Computer System Administration

Gwyneth Dionne

First Edition, 2012

ISBN 978-81-323-4114-7

Published by:
White Word Publications
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter 1

Out-of-Band Management

In computing, **out-of-band management** (sometimes called **lights-out management** or **LOM**) involves the use of a dedicated management channel for device maintenance. It allows a system administrator to monitor and manage servers and other network equipment by remote control regardless of whether the machine is powered on.

By contrast, in-band management is the use of regular data channels (usually through Ethernet) to manage devices. A significant limitation of in-band management is its vulnerability to problems from the very devices that are being managed. To manage network servers and routers remotely, IT administrators need network access when problems occur. However, the same problems that cause the network to go down also result in the loss of management access to those devices.

Out-of-band management addresses this limitation by employing a management channel that is physically isolated from the data channel.

History

In the early 1980s, the concept of out-of-band was adapted for its natural application across the emerging **data** transmission network structures being introduced with the onset of Ethernet and cost-effective wide area networks. Network architects recognized that this out-of-band alternative pathway was a key requirement in service availability, and they could readily apply many of the lessons learned within the telecoms industry for the previous 30 years. Some of the earliest implementations of a **data** network out-of-band structure included the attachment of a single modem to any given server—in essence creating a very small out-of-band infrastructure (OOBI). Vendors such as IBM, DEC, HP, and Data General made very lucrative service businesses by providing such out-of-band tools as subscription-based services products. The ability to interact remotely with servers that were otherwise compromised was a powerful one which gave rise to the growth of out-of-band as a more general tool for data networks.

In the mid-1980s, Encore Computer released the Annex terminal server, later purchased by Xylogics. The Annex was capable of serving one parallel printer and up to 16 serial printers, terminals, modems, or serial consoles to various equipment. Later models

supported more serial lines, making it the first OOB management server. It also supported a "reverse telnet" (aka. rtelnet) feature that, through a daemon, created a character device file on the Unix host where it ran. Opening this device created a connection to the pre-configured port on the Annex, thus supporting remote kernel debugging, remote modems, etc.

Beginning in the year 2000, the concept was formalized by many early out-of-band infrastructure data pioneers. It was quite clear that this technology was quickly becoming a core IT requirement when dealing with service levels across hundreds or thousands of geographically dispersed IT assets. **OOBI**, uses many of the same concepts and provides similar features to the telecom industry's out-of-band infrastructures. Vendors of **OOBI** solutions began offering these cost-effective alternatives to local administration for data system and network management.

In its original conception, OOBI referred to the physical architecture and components that were used to construct an out-of-band network. A more accurate description would be a *service port network* since the OOBI connected to the service ports rather than the data ports on the target devices. This network provided the platform to implement out-of-band management (or service port management). Just as in the past, a data OOBI provides alternate paths into the production infrastructure for the purpose of allowing disconnected assets to be remotely reconnected and subsequently returned to normal operation, in most cases eliminating the need for costly local administration. Some OOBI implementations include inherent enterprise-class security while others are constrained to the attributes of limited or proprietary mechanisms. An OOBI can improve operational efficiencies, cut costs, improve productivity and, in many cases, improve service levels and asset availability. Conceptually, data OOB infrastructures virtually guarantee a data "dial tone."

An example of a modern, open remote management interface is IPMI, which in physical terms uses data ports but allows their use as service ports, with a low-level route to a baseboard management controller of some sort configured such that OOB services can be accessed separately over the same IP network. While IPMI does provide remote access to computers even when the OS is down, it actually does not provide completely out-of-band access, because it shares the Network Interface Controller with the data pathway. True out of band access is provided by using a dedicated NIC . This is accomplished by using a **Remote Access Card** (RAC), or, more recently, through a Soft Processor leveraging a highly integrated BMC (iBMC).

Types of management systems

A complete LOM system consists of a hardware component called the LOM module and a program that facilitates the continuous monitoring of variables such as microprocessor temperature and utilization. The program also allows for such remote operations as rebooting, shutdown, troubleshooting, alarm setting, fan speed control, and operating system reinstallation. The program often integrates into traditional infrastructure in-band management tools such as HP Openview, Computer Associates, BMC, and Tivoli.

The most common out-of-band management solution involves connecting each device's serial console port to a console server. This implementation allows the monitoring of hardware self-test information and console access that is not available using typical in-band management.

Another type of management solution, a **Remote Access Card** (RAC), involves an expansion card for a computer which has its own processor, memory, battery, network connection, and access to the system bus. This system is effective but costly, and is being progressively supplanted by the use of dedicated Systems on Chip, also called Integrated Baseboard Management Controllers.

Some LOM systems function with more than one server, especially if combined with a KVM switch. When combined with a terminal server, administrators may access all serial console ports in a network or server farm from a single station. If the terminal server is also configured with network, Internet, and dial-up access, administrators will be able to manage network problems from any remote location, even if the network connection has been lost.

Communication between the controller and the remote servers sometimes takes place through an independent dial-up connection. More commonly nowadays, the LOM modules are connected by serial links to a separate management host; or the LOM module accepts telnet connections over an Ethernet connection. Either way, the LOM can then be remotely accessed over the Internet (through SSH to the management host, and/or a VPN). The LOM module keeps a record of all the operations (known as the event log), allowing the administrator to check instantly any or all of several hundred systems.

SoC based service Processors

Modern systems based on System-on-Chip , or iBMCs, usually have separate Ethernet connection that can be implemented either through dedicated or shared Ethernet port. This connection has its own IP address and other network settings. It remains functional also if the server is powered down and can power it up when required. Systems provide remote screen view (both graphical and text modes), remote mouse and keyboard and remote virtual media (including media that exists only as .iso images on the administrator machine). This access is not dependent from the operating system on the server and also works when managing remotely BIOS settings, for instance. IPMI connection is normally encrypted. Recent server boards have all this functionality built-in and do not require any additional extension cards. IPMI system can also be accessed from inside the server if required (for instance, to read the hardware sensor values). Interaction with the system on remote side can be implemented through web browser (including Java applets or JNLP) or specialized tools that may be cross platform. Open source software like FreeIpmi is also available. The most commonly used iBMCs include ASPEED AST 2050, Nuvoton WC450 (Hermon), Renesas 2164, Server Engines Pilot II and Pilot III.

Console redirection

Embedded firmware of most server motherboards support (BIOS) serial console redirection. And boot parameters of modern operating systems can be changed through the boot loader console which supports redirection as well (in Linux this is LILO, GRUB, or SYSLINUX). A Microsoft Windows feature is EMS. Furthermore, Unix-like systems can be configured to log kernel messages to their (serial) console too. The Linux kernel for example logs all messages to all its configured consoles, which can be a combination of virtual terminals (graphics card/keyboard combination), serial ports, parallel ports, etc. Management software such as the Conserver automatically captures this data, and can replay it if needed. When using serial console servers care should be taken not to send any unsolicited BREAK over the line (especially with Sun hardware, and also Linux if SysRq is enabled) as it can put the machine in "LOM mode" otherwise. Some solutions use a Java applet to display remote console view to the administrator.

Limitations

Servicing and managing computer servers in a remote data center can require the physical presence of a system administrator. For example, the loading or removal of media, or direct interaction with the server through a console and keyboard (which should only ever be needed if the CMOS NVRAM becomes corrupted). Such access requirements depend on a system administrator being co-located with the data center, often an additional business expense.

Specific implementations

- Advanced Lights Out Management (ALOM), Sun Microsystems-specific and comes standard on newer Sun servers (SunFire V125/V210/V215/V240/V245/V250/V440/T1000, Sun Netra 210/240/440)
- Integrated Lights Out Management (ILOM), Sun Microsystems's ALOM replacement on Sun x64 server SunFire X4100(M2)/X4200(M2)/X4600(M2)/X4140/X4240/X4440/X4150/X4250/X4450 /X4170/X4270/X2250/X2270, Sun Blade 6000 Chassis Management Module/Blade Module(X6220/X6420/X6240/X6440/X6250/X6450/X6270/X6275), Sun CMT servers/blades (Sun T5120, T5220, T5240, T6340, T6320)
- American Megatrends' MegaRAC-SP firmware, which powers a number of OEM LOM implementations
- Apple Computer's Xserve, which provides lights-out management through the Ethernet network interface.
- Dell DRAC, the Dell-specific "Dell Remote Access Controller;" also called "Dell Remote Assistance Card" or "Dell Remote Access Card" depending on to which version one is referring
- Fujitsu Integrated Remote Management Controller (iRMC) (manual)
- HP Integrated Lights-Out (HP/Compaq specific)
- IBM Remote supervisor adapter (IBM specific)

- Intel Active Management Technology (iAMT)
- Intelligent Platform Management Interface
- LOM port, Sun Microsystems specific used on their older products (Netras)
- Open Platform Management Architecture (Generic interface)
- PC Weasel 2000 (Personal Computer hardware)
- Remote System Control (RSC) on Sun Microsystems SunFire 280R/V480/V490/V880/V890/VSP servers.
- Winbond Hermon
- Baseboard Management Controller (BMC) for IPMI
- Network Console on Acid (Unix tty redirection over SSH)

Chapter 2

Intrusion Detection System

An **intrusion detection system** (**IDS**) is a device or software application that monitors network and/or system activities for malicious activities or policy violations and produces reports to a Management Station. Intrusion prevention is the process of performing intrusion detection and attempting to stop detected possible incidents. Intrusion detection and prevention systems (IDPS) are primarily focused on identifying possible incidents, logging information about them, attempting to stop them, and reporting them to security administrators. In addition, organizations use IDPSs for other purposes, such as identifying problems with security policies, documenting existing threats, and deterring individuals from violating security policies. IDPSs have become a necessary addition to the security infrastructure of nearly every organization.

IDPSs typically record information related to observed events, notify security administrators of important observed events, and produce reports. Many IDPSs can also respond to a detected threat by attempting to prevent it from succeeding. They use several response techniques, which involve the IDPS stopping the attack itself, changing the security environment (e.g., reconfiguring a firewall), or changing the attack's content.

Terminology

- **Alert/Alarm:** A signal suggesting that a system has been or is being attacked.
- **True Positive:** A legitimate attack which triggers an IDS to produce an alarm.
- **False Positive:** An event signaling an IDS to produce an alarm when no attack has taken place.
- **False Negative:** A failure of an IDS to detect an actual attack.
- **True Negative:** When no attack has taken place and no alarm is raised.
- **Noise:** Data or interference that can trigger a false positive.
- **Site policy:** Guidelines within an organization that control the rules and configurations of an IDS.
- **Site policy awareness:** The ability an IDS has to dynamically change its rules and configurations in response to changing environmental activity.
- **Confidence value:** A value an organization places on an IDS based on past performance and analysis to help determine its ability to effectively identify an attack.

- **Alarm filtering:** The process of categorizing attack alerts produced from an IDS in order to distinguish false positives from actual attacks.
- **Attacker** or **Intruder:** An entity who tries to find a way to gain unauthorized access to information, inflict harm or engage in other malicious activities.
- **Masquerader:** A user who does not have the authority to a system, but tries to access the information as an authorized user. They are generally outside users.
- **Misfeasor:** They are commonly internal users and can be of two types:
    1. An authorized user with limited permissions.
    2. A user with full permissions and who misuses their powers.
- **Clandestine user:** A user who acts as a supervisor and tries to use his privileges so as to avoid being captured.

Types

For the purpose of dealing with IT, there are two main types of IDS:

Network intrusion detection system (NIDS)
: It is an independent platform that identifies intrusions by examining network traffic and monitors multiple hosts. Network intrusion detection systems gain access to network traffic by connecting to a network hub, network switch configured for port mirroring, or network tap. In a NIDS, sensors are located at choke points in the network to be monitored, often in the demilitarized zone (DMZ) or at network borders. Sensors captures all network traffic and analyzes the content of individual packets for malicious traffic. An example of a NIDS is Snort.

Host-based intrusion detection system (HIDS)
: It consists of an agent on a host that identifies intrusions by analyzing system calls, application logs, file-system modifications (binaries, password files, capability databases, Access control lists, etc.) and other host activities and state. In a HIDS, sensors usually consist of a software agent. Some application-based IDS are also part of this category. An example of a HIDS is OSSEC.

Intrusion detection systems can also be system-specific using custom tools and honeypots. In the case of physical building security, IDS is defined as an alarm system designed to

Passive and/or reactive systems

In a **passive system**, the intrusion detection system (IDS) sensor detects a potential security breach, logs the information and signals an alert on the console and or owner. In a **reactive system**, also known as an intrusion prevention system (IPS), the IPS auto-responds to the suspicious activity by resetting the connection or by reprogramming the firewall to block network traffic from the suspected malicious source. The term IDPS is commonly used where this can happen automatically or at the command of an operator; systems that both "detect" (alert) and/or "prevent."

Comparison with firewalls

Though they both relate to network security, an intrusion detection system (IDS) differs from a firewall in that a firewall looks outwardly for intrusions in order to stop them from happening. Firewalls limit access between networks to prevent intrusion and do not signal an attack from inside the network. An IDS evaluates a suspected intrusion once it has taken place and signals an alarm. An IDS also watches for attacks that originate from within a system. This is traditionally achieved by examining network communications, identifying heuristics and patterns (often known as signatures) of common computer attacks, and taking action to alert operators. A system that terminates connections is called an intrusion prevention system, and is another form of an application layer firewall.

Statistical anomaly and signature based IDSes

All Intrusion Detection Systems use one of two detection techniques:

Statistical anomaly-based IDS

A statistical anomaly-based IDS determines normal network activity like what sort of bandwidth is generally used, what protocols are used, what ports and devices generally connect to each other- and alert the administrator or user when traffic is detected which is anomalous(not normal).

Signature-based IDS

Signature based IDS monitors packets in the Network and compares with preconfigured and predetermined attack patterns known as signatures. The issue is that there will be lag between the new threat discovered and Signature being applied in IDS for detecting the threat.During this lag time your IDS will be unable to identify the threat.

Limitations

- Noise can severely limit an Intrusion detection system's effectiveness. Bad packets generated from software bugs, corrupt DNS data, and local packets that escaped can create a significantly high false-alarm rate.
- It is not uncommon for the number of real attacks to be far below the false-alarm rate. Real attacks are often so far below the false-alarm rate that they are often missed and ignored.
- Many attacks are geared for specific versions of software that are usually outdated. A constantly changing library of signatures is needed to mitigate threats. Outdated signature databases can leave the IDS vulnerable to new strategies.

Evasion techniques

Intrusion detection system evasion techniques bypass detection by creating different states on the IDS and on the targeted computer. The adversary accomplishes this by manipulating either the attack itself or the network traffic that contains the attack.

Development

A preliminary concept of an IDS began and reviews of audit trails. An example of an audit trail would be a log of user access.

Fred Cohen noted in 1984 that it is impossible to detect an intrusion in every case and that the resources needed to detect intrusions grows with the amount of usage.

Dorothy E. Denning, assisted by Peter G. Neumann, published a model of an IDS in 1986 that formed the basis for many systems today. Her model used statistics for anomaly detection, and resulted in an early IDS at SRI International named the Intrusion Detection Expert System (IDES), which ran on Sun workstations and could consider both user and network level data. IDES had a dual approach with a rule-based Expert System to detect known types of intrusions plus a statistical anomaly detection component based on profiles of users, host systems, and target systems. Lunt proposed adding an Artificial neural network as a third component. She said all three components could then report to a resolver. SRI followed IDES in 1993 with the Next-generation Intrusion Detection Expert System (NIDES).

The Multics intrusion detection and alerting system (MIDAS), an expert system using P-BEST and Lisp, was developed in 1988 based on the work of Denning and Neumann. Haystack was also developed this year using statistics to reduce audit trails.

Wisdom & Sense (W&S) was a statistics-based anomaly detector developed in 1989 at the Los Alamos National Laboratory. W&S created rules based on statistical analysis, and then used those rules for anomaly detection.

In 1990, the Time-based Inductive Machine (TIM) did anomaly detection using inductive learning of sequential user patterns in Common Lisp on a VAX 3500 computer. The Network Security Monitor (NSM) performed masking on access matrices for anomaly detection on a Sun-3/50 workstation. The Information Security Officer's Assistant (ISOA) was a 1990 prototype that considered a variety of strategies including statistics, a profile checker, and an expert system. ComputerWatch at AT&T Bell Labs used statistics and rules for audit data reduction and intrusion detection.

Then, in 1991, researchers at the University of California, Davis created a prototype Distributed Intrusion Detection System (DIDS), which was also an expert system. The Network Anomaly Detection and Intrusion Reporter (NADIR), also in 1991, was a prototype IDS developed at the Los Alamos National Laboratory's Integrated Computing

Network (ICN), and was heavily influenced by the work of Denning and Lunt. NADIR used a statistics-based anomaly detector and an expert system.

The Lawrence Berkeley National Laboratory announced Bro in 1998, which used its own rule language for packet analysis from libpcap data. Network Flight Recorder (NFR) in 1999 also used libpcap. APE was developed as a packet sniffer, also using libpcap, in November, 1998, and was renamed Snort one month later, and has since become the world's largest used IDS/IPS system with over 300,000 active users.

The Audit Data Analysis and Mining (ADAM) IDS in 2001 used tcpdump to build profiles of rules for classifications.

In 2003, Dr. Yongguang Zhang and Dr. Wenke Lee argue for the importance of IDS in networks with mobile nodes.

Chapter 3

Computer Performance and Network Management

Computer performance

Computer performance is characterized by the amount of useful work accomplished by a computer system compared to the time and resources used.

Depending on the context, good computer performance may involve one or more of the following:

- Short response time for a given piece of work
- High throughput (rate of processing work)
- Low utilization of computing resource(s)
- High availability of the computing system or application
- Fast (or highly compact) data compression and decompression
- High bandwidth / short data transmission time

Performance metrics

Computer performance metrics include availability, response time, channel capacity, latency, completion time, service time, bandwidth, throughput, relative efficiency, scalability, performance per watt, compression ratio, instruction path length and speed up. CPU benchmarks are available.

Aspect of software quality

Computer software performance, particularly software application response time, is an aspect of software quality that is important in human–computer interactions.

Technical and non-technical definitions

The performance of any computer system can be evaluated in measurable, technical terms, using one or more of the metrics listed above. This way the performance can be

- compared relative to other systems or the same system before/after changes
- defined in absolute terms, e.g. for fulfilling a contractual obligation

Whilst the above definition relates to a scientific, technical approach, the following definition given by Arnold Allen would be useful for a non-technical audience:

The word performance *in computer performance means the same thing that performance means in other contexts, that is, it means "How well is the computer doing the work it is supposed to do?"*

Technical performance metrics

There is a wide variety of technical performance metrics that indirectly affect overall computer performance.

Because there are too many programs to test a CPU's speed on all of them, benchmarks were developed. The most famous benchmarks are the SPECint and SPECfp benchmarks developed by Standard Performance Evaluation Corporation and the ConsumerMark benchmark developed by the Embedded Microprocessor Benchmark Consortium EEMBC.

Some important measurements include:

- Instructions per second – Most consumers pick a computer architecture (normally Intel IA32 architecture) to be able to run a large base of pre-existing, pre-compiled software. Being relatively uninformed on computer benchmarks, some of them pick a particular CPU based on operating frequency.
- FLOPS – The number of floating-point operations per second is often important in selecting computers for scientific computations.
- Performance per watt – System designers building parallel computers, such as Google, pick CPUs based on their speed per watt of power, because the cost of powering the CPU outweighs the cost of the CPU itself.
- Some system designers building parallel computers pick CPUs based on the speed per dollar.
- System designers building real-time computing systems want to guarantee worst-case response. That is easier to do when the CPU has low interrupt latency and when it has deterministic response. (DSP)
- Computer programmers who program directly in assembly language want a CPU to support a full-featured instruction set.
- Low power – For systems with limited power sources (e.g. solar, batteries, human power).

