***** Computer Select, July 1993 : Articles *****

Journal: Computer Shopper July 1993 v13 n7 p528(4)

Title: Year of the OS, part two: Unix versus Windows NT. (operating system)

Author: Bryan, John

Abstract: There are several differences between two of the most powerful desktop operating systems (OS), Microsoft Corp's Microsoft Windows NT and Unix. While Windows NT is designed to function as a standalone, prioritized multitasking system, most versions of Unix are multiuser, time-sharing and multi-tasking systems. Because of the their many powerful elements, both OSs require a substantial amount of computing resources to reach their potential. NT is an event-driven system; Unix is process-driven. Event-driven means NT prioritizes processor functions by events. An event is the initiation of a process caused by any external signal. NT is a true graphical operating environment, while Unix is character-based. The two main standards on which most Unix packages are based are System V Release 4 and OSF/1. NT probably has the best chance of becoming the dominant OS because it is a new system based on the best elements of its predecessors.

Full Text:

Last month, we defined the desirable features of a modern operating system, and compared those features to Windows/DOS, Unix, OS/2, NT, and System 7. Our wish list included: system-to-system communications, transparent application-to-application data exchange, multiple concurrent applications, a linear memory scheme with application protection, virtual memory, and support for multiple processors. When push came to shove, only two OSs have made it to part two: Unix and NT. This month, we'll take apart both operating systems and see just where the real power lies.

THE UNIX AND NT DIFFERENCE

Several fundamental differences exist between Unix, in any form, and Windows NT. First, Unix is a multiuser, multitasking, and time-sharing operating system, while NT is designed primarily as a single-user, prioritized multitasking OS. Actually, that is not quite true of NT: Though it does not support a terminal mode of many users accessing the same central processor, it does support client/server computing. The client is not restricted to using Windows for Workgroups, but may be run on distributed systems running DOS or Windows, or on a Macintosh or workstation running Unix. Both NT and Unix, however, are high-end operating systems that require a significant commitment in terms of computing power in order to be effective.

NT borrows a concept from System 7's design that produces a second functional difference between NT and Unix: NT is an event-driven operating system. Unix and DOS, and most other systems, are process-driven. An event-driven system prioritizes processor activity by events--that is, any external stimuli which signals the initiation of a process. This consideration has a further-reaching impact than one might at first think, though the most notable is the priority of the active window in the graphical environment.

Another conceptual difference is that NT is a truly graphical operating system, rather than a character-based OS with a graphical interface pasted on top, as is the case with DOS/Windows and Unix/Motif or Unix with X Windows. This produces a distinctly different look and feel in the manner in which the system handles output and application responsiveness. All screen output under NT is handled by the Win32 sub-system, with other application environments translated. Under Unix, an application may or may not be supported by the graphical interface, and many applications will manipulate the screen directly.

UNIX BY FLAVORS

Unix is in a process of unification toward a standard API, or Application Program Interface, but it isn't there yet. The vast majority of Unix flavors fall into one of two versions. The first is System V Release 4, or SVR4. SVR4 is AT&T's version of the complete operating system. SVR4 brings together most of the functions and systems of AT&T, Berkeley BSD, and Sun OS releases into a single package standardized under the SVID (System V Interface Definition) suite of benchmarks.

This other standard of Unix (I hesitate to say "second standard") is OSF/1. OSF, the Open Software Foundation, is a consortium of vendors led by IBM, DEC, and HP, all of whom were concerned about AT&T's agreements with Sun Microsystems regarding the direction and development of Unix. The group banded together to push OSF/1, a version of Unix based on the Mach kernel developed at Carnegie-Mellon University. The primary differences between OSF