal, Vol 10, Issue 3, August 10, 2006, http://download.intel.com/technology/itj/2006/v10i3/v10-i3-art02.pdf

[24] Wikipedia, "Electronic flight bag," http://en.wikipedia.org/wiki/Electronic_flight_bag