- Small size or low weight - for portable embedded systems, systems for spacecraft.
- Environmental impact – Minimizing environmental impact of computers during manufacturing and recycling as well as during use. Reducing waste, reducing hazardous materials.
- Giga-updates per second - a measure of how frequently the RAM can be updated

Occasionally a CPU designer can find a way to make a CPU with better overall performance by improving one of these technical performance metrics without sacrificing any other (relevant) technical performance metric—for example, building the CPU out of better, faster transistors. However, sometimes pushing one technical performance metric to an extreme leads to a CPU with worse overall performance, because other important technical performance metrics were sacrificed to get one impressive-looking number—for example, the megahertz myth.

The total amount of time (**t**) required to execute a particular benchmark program is

$$t = N * C / f$$

where

- **N** is the number of instructions actually executed (the instruction path length). The code density of the instruction set strongly affects N. The value of N can either be determined **exactly** by using an instruction set simulator (if available) or by estimation—itself based partly on estimated or actual frequency distribution of input variables and by examining generated machine code from an HLL compiler. It cannot be determined from the number of lines of HLL source code. N is not affected by other processes running on the same processor. The significant point here is that hardware normally does not keep track of (or at least make easily available) a value of N for executed programs. The value can therefore only be accurately determined by instruction set simulation, which is rarely practiced.

- **f** is the clock frequency in cycles per second.

- **C** is the average cycles per instruction (CPI) for this benchmark.

Even on one machine, a different compiler or the same compiler with different compiler optimization switches can change N and CPI—the benchmark executes faster if the new compiler can improve N or C without making the other worse, but often there is a trade-off between them—is it better, for example, to use a few complicated instructions that take a long time to execute, or to use instructions that execute very quickly, although it takes more of them to execute the benchmark?

A CPU designer is often required to implement a particular instruction set, and so cannot change N. Sometimes a designer focuses on improving performance by making significant improvements in f (with techniques such as deeper pipelines and faster caches), while (hopefully) not sacrificing too much C—leading to a speed-demon CPU

design. Sometimes a designer focuses on improving performance by making significant improvements in CPI (with techniques such as out-of-order execution, superscalar CPUs, larger caches, caches with improved hit rates, improved branch prediction, speculative execution, etc), while (hopefully) not sacrificing too much clock frequency—leading to a brainiac CPU design.

Network management

Network management refers to the activities, methods, procedures, and tools that pertain to the operation, administration, maintenance, and provisioning of networked systems.

- Operation deals with keeping the network (and the services that the network provides) up and running smoothly. It includes monitoring the network to spot problems as soon as possible, ideally before users are affected.
- Administration deals with keeping track of resources in the network and how they are assigned. It includes all the "housekeeping" that is necessary to keep the network under control.
- Maintenance is concerned with performing repairs and upgrades—for example, when equipment must be replaced, when a router needs a patch for an operating system image, when a new switch is added to a network. Maintenance also involves corrective and preventive measures to make the managed network run "better", such as adjusting device configuration parameters.
- Provisioning is concerned with configuring resources in the network to support a given service. For example, this might include setting up the network so that a new customer can receive voice service.

A common way of characterizing network management functions is FCAPS—Fault, Configuration, Accounting, Performance and Security.

Functions that are performed as part of network management accordingly include controlling, planning, allocating, deploying, coordinating, and monitoring the resources of a network, network planning, frequency allocation, predetermined traffic routing to support load balancing, cryptographic key distribution authorization, configuration management, fault management, security management, performance management, bandwidth management, Route analytics and accounting management.

Data for network management is collected through several mechanisms, including agents installed on infrastructure, synthetic monitoring that simulates transactions, logs of activity, sniffers and real user monitoring. In the past network management mainly consisted of monitoring whether devices were up or down; today performance management has become a crucial part of the IT team's role which brings about a host of challenges—especially for global organizations.

Note: Network management does not include user terminal equipment.

Technologies

A small number of accessories methods exist to support network and network device management. Access methods include the SNMP, command-line interface (CLIs), custom XML, CMIP, Windows Management Instrumentation (WMI), Transaction Language 1, CORBA, NETCONF, and the Java Management Extensions (JMX).

Schemas include the WBEM, the Common Information Model, and MTOSI amongst others.

Medical Service Providers provide a niche marketing utility for managed service providers; as HIPAA legislation consistently increases demands for knowledgeable providers. Medical Service Providers are liable for the protection of their clients confidential information, including in an electronic realm. This liability creates a significant need for managed service providers who can provide secure infrastructure for transportation of medical data.

Chapter 4

Automounter

An **automounter** is any program or software facility which automatically mounts filesystems in response to access operations by user programs. An automounter system utility (daemon under Unix), when notified of file and directory access attempts under selectively monitored subdirectory trees, dynamically and transparently makes local or remote devices accessible.

The automounter has the purpose of conserving local system resources and of reducing the coupling between systems which share filesystems with a number of servers. For example, a large to mid-sized organization might have hundreds of file servers and thousands of workstations or other nodes accessing files from any number of those servers at any time. Usually, only a relatively small number of remote filesystems (*exports*) will be active on any given node at any given time. Deferring the mounting of such a filesystem until a process actually needs to access it reduces the need to track such mounts, increasing reliability, flexibility and performance.

Frequently, one or more fileservers will become inaccessible (down for maintenance, on a remote and temporarily disconnected network, or accessed via a congested link). Administrators also often find it necessary to relocate data from one file server to another - to resolve capacity issues and balance the load. Having data mount-points automated makes it easier to reconfigure client systems in such cases. In addition, end-users should only have the ability to mount some removable media devices - such as floppies, CD-ROMs, and USB keys - when the device is attached to the system.

These factors combine to pose challenges to older "static" management methods of filesystem mount tables (the `fstab` files on Unix systems). Automounter utilities address these challenges and allow sysadmins to consolidate and centralize the associations of mountpoints (directory names) to the exports. When done properly, users can transparently access files and directories as if all of their workstations and other nodes attach to a single enterprise-wide filesystem.

One can also use automounters to define multiple repositories for read-only data; client systems can automatically choose which repository to mount based on availability, file-server load, or proximity on the network.

Home directories

Many establishments will have a number of file servers which host the home directories of various users. All workstations and other nodes internal to such organizations (typically all those behind a common firewall separating them from the Internet) will be configured with automounter services so that any user logging into any node implicitly triggers access to his or her own home directory which, consequently, is mounted at a common mountpoint, such as `/home/user`. This allows users to access their own files from anywhere in the enterprise, which is extremely useful in Unix environments, where users may frequently invoke commands on many remote systems via various job-dispatching commands such as `ssh`, `telnet`, `rsh` or `rlogin`, or via the X11 or VNC protocols.

/net

A very common default automounter local path is of the form `/net/hostname/nfspath` where `hostname` is the host name of the remote machine and `nfspath` is the path that is exported over NFS on the remote machine. This notation generally frees the system manager from having to manage each exported path explicitly via a central automounter map.

Software shares and repositories

In some computing environments, user workstations and computing nodes do not host installations of the full range of software that users might want to access. Systems may be "imaged" with a minimal or typical cross-section of the most commonly used software. Also, in some environments, users might require specialized or occasional access to older versions of software (for instance, developers may need to perform bug fixes and regression testing, or some users may need access to archived data using outdated tools).

Commonly, organizations will provide repositories or "depots" of such software, ready for installation as required. These also may include full copies of the system images from which machines have their operating systems initially installed, or available for repair of any system files that may get corrupted during a machine's lifecycle.

Some software may require quite substantial storage space or might be undergoing rapid (perhaps internal) development. In those cases the software may be installed on, and configured to be run directly from, the file servers.

Dynamically variant automounts

In the simplest case, a fileserver houses data and perhaps scripts which can be accessed by any system in an environment. However, certain types of files (executable binaries and shared libraries, in particular) can only be used by specific types of hardware or specific versions of specific operating systems.

For situations like this, automounter utilities generally support some means of "mapping" or "interpolating" variable data into the mount arguments.

For example, an organization with a mixture of Linux and Solaris systems might arrange to host their software package repositories for each on a common file-server using export names like `depot:/export/linux` and `depot:/export/solaris` respectively. Thereunder they might have directories for each of the OS versions that they support. Using the dynamic variation features in their automounter, they might then configure all their systems so that any administrator on any machine in their enterprise could access available software updates under `/software/updates`. A user on a Solaris system would find the Solaris compiled packages under `/software`, while a Red Hat or CentOS (Linux) user would find RPMs for their particular OS version thereunder. Moreover, a Solaris user on a SPARC workstation would have his `/software/updates` mapped to an appropriate export for that system's architecture, while a Solaris user on an x86 PC would transparently find his `/software/updates` directory containing packages suited to his system. Some software (written in scripting languages such as Perl or Python) can be installed and/or run on any supported platform without porting, recompilation or re-packaging of any sort. A systems administrator might conceivably locate such software in a `/software/common` export.

In some cases, organizations may also use regional or location-based variable/dynamic mappings — so that users in one building or site are directed to a closer file server which hosts replications of the resources that are hosted at other locations.

In all of these cases, automounter utilities allow the users to access files and directories without regard for the actual physical location. Using an automounter, the users and systems administrators can usually access files where they are "supposed to be" and find that they appear to be there.

Software

Tom Lyon developed the original automount software at Sun Microsystems: SunOS 4.0 made automounting available in 1988. Sun Microsystems eventually licensed this implementation to other commercial UNIX distributions. Solaris 2.0, first released in 1992, implemented its automounter with a pseudofilesystem called `autofs`, which communicates with a user-mode daemon that performs mounts. Other Unix-like systems have adopted that implementation of the automounter - including AIX, HP-UX, and Mac OS X 10.5 and later.

Linux has an independent implementation of an autofs-based automounter; version 5 of that automounter generally operates compatibly with the Solaris automounter.

In December 1989 Jan-Simon Pendry released *amd*, an automounter "based in spirit" on the SunOS automount program. *amd* has also become known as the Berkeley Automounter.

Some operating systems also support automatic mounting of external drives (such as disk drives or flash drives that use FireWire or USB connections) and removable media (such as CDs and DVDs). This technology differs from the automounting described here; it involves mounting local media when the user attaches them to or inserts them into the system, rather than mounting directories from remote file servers when a reference is made to them. Linux currently (as of Linux 2.6) uses the user-space program udev for this form of automounting. Some automounting functions have been implemented in the separate program HAL, but As of 2010 are being merged into udev. OpenBSD has hotplugd(8) which triggers special scripts on attach or detach of removable devices, so that user can easily add mounting of removable drives. In Mac OS X `diskarbitrationd` carries out this form of automatic mounting.

Disadvantages and caveats

While automounter utilities (and remote filesystems in general) can provide centrally managed, consistent and largely transparent access to an organization's storage services, they also can have their downsides:

- Access to automounted directories can trigger delays while the automounter resolves the mapping and mounts the export into place.
- Timeouts can cause the unmounting of mounted directories (which situation can later result in mount delays upon the next attempted access).
- The mapping of mountpoint to export arguments is usually done via some directory service such as LDAP or NIS, which constitutes another dependency (potential point of failure).
- When some systems require frequent access to some resources, while others only need occasional access, this can pose difficult or impossible problems in implementing a consistent, enterprise-wide mixture of locally "mirrored" (replicated) and automounted directories.
- When data is migrated from one file server (export) to another, there can be an indeterminate number of systems which, for various reasons, still have an active mount on the old location ("stale NFS mounts"); these can cause issues which may even necessitate the reboot of otherwise perfectly stable hosts.
- Organizations can find that they've created a "spaghetti" of mappings which can entail considerable management overhead and sometimes quite a bit of confusion among users and administrators.
- Users can become so accustomed to the transparency of automounted resources that they neglect to consider some of the differences in access semantics that may apply to networked filesystems, as compared to locally mounted devices. In particular, programmers may attempt to use "locking" techniques which are safe and provide the desired atomicity guarantees on local filesystems, but which are documented as inherently vulnerable to race conditions when used on NFS.

Chapter 5

Application Performance Management and Business Transaction Management

Application performance management

Application performance management, or **APM**, refers to the discipline within systems management that focuses on monitoring and managing the performance and service availability of software applications.

APM can be defined as process and use of related IT tools to detect, diagnose, remedy and report application's performance to ensure that it meets or exceeds end-users' and businesses' expectations. Application performance relates to how fast transactions are completed on behalf of, or information is delivered to the end user by the application via a particular network, application and/or web services infrastructure.

Methods for Measuring Performance

There are two main methods by which applications performance is assessed for production applications. The first is measuring the resources used by the application, has been in use since computers have been used for business applications, and is still in use. The second is measuring the response time of applications from the perspective of the end user.

Application performance management is related to end-user experience management and real user management in that measuring the experience of real users in the use of an application in production is considered by many as being the most valid method of assessing the performance of an application in production.

Platforms

The use of application performance management is common for web applications written to JEE and Microsoft .NET platforms. All of the leading systems management vendors

have JEE and .NET APM products in their portfolios. These APM for JEE and .NET based applications have the advantage of being able to measure response time from the perspective of the web server, and being able to provide root cause analysis for the likely causes of performance issues within the applications code executing in the JEE or .NET environment. Many of these products also have connectors that monitor the transaction flow from the business logic layer of the application to the database server, or to external interfaces like web services. Some of these vendors also have HTTP appliances in their product line that can decode transaction specific response times at the web server layer.

Dependency injection software development frameworks on JEE instrument an application to provide performance metrics automatically. For example, Spring-based JEE applications support management protocols to provide observed issues in application operation to a performance management tool/dashboard. SpringSource acquired APM-player Hyperic in 2009 to combine application development, automatic application instrumentation, and application performance management. Aspect Oriented Programming on JEE platforms enables automatic performance monitoring without instrumentation of the application. PushToTest TestMaker is an open source load testing solution that integrates with Glassbox, an open source application performance monitoring and troubleshooter application.

Current Issues

The difficult issues in APM currently revolve around two trends in the IT industry. The first is that for many enterprises, only a small fraction of their business critical applications are web based and written to JEE or .NET. For these enterprises who may have business critical applications like SAP that use "fat" Win32 clients, their APM need can only be met by engaging with vendors offering deep End User Experience monitoring for a specific set of enterprise applications. The second issue is that many applications systems are being virtualized, which has the effect of breaking the validity of time based performance metrics gathered within the guest OS where the application is running. This requires a totally new approach to APM tuned to the requirements of virtualized systems.

Five Functional Dimensions of APM

According to Gartner research, Application Performance Management includes 5 distinct functional dimensions:

- End-user experience monitoring
- Application runtime architecture discovery and modeling
- User-defined transaction profiling (Also called Business Transaction Management)
- Application component deep-dive monitoring
- Application data analytics

Business transaction management

Business transaction management (BTM), also known as **business transaction monitoring**, **application transaction profiling** or **user defined transaction profiling**, is the practice of managing information technology (IT) from a business transaction perspective. It provides a tool for tracking the flow of transactions across IT infrastructure, in addition to detection, alerting, and correction of unexpected changes in business or technical conditions. BTM provides visibility into the flow of transactions across infrastructure tiers, including a dynamic mapping of the application topology.

Using BTM, application support teams are able to search for transactions based on message context and content – for instance, time of arrival or message type – providing a way to isolate causes for common issues such as application exceptions, stalled transactions, and lower-level issues such as incorrect data values.

The ultimate goal of BTM is to improve service quality for users conducting business transactions while improving the effectiveness of the IT applications and infrastructure across which those transactions execute. The main benefit of BTM is its capacity to identify precisely where transactions are delayed within the IT infrastructure. BTM also aims to provide proactive problem prevention and the generation of business service intelligence for optimization of resource provisioning and virtualization.

A number of factors have led to the demand for the development of BTM software:

- Modern applications have become more complex, modular, distributed, interdependent and sensitive to environmental conditions.
- IT infrastructure has become a complex multi-tier environment.
- The rise of service-oriented architecture in systems development .
- The proliferation of service level agreements.

Applications

BTM solutions capture all of the transaction instances in the production environment and as such can be used for monitoring as well as for analysis and planning. Some applications include:

- Outage avoidance and problem isolation: Identification and isolation of tier-specific performance and availability issues.
- Service level management: Monitoring of SLAs and alerting of threshold breaches both at the end-user and infrastructure tier level.
- Infrastructure optimization: Modification of the configuration of data center infrastructure to maximize utilization and improve performance.
- Capacity planning: Analysis of usage and performance trends in order to estimate future capacity requirements.
- Change management: Analysis of the impact of change on transaction execution.

- Cloud management: Track the end-to-end transaction flow across both cloud (private, hybrid, public) and dedicated (on-premise, off-premise) infrastructure.

Transaction discovery methods

BTM systems track each of the hops in the transaction path using a variety of data collection methods including OS-level sockets, network packet sniffing, log parsing, agent-based middleware protocol sniffing, and others.

Relationship to application performance management

BTM is sometimes categorized as a form of application performance monitoring or management. It works alongside other IT monitoring systems including End-User Experience Monitoring, Synthetic Transaction Monitoring, Deep-Dive Monitoring and Business Activity Monitoring (BAM) solutions. According to Gartner, BTM and deep dive monitoring are "fundamentally distinct and their associated processes are typically carried out by different communities with different skill sets… The buyer should still implement multiple products, even if it means greater architectural complexity and apparent functional overlap."

Relationship to virtualization and cloud computing

BTM dynamically maps the execution of a user transaction as it traverses the data center. In both virtualized and cloud environments, the relationship between the application and infrastructure is to some degree dynamically allocated or defined. BTM discovers the infrastructure currently executing each transaction instance for purposes of problem identification, resolution, and infrastructure tuning. In public and hybrid cloud architectures, BTM has the ability to profile transactions from the datacenter, to the cloud provider, and back.

Chapter 6

Deep Freeze (Software) and Multiseat Configuration

Deep Freeze (software)

Deep Freeze, by Faronics, is an application available for the Microsoft Windows, Mac OS X, and SUSE Linux operating systems which allows system administrators to protect the core operating system and configuration files on a workstation or server by restoring a computer back to its original configuration each time the computer restarts.

Operation

Deep Freeze is a kernel-level driver that protects hard drive integrity by redirecting information being written to the hard drive or partition, leaving the original data intact. This redirected information is no longer referenced once the computer is restarted, thus restoring the system to its original state at the disk sector level. This allows users to make 'virtual' changes to the system, giving them the appearance that they can modify core files or even delete them, and even make the system unusable to themselves, but upon reboot the originally configured 'frozen' state of the operating system is restored.

To make changes, a system administrator must 'thaw' the protected partition by disabling Deep Freeze, make any needed changes, and then 'freeze' it again by re-enabling Deep Freeze. These changes become part of the protected partition and will be maintained after restarts. 'Freezing' and 'thawing' can be done at the workstation level or remotely via either the Faronics Core management platform or the Deep Freeze Enterprise Console. Users of the Enterprise version can also create virtual partitions called ThawSpaces (of up to 1 TB on an NTFS-formatted drive) to retain data on "frozen" hard drives after restarts.

Deep Freeze can also protect a computer from harmful malware as it automatically deletes (or rather, no longer 'sees') downloaded files when the computer is restarted. The advantage to using Deep Freeze as an antivirus/antimalware application is that it uses

almost no system resources, and does not slow down the computer noticeably. The disadvantage is that it does not provide real-time protection, therefore an infected computer would have to be restarted in order to remove malware.

Limitations and security

Deep Freeze only protects workstations in a "fresh-booted" state. That is, Deep Freeze prevents permanent tampering with protected hard drives/partitions across reboots, but user activity between restarts is not limited by the program. For example, Deep Freeze does not prevent application installation; a user can install a modified version of a Web browser (but seemingly harmless to the unknowing user) designed to secretly send users' passwords to a server connected to the Internet. As a workaround, Deep Freeze can be configured to restart after user logout, shutdown after a chosen period of inactivity, or restart/shutdown at a scheduled time in an attempt to ensure that no such installations are retained (as rebooting the system returns the system to its original, unmodified state).

Deep Freeze cannot protect the operating system and hard drive upon which it is installed if the computer is booted from another medium (such as an external hard drive, a USB device, optical media, or network server). In such cases, a user would have real access to the contents of the (supposedly) frozen system. On a Windows-based computer, this scenario may be prevented by configuring the CMOS (nonvolatile BIOS memory) on the workstation to boot only to the hard drive to be protected, then password protecting the CMOS. This is a normal precaution for most public access computers. A further precaution would be to lock the PC case shut with a physical lock or tiedown cable system to prevent access to motherboard jumpers.

Deep Freeze can only protect hard drive partitions of up to a 2 TB capacity (using NTFS).

Competitors

There are sandboxing and virtualization products which have similar features to what Deep Freeze offers but do not employ the same redirection process. These include:

- Rollback Rx
- Clean Slate (Fortres Grand)
- HDGUARD
- Returnil Virtual System (Returnil)
- Sandboxie (Ronen Tzur)
- Shadow Defender (ShadowDefender.Net)
- SmartShield (Centurion Technologies)
- Windows SteadyState (Microsoft)
- System Revert

Multiseat configuration

A four-head multiterminal.

A **multiseat**, **multi-station** or **multiterminal** configuration is a single computer which supports multiple independent users at the same time. In modern usage the terms refer to multiple users using one personal computer, each with their own console, consisting of a keyboard a mouse, a monitor, and possibly headphones.

Motivation

With the increasing capacity of processors and memory, commodity personal computers can now perform significant numbers of tasks simultaneously without slowing down. However, using standard computer configurations, only one user is able to use the computer at a time, limiting the effectiveness of the system as it remains idle most of the time. With a multiterminal, a lot of users can share the same computer, so more of its total capacity is going to be used. For example, if someone is just using a web browser or word processor, no one else can use the computer and 90% of the system's resources may be idle - but with multiterminals, other people will be able to use the otherwise idle resources. However, if someone is using all of the system's resources (playing a resource-intensive computer game, for example) the other users will have a very slow system.

Multiseats are also more cost-effective: it is not necessary to buy separate motherboards, microprocessors, RAM, hard disks and other components for each user. For example, buying one high speed CPU usually costs less than buying several slower CPUs.

History

In the 1970s, it was very commonplace to connect multiple computer terminals to a single mainframe computer, even graphical terminals. Early terminals were connected with RS-232 type serial connections, either directly, or through modems. With the advent of Internet Protocol based networking, it became possible for multiple users to log in to a

host using telnet or – for a graphic environment – a X Window "server". These systems would retain a physically secure "root console" for system administration and direct access to the host machine.

Support for multiple consoles in a PC running the X Window interface was implemented in 2001 by Miguel Freitas, using the Linux operating system and the X11 graphical system (in that age maintained by XFree86). This was done using a patch in the X server to execute several instances of X at the same time such that each one captures specific mouse and keyboard events and the graphical content. This method received the name of multiseat or multiterminal.

In 2002 a Canadian company, Userful Corporation, released Userful Multiplier, a multiseat Linux software solution that enables up to 10 users to simultaneously share one computer. Earlier they worked on a kernel-based approach to a multi-station platform computer, but abandoned the idea due to a problem with multiple video card support.

Other solutions appeared in 2003, such Svetoslav Slavtchev, Aivils Stoss and James Simmons worked, with the evdev and Faketty approach modifying the kernel Linux and letting more than one user independently use the same machine. In that time, the Linux Console Project also proposed an idea to use multiple independent consoles and then multiple independent keyboards and mice in a project called "Backstreet Ruby". Backstreet Ruby is a kernel patch for the Linux kernel. It is a back port to Linux-2.4 of the Ruby kernel tree. The aim of the Linux Console developers is to enhance and reorganize the input, the console and the framebuffer subsystems in the Linux kernel, so they can work independent from each other and to allow multi-desktop operation. The Backstreet Ruby idea was never finished.

In 2005, the team of C3SL (Center for Scientific Computing and Free Software), from Federal University of Parana in Brazil, created the solution based with nested X servers, such Xnest and Xephyr. With this solution, each nested X server runs in each screen of a host X server (e.g. Xorg) and a modification in the nested servers let it get the exclusivity of each set of mouse and keyboard. In 2008, the C3SL group releases the Multiseat Display Manager (MDM) to ease the process of installation and configuration of a multiseat box. This group, also in 2008, conceived a live-CD for tests purposes.

Multiseat was a planned feature for Fedora 12 but did not materialize and is currently pending.

Time line, commercial multiseat software evolution

- 1996, ThinSoft/BeTwin
- 2002, Userful Corporation
- 2004, Open-Sense Solutions (Groovix)
- 2006, NComputing
- 2010, Microsoft
- 2011, Black Box VirtuaCore

Requirements

Hardware requirements

Each monitor will need to be connected to a graphics output from a video card. For example, to make a *four-head* (four users), would require four monitors, four keyboards, four mice and two dual or one quad output video card (five users can be accommodated if onboard video off the motherboard is available). USB keyboards and mice are typically recommended instead of PS/2 connections, as they can be connected to a USB hub, or USB/audio hub. Additional devices and peripherals such as cameras, flash storage drives, card readers, touch screens, etc. could also be assigned to each seat. An alternative to multiple physical video cards and connections is DisplayLink over USB.

Windows

For Windows 2000, XP and Vista operating systems, there are several commercial products to implement multiseat configurations for two or more seats.

Windows MultiPoint Server 2010 was announced on February 24, 2010. It uses Remote Desktop (Terminal Services) technologies in Windows Server 2008 R2.

Case studies

Paraná Digital project

One of multiterminal's successful cases is happening at Paraná Digital project. It is creating multiterminal laboratories on 2000 public schools of the state of Paraná (Brazil). More than 1.5 million users will benefit from the 40,000 terminals when the project is finished. The laboratories have four-head multiterminals running Debian. The cost of all the hardware is 50% less than the normal price, and there is absolutely no cost with software. This project developer is C3SL (Center for Scientific Computing and Free Software).

Michigan State University Research in Tanzania

Since 2008, electrical and computer engineering students from Michigan State University have installed multiterminal systems with internet access in three schools in Mto wa Mbu, Tanzania. The purpose of the project is to study the impact of having computer systems with internet access in an education system that cannot afford other educational resources such as books. The computer systems run Ubuntu 8.04 32-bit and utilize the open source Multiseat Display Manager created by C3SL. The research will eventually be used to present to government officials of third world countries in effort to showcase the positive impact of having cost-effective computing systems in schools. The project is sponsored by George and Vickie Rock and the Dow Chemical Company.

Notable installations

- Userful announced a deployment of 356,800 Linux-based virtual desktops in Brazil (February 2009)
- NComputing provided 180,000 one to one computing seats for K–12 students in the country of Macedonia

Chapter 7

Intelligent Platform Management Interface

The **Intelligent Platform Management Interface** (IPMI) is a standardized computer system interface used by system administrators to manage a computer system and monitor its operation.

The development of this interface specification was led by Intel Corporation and is supported by more than two hundred computer systems vendors.. Dell, Hewlett-Packard, Intel, and NEC Corporation announced IPMI v1.0 on 1998-09-16, v1.5 on 2001-03-01, and v2.0 on 2004-02-14. The technology is now considered a de-facto standard for computer system management.

Functionality

An IPMI sub-system operates independently of the operating system and allows administrators to manage a system remotely in the absence of an operating system or of the system management software. The monitored system may be powered off, but must be connected to a power source and the monitoring medium, typically a local area network connection. IPMI can also function after the operating system has started, and exposes management data and structures to the system management software. IPMI prescribes only the structure and format of the interfaces as a standard, while detailed implementations may vary.

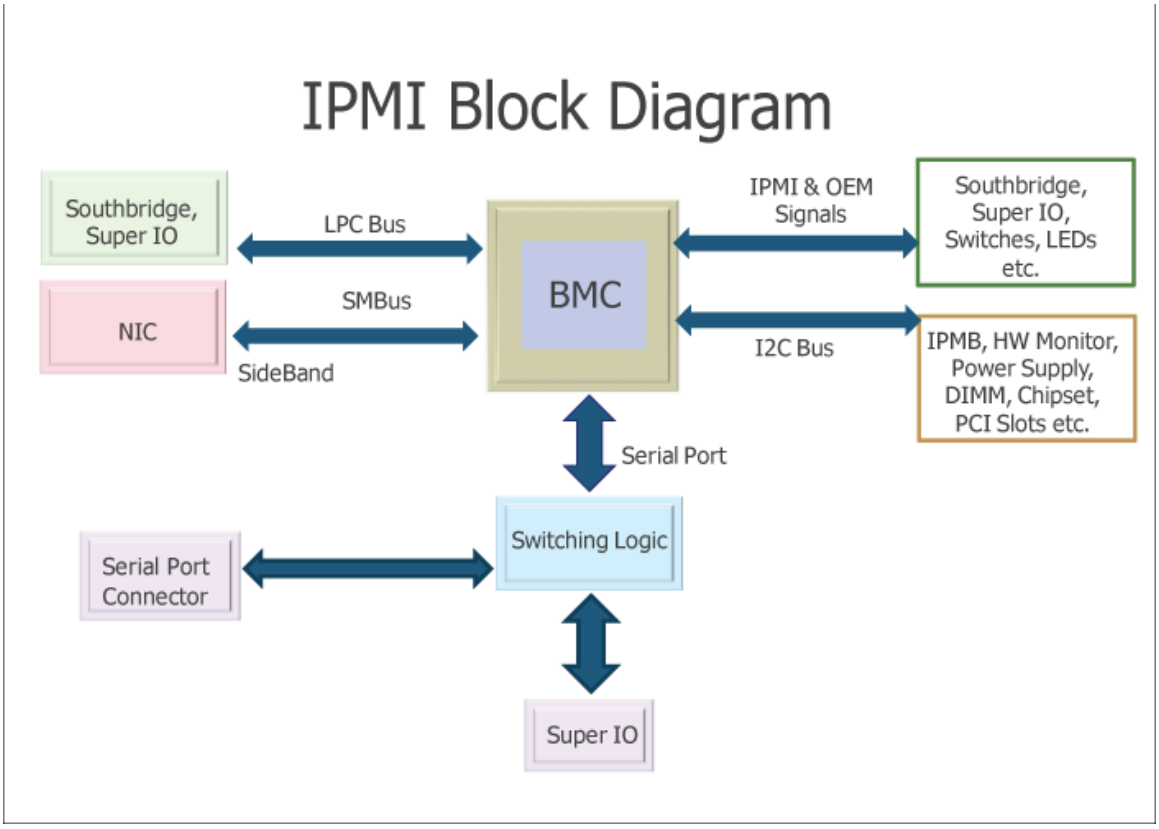
An implementation of IPMI version 1.5 can send out alerts via a direct serial connection or via a side-band local area network (LAN) connection to a remote client. The side-band LAN connection utilizes the board network interface controller (NIC). This solution is less expensive than a dedicated LAN connection but also has limited bandwidth. Systems compliant with IPMI version 2.0 can also send alerts via serial over LAN. System administrators can then use IPMI messaging to query platform status, to review hardware logs, or to issue other requests from a remote console through the same connections. The standard also defines an alerting mechanism for the system to send a simple network management protocol (SNMP) platform event trap (PET).

Side-band and out-of-band

IPMI implements what is often called a side-band management LAN connection. This connection utilizes a System Management Bus (SMBUS) interface between the BMC (Baseboard Management Controller) and the board Network Interface Controller (NIC). This solution has the advantage of reduced costs but also provides limited bandwidth – sufficient for text console redirection but not for video redirection. In other words, when a remote computer is down the system administrator can access it through IPMI and utilize a text console. This will be sufficient for a few vital functions, such as checking the event log, accessing the BIOS setup and perform power on, power off or power cycle. However, more advanced functions, such as remote re-installation of an operating system, may require an out-of-band management approach utilizing a dedicated LAN Connection.

IPMI components

An IPMI sub-system consists of a main controller, called the baseboard management controller (BMC) and other management controllers distributed among different system modules that are referred to as satellite controllers. The satellite controllers within the same chassis connect to the BMC via the system interface called Intelligent Platform Management Bus/Bridge (IPMB) — an enhanced implementation of I²C (Inter-Integrated Circuit). The BMC connects to satellite controllers or another BMC in another chassis via the Intelligent Platform Management Chassis (IPMC) bus or bridge. It may be managed with the Remote Management Control Protocol (RMCP), a specialized wire protocol defined by this specification.



Interfaces to the baseboard management controller

Several vendors develop and market BMC chips. A BMC utilized for embedded applications may have limited memory and require optimized firmware code for implementation of the full IPMI functionality. Highly integrated BMCs can provide complex instructions and provide the complete out-of-band functionality of a service processor. The firmware implementing the IPMI interfaces is provided by various vendors. A field replaceable unit (FRU) holds the inventory, such as vendor ID and manufacturer, of potentially replaceable devices. A sensor data record (SDR) repository provides the properties of the individual sensors present on the board. For example, the board may contain sensors for temperature, fan speed, and voltage.

Baseboard management controller

The *baseboard management controller* is the intelligence in the IPMI architecture. It is a specialized microcontroller embedded on the motherboard of a computer, generally a server. The BMC manages the interface between system management software and platform hardware.

Different types of sensors built into the computer system report to the BMC on parameters such as temperature, cooling fan speeds, power status, operating system (OS) status, etc. The BMC monitors the sensors and can send alerts to a system administrator via the network if any of the parameters do not stay within preset limits, indicating a

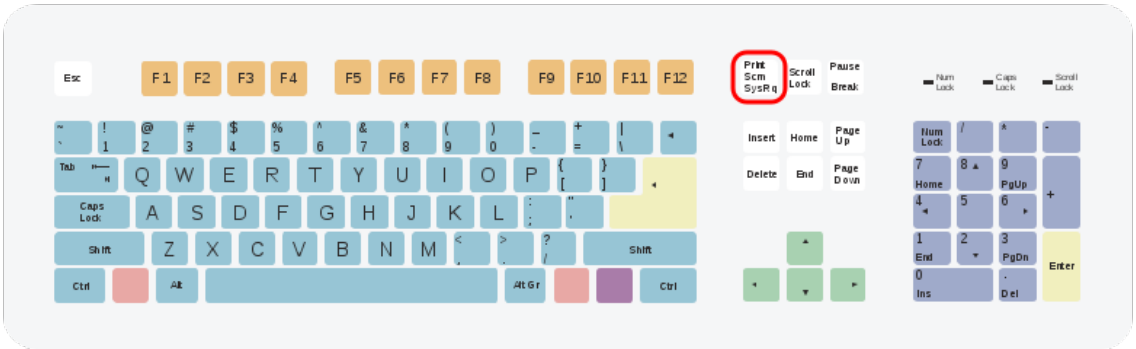
potential failure of the system. The administrator can also remotely communicate with the BMC to take some corrective action such as resetting or power cycling the system to get a hung OS running again. These abilities save on the total cost of ownership of a system.

Physical interfaces to the BMC include SMBus busses, an RS-232 serial console, address and data lines and an Intelligent Platform Management Bus (IPMB), that enables the BMC to accept IPMI request messages from other management controllers in the system.

A direct serial connection to the BMC is not encrypted as the connection itself is secure. Connection to the BMC over LAN may or may not use encryption depending on the security concerns of the user.

Chapter 8

Magic SysRq Key



The SysRq key

The **magic SysRq key** is a key combination understood by the Linux kernel, which allows the user to perform various low level commands regardless of the system's state. It is often used to recover from freezes, or to reboot a computer without corrupting the filesystem.

To be able to use this functionality the `CONFIG_MAGIC_SYSRQ` option has to be enabled at kernel compile time.

Purpose

Much like Sun Microsystems's Open Firmware (OpenBoot), this key combination provides access to powerful tools for software development and disaster recovery. In this sense, it can be considered a form of escape sequence. Principal among the offered commands are means to forcibly unmount file systems, kill processes, recover keyboard state, and write unwritten data to disk. With respect to these tasks, this feature serves as a tool of last resort.

Magic commands

The key combination consists of Alt, SysRq and another key, which controls the command issued (as shown in the table below). Users with a keyboard layout other than QWERTY have to remember that their layout becomes QWERTY when they use one of these combinations. For example, on a Dvorak keyboard, the key below '9' and '0' counts as an 'o', not as an 'r', so it shuts the system down instead of switching the keyboard to raw mode. Furthermore, some keyboards may not provide a separate SysRq key. In this case, a separate "Print Screen" key should be present. Under graphical environments (such as Gnome or KDE) 'Alt'+'PrintScrn/SysRq'+key combination generally only leads to a screenshot being dumped. To avoid this Print Screen feature the magic SysRq combination should include the Ctrl, becoming 'Ctrl'+'Alt'+'SysRq'+key. For the same purposes the AltGr key, if present, can be used in place of the Alt key. The magic SysRq can also be accessed from the serial console.

Action	QWERTY	Dvorak	AZERTY
Set the console log level, which controls the types of kernel messages that are output to the console	**0** through **9**	**0** through **9**	**0** through **9** (*without* using shift)
Immediately reboot the system, without unmounting partitions or syncing	**b**	**x**	**b**
Reboot kexec and output a crashdump	**c**	**j**	**c**
Display all currently held Locks	**d**	**e**	**d**
Send the SIGTERM signal to all processes except init (PID 1)	**e**	**.**	**e**
Call oom_kill, which kills a process to alleviate an OOM condition	**f**	**u**	**f**
When using Kernel Mode Setting, provides emergency support for switching back to the kernel's framebuffer console	**g**	**i**	**g**
Output a terse help document to the console Any key which is not bound to a command should also perform this action	**h**	**d**	**h**
Send the SIGKILL signal to all processes except init	**i**	**c**	**i**
Kill all processes on the current virtual console (Can be used to kill X and svgalib programs, see below) This was originally designed to imitate a Secure Access Key	**k**	**t**	**k**
Output current memory information to the console	**m**	**m**	**,**

Reset the nice level of all high-priority and real-time tasks	**n**	**b**	**n**
Shut off the system	**o**	**r**	**o**
Output the current registers and flags to the console	**p**	**l**	**p**
Display all active high-resolution timers and clock sources.	**q**	**'**	**a**
Switch the keyboard from raw mode, the mode used by programs such as X11 and svgalib, to XLATE mode	**r**	**p**	**r**
Sync all mounted filesystems	**s**	**o**	**s**
Output a list of current tasks and their information to the console	**t**	**y**	**t**
Remount all mounted filesystems in read-only mode	**u**	**g**	**u**
Output Voyager SMP processor information	**v**	**k**	**v**
Display list of blocked (D state) tasks	**w**	**,**	**z**

Common usage

Command line access and configuration

While this was originally implemented as part of the kernel's keyboard handler for debugging, the functionality has been also exposed via the proc filesystem and is commonly used to provide extended management capabilities to headless and remote systems. As an example, shell script can be simply used:

```
echo b > /proc/sysrq-trigger
```

This is equivalent to the key combination Alt + SysRq + B which reboots the machine.

The feature is controlled both by a compile-time option in the kernel configuration, CONFIG_MAGIC_SYSRQ, and a sysctl kernel parameter, kernel.sysrq.

"Raising Elephants" mnemonic device

A common use of the magic SysRq key is to preform a safe reboot of a Linux computer which has otherwise locked up. This can prevent a fsck being required on reboot and gives some programs a chance to save emergency backups of unsaved work. The QWERTY (or AZERTY) mnemonic "**R**aising **E**lephants **I**s **S**o **U**tterly **B**oring", "**R**eboot

Even **I**f **S**ystem **U**tterly **B**roken" or simply remembering the word "BUSIER" backwards, are often used. It stands for:

```
unRaw      (take control of keyboard back from X),
 tErminate (send SIGTERM to all processes, allowing them to terminate
gracefully),
 kIll      (send SIGKILL to all processes, forcing them to terminate
immediately),
  Sync     (flush data to disk),
  Unmount  (remount all filesystems read-only),
reBoot.
```

In practice, each command may require a few seconds to complete, especially if feedback is unavailable from the screen due to a freeze or display corruption.

Remote access

The linux daemon sysrqd provides a method of accessing SysRq features over TCP/IP port 4094 after authenticating with a plain-text password.

Graphical programs

When magic SysRq keys are used to kill a frozen graphical program, the program has no chance to restore text mode. This can make everything unreadable. The commands `textmode` (part of SVGAlib) and `reset` can restore text mode and make the console readable again.

In hypervisors

The Xen hypervisor has functionality to send magic commands to hosted domains via its "xm sysrq" command.

Security concerns

Some people view this key as giving access to dangerous system-level commands to anyone who has physical access to the keyboard or serial console. It has been argued that this perceived security is illusory, as anyone with physical access to the computer would already have the capability to compromise its security. The advent of the procfs interface has rekindled debate over this subject.

Disabling SysRq key

The SysRq key can be disabled with the following command:

```
echo 0 > /proc/sys/kernel/sysrq
```

To re-enable:

```
echo 1 > /proc/sys/kernel/sysrq
```

On newer kernels (exact version unknown), it is possible to have a more fine-grained control. On these machines, the number written to /proc/sys/kernel/sysrq can be zero, one, or a number greater than one which is a bitmap indicating which features to allow.

Possible values are:

- 0 - disable sysrq
- 1 - enable sysrq completely
- >1 - bitmask of enabled sysrq functions:
    - 2 - control of console logging level
    - 4 - control of keyboard (SAK, unraw)
    - 8 - debugging dumps of processes etc.
    - 16 - sync command
    - 32 - remount read-only
    - 64 - signalling of processes (term, kill, oom-kill)
    - 128 - reboot/poweroff
    - 256 - nicing of all RT tasks

Chapter 9

Password Cracking and Remote Administration

Password cracking

Password cracking is the process of recovering passwords from data that has been stored in or transmitted by a computer system. A common approach is to repeatedly try guesses for the password. The purpose of password cracking might be to help a user recover a forgotten password (though installing an entirely new password is less of a security risk, but involves system administration privileges), to gain unauthorized access to a system, or as a preventive measure by system administrators to check for easily crackable passwords. On a file-by-file basis, password cracking is utilized to gain access to digital evidence for which a judge has allowed access but the particular file's access is restricted.

Time needed for password searches

The time to crack a password is related to bit strength, which is a function of the password's information entropy. Most methods of password cracking require the computer to produce many candidate passwords, each of which is checked. Brute force cracking, in which a computer tries *every* possible key or password until it succeeds, is the lowest common denominator of password cracking. More common methods of password cracking, such as dictionary attacks, pattern checking, word list substitution, etc., attempt to reduce the number of trials required and will usually be attempted before brute force.

The ability to crack passwords using computer programs is a function of the number of possible passwords per second which can be checked. If a hash of the target password is available to the attacker, this number can be quite large. If not, the rate depends on whether the authentication software limits how often a password can be tried, either by time delays, CAPTCHAs, or forced lockouts after some number of failed attempts.

Individual desktop computers can test anywhere between one million to fifteen million passwords per second against a password hash for weaker algorithms, such as DES or LanManager. See: John the Ripper benchmarks A user-selected eight-character password with numbers, mixed case, and symbols, reaches an estimated 30-bit strength, according to NIST. 2^{30} is only one billion permutations and would take an average of 16 minutes to crack. When ordinary desktop computers are combined in a cracking effort, as can be done with botnets, the capabilities of password cracking are considerably extended. In 2002, distributed.net successfully found a 64-bit RC5 key in four years, in an effort which included over 300,000 different computers at various times, and which generated an average of over 12 billion keys per second. Graphics processors can speed up password cracking by a factor of 50 to 100 over general purpose computers. As of 2011, commercial products are available that claim the ability to test up to 2,800,000,000 passwords a second on a standard desktop computer using a high-end graphics processor. Such a device can crack a 10 letter single-case password in one day. Note that the work can be distributed over many computers for an additional speedup proportional to the number of available computers with comparable GPUs.

If a cryptographic salt is not used in the password system, the attacker can pre-compute hash values for common passwords variants and for all passwords shorter than a certain length, allowing very rapid recovery. Long lists of pre-computed password hashes can be efficiently stored rainbow tables. Such tables are available on the Internet for several common password authentication systems.

Another situation where quick guessing is possible is when the password is used to form a cryptographic key. In such cases, an attacker can quickly check to see if a guessed password successfully decodes encrypted data. For example, one commercial product claims to test 103,000 WPA PSK passwords per second.

Despite their capabilities, desktop CPUs are slower at cracking passwords than purpose-built password breaking machines. In 1998, the Electronic Frontier Foundation (EFF) built a dedicated password cracker using FPGAs, as opposed to general purpose CPUs. Their machine, Deep Crack, broke a DES 56-bit key in 56 hours, testing over 90 billion keys per second. In 2010, the Georgia Tech Research Institute developed a method of using GPGPU to crack passwords, coming up with a minimum secure password length of 12 characters.

Perhaps the fastest way to crack passwords is through the use of pre-computed rainbow tables. These encode the hashes of common passwords based on the most widely used hash functions and can crack passwords in a matter of seconds. However, they are only effective on systems that do not use a salt, such as Windows LAN Manager and some application programs.

Prevention

The best method of preventing password cracking is to ensure that attackers cannot get access even to the encrypted password. For example, on the Unix operating system,

encrypted passwords were originally stored in a publicly accessible file `/etc/passwd`. On modern Unix (and similar) systems, on the other hand, they are stored in the file `/etc/shadow`, which is accessible only to programs running with enhanced privileges (ie, 'system' privileges). This makes it harder for a malicious user to obtain the encrypted passwords in the first instance. Unfortunately, many common network protocols transmit passwords in cleartext or use weak challenge/response schemes.

Modern Unix systems have replaced traditional DES-based password hashing with stronger methods based on MD5 and Blowfish. Other systems have also begun to adopt these methods. For instance, the Cisco IOS originally used a reversible Vigenère cipher to encrypt passwords, but now uses md5-crypt with a 24-bit salt when the "enable secret" command is used. These newer methods use large salt values which prevent attackers from efficiently mounting offline attacks against multiple user accounts simultaneously. The algorithms are also much slower to execute which drastically increases the time required to mount a successful offline attack.

Many hashes used for storing passwords, such as MD5 and the SHA family, are designed for fast computation and efficient implementation in hardware. Using key stretching algorithms, such as PBKDF2, to form password hashes can significantly reduce the rate at which passwords can be tested.

Solutions like a security token give a formal proof answer by constantly shifting password. Those solutions abruptly reduce the timeframe for brute forcing (attacker needs to break and use the password within a single shift) and they reduce the value of the stolen passwords because of its short time validity.

Software

Main category: Password cracking software.

There are many password cracking software tools, but the most popular are Cain and Abel, John the Ripper, Hydra, ElcomSoft and Lastbit. Many litigation support software packages also include password cracking functionality. Most of these packages employ a mixture of cracking strategies, with brute force and dictionary attacks proving to be the most productive.

Remote administration

Remote administration refers to any method of controlling a computer from a remote location.

Software that allows remote administration is becoming increasingly common and is often used when it is difficult or impractical to be physically near a system in order to use it, or in order to access web material that is not available in one's location, for example viewing the BBC iPlayer from outside the United Kingdom. A remote location may refer

to a computer in the next room or one on the other side of the world. It may also refer to both legal and illegal (i.e. hacking) remote administration.

Requirements

Internet connection

Any computer with an Internet connection, TCP/IP or on a Local Area Network can be remotely administered.

For non-malicious administration, the user must install or enable server software on the host system in order to be viewed. Then the user/client can access the host system from another computer using the installed software.

Usually, both systems should be connected to the internet, and the IP address of the host/server system must be known. Remote administration is therefore less practical if the host uses a dial-up modem, which is not constantly online and often has a Dynamic IP.

Connecting

When the client connects to the host computer, a window showing the Desktop of the host usually appears. The client may then control the host as if he/she were sitting right in front of it.

Certain versions of Windows XP have a built-in remote administration package called Remote Desktop Connection. A free cross-platform alternative is VNC, which offers similar functionality.

Common tasks for which remote administration is used

General

Controlling one's own computer from a remote location (e.g. to access the software on a personal computer from an internet café).

Shutdown

- Shutting down or rebooting another computer over a network

Accessing Peripherals

- Using a network device, like printer
- Retrieving streaming data, much like a CCTV system

Modifying

- Editing another computer's registry settings
- Modifying system services
- Installing software on another machine
- Modifying logical groups

Viewing

- Remotely assisting others
- Supervising computer or internet usage
- Access to a remote system's "Computer Management" snap-in

Popular software

Windows

Windows Server 2003/2008, Tablet PC Editions, and Windows Vista Ultimate, Enterprise and Business editions come with Microsoft's Microsoft Management Console, Windows Registry Editor and various command-line utilities that may be used to administrate a remote machine. One form of remote administration is remote desktop software, and Windows includes a Remote Desktop Connection client for this purpose.

Windows XP comes with a built-in remote administration tools called Remote Assistance and Remote Desktop, these are restricted versions of the Windows Server 2003 Terminal Services meant only for helping users and remote administration. With a simple hack/patch (derived from the beta version of Windows XP) it's possible to "unlock" XP to a fully featured Terminal Server, one good and easy example is Sala´s Terminal Server Patch. With this patch it is possible to make a terminal server out of Windows XP Professional, Multimeda Center, and Tablet PC Edition. Windows XP Home can be made to run a full-featured Terminal Service as well, with additional patching. .

Windows Server 2003 comes with built-in remote administration tools, including a web application and a simplified version of Terminal Services designed for Remote administration.

Active Directory and other features found in Microsoft's Windows NT Domains allow for remote administration of computers that are members of the domain, including editing the registry and modifying system services and access to the system's "Computer Management" Microsoft Management Console snap-in.

Some third-party remote desktop software programs perform the same job.

Remote Server Administration Tools for Windows 7 enables IT administrators to manage roles and features that are installed on remote computers that are running Windows Server 2008 R2

Non-Windows

VNC can be used for remote administration of computers, however it is increasingly being used as an equivalent of Terminal Services and Remote Desktop Protocol for multi-user environments.

Back Orifice, whilst commonly used as a Script Kiddie tool, claims to be a remote-administration and system management tool. Critics have previously stated that the capabilities of the software require a very loose definition of what "administration" entails.

Linux, UNIX and BSD support remote administration via remote login, typically via SSH (The use of the Telnet protocol has been phased out due to security concerns). X-server connection forwarding, often tunnelled over SSH for security, allows GUI programs to be used remotely. VNC is also available for these operating systems.

Apple Remote Desktop provides Macintosh users with remote administration capabilities.

Scriptlogic's Desktop Authority encompasses remote control as a part of remote management. This solution includes: secure web-based access to client machines, real-time diagnostics and troubleshooting, management of the file system, users/groups, registry, virtual memory, reboots and more - without user interaction, interactive remote monitoring and control of the desktop, supports clients running Windows 98 through XP/2003/Vista.

NX and its Google fork NeatX are free graphical Desktop sharing solutions for the X Window System with Clients for different platforms like Linux, Windows and Mac OS X. There is also a enhanced commercial version of NX Server available.

Wireless Remote Administration

Remote administration software has recently started to appear on wireless devices such as the BlackBerry, Pocket PC, and Palm devices, as well as some mobile phones.

Generally these solutions do not provide the full remote access seen on software such as VNC or Terminal Services, but do allow administrators to perform a variety of tasks, such as rebooting computers, resetting passwords, and viewing system event logs, thus reducing or even eliminating the need for system administrators to carry a laptop or be within reach of the office.

Chapter 10

Software Deployment and System Console

Software deployment

Software deployment is all of the activities that make a software system available for use.

The general deployment process consists of several interrelated activities with possible transitions between them. These activities can occur at the producer site or at the consumer site or both. Because every software system is unique, the precise processes or procedures within each activity can hardly be defined. Therefore, "deployment" should be interpreted as a *general process* that has to be customized according to specific requirements or characteristics. A brief description of each activity will be presented later.

Deployment activities

Release

> The release activity follows from the completed development process. It includes all the operations to prepare a system for assembly and transfer to the customer site. Therefore, it must determine the resources required to operate at the customer site and collect information for carrying out subsequent activities of deployment process.

Install and activate

> Activation is the activity of starting up the executable component of software. For simple system, it involves establishing some form of command for execution. For complex systems, it should make all the supporting systems ready to use.
> In larger software deployments, the working copy of the software might be installed on a production server in a production environment. Other versions of the deployed software may be installed in a test environment, development environment and disaster recovery environment.

Deactivate

Deactivation is the inverse of activation, and refers to shutting down any executing components of a system. Deactivation is often required to perform other deployment activities, e.g., a software system may need to be deactivated before an update can be performed. The practice of removing infrequently used or obsolete systems from service is often referred to as application retirement or application decommissioning.

Adapt

The adaptation activity is also a process to modify a software system that has been previously installed. It differs from updating in that adaptations are initiated by local events such as changing the environment of customer site, while updating is mostly started from remote software producer.

Update

The update process replaces an earlier version of all or part of a software system with a newer release.

Built-In

Mechanisms for installing updates are built into some software systems. Automation of these update processes ranges from fully automatic to user initiated and controlled. Norton Internet Security is an example of a system with a semi-automatic method for retrieving and installing updates to both the antivirus definitions and other components of the system. Other software products provide query mechanisms for determining when updates are available.

Version tracking

Version tracking systems help the user find and install updates to software systems installed on PCs and local networks.

- Web based version tracking systems notify the user when updates are available for software systems installed on a local system. For example: VersionTracker Pro checks software versions on a user's computer and then queries its database to see if any updates are available.
- Local version tracking system notifies the user when updates are available for software systems installed on a local system. For example: Software Catalog stores version and other information for each software package installed on a local system. One click of a button launches a browser window to the upgrade web page for the application, including auto-filling of the user name and password for sites that require a login.
- Browser based version tracking systems notify the user when updates are available for software packages installed on a local system. For example: wfx-Versions is a Firefox extension which helps the user find the current version number of any program listed on the web.

Uninstall

Uninstallation is the inverse of installation. It is the removal of a system that is no longer required. It also involves some reconfiguration of other software systems in order to remove the uninstalled system’s files and dependencies.

Retire

Ultimately, a software system is marked as obsolete and support by the producers is withdrawn. It is the end of the life cycle of a software product.

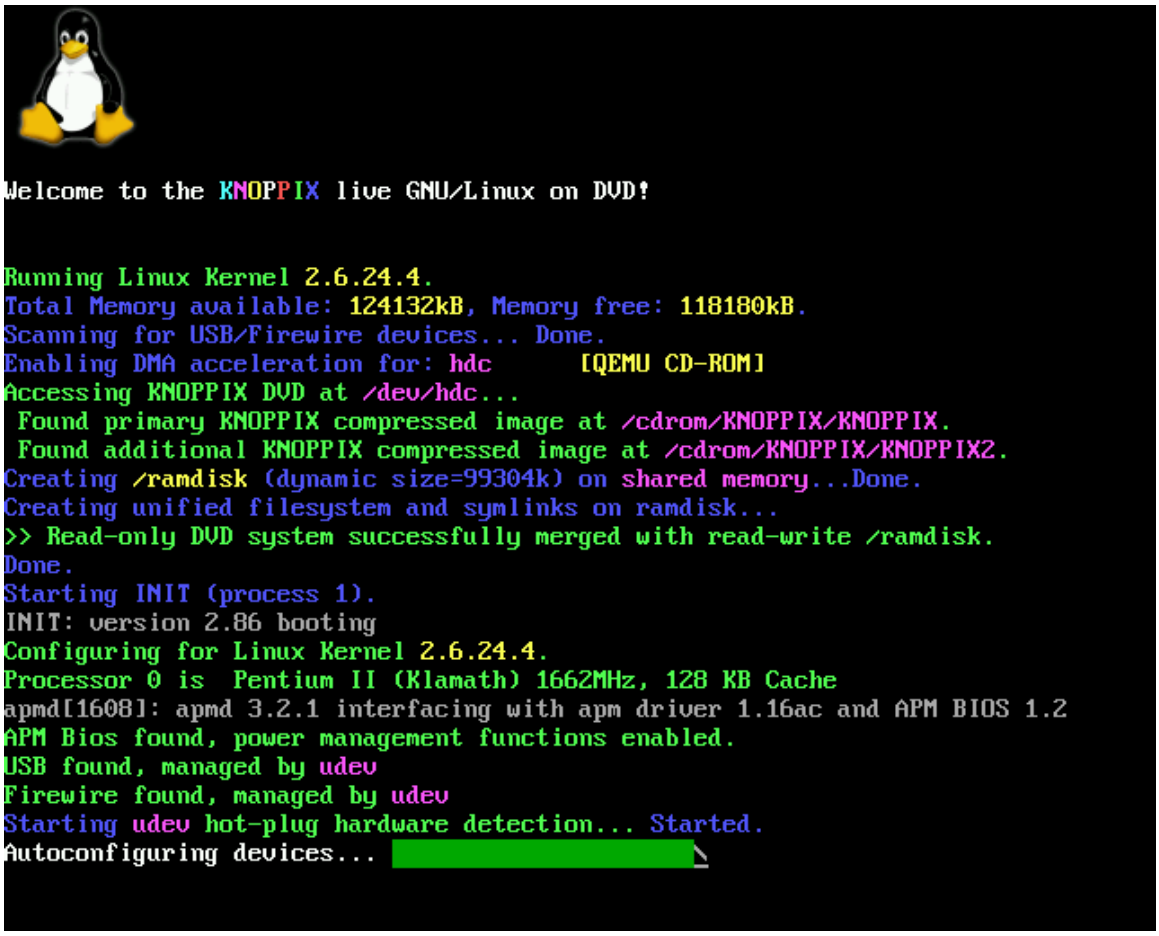
Deployment roles

The complexity and variability of software products has necessitated the creation of specialized roles for coordinating and engineering the deployment process. For desktop systems, an end user is frequently also the "software deployer" when they install the software package on their machine. For enterprise software, there are many more roles involved. Additionally, the roles involved typically change as the application progresses from test (pre-production) to production environments. The typical roles involved in software deployments for enterprise applications are:

- Pre-production environments
  - Application developers
  - Build and release engineers
  - Release managers
  - Deployment coordinators

- Production environments
  - System administrator
  - Database administrator
  - Release coordinators
  - Operations project managers

System console



Knoppix system console showing the boot process

The **system console**, **root console** or simply **console** is the text entry and display device for system administration messages, particularly those from the BIOS or boot loader, the kernel, from the init system and from the system logger. It is a physical device consisting of a keyboard and a screen.

On traditional minicomputers, the console was a **serial console**, an RS-232 serial link to a terminal such as a DEC VT100. This terminal was usually kept in a secured room since it could be used for certain privileged functions such as halting the system or selecting which media to boot from. Large midrange systems, e.g. those from Sun Microsystems, Hewlett-Packard and IBM, still use serial consoles. In larger installations, the console ports are attached to multiplexers or network-connected multiport serial servers that let an operator connect a terminal to any of the attached servers. Today, serial consoles are often used for accessing headless systems, usually with a terminal emulator running on a laptop. Also, routers, enterprise network switches and other telecommunication equipment have RS-232 serial console ports.

On PCs and workstations, the computer's attached keyboard and monitor have the equivalent function. Since the monitor cable carries video signals, it cannot be extended very far. Often, installations with many servers therefore use keyboard/video multiplexers (KVM switches) and possibly video amplifiers to centralize console access. In recent years, KVM/IP devices have become available that allow a remote computer to view the video output and send keyboard input via any TCP/IP network and therefore the Internet.

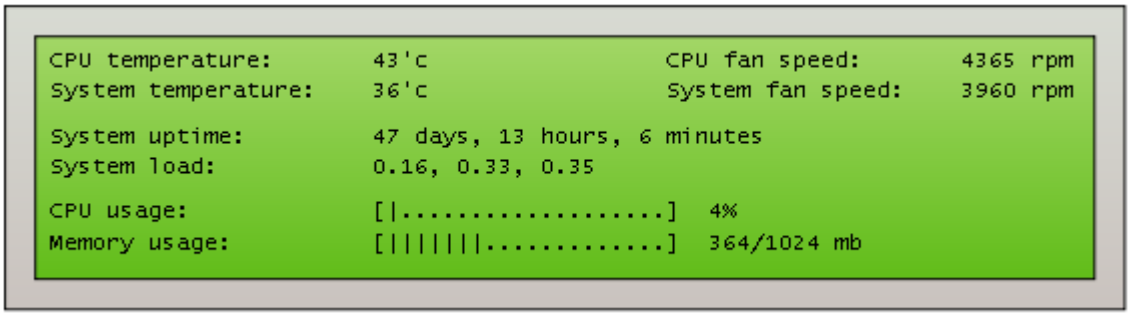
Some PC BIOSes, especially in servers, also support serial consoles, giving access to the BIOS through a serial port so that the simpler and cheaper serial console infrastructure can be used. Even where BIOS support is lacking, some operating systems, e.g. FreeBSD and Linux, can be configured for serial console operation either during bootup, or after startup.

It is usually possible to log in from the console. Depending on configuration, the operating system may treat a login session from the console as being more trustworthy than a login session from other sources.

Chapter 11

System Monitor and System Profiler

System monitor



Simulated LCD panel for display of resource usage.

A **system monitor** is a hardware- or software- based system used to monitor resources and performance in a computer system.

Software monitors occur more commonly, sometimes as a part of a widget engine. These monitoring systems are often used to keep track of system resources, such as CPU usage and frequency , or the amount of free RAM. They are also used to display items such as free space on one or more hard drives, the temperature of the CPU and other important components, and networking information including the system IP address and current rates of upload and download. Other possible displays may include the date and time, system uptime, computer name, username, hard drive S.M.A.R.T data, fan speeds, and the voltages being provided by the power supply.

Less common are hardware-based systems monitoring similar information. Customarily these occupy one or more drive bays on the front of the computer case, and either interface directly with the system hardware or connect to a software data-collection system via USB. With either approach to gathering data, the monitoring system displays information on a small LCD panel or on series of small analog or LED numeric displays.

Some hardware-based system monitors also allow direct control of fan speeds, allowing the user to quickly customize the cooling in the system.

A few very high-end models of hardware system monitor are designed to interface with only a specific model of motherboard. These systems directly utilize the sensors built into the system, providing more detailed and accurate information than less-expensive monitoring systems customarily provide.

Software versions are becoming more common, with even the Microsoft Windows Vista sidebar including a meter to monitor CPU and RAM usage. Hardware versions are very rare on OEM computers, with the exception of high-end servers. However, computer enthusiasts often install such hardware-based monitors.

Software

- CPU-Z, detects live changes in CPU attributes
- Conky
- frysk - analyzes and monitors system processes
- Ganglia (software)
- GKrellM
- htop (Unix)
- Iostat
- Motherboard Monitor
- Nmon
- Sar in UNIX
- SpeedFan
- top (software)
- Vmstat
- Windows Sidebar on Windows Vista
- WinBar
- Activity monitor on Mac OS X.

System profiler

A **system profiler** is a program that can provide detailed information about the software installed and hardware attached to the computer. In older versions of Apple Computer's Mac OS, this was done by an application called Apple System Profiler. Mac OS X's profiler is simply called System Profiler. In Microsoft Windows some similar information may be found by getting properties on My Computer on the desktop.

List of system profiler software

Microsoft Windows

- Sisoft SANDRA (System Analyser, Diagnostic and Reporting Assistant) system profiling software
- HWiNFO32 - Freeware system information tool.
- Lavalys EVEREST Ultimate Edition (formerly AIDA32) - Shareware System information and benchmarking tool.
- FreshDiagnose - Freeware system information tool.
- CPU-Z, useful when overclocking processors
- SIW - System Information for Windows - portable freeware (does not require installation) - software, hardware, network information, tools and real-time monitors
- Belarc Freeware for personal use PC Auditing Software lists hardware, as well as software installed on the local machine and displays as a local webpage. Belarc also makes a security assessment for checking how secure a system is, and links missing updates directly to a Microsoft website for download.
- systeminfo native windows command line, returns OS version, uptime, CPU, physical memory, network cards, etc ...
- msinfo32 or winmsd comprehensive view of hardware, system components, and software environment
- PsInfo command line, from the Sysinternals suite
- SekChek Local - an automated security audit tool which scans multiple Windows workstations and servers, from the network. It creates a security assessment report file which is presented as an Microsoft Access dataset.
- CPU Speed Pro is a Microsoft Windows software application which will test the speed of the central processing unit (CPU)
- SlimWare Utilities - PC data and registry cleaner, driver diagnostic, system analyzer and optimization software.

Linux

- LSHW
- HardInfo
- SekChek Classic - The UNIX extract tool extracts control data regarding security policies and objects defined on the target host.

DOS

- HWiNFO - still updated DOS version of HWiNFO32

OS Independent

- Hardware Detection Tool (HDT)

Chapter 12

Systems Management and Windows Management Instrumentation

Systems management

Systems management refers to enterprise-wide administration of distributed systems including (and commonly in practice) computer systems. Systems management is strongly influenced by network management initiatives in telecommunications.

Centralized management has a time and effort tradeoff that is related to the size of the company, the expertise of the IT staff, and the amount of technology being used:

- For a small-business startup with ten computers, automated centralized processes may take more time to learn how to use and implement than just doing the management work manually on each computer.
- A very large business with thousands of similar employee computers may clearly be able to save time and money, by having IT staff learn to do systems management automation.
- A small branch office of a large corporation may have access to a central IT staff, with the experience to set up automated management of the systems in the branch office, without need for local staff in the branch office to do the work.

System management may involve one or more of the following tasks:

- Hardware inventories.
- Server availability monitoring and metrics.
- Software inventory and installation.
- Anti-virus and anti-malware management.
- User's activities monitoring.
- Capacity monitoring.
- Security management.
- Storage management.

- Network capacity and utilization monitoring.
- Anti-manipulation management

Functions

Functional groups are provided according to International Telecommunication Union Telecommunication Standardization Sector (ITU-T) Common management information protocol (X.700) standard. This framework is also known as Fault, Configuration, Accounting, Performance, Security (FCAPS).

Fault management

- Troubleshooting, error logging and data recovery

Configuration management

- Hardware and software inventory

- *As we begin the process of automating the management of our technology, what equipment and resources do we have already?*
- *How can this inventorying information be gathered and updated automatically, without direct hands-on examination of each device, and without hand-documenting with a pen and notepad?*
- *What do we need to upgrade or repair?*
- *What can we consolidate to reduce complexity or reduce energy use?*
- *What resources would be better reused somewhere else?*
- *What commercial software are we using that is improperly licensed, and either needs to be removed or more licenses purchased?*

- Provisioning

- *What software will we need to use in the future?*
- *What training will need to be provided to use the software effectively?*

- Software deployment

- *What steps are necessary to install it on perhaps hundreds or thousands of computers?*

- Package management

- *How do we maintain and update the software we are using, possibly through automated update mechanisms?*

Accounting management

- Billing and statistics gathering

Performance management

- Software metering

- *Who is using the software and how often?*
- *If the license says only so many copies may be in use at any one time but may be installed in many more places than licensed, then track usage of those licenses.*
- *If the licensed user limit is reached, either prevent more people from using it, or allow overflow and notify accounting that more licenses need to be purchased.*

- Event and metric monitoring

- *How reliable are the computers and software?*
- *What errors or software bugs are preventing staff from doing their job?*
- *What trends are we seeing for hardware failure and life expectancy?*

Security management

- Identity management
- Policy management

However this standard should not be treated as comprehensive, there are obvious omissions. Some are recently emerging sectors, some are implied and some are just not listed. The primary ones are:

- Business Impact functions (also known as Business Systems Management)
- Capacity Management
- Real-time Application Relationship Discovery (which supports Configuration Management)
- Security Information and Event Management functions (SIEM)
- Workload Scheduling

Performance Management functions can also be split into end-to-end performance measuring and infrastructure component measuring functions. Another recently emerging sector is Operational Intelligence which focuses on real-time monitoring of business events that relate to business processes, not unlike Business Activity Monitoring.

Standards

Distributed Management Task Force (DMTF)

Alert Standard Format (ASF)
Common Information Model (CIM)
Desktop and mobile Architecture for System Hardware DASH

Directory Enabled Networking (DEN)
Systems Management Architecture for Server Hardware (SMASH)
Java Management Extensions (JMX)

Windows Management Instrumentation

Management Instrumentation (**WMI**) is a set of extensions to the Windows Driver Model that provides an operating system interface through which instrumented components provide information and notification. WMI is Microsoft's implementation of the Web-Based Enterprise Management (WBEM) and Common Information Model (CIM) standards from the Distributed Management Task Force (DMTF).

WMI allows scripting languages like VBScript or Windows PowerShell to manage Microsoft Windows personal computers and servers, both locally and remotely. WMI is preinstalled in Windows 2000 and newer OSs. It is available as a download for Windows NT, Windows 95 and Windows 98.

Microsoft also provides a command line interface to WMI called **Windows Management Instrumentation Command-line** (**WMIC**).

Purpose of WMI

The purpose of WMI is to define a non-proprietary set of environment-independent specifications which allow management information to be shared between management applications. WMI prescribes enterprise management standards and related technologies that work with existing management standards, such as Desktop Management Interface (DMI) and SNMP. WMI complements these other standards by providing a uniform model. This model represents the managed environment through which management data from any source can be accessed in a common way.

Development process

Because WMI abstracts the manageable entities with CIM and a collection of providers, the development of a provider implies several steps. The major steps can be summarized as follows:

Step 1 – Create the manageable entity model

- Define a model
- Implement the model

Step 2 – Create the WMI Provider

- Determine the provider type to implement
- Determine the hosting model of the provider
- Create the provider template with the ATL wizard
- Implement the code logic in the provider
- Register the provider with WMI and the system

Step 3 – Test the provider

Step 4 – Create consumer sample code

Importance of WMI providers

Since the release of the first WMI implementation during the Windows NT 4.0 SP4 era (as an out-of-band download), Microsoft has consistently added WMI providers to Windows. Under Windows NT 4.0, Microsoft had roughly 15 WMI providers available once WMI was installed. When Windows 2000 was released, there were 29 WMI providers as part of the operating system installation. With the release of Windows Server 2003, Microsoft included in the platform more than 80 WMI providers. Windows Vista includes 13 new WMI providers, taking the number close to around 100 in all, and Windows Server 2008 includes some more including providers for IIS 7, PowerShell and virtualization. This has been a sign for many customers that WMI became at Microsoft, the "ubiquitous" management layer of Windows, even if this commitment has never been explicit from Microsoft.

During these last years, due to a constant increasing exposure of management data through WMI in Windows, more and more people in the IT systems management field started to develop scripts and automation procedures based on WMI. Beyond the scripting needs, most leading management software in the world, such as MOM, SMS, ADS, HP OpenView for Windows (HPOV), BMC Software or CA, Inc. are WMI-enabled and capable to consume and provide WMI information through various *User Interfaces*. This enables administrators and operators not capable of scripting or programming on top of WMI to enjoy the benefits of WMI without even learning about it. However, if they want to, because WMI is scriptable, it gives them the opportunity to consume WMI information from scripts or from any Enterprise Management software that is WMI-aware.

Features

For someone willing to develop one or many WMI providers, WMI offers many features out of the box. Here are the most important advantages:

1. *Automation interfaces:* Because WMI comes with a set of automation interfaces ready to use, all management features supported by a WMI provider and its set of classes get the scripting support for free out-of-the box. Beyond the WMI class

design and the provider development, the Microsoft development and test teams are not required to create, validate and test a scripting model as it is already available from WMI.

2. *.NET Management interfaces:* Because the System.Management namespace relies on the existing COM/DCOM plumbing, the created WMI provider and its set of WMI classes becomes automatically available to all .NET applications independently of the language used (e.g. C#, VB.NET). Beyond the WMI class design and the provider development, like for scripting, the Microsoft development and test teams are not required to create, validate and test new assemblies to support a new namespace in the .NET Framework as this support is already available from WMI for free.
3. *C/C++ COM/DCOM programming interfaces:* Like most components in Windows, COM/DCOM programmers can leverage the features of the provider they develop at the COM/DCOM interfaces level. Like in previous environments (scripting and .NET Framework), a COM/DCOM consumer just needs to interact with the standard set of WMI COM interfaces to leverage the WMI provider capabilities and its set of supported WMI classes. To make all management information available from the native APIs, the WMI provider developer just needs to interact with a set of pre-defined WMI COM interfaces. This will make the management information available at the WMI COM level automatically. Moreover, the scripting COM interface object model is very similar to the COM/DCOM interface object model, which makes it easy for developers to be familiar with the scripting experience.
4. *Remoting capabilities over DCOM and SOAP:* More than simply offering local COM capabilities, as management is all about remoting, WMI offers the DCOM transport. In addition, SOAP transport will be available in Windows Server 2003 **R2** through the WS-Management initiative led by Microsoft, Intel, Sun Microsystems and Dell. This initiative allows to run any scripts remotely or to consume WMI data through a specific set of interfaces handling SOAP requests/responses. The advantage for the WMI provider developer is that when he exposes all his features through WMI, *Windows Remote Management*/WS-Management can in turn consume that information as well (embedded objects in WMI instances are not supported in Windows Server 2003 R2. It is however a target for Vista). All the layering to WS-Management and the mapping of the CIM data model to SOAP comes for free out of the WMI/WS-Management solution. In the event DCOM must be used, implementing DCOM requires the presence of a proxy DLL deployed on each client machine. As WMI is available in the Windows operating system since Windows 2000, these issues are eliminated.
5. *Support for Queries:* WMI offers support for WQL queries out of the box. This means that if a provider is not designed to support queries, WMI supports it by using an enumeration technique out of the provider.
6. *Eventing capabilities:* WMI offers the capability to notify a subscriber for any event it is interested in. WMI uses the WMI Query Language (WQL) to submit WQL event queries and defines the type of events to be returned. The eventing mechanism, with all related callbacks, is part of the WMI COM/DCOM and

automation interfaces. Anyone writing a WMI provider can have the benefit of this functionality at no cost for his customers. It will be up to the consumer to decide how it wants to consume the management information exposed by the WMI provider and its related set of WMI classes.

7. *Code template generator:* To speed up the process of writing a WMI provider including all COM/DCOM interfaces and related definitions, the WMI team developed the *WMI ATL Wizard* to generate the code template implementing a provider. The code generated is based on the WMI class model initially designed by the developer. The WMI provider developer will be able to interface the pre-defined COM/DCOM interfaces for the WMI provider with its set of native APIs retrieving the management information to expose. The exercise consists in filling the "gaps" in the provider code to create the desired interfacing logic.
8. *Predictability:* Predictability is an important concern for IT professionals because it defines the capability of someone having an experience with a set of interfaces managing a Windows component to apply this knowledge right away, intuitively, to any other manageable Windows component without having relearn everything from ground up. Predictability for a customer is a real gain as it increases the Return of Investment (ROI). A person facing such a situation simply expects things to work the same way based on his previous experience. The constant increase of COM programming/scriptable interfaces has a huge impact on the predictability, as this makes it difficult for customers to automate, manage Windows and leverage their existing knowledge. WMI with CIM address this problem by always exposing the same programming object model (COM/DCOM, Automation, .NET) whatever the manageable entity is.
9. *Protect existing customer investments:* Protecting customers and partners investment motivates customers to invest in technologies. As Microsoft did invest a lot these past years in writing WMI providers, customers and partners invested in tools leveraging the WMI capabilities of Windows. Therefore, they naturally continue to exploit these capabilities instead of having to use a new set of specific interfaces for each Windows manageable component. A specific set of interfaces means having a specific set of agents or in-house developed software based on a new model or set of interfaces especially dedicated to a component or technology. By leveraging the capabilities of WMI today, customers and partners can leverage the work investment made in the past while minimizing their costs in developments, learning curves and new discoveries. This will also have a great impact on the stability and reliability of their infrastructure as they continue to leverage an existing implementation with an improved technology.
10. *Provide a logical and unified administration model:* As briefly described before in the introduction, this model is based on an industry standard called CIM defined by the DMTF. The CIM class-based schema is defined by a consortium of constructors and software developers that meets the requirements of the industry. This implies that not only Microsoft leverages the WMI capabilities, but also any other third party constructors or developers write their own code to fit into the model. For instance, Intel is doing this for some their network driver adapters and software. HP is leveraging existing WMI providers and implementing their own WMI providers in their HP Open View Enterprise Management software. IBM

consumes WMI from the Tivoli management suite, MOM and SMS are also consuming and providing WMI information. Lastly, Windows XP SP2 leverages WMI to get information status from anti-virus software and firewalls.

WMI tools

Some WMI tools can also be useful during the design and development phases. These tools are:

- *The MOF compiler (MOFComp.exe):* The Managed Object Format (MOF) compiler parses a file containing Managed Object Format statements and adds the classes and class instances defined in the file to the CIM repository. The MOF format is a specific syntax to define CIM class representation in an ASCII file (e.g. MIB are to SNMP what MOF files are to CIM). MOFComp.exe is included in every WMI installation. Every definition existing in the CIM repository is initially defined in an MOF file. MOF files are located in %SystemRoot%\System32\WBEM. During the WMI setup, they are loaded in the CIM repository.
- *The WMI Administrative Tools:* The WMI Administrative Tools are made of four tools: WMI CIM Studio, WMI Object Browser, WMI Event Registration and WMI Event Viewer. WMI Administrative Tools can be downloaded here. The most important tool for a WMI provider developer is WMI CIM Studio as it helps in the initial WMI class creation in the CIM repository. It uses a web interface to display information and relies on a collection of ActiveX components installed on the system when it runs for the first time. WMI CIM Studio provides the ability to:
    - Connect to a chosen system and browse the CIM repository in any namespace available.
    - Search for classes by their name, by their descriptions or by property names.
    - Review the properties, methods and associations related to a given class.
    - Perform Queries in the WQL language.
    - Generate an MOF file based on selected classes.
    - Compile an MOF file to load it in the CIM repository.
- *WinMgmt.exe:* WinMgmt.exe is not a tool; it is the executable that implements the WMI Core service. Under the Windows NT family of operating systems, WMI runs as a service. On computers running Windows 98, Windows 95 or Windows Me, WMI runs as an application. Under the Windows NT family of operating systems, it is also possible to run this executable as an application, in which case, the executable runs in the current user context. For this, the WMI service must be stopped first. The executable supports some switches that can be useful when starting WMI as a service or as an application. WMI provider developers who may want to debug their providers essentially need to run the WMI service as an application.
- *WBEMTest.exe:* WBEMTest.exe is a WMI tester tool, which is delivered with WMI. This tool allows an administrator or a developer to perform most of the

tasks from a graphical interface that WMI provides at the API level. Although available under all Windows NT-based operating systems, this tool is not officially supported by Microsoft. WBEMTest provides the ability to:
    - Enumerate, open, create and delete classes.
    - Enumerate, open, create and delete instances of classes.
    - Select a namespace.
    - Perform data and event queries.
    - Execute methods associated to classes or instances.
    - Execute every WMI operation asynchronously, synchronously or semi-asynchronously.
- The WMI command line tool (WMIC): WMIC is a command-line tool designed to ease WMI information retrieval about a system by using some simple keywords (aliases). WMIC.exe is only available under Windows XP Professional, Windows Server 2003, Windows Vista and Windows Server 2008. By typing "WMIC /?" from the command-line, a complete list of the switches and reserved keywords is available.
    - There is a Linux port of WMI command line tool, written in Python, based on Samba4 called 'wmi-client'
- *WBEMDump.exe:* WBEMDump is a tool delivered with the Platform SDK. This command line tool comes with its own Visual C++ project. The tool can show the CIM repository classes, instances, or both. It is possible to retrieve the same information as that retrieved with WMIC. WBEMDump.exe requires more specific knowledge about WMI, as it doesn't abstract WMI as WMIC. However, it runs under Windows NT 4.0 and Windows 2000. It is also possible to execute methods exposed by classes or instances. Even if it is not a standard WMI tool delivered with the system installation, this tool can be quite useful for exploring the CIM repository and WMI features.

Wireless networking example

In the .NET framework, the ManagementClass class represents a Common Information Model (CIM) management class. A WMI class can be a Win32_LogicalDisk in the case of a disk drive, or a Win32_Process, such as a running program like Notepad.exe.

This example shows how "MSNdis_80211_ServiceSetIdentifier" WMI class is used to find the SSID of the Wi-Fi network that the system is currently connected to in the language C#:

```
ManagementClass mc = new ManagementClass("root\\WMI",
"MSNdis_80211_ServiceSetIdentifier", null);
ManagementObjectCollection moc = mc. GetInstances();

foreach (ManagementObject mo in moc)
{
   string wlanCard = (string)mo["InstanceName"];
   bool active;
   if (!bool.TryParse((string)mo["Active"], out active))
   {
```

```
      active = false;
   }
   byte[] ssid = (byte[])mo["Ndis80211SsId"];
}
```

The "MSNdis_80211_ServiceSetIdentifier" WMI class is only supported on Windows XP and Windows Server 2003.

WMI driver extensions

The WMI extensions to WDM provide kernel-level instrumentation such as publishing information, configuring device settings, supplying event notification from device drivers and allowing administrators to set data security through a WMI provider known as the *WDM provider*. The extensions are part of the WDM architecture; however, they have broad utility and can be used with other types of drivers as well (such as SCSI and NDIS). The WMI Driver Extensions service monitors all drivers and event trace providers that are configured to publish WMI or event trace information. Instrumented hardware data is provided by way of drivers instrumented for WMI extensions for WDM. WMI extensions for WDM provide a set of Windows device driver interfaces for instrumenting data within the driver models native to Windows, so OEMs and IHVs can easily extend the instrumented data set and add value to a hardware/software solution. The WMI Driver Extensions, however, are not supported by Windows Vista and later operating systems.

Chapter 13

Unix Shell

tcsh and sh shell windows on a Mac OS X desktop..

A **Unix shell** is a command-line interpreter or shell that provides a traditional user interface for the Unix operating system and for Unix-like systems. Users direct the operation of the computer by entering command input as text for a command line interpreter to execute or by creating text scripts of one or more such commands.

The most influential Unix shells have been the Bourne shell and the C shell. The Bourne shell, sh, was written by Stephen Bourne at AT&T as the original Unix command line interpreter; it introduced the basic features common to all the Unix shells, including piping, here documents, command substitution, variables, control structures for

condition-testing and looping and filename wildcarding. The language, including the use of a reversed keyword to mark the end of a block, was influenced by ALGOL 68.

The C shell, csh, was written by Bill Joy while a graduate student at University of California, Berkeley. The language, including the control structures and the expression grammar, was modeled on C. The C shell also introduced a large number of features for interactive work, including the history and editing mechanisms, aliases, directory stacks, tilde notation, cdpath, job control and path hashing.

Both shells have been used as coding base and model for many derivative and work-alike shells with extended feature sets.

Concept

The most generic sense of the term *shell* means any program that users employ to type commands. In the Unix operating system users may select which shell to use for interactive sessions. When the user logs in to the system the shell program is automatically executed. Many types of shells have been developed for this purpose. The program is called a "shell" because it hides the details of the underlying operating system behind the shell's interface. The shell manages the technical details of the operating system kernel interface, which is the lowest-level, or 'inner-most' component of an operating system.

Similarly, graphical user interfaces for Unix, such as GNOME, KDE, and Xfce can be called *visual shells* or *graphical shells*.

By itself, the term *shell* is usually associated with the command line. In Unix, users who want to use a different syntax for typing commands can specify a different program as their shell, though in practice this usually requires administrator rights.

The Unix shell was unusual when it was first created. Since it is both an interactive command language as well as a scripting programming language it is used by Unix as the facility to control the execution of the system.

Many shells created for other operating systems offer rough equivalents to Unix shell functionality.

On systems using a windowing system, some users may never use the shell directly. On Unix systems, the shell is still the implementation language of system startup scripts, including the program that starts the windowing system, the programs that facilitate access to the Internet, and many other essential functions.

Many users of a Unix system still find a modern command line shell more convenient for many tasks than any GUI application.

Due to the recent movement in favor of free and open source software, most Unix shells have at least one version that is distributed under an open source or free software license.

Bourne shell

The Bourne shell was one of the major shells used in early versions of the Unix operating system and became a *de facto* standard. It was written by Stephen Bourne at Bell Labs and was first distributed with Version 7 Unix, circa 1977. Every Unix-like system has at least one shell compatible with the Bourne shell. The Bourne shell program name is `sh` and it is typically located in the Unix file system hierarchy at `/bin/sh`. On many systems, however, `/bin/sh` may be a symbolic link or hard link to a compatible, but more feature-rich shell than the Bourne shell. The POSIX standard specifies its standard shell as a strict subset of the Korn shell. From a user's perspective the Bourne shell was immediately recognized when active by its characteristic default command line prompt character, the dollar sign (*$*).

C shell

The C shell was developed by Bill Joy for the Berkeley Software Distribution, a line of Unix operating systems derived from Unix and developed at the University of California, Berkeley. It was originally derived from the 6th Edition Unix shell (Thompson shell). Its syntax is modeled after the C programming language. It is used primarily for interactive terminal use, but less frequently for scripting and operating system control. C shell has many interactive commands.

Shell categories

Unix shells can be broadly divided into four categories: Bourne-like, C shell-like, nontraditional, and historical.

Bourne shell compatible

- Bourne shell (sh) -- Written by Steve Bourne, while at Bell Labs. First distributed with Version 7 Unix, circa 1978, and enhanced over the years.
- Almquist shell (ash) -- Written as a BSD-licensed replacement for the Bourne Shell; often used in resource-constrained environments. The sh of FreeBSD, NetBSD (and their derivatives) are based on ash that has been enhanced to be POSIX conformant for the occasion.
- Bourne-Again shell (bash) -- Written as part of the GNU Project to provide a superset of Bourne Shell functionality.
- Debian Almquist shell (dash) -- Dash is a modern replacement for ash in Debian.
- Korn shell (ksh) -- Written by David Korn, while at Bell Labs.
- mksh -- Descendant of the OpenBSD /bin/ksh and pdksh, developed as part of MirOS BSD.

- Z shell (zsh) -- considered as the most complete (read: the one with the most features) shell: it is the closest thing that exists to a superset of sh, ash, bash, csh, ksh, and tcsh.
- Busybox -- Tiny utilities for small and embedded systems, include a shell.

C shell compatible

- C shell (csh) Written by Bill Joy, while at the University of California, Berkeley. First distributed with BSD in 1978.
- TENEX C shell (tcsh)
- Hamilton C shell written by Nicole Hamilton, first distributed on OS/2 in 1988 and on Windows since 1992.

Other or exotic

- es, a functional programming rc-compatible shell written in the mid-1990s.
- fish (friendly interactive shell), first released in 2005.
- mudsh, an "intelligent" game-like shell that operates like a MUD.
- psh (Perl shell), a shell for Unix-like and Windows operating systems, combining aspects of bash (and other Unix shells) with the power of the Perl scripting language.
- pysh, a special profile of the IPython project, tries to integrate a heavily enhanced Python shell and system shell into a seamless experience.
- rc, the default shell on Plan 9 from Bell Labs and Version 10 Unix written by Tom Duff. Ports have been made to various Unix-like operating systems.
- scsh, a Scheme Shell.
- wish, a windowing shell for Tcl/Tk.
- zoidberg, a modular Perl shell written, configured, and operated entirely in Perl.

Configuration files for shells

Shells read configuration files on multiple circumstances which differ depending on the shell. This table shows the configuration files for popular shells:

	sh	ksh	csh	tcsh	bash	zsh
/etc/.login			login	login		
/etc/csh.cshrc			yes	yes		
/etc/csh.login			login	login		
~/.tcshrc				yes		
~/.cshrc			yes	yes		
~/.login			login	login		
~/.logout			login	login		
/etc/profile	login	login			login	
~/.profile	login	login			login	

~/.bash_profile					login	
~/.bash_login					login	
~/.bash_logout					login	
~/.bashrc					int.+n/login	
/etc/zshenv						yes
/etc/zprofile						login
/etc/zshrc						int.
/etc/zlogin						login
/etc/zlogout						login
~/.zshenv						yes
~/.zprofile						login
~/.zshrc						int.
~/.zlogin						login
~/.zlogout						login

Explanation:

- blank means a file is not read by a shell at all.
- "yes" means a file is always read by a shell upon startup.
- "login" means a file is read if the shell is a login shell.
- "n/login" means a file is read if the shell is not a login shell.
- "int." means a file is read if the shell is interactive.

Historic

- Thompson shell (sh) -- The first Unix shell, written by Ken Thompson at Bell Labs. Distributed with Versions 1 through 6 of Unix, from 1971 to 1975. Considered very rudimentary by modern standards and not used on current systems, though available as part of some Ancient UNIX Systems.
- PWB shell or Mashey shell (sh) -- A version of the Thompson shell, augmented by John Mashey and others, while at Bell Labs. Distributed with the Programmer's Workbench UNIX, circa 1976.

Chapter 14

Unix Security

Unix security refers to the means of securing a Unix or Unix-like operating system. A secure environment is achieved not only by the design concepts of these operating systems, but also through vigilant user and administrative practices.

Design concepts

Permissions

A core security feature in these systems is the permissions system. All files in a typical Unix-style filesystem have permissions set enabling different access to a file.

Permissions on a file are commonly set using the chmod command and seen through the ls command. For example:

```
-r-xr-xr-x  1 root  wheel  745720 Sep  8  2002 /bin/sh
```

Unix permissions permit different users access to a file. Different *user groups* have different permissions on a file.

More advanced Unix filesystems include the *Access Control List* concept which allows permissions to be granted to multiple users or groups. An *Access Control List* may be used to grant permission to additional individual users or groups. For example:

```
/pvr [u::rwx,g::r-x,o::r-x/u::rwx,u:sue:rwx,g::r-x,m::rwx,o::r-x]
```

In this example, which is from the chacl command on the Linux operating system, the user **sue** is granted *write* permission to the */pvr* directory.

User groups

Users under Unix style operating systems often belong to managed groups with specific access permissions. This enables users to be grouped by the level of access they have to this system. Many Unix implementations add an additional layer of security by requiring

that a user be a member of the *wheel* user privileges group in order to access the `su` command.

Issues

Most Unix and Unix-like systems have an account or group which enables a user to exact complete control over the system, often known as a root account. If access to this account is gained by an unwanted user, this results in a complete breach of the system. A root account however is necessary for administrative purposes, and for the above security reasons the root account is *seldom* used for day to day purposes (the sudo program is more commonly used), so usage of the root account can be more closely monitored.

Root access "as it should be" can be visualised by those familiar with the Superman stories using the following analogy:

Using a root account is rather like being Superman; an administrator's regular user is more like Clark Kent. Clark Kent becomes Superman for only as long as necessary, in order to save people. He then reverts to his "disguise". Root access should be used in the same fashion. The Clark Kent disguise doesn't really restrict him though, as he is still able to use his super powers. This is analogous to using the sudo program.

User and administrative techniques

Unix has many tools that can improve security if used properly by users and administrators.

Passwords

Selecting a strong password and guarding it properly are probably the most important things a user can do to improve Unix security. In Unix systems, the essential information about users is stored under the file `/etc/passwd`. This file keeps track of the users registered in the system and their main definitions. Passwords, or more correctly, the hash of the password, can also be stored in the same place. The entries in `/etc/passwd` occupy exactly one line each, and have the following form:

```
nickname:password_hash:UserID:GroupID:Complete_Name:home_dir:shell_bin
```

An example would be:

```
xfze:$1$zuW2nX3sslp3qJm9MYDdglEApAc36r/:1000:100:José Carlos D. S.
Saraiva:/home/xfze:/bin/bash
```

Since all users must have read access to this file, in order for the system to check the login password, one security issue was raised: anyone could have read the file and retrieve the password hashes of other users. To solve this problem, the file `/etc/shadow` was created to store the password hashes, with only root having read access. Under

password shadowing, the `/etc/passwd` the 2nd field (password hash) is replaced by an 'x' which tells the system to retrieve the corresponding user's password via the `/etc/shadow` file.

The `/etc/shadow` file usually only contains the first two fields:

```
xfze:$1$zuW2nX3sslp3qJm9MYDdglEApAc36r/:::::
```

The remaining fields in the `/etc/shadow` file include:

1. The minimum number of days between password changes
2. The maximum number of days until the password must be changed
3. The number of days of warning given before the password must be changed
4. The number of days after the password must be changed when the account becomes unusable
5. The date (expressed as the number of days since January 1st, 1970) when the account is expired

These fields may be used to improve Unix security by enforcing a password security policy.

Users and accounts

Administrators should delete old accounts promptly.

- su, sudo, ssh only, no remote root logins

Software Maintenance

Patching

Operating systems, like all software, may contain bugs in need of fixing or may be enhanced with the addition of new features. Patching the operating system in a secure manner requires that the software come from a trustworthy source and not have been altered since it was packaged. Common methods for verifying that operating system patches have not been altered include the use of cryptographic hash, such as an MD5 based checksum, or the use of read-only media.

From a security standpoint, the specific packaging method, such as the RPM Package Manager format originally from Red Hat Linux is not as important as the use of features which ensure the integrity of the patch itself.

Source Distributions

Source distributions include the ability to examine the code for suspicious content. The drawback, absent an accompanying cryptographic hash value, is that the user must be able to perform a security analysis of the code themselves.

RPM Packages

Linux distributions which use the RPM Package Manager format for providing base functionality and software updates make use of MD5 and GPG to ensure content integrity. The hash values are packaged with the RPM file and verified when the package is installed.

Debian Packages

Linux distributions which use the Debian .deb package format for providing base functionality and software updates make use of GPG signatures to ensure content integrity. A signature is computed when the package is constructed and verified later when the package is installed.

Other vendors and distributions

Regardless of the vendor or distribution, all software distributions should provide a mechanism for verifying that the software is legitimate and has not been modified since it was originally packaged.

Services

Unnecessary system software should not be installed or configured on a system. Software which is no longer required should be removed completely, if possible.

- Identify what services are running
    - netstat -na
    - lsof
    - nmap
    - sockstat -4 (FreeBSD)

The commands inetd and xinetd act as super-servers for a variety of network protocols such as rlogin, telnet and ftp.

Turning off unnecessary services

- using *chkconfig* on Red Hat Linux
- using */etc/rc.conf* and */usr/local/etc/rc.d* on FreeBSD (mention /etc/rc.local)
- using *rc-update* on Gentoo Linux

This approach is usually called *proactive security*. There are some operating systems which are *secure by default*. Amongst others, the free BSD flavours (FreeBSD, NetBSD, and OpenBSD) are proactively secure. For example, the output of netstat on a NetBSD 3.0 workstation clearly outlines this technique:

```
$ netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address
State
tcp        0      0  localhost.smtp         *.*
LISTEN
tcp        0      0  *.ssh                  *.*
LISTEN
Active Internet6 connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address
(state)
tcp6       0      0  localhost.smtp         *.*
LISTEN
tcp6       0      0  *.ssh                  *.*
LISTEN
Active UNIX domain sockets
Address  Type   Recv-Q Send-Q    Inode     Conn     Refs  Nextref Addr
c0d10d80 dgram       0      0        0 c0cd8680        0 c0cb7000 ->
/var/run/log
c0cb7000 dgram       0      0        0 c0cd8680        0        0 ->
/var/run/log
c0cd8680 dgram       0      0 cb9639e8        0 c0d10d80        0
/var/run/log
```

The following example from a BSD system

```
$ sockstat -4
USER     COMMAND    PID   FD PROTO  LOCAL ADDRESS         FOREIGN
ADDRESS
root     sendmail   569    4 tcp    localhost.smtp        *.*
root     sshd       593    4 tcp    *.ssh                 *.*
```

Shows that on this machine only the SSH service is listening to the public network interface of the computer. sendmail is listening to the loopback interface only. Access to a service may be further restricted by using a firewall.

File Systems

File system security

File system security within UNIX and Unix-like systems is based on 9 permission bits, set user and group ID bits, and the sticky bit, for a total of 12 bits. These permissions apply almost equally to all filesystem objects such as files, directories and devices.

The 9 permission bits are divided into three groups of three bits each. The first group describes the permissions of the file owner, the second group describes the permissions of

a group associated with the file owner or the directory containing the file, and the third group describes the permissions associated with any process which does not have the same user ID as the file. Each group of three bits contains a bit indicating the read, write or execute access is granted. In the case of directories, execute access is interpreted as the permission to perform a filename lookup within the directory.

The set user ID and set group ID bits, commonly abbreviated *set-UID* and *set-GID* respectively, are used to change the identity of the process which executes a file having either or both of those bits set. A file having the *set-UID* permission bit set will cause a process which executes that file to temporarily switch the effective user ID to that of the file owner. A file having the *set-GID* permission bit set will cause a process which executes that file to temporarily switch the effective group ID to that of the file group. A process may then alternate between the effective user or group ID which it inherited from the file and the real user or group ID which it inherited when the user logged on to the system. This provides a mechanism by which a process may limit the access rights it possesses to those code regions which require those access rights. This is a form of a security technique known as privilege separation and improves program security by limiting the unintended or undesirable actions of a processes.

A directory having the *set-GID* permission bit set will cause a newly created file to have an initial file group value equal to the file group of the directory. This provides a mechanism whereby a subsystem, such as the system's mail subsystem, can create files which have a common file group value so that *set-GID* processes within that subsystem are then able to read or write the file.

The *sticky bit*, formally known as the *save text on swap* bit, derives its name from its original purpose. Originally the *sticky bit* caused a process's initial memory image to be stored as a contiguous image on the disk drive which was used to store real memory pages when they were not in use. This improved the performance of commonly executed commands by making the initial memory image readily available. Modern UNIX systems no longer perform that function when the bit is set, but the name has been preserved nonetheless. In the case of files, the *sticky-bit* may be used by the system to indicate the style of file locking to be performed. In the case of directories, the *sticky bit* prevents any process, other than one which has super-user privileges or one having an effective user ID of the file owner, from deleting a file within that directory. The *sticky bit* is most commonly used on publicly writable directories, such as the various temporary working space directories on the system.

Viruses and Virus Scanners

Unix-like operating systems are immune to most Microsoft Windows viruses because binaries created to run on Windows generally won't run on other platforms. However, many Unix like installations provide file storage services to Microsoft Windows clients, such as through the use of Samba software, and may unintentionally become a repository for viruses stored by users. It is common for Unix servers to act as Mail Transfer Agents consequently email virus scanning is often installed. The ClamAV virus scanner is

available in source code form and may be used to scan Unix file systems for viruses which infect other operating systems.

There are viruses and worms that target Unix-like operating systems. In fact, the first computer worm -- the Morris worm -- targeted Unix systems.

Firewalls

A *firewall* derives its name from physical construction methods in which a reinforced, fire-resistant wall is used to protect opposite sides of the wall from a fire occurring on one side of the wall. In the same way, a network firewall protects systems and networks from network threats which exist on the opposite side of the firewall.

iptables

iptables is the current user interface for interacting with Linux kernel netfilter functionality. It replaced ipchains. Other Unix like operating systems may provide their own native functionality and other open source firewall products exist. More detailed information about iptables is contained elsewhere. A brief discussion is contained here in order to describe how iptables may be used to configure a Linux firewall.

netfilter provides a state-full packet filter which can be configured according to network interface, protocol, source and/or destination address, source and/or destination port and the state of the packet. A network packet traverses several *chains* between the time it is received by a network interface and the time it is accepted by the host or forwarded to another host. The common chains are **INPUT**, **OUTPUT** and **FORWARD**. The **INPUT** *chain* is traversed for all packets as they are received by a network interface, regardless of whether they are to be accepted by the host or forwarded to another host. The **OUTPUT** *chain* is traversed for all packets as they are transmitted by a network interface. The **FORWARD** chain is traversed for those packets are being routed through the host from one network interface to another, such as is the case for a multi-homed system (a system with more than one physical network interface).

Each of the built-in chains has a default *policy* which defines what action is taken for a packet which reaches the end of the chain. Packet traversal ends when a *rule* matches the packet and has an action of **ACCEPT**, **DROP**, **REJECT** or **RETURN**.

The simplest iptables firewall consists of *rules* for each desired service, followed by a rule which indicates that any packets which reach this rule are dropped. A system which only permitted, for example, incoming email traffic would have a rule which accepted connections on the SMTP port, and then dropped others. A rule would be required which indicated that all established connections were also permitted so that outgoing connections would receive responses from other systems.

INPUT chain

The following example shows a simple packet filter for the **INPUT** chain for the above described example:

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
destination
    0     0 ACCEPT     all  --  any    any     anywhere
anywhere            state ESTABLISHED
    0     0 ACCEPT     tcp  --  any    any     anywhere
anywhere            tcp dpt:smtp
    0     0 LOG        all  --  any    any     anywhere
anywhere            LOG level warning
    0     0 DROP       all  --  any    any     anywhere
anywhere
```

The addition of an explicit **DROP** action ensures that the packets are discarded should the default policy of the **INPUT** chain accidentally be changed to **ACCEPT**.

OUTPUT chain

There is less need for an **OUTPUT** chain and the default *policy* of the **OUTPUT** chain can safely be set to **ACCEPT**. In some instances it may be desirable for a firewall to limit certain outgoing connections to a certain set of approved systems. This is known as egress filtering and may be used to prevent viruses within the firewall from escaping to other systems. For example, it may be the policy of a network to limit outgoing email connections to a single authorized email servers as a way of combating e-mail spam. This could be achieved by the following example:

```
Chain OUTPUT (policy ACCEPT)
 pkts bytes target     prot opt in     out     source
destination
    0     0 DROP       tcp  --  any    any    !server
anywhere            tcp dpt:smtp
```

There is no need to include any other rules in this example as the default policy for the **OUTPUT** chain is **ACCEPT**. Note also that this rule assumes that the host which is acting as the firewall will not be sending email itself, such as to the email server. This is a good assumption as typically a firewall system contains the minimal amount of system code needed to act as a firewall.

A more restrictive **OUTPUT** chain would contain permissive (**ACCEPT**) entries for those services which may be accessed outside the firewall and then a restrictive (**DROP**) policy for the chain itself.

General

Secure network communication:

- Layer 7: GPG/PGP
- Layers 4,5: SSL/TLS/Stunnel/S/MIME
- Layer 3: VPN, IPsec
- Layer 2: PPTP

Packet sniffing:

- tcpdump, Wireshark

Attacks:

- Man in the middle attack
- land ping of death xmas Denial-of-service attack et al.

Advanced

- rootkits, kernel modules, chkrootkit
- exploit details, buffer overflows, local vs remote
- Security-Enhanced Linux

Service details

- banners
- SMTP - spam
- Sendmail - banners help header version etc.
- Domain Name System - reverse mapping dnssec

Chapter 15

Quattor

Quattor' is a generic open-source tool-kit used to install, configure, and manage computers. Quattor was originally developed in the framework of European Data Grid project (2001-2004). Since its first release in 2003, Quattor has been maintained and extended by a volunteer community of users and developers, primarily from the community of grid system administrators. The Quattor tool-kit, like other configuration management systems, reduces the manpower required to maintain a cluster and facilitates reliable change management. However, three unique features make it particularly attractive for managing grid resources:

- Federated Management: The open, modular nature of the tool-kit permits system administrators at different institutes to share the management of their distributed resources.
- Shared Configuration and Management Efficiency: Quattor encourages the re-use of configuration information in such a way that it can be distributed and used with little or no modification at different sites, facilitating the distribution of best practices without the need for each site to implement configuration changes.
- Coherent Site Model: Quattor allows an administrator to develop a site model that, once constructed, can be used to manage a range of different resources, such as real machines, virtual machines and cloud resources.

These features are also attractive beyond the grid context. This has been confirmed by the growing adoption of Quattor, by both commercial companies and academic institutions, most of them using the tool-kit to manage consistently their grid and non-grid systems.

Principles

The challenge of structuring and sharing components in a collaborative system is not new; over the years programming language designers have attacked this problem from many angles. While trends change, the basic principles are well understood. Features such as encapsulation, abstraction, modularity, and typing produce clear benefits. We believe that similar principles apply when sharing configuration information across administrative domains.

The Quattor configuration tool-kit derives its architecture from LCFG, improving it in several aspects. At the core of Quattor is Pan, a high-level, typed language with flexible include mechanisms, a range of data structures, and validation features familiar to modern programmers. Pan allows collaborative administrators to build up a complex set of configuration templates describing service types, hardware components, configuration parameters, users etc. The use of a high-level language facilitates code reuse in a way that goes beyond cut-and-paste of configuration snippets.

The principles embodied in Quattor are in line with those established within the system administration community. In particular, all managed nodes retrieve their configurations from a configuration server backed by a source-control system (or systems in the case of devolved management). This allows individual nodes to be recreated in the case of hardware failure. Quattor handles both distributed and traditional (single-site) infrastructures.

Devolved management includes the following features: consistency over a multi-site infrastructure, multiple management points, and the ability to accommodate the specific needs of constituent sites. There is no single “correct” model for a devolved infrastructure, thus great flexibility is needed in the architecture of the configuration system itself. Sometimes a set of highly autonomous sites wish to collaborate loosely. In this case each site will host a fairly comprehensive set of configuration servers, with common configuration information being retrieved from a shared database and integrated with the local configuration.

Distributing the management task can potentially introduce new costs. For example, transmitting configuration information over the WAN introduces latency and security concerns. Quattor allows servers to be placed at appropriate locations in the infrastructure to reduce latency, and the use of standard tools and protocols means that existing security systems (such as a public key infrastructure) can be harnessed to encrypt and authenticate communications.

Quattor Architecture

Configuration management system

Quattor’s configuration management system is composed of a configuration database that stores high-level configuration templates, the Pan compiler that validates templates and translates them to XML profiles, and a machine profile repository that serves the profiles to client nodes. Only the Pan compiler is strictly necessary in a Quattor system; the other two subsystems can be replaced by any service providing similar functionality.

Devolved management in a cross-domain environment requires users to be authenticated and their operations to be authorized. For the configuration database, we chose to adopt X.509 certificates1 because of the support offered by many standard tools, and access control lists (ACLs) because they allow a fine-grained control (an ACL can be attached to each template). When many users interact with the system, conflicts and

misconfiguration may arise which require a roll back mechanism; to this purpose, a simple concurrent transaction mechanism, based on standard version control systems, was implemented.

Quattor's modular architecture allows the three configuration management subsystems to be deployed in either a distributed or centralized fashion. In the distributed approach, profile compilation (at development stage) is carried out on client systems, templates are then checked in to a suitable database, and finally the deployment is initiated by invoking a separate operation on the server. The centralized approach provides strict control of configuration data. The compilation burden is placed onto the central server, and users can only access and modify templates via a dedicated interface.

Since the two paradigms provide essentially the same functionality, the choice between them depends on which fits the management model of an organization better. For instance, the centralized approach fits large computer centres well because of its strictly controlled work-flow, whereas multi-site organizations such as GRIF prefer the distributed approach because it allows different parts of the whole configuration set to be handled autonomously.

Pan language

The Pan language compiler sits at the core of the Quattor tool-kit. It compiles machine configurations written in the Pan configuration language by system administrators and produces XML files (profiles) that are easily consumed by Quattor clients. The Pan language itself has a simple, declarative syntax that allows simultaneous definition of configuration information and an associated schema. In this section, we focus only on the Pan features that are relevant to devolved management of distributed sites: validation, configuration reuse, and modularization.

Validation. The extensive validation features in the Pan language maximize the probability of finding configuration problems at compile time, minimizing costly clean-ups of deployed misconfiguration. Pan enables system administrators to define atomic or compound types with associated validation functions; when a part of the configuration schema is bound to a type, the declared constraints are automatically enforced.

Configuration reuse. Pan allows identification and reuse of configuration information through "structure templates." These identify small, reusable chunks of Pan-level configuration information which can be used whenever an administrator identifies an invariant (or nearly invariant) configuration sub-tree.

Modularization. With respect to the original design, two new features have been developed to promote modularization and large-scale reuse of configurations: the name-spacing and load-path mechanisms.

A full site configuration typically consists of a large number of templates organized into directories and subdirectories. The Pan template name-spacing mimics (and enforces) this

organization much as is done in the Java language. The name-space hierarchy is independent of the configuration schema. The configuration schema is often organized by low-level services such as firewall settings for ports, account generation, log rotation entries, cron entries, and the like. In contrast, the Pan templates are usually organized based on other criteria like high-level services (web server, mail server, etc.) or by responsible person/group.

The name-spacing allows various parts of the configuration to be separated and identified. To effectively modularize part of the configuration for reuse, administrators must be able to import the modules easily into a site's configuration and to customize them. Users of the Pan compiler combine a load-path with the name-spacing to achieve this. The compiler uses the load-path to search multiple root directories for particular, named templates; the first version found on the load-path is the one that is used by the compiler. This allows modules to be kept in a pristine state while allowing sites to override any particular template.

Further, module developers can also expose global variables to parameterize the module, permitting a system administrator to use a module without having to understand the inner workings of the module's templates.

Quattor Working Group (QWG) templates are used to configure grid middleware services. The QWG templates use all of the features of Pan to allow distributed sites to share grid middleware expertise.

Automated installation management

A key feature for administering large distributed infrastructures is the ability to automatically install machines, possibly from a remote location. To this purpose, Quattor provides a modular framework called the Automated Installation Infrastructure (AII). This framework is responsible for translating the configuration parameters embodied in node profiles into installation instructions suitable for use by standard installation tools. Current AII modules use node profiles to configure DHCP servers, PXE boot and Kickstart-guided installations.

Normally AII is set up with an install server at each site. However, the above mentioned technologies allow the transparent implementation of multi-site installations, by setting up a central server and appropriate relays using standard protocols.

Node configuration management

In Quattor, managed nodes handle their configuration process autonomously; all actions are initiated locally, once the configuration profile has been retrieved from the repository. Each node has a set of configuration agents (components) that are each registered with a particular part of the configuration schema. For example, the component that manages user accounts is registered with the path /software/components/accounts. A dispatcher program running on the node performs an analysis of the freshly retrieved configuration

for changes in the relevant sections, and triggers the appropriate components. Run-time dependencies may be expressed in the node's profile, so that a partial order can be enforced on component execution. For example, it is important that the user accounts component runs before the file creation component, to ensure that file ownership can be correctly specified.

By design, no control loop is provided for ensuring the correct execution of configuration components. Site administrators typically use standard monitoring systems to detect and respond to configuration failures. Nagios and Lemon are both being used at Quattor sites for this purpose. In fact, Lemon has been developed in tandem with Quattor, and provides sensors to detect failures in Quattor component execution.

While nodes normally update themselves automatically, administrators can configure the system to disable automatic change deployment. This is crucial in a devolved system where the responsibilities for, respectively, modifying and deploying the configuration may be separated. A typical scenario is that top-level administrators manage the shared configuration of multiple remote sites and local managers apply it according to their policies. For instance, software updates might be scheduled at different times.

Chapter 16

Advanced Configuration and Power Interface

In computing, the **Advanced Configuration and Power Interface** (**ACPI**) specification provides an open standard for unified operating system-centric device configuration and power management. ACPI, first released in December 1996, defines platform-independent interfaces for hardware discovery, configuration, power management and monitoring. The specification is central to *Operating System-directed configuration and Power Management* (OSPM); a term used to describe a system implementing ACPI, which therefore removes device management responsibilities from legacy firmware interfaces. The standard was originally developed by Intel, Microsoft, and Toshiba - later joined by HP and Phoenix. - and last (as of 2010) published as "Revision 4.0a", on April 5, 2010.

Overview

ACPI aims to consolidate and improve upon existing power and configuration standards for hardware devices. It provides a transition from existing standards to entirely ACPI-compliant hardware, with some ACPI operating systems already removing support for legacy hardware. With the intention of replacing Advanced Power Management, the MultiProcessor Specification and the Plug and Play BIOS Specification, the standard brings power management into operating system control (OSPM), as opposed to the previous BIOS central system, which relied on platform-specific firmware to determine power management and configuration policy.

The ACPI specification contains numerous related components for hardware and software programming, as well as a unified standard for device/power interaction and bus configuration. As a document that unifies many previous standards it covers many areas, for system and device builders as well as system programmers. Some software developers have trouble implementing ACPI and express concerns about the requirements that bytecode from an external source must be run by the system with full privileges. Linus Torvalds, creator of the Linux kernel, once described it as "a complete design disaster in every way", in relation to his view that "modern PCs are horrible".

Microsoft Windows 98 was the first operating system with full support for ACPI, with Windows 2000, Windows XP, Windows Vista, Windows 7, eComStation, FreeBSD,

NetBSD, OpenBSD, HP-UX, OpenVMS, Linux and PC versions of SunOS all having at least some support for ACPI.

OSPM responsibilities

ACPI requires that, once an OSPM-compatible operating system has activated ACPI on a computer, it then takes over and has exclusive control of all aspects of power management and device configuration. The OSPM implementation must expose an ACPI-compatible environment to device drivers, which exposes certain system, device and processor states.

Power States

Global states

The ACPI specification defines the following seven states (so-called global states) for an ACPI-compliant computer-system:

- **G0** (**S0**): *Working*
- **G1**, *Sleeping* subdivides into the four states S1 through S4:
  - **S1**: All processor caches are flushed, and the CPU(s) stop executing instructions. Power to the CPU(s) and RAM is maintained; devices that do not indicate they must remain on may be powered down.
  - **S2**: CPU powered off
  - **S3**: Commonly referred to as *Standby*, *Sleep*, or *Suspend to RAM*. RAM remains powered
  - **S4**: *Hibernation* or *Suspend to Disk*. All content of main memory is saved to non-volatile memory such as a hard drive, and is powered down.
- **G2** (**S5**), *Soft Off*: G2 is almost the same as G3 *Mechanical Off*, but some components remain powered so the computer can "wake" from input from the keyboard, clock, modem, LAN, or USB device.
- **G3**, *Mechanical Off*: The computer's power consumption approaches close to zero, to the point that the power cord can be removed and the system is safe for dis-assembly (typically, only the real-time clock is running off its own small battery).

Furthermore, the specification defines a *Legacy* state: the state on an operating system which does not support ACPI. In this state, the hardware and power are not managed via ACPI, effectively disabling ACPI.

Device states

The device states *D0-D3* are device-dependent:

- **D0** *Fully On* is the operating state.
- **D1** and **D2** are intermediate power-states whose definition varies by device.

- **D3** *Off* has the device powered off and unresponsive to its bus.

Processor states

The CPU power states *C0*-*C3* are defined as follows:

- **C0** is the operating state.
- **C1** (often known as *Halt*) is a state where the processor is not executing instructions, but can return to an executing state essentially instantaneously. All ACPI-conformant processors must support this power state. Some processors, such as the Pentium 4, also support an Enhanced C1 state (**C1E** or Enhanced Halt State) for lower power consumption.
- **C2** (often known as *Stop*-*Clock*) is a state where the processor maintains all software-visible state, but may take longer to wake up. This processor state is optional.
- **C3** (often known as *Sleep*) is a state where the processor does not need to keep its cache coherent, but maintains other state. Some processors have variations on the C3 state (Deep Sleep, Deeper Sleep, etc.) that differ in how long it takes to wake the processor. This processor state is optional.

Performance states

While a device or processor operates (D0 and C0, respectively), it can be in one of several power-performance states. These states are implementation-dependent, but P0 is always the highest-performance state, with P1 to P*n* being successively lower-performance states, up to an implementation-specific limit of *n* no greater than 16.

P-states have become known as SpeedStep in Intel processors, as PowerNow! or Cool'n'Quiet in AMD processors, and as PowerSaver in VIA processors.

- **P0** max power and frequency
- **P1** less than P0, voltage/frequency scaled
- **Pn** less than P(n-1), voltage/frequency scaled

Hardware Interface

ACPI-compliant systems interact with hardware through either a "Function Fixed Hardware (FFH) Interface" or a platform-independent hardware programming model which relies on platform-specific ACPI Machine Language (AML) provided by the original equipment manufacturer (OEM).

Function Fixed Hardware interfaces are platform-specific features, provided by platform manufacturers for the purposes of performance and failure recovery. Standard Intel-based PCs have a fixed function interface defined by Intel, which provides a set of core functionality that reduces an ACPI-compliant system's need for full driver stacks for providing basic functionality during boot time or in the case of major system failure.

Firmware interface

ACPI defines a large number of tables that provide the interface between an ACPI-compliant operating system and system firmware. For example:

- DSDT – Differentiated System Description Table
- SSDT – Secondary System Description Table
- SRAT – System Resource Affinity Table

The tables allow description of system hardware in a platform-independent manner, and are presented as either fixed-formatted data structures or in AML. The main AML table is the DSDT (differentiated system description table).

The Root System Description Pointer is located in a platform-dependent manner, and describes the rest of the tables.

ACPI Component Architecture (ACPICA)

The ACPI Component Architecture (ACPICA) provides an open-source OS-independent reference implementation of the ACPI specification.

History

The first revision of the ACPI specification was released in December 1996 supporting 16 and 32-bit addressing spaces. It wasn't until August 2000 that ACPI received 64-bit address support as well as support for multiprocessor workstations and servers with revision 2.0. In September 2004, revision 3.0 gave the ACPI specification support for SATA connectors, PCI Express bus, >256 multiprocessor support, ambient light sensors and user-presence devices, as well as extending the Thermal model beyond the previous processor centric support. The latest of the major publications is that of revision 4.0. Released in June 2009, the 4.0 specification added many new features to the design; most notable are USB 3.0 support, logical processor idling support, and x2APIC support.

Chapter 17

Time-Sharing

Time-sharing is the sharing of a computing resource among many users by means of multiprogramming and multi-tasking. Its introduction in the 1960s, and emergence as the prominent model of computing in the 1970s, represents a major technological shift in the history of computing.

By allowing a large number of users to interact concurrently with a single computer, time-sharing dramatically lowered the cost of providing computing capability, made it possible for individuals and organizations to use a computer without owning one, and promoted the interactive use of computers and the development of new interactive applications.

History

Batch processing

The earliest computers were extremely expensive devices, and very slow. Machines were typically dedicated to a particular set of tasks and operated by control panel, the operator manually entering small programs via switches in order to load and run other programs. These programs might take hours, even weeks, to run. As computers grew in speed, run times dropped, and suddenly the time taken to start up the next program became a concern. The batch processing methodologies evolved to decrease these dead times, queuing up programs so that as soon as one completed, the next would start.

To support a batch processing operation, a number of card punch or paper tape writers would be used by programmers, who would use these inexpensive machines to write their programs "offline". When they completed typing them, they were submitted to the operations team, who would schedule them for running. Important programs would be run quickly, less important ones were unpredictable. When the program was finally run, the output, generally printed, would be returned to the programmer. The complete process might take days, during which the programmer might never see the computer.

The alternative, allowing the user to operate the computer directly, was generally far too expensive to consider. This was because the user had long delays where they were simply

sitting there entering code. This limited developments in direct interactivity to organizations that could afford to waste computing cycles, large universities for the most part. Programmers at the universities decried the inhumanist behaviors that batch processing imposed, to the point that Stanford students made a short film humorously critiquing it. They experimented with new ways to directly interact with the computer, a field today known as human-computer interaction.

Time-sharing

Time-sharing developed out of the realization that while any single user was inefficient, a large group of users together were not. This was due to the pattern of interaction; in most cases users entered bursts of information followed by long pause, but a group of users working at the same time would mean that the pauses of one user would be used up by the activity of the others. Given an optimal group size, the overall process could be very efficient. Similarly, small slices of time spent waiting for disk, tape, or network input could be granted to other users.

Implementing a system able to take advantage of this would be difficult. Batch processing was really a methodological development on top of the earliest systems; computers still ran single programs for single users at any time, all that batch processing changed was the time delay between one program and the next. Developing a system that supported multiple users at the same time was a completely different concept; the "state" of each user and their programs would have to be kept in the machine, and then switched between quickly. This would take up computer cycles, and on the slow machines of the era this was a concern. However, as computers rapidly improved in speed, and especially in size of core memory in which users' states were retained, the overhead of time-sharing continually decreased, relatively.

The concept was first described publicly in early 1957 by Bob Bemer as part of an article in *Automatic Control Magazine*. The first project to implement a time-sharing system was initiated by John McCarthy in late 1957, on a modified IBM 704, and later on an additionally modified IBM 7090 computer. Although he left to work on Project MAC and other projects, one of the results of the project, known as the *Compatible Time-Sharing System* or CTSS, was demonstrated in November 1961. CTSS has a good claim to be the first time-sharing system and remained in use until 1973. Another contender for the first demonstrated time-sharing system was PLATO II, created by Donald Bitzer at a public demonstration at Robert Allerton Park near the University of Illinois in early 1961. Bitzer has long said that the PLATO project would have gotten the patent on time-sharing if only the University of Illinois had known how to process patent applications faster, but at the time university patents were so few and far between, they took a long time to be submitted. The first commercially successful time-sharing system was the Dartmouth Time Sharing System.

Development

Throughout the late 1960s and the 1970s, computer terminals were multiplexed onto large institutional mainframe computers (central computer systems), which in many implementations sequentially polled the terminals to see if there was any additional data or action requested by the computer user. Later technology in interconnections were interrupt driven, and some of these used parallel data transfer technologies such as the IEEE 488 standard. Generally, computer terminals were utilized on college properties in much the same places as *desktop computers* or *personal computers* are found today. In the earliest days of personal computers, many were in fact used as particularly smart terminals for time-sharing systems.

With the rise of microcomputing in the early 1980s, time-sharing faded into the background because the individual microprocessors were sufficiently inexpensive that a single person could have all the CPU time dedicated solely to their needs, even when idle.

The Internet has brought the general concept of time-sharing back into popularity. Expensive corporate server farms costing millions can host thousands of customers all sharing the same common resources. As with the early serial terminals, websites operate primarily in bursts of activity followed by periods of idle time. This bursting nature permits the service to be used by many website customers at once, and none of them notice any delays in communications until the servers start to get very busy.

Time-sharing business

In the 1960s, several companies started providing time-sharing services as service bureaus. Early systems used Teletype K/ASR-33s or K/ASR-35s in ASCII environments, and IBM Selectric typewriter-based terminals in EBCDIC environments. They would connect to the central computer by dial-up Bell 103A modem or acoustically coupled modems operating at 10–15 characters per second. Later terminals and modems supported 30–120 characters per second. The time-sharing system would provide a complete operating environment, including a variety of programming language processors, various software packages, file storage, bulk printing, and off-line storage. Users were charged rent for the terminal, a charge for hours of connect time, a charge for seconds of CPU time, and a charge for kilobyte-months of disk storage.

Common systems used for time-sharing included the SDS 940, the PDP-10, and the IBM 360. Companies providing this service included GE's GEISCO, IBM subsidiary The Service Bureau Corporation, Tymshare (founded in 1966), National CSS (founded in 1967 and bought by Dun & Bradstreet in 1979), Dial Data (bought by Tymshare in 1968), and Bolt, Beranek, and Newman. By 1968, there were 32 such service bureaus serving the NIH alone. The Auerbach Guide to Timesharing 1973 edition lists 125 different timesharing services using equipment from Burroughs, CDC, DEC, HP, Honeywell, IBM , RCA, Univac and XDS.

The computer utility

A great deal of thought was given in the 1970s to centralized computer resources being offered as computing utilities, the same as the electrical or telephone utilities. Ted Nelson's original "Xanadu" hypertext repository was envisioned as such a service. It became clear as the computer industry grew that no such consolidation of computing resources would occur as timesharing systems. Some argue that the move through client-server computing to centralized server farms and virtualization presents a market for computing utilities again.

Security

Security had not been a major issue for the centralized batch processing systems that were common when the time-sharing paradigm emerged. Neither was much more than username security required on many campuses. Commercial users, especially those in the financial and retail categories, demanded much higher security and also raised the issues that are being addressed today as companies consider the outsourcing of services. The first international conference on computer security in London in 1971 was primarily driven by the time-sharing industry and its customers. The same issues are still being tackled today on the Web and with SaaS products.

Time-sharing systems

Significant early timesharing systems:

Also see: Time-sharing system evolution

- Allen-Babcock RUSH Time-sharing System
- BBN PDP-1 Time-sharing System -> Massachusetts General Hospital PDP-1D -> MUMPS
- BBN TENEX -> DEC TOPS-20, Foonly FOONEX, MAXC OS at PARC, Stanford LOTS
- Burroughs Time-sharing MCP -> HP 3000 MPE
- UC Berkeley GENIE -> SDS 940 (Tymshare, BBN, SRI, Community Memory) -> BCC 500 -> MAXC at PARC
- UC Berkeley CAL-TSS (ran on CDC 6400)
- UC Berkeley BSD UNIX
- CDC KRONOS
- Compu-Time, Inc (Ran on a Honeywell 400/4000) Started 1968 in Ft Lauderdale, FL, moved to Daytona Beach in 1970.
- Dartmouth Time Sharing System (DTSS) -> GE Time-sharing -> GEnie
- DEC PDP-6 Time-sharing Monitor -> TOPS-10 -> TSS-8, RSTS-11, RSX-11 -> VAX/VMS
- HP-2000 Timeshared BASIC
- IBM TSS/360
- IBM CP-67 -> VM/CMS

- IBM CALL/360, CALL/OS - using IBM 360/50
- International Timesharing Corporation
- Michigan Terminal System
- Michigan State University CDC SCOPE/HUSTLER System
- MIT CTSS -> MULTICS (MIT/GE/Bell Labs) -> UNIX, PRIMOS
- MIT PDP-1 Time-sharing System -> ITS
- MUSIC/SP -> McGill University System for Interactive Computing
- National CSS -> VP/CSS (ran on IBM 360 series; originally based on IBM's CP/CMS
- Oregon State University OS-3 (ran on CDC 3000 series)
- RAND JOSS -> JOSS-2 -> JOSS-3
- Service in Informatics and Analysis (SIA) (ran on CDC 6600 Kronos system)
- SDC Q-32 Time-sharing System
- Stanford PDP-1 Time-sharing System -> SAIL -> WAITS
- Time Sharing Ltd. First commercial Time-sharing system in Europe and first dual (fault tolerant) Time-sharing system.
- Tymshare SDS-940 -> Tymcom X -> Tymcom XX
- XDS CP-V -> Honeywell CP-6

Chapter 18

Simple Network Management Protocol

Simple Network Management Protocol (**SNMP**) is an "Internet-standard protocol for managing devices on IP networks. Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks, and more." It is used mostly in network management systems to monitor network-attached devices for conditions that warrant administrative attention. SNMP is a component of the Internet Protocol Suite as defined by the Internet Engineering Task Force (IETF). It consists of a set of standards for network management, including an application layer protocol, a database schema, and a set of data objects.

SNMP exposes management data in the form of variables on the managed systems, which describe the system configuration. These variables can then be queried (and sometimes set) by managing applications.

Overview and basic concepts

In typical SNMP use, one or more administrative computers called managers have the task of monitoring or managing a group of hosts or devices on a computer network. Each managed system executes, at all times, a software component called an *agent* which reports information via SNMP to the manager.

Essentially, SNMP agents expose management data on the managed systems as variables. The protocol also permits active management tasks, such as modifying and applying a new configuration through remote modification of these variables. The variables accessible via SNMP are organized in hierarchies. These hierarchies, and other metadata (such as type and description of the variable), are described by Management Information Bases (MIBs).

An SNMP-managed network consists of three key components:

- Managed device
- Agent — software which runs on managed devices
- Network management system (NMS) — software which runs on the manager

A *managed device* is a network node that implements an SNMP interface that allows unidirectional (read-only) or bidirectional access to node-specific information. Managed devices exchange node-specific information with the NMSs. Sometimes called network elements, the managed devices can be any type of device, including, but not limited to, routers, access servers, switches, bridges, hubs, IP telephones, IP video cameras, computer hosts, and printers.

An *agent* is a network-management software module that resides on a managed device. An agent has local knowledge of management information and translates that information to or from an SNMP specific form.

A *network management system* (NMS) executes applications that monitor and control managed devices. NMSs provide the bulk of the processing and memory resources required for network management. One or more NMSs may exist on any managed network.

Management information base (MIB)

SNMP itself does not define which information (which variables) a managed system should offer. Rather, SNMP uses an extensible design, where the available information is defined by management information bases (MIBs). MIBs describe the structure of the management data of a device subsystem; they use a hierarchical namespace containing object identifiers (OID). Each OID identifies a variable that can be read or set via SNMP. MIBs use the notation defined by ASN.1.

Protocol details

SNMP operates in the Application Layer of the Internet Protocol Suite (Layer 7 of the OSI model). The SNMP agent receives requests on UDP port 161. The manager may send requests from any available source port to port 161 in the agent. The agent response will be sent back to the source port on the manager. The manager receives notifications (*Traps* and *InformRequests*) on port 162. The agent may generate notifications from any available port.

SNMPv1 specifies five core protocol data units (PDUs). Two other PDUs, *GetBulkRequest* and *InformRequest* were added in SNMPv2 and carried over to SNMPv3.

All SNMP PDUs are constructed as follows:

IP header	UDP header	version	community	PDU-type	request-id	error-status	error-index	variable bindings

The seven SNMP protocol data units (PDUs) are as follows:

GetRequest

A manager-to-agent request to retrieve the value of a variable or list of variables. Desired variables are specified in variable bindings (values are not used). Retrieval of the specified variable values is to be done as an atomic operation by the agent. A *Response* with current values is returned.

SetRequest

A manager-to-agent request to change the value of a variable or list of variables. Variable bindings are specified in the body of the request. Changes to all specified variables are to be made as an atomic operation by the agent. A *Response* with (current) new values for the variables is returned.

GetNextRequest

A manager-to-agent request to discover available variables and their values. Returns a *Response* with variable binding for the lexicographically next variable in the MIB. The entire MIB of an agent can be walked by iterative application of *GetNextRequest* starting at OID 0. Rows of a table can be read by specifying column OIDs in the variable bindings of the request.

GetBulkRequest

Optimized version of *GetNextRequest*. A manager-to-agent request for multiple iterations of *GetNextRequest*. Returns a *Response* with multiple variable bindings walked from the variable binding or bindings in the request. PDU specific *non-repeaters* and *max-repetitions* fields are used to control response behavior. *GetBulkRequest* was introduced in SNMPv2.

Response

Returns variable bindings and acknowledgement from agent to manager for *GetRequest*, *SetRequest*, *GetNextRequest*, *GetBulkRequest* and *InformRequest*. Error reporting is provided by *error-status* and *error-index* fields. Although it was used as a response to both gets and sets, this PDU was called *GetResponse* in SNMPv1.

Trap

Asynchronous notification from agent to manager. Includes current *sysUpTime* value, an OID identifying the type of trap and optional variable bindings. Destination addressing for traps is determined in an application-specific manner typically through trap configuration variables in the MIB. The format of the trap message was changed in SNMPv2 and the PDU was renamed *SNMPv2-Trap*.

InformRequest

Acknowledged asynchronous notification from manager to manager. This PDU uses the same format as the SNMPv2 version of *Trap*. Manager-to-manager notifications were already possible in SNMPv1 (using a *Trap*), but as SNMP commonly runs over UDP where delivery is not assured and dropped packets are not reported, delivery of a *Trap* was not guaranteed. *InformRequest* fixes this by sending back an acknowledgement on receipt. Receiver replies with *Response* parroting all information in the *InformRequest*. This PDU was introduced in SNMPv2.

Development and usage

Version 1

SNMP version 1 (SNMPv1) is the initial implementation of the SNMP protocol. SNMPv1 operates over protocols such as User Datagram Protocol (UDP), Internet Protocol (IP), OSI Connectionless Network Service (CLNS), AppleTalk Datagram-Delivery Protocol (DDP), and Novell Internet Packet Exchange (IPX). SNMPv1 is widely used and is the de facto network-management protocol in the Internet community.

The first RFCs for SNMP, now known as SNMPv1, appeared in 1988:

- RFC 1065 — Structure and identification of management information for TCP/IP-based internets
- RFC 1066 — Management information base for network management of TCP/IP-based internets
- RFC 1067 — A simple network management protocol

These protocols were obsoleted by:

- RFC 1155 — Structure and identification of management information for TCP/IP-based internets
- RFC 1156 — Management information base for network management of TCP/IP-based internets
- RFC 1157 — A simple network management protocol

After a short time, RFC 1156 (MIB-1) was replaced by more often used:

- RFC 1213 — Version 2 of management information base (MIB-2) for network management of TCP/IP-based internets

Version 1 has been criticized for its poor security. Authentication of clients is performed only by a "community string", in effect a type of password, which is transmitted in cleartext. The '80s design of SNMP V1 was done by a group of collaborators who viewed the officially sponsored OSI/IETF/NSF (National Science Foundation) effort (HEMS/CMIS/CMIP) as both unimplementable in the computing platforms of the time as

well as potentially unworkable. SNMP was approved based on a belief that it was an interim protocol needed for taking steps towards large scale deployment of the Internet and its commercialization. In that time period Internet-standard authentication/security was both a dream and discouraged by focused protocol design groups.

Version 2

SNMPv2 (RFC 1441–RFC 1452), revises version 1 and includes improvements in the areas of performance, security, confidentiality, and manager-to-manager communications. It introduced *GetBulkRequest*, an alternative to iterative GetNextRequests for retrieving large amounts of management data in a single request. However, the new party-based security system in SNMPv2, viewed by many as overly complex, was not widely accepted.

Community-Based Simple Network Management Protocol version 2, or *SNMPv2c*, is defined in RFC 1901–RFC 1908. In its initial stages, this was also informally known as *SNMPv1.5*. SNMPv2c comprises SNMPv2 *without* the controversial new SNMP v2 security model, using instead the simple community-based security scheme of SNMPv1. While officially only a "Draft Standard", this is widely considered the *de facto* SNMPv2 standard.

User-Based Simple Network Management Protocol version 2, or *SNMPv2u*, is defined in RFC 1909–RFC 1910. This is a compromise that attempts to offer greater security than SNMPv1, but without incurring the high complexity of SNMPv2. A variant of this was commercialized as *SNMP v2**, and the mechanism was eventually adopted as one of two security frameworks in SNMP v3.

SNMPv1 & SNMPv2c interoperability

As presently specified, SNMPv2 is incompatible with SNMPv1 in two key areas: message formats and protocol operations. SNMPv2c messages use different header and protocol data unit (PDU) formats from SNMPv1 messages. SNMPv2c also uses two protocol operations that are not specified in SNMPv1. Furthermore, RFC 2576 defines two possible SNMPv1/v2c coexistence strategies: proxy agents and bilingual network-management systems.

Proxy agents

A SNMPv2 agent can act as a proxy agent on behalf of SNMPv1 managed devices, as follows:

- A SNMPv2 NMS issues a command intended for a SNMPv1 agent.
- The NMS sends the SNMP message to the SNMPv2 proxy agent.
- The proxy agent forwards `Get`, `GetNext`, and `Set` messages to the SNMPv1 agent unchanged.

- GetBulk messages are converted by the proxy agent to `GetNext` messages and then are forwarded to the SNMPv1 agent.

The proxy agent maps SNMPv1 trap messages to SNMPv2 trap messages and then forwards them to the NMS.

Bilingual network-management system

Bilingual SNMPv2 network-management systems support both SNMPv1 and SNMPv2. To support this dual-management environment, a management application in the bilingual NMS must contact an agent. The NMS then examines information stored in a local database to determine whether the agent supports SNMPv1 or SNMPv2. Based on the information in the database, the NMS communicates with the agent using the appropriate version of SNMP.

Version 3

Although SNMPv3 makes no changes to the protocol aside from the addition of cryptographic security, its developers have managed to make things look much different by introducing new textual conventions, concepts, and terminology.

SNMPv3 primarily added security and remote configuration enhancements to SNMP.

Security has been the biggest weakness of SNMP since the beginning. Authentication in SNMP Versions 1 and 2 amounts to nothing more than a password (community string) sent in clear text between a manager and agent. Each SNMPv3 message contains security parameters which are encoded as an octet string. The meaning of these security parameters depends on the security model being used.

SNMPv3 provides important security features:

- Confidentiality - Encryption of packets to prevent snooping by an unauthorized source.
- Integrity - Message integrity to ensure that a packet has not been tampered with in transit including an optional packet replay protection mechanism.
- Authentication - to verify that the message is from a valid source.

As of 2004 the IETF recognizes *Simple Network Management Protocol version 3* as defined by RFC 3411–RFC 3418 (also known as STD0062) as the current standard version of SNMP. The IETF has designated SNMPv3 a full Internet standard, the highest maturity level for an RFC. It considers earlier versions to be obsolete (designating them "Historic").

In practice, SNMP implementations often support multiple versions: typically SNMPv1, SNMPv2c, and SNMPv3.

Implementation issues

SNMP implementations vary across platform vendors. In some cases, SNMP is an added feature, and is not taken seriously enough to be an element of the core design. Some major equipment vendors tend to over-extend their proprietary command line interface (CLI) centric configuration and control systems.

SNMP's seemingly simple tree structure and linear indexing may not always be understood well enough within the internal data structures that are elements of a platform's basic design. As a result, processing SNMP queries on certain data sets may result in higher CPU utilization than necessary. One example of this would be large routing tables, such as BGP or IGP.

Resource indexing

Modular devices may dynamically increase or decrease their SNMP indices (aka instances) whenever slotted hardware is added or removed. Although this is most common with hardware, virtual interfaces have the same effect. Index values are typically assigned at boot time and remain fixed until the next reboot. Hardware or virtual entities added while the device is 'live' may have their indices assigned at the end of the existing range and possibly reassigned at the next reboot. Network inventory and monitoring tools need to have the device update capability by properly reacting to the cold start trap from the device reboot in order to avoid corruption and mismatch of polled data.

Index assignments for an SNMP device instance may change from poll to poll mostly as a result of changes initiated by the system admin. If information is needed for a particular interface, it is imperative to determine the SNMP index before retrieving the data needed. Generally, a description table like ifDescr will map a user friendly name like Serial 0/1 (Blade 0, port 1) to a SNMP index.

Security implications

- SNMP versions 1 and 2c are subject to packet sniffing of the clear text community string from the network traffic, because they do not implement encryption.
- All versions of SNMP are subject to brute force and dictionary attacks for guessing the community strings, authentication strings, authentication keys, encryption strings, or encryption keys, because they do not implement a challenge-response handshake. Entropy is an important consideration when selecting keys, passwords and/or algorithms.
- Although SNMP works over TCP and other protocols, it is most commonly used over UDP that is connectionless and vulnerable to IP spoofing attacks. Thus, all versions are subject to bypassing device access lists that might have been implemented to restrict SNMP access, though SNMPv3's other security mechanisms should prevent a successful attack.

- SNMP's powerful configuration (write) capabilities are not being fully utilized by many vendors, partly due to lack of security in SNMP versions before SNMPv3 and partly due to the fact that many devices simply are not capable of being configured via individual MIB object changes.
- SNMP tops the list of the SANS Institute's Common Default Configuration Issues with the issue of default SNMP community strings set to ‘public’ and ‘private’ and was number ten on the SANS Top 10 Most Critical Internet Security Threats for the year 2000.

Autodiscovery

SNMP by itself is simply a protocol for collecting and organizing information. Most toolsets implementing SNMP offer some form of discovery mechanism, a standardized collection of data common to most platforms and devices, to get a new user or implementor started. One of these features is often a form of automatic discovery, where new devices discovered in the network are polled automatically. For SNMPv1 and SNMPv2c, this presents a security risk, in that your SNMP read communities will be broadcast in cleartext to the target device. While security requirements and risk profiles vary from organization to organization, care should be taken when using a feature like this, with special regard to common environments such as mixed-tenant datacenters, server hosting and colocation facilities, and similar environments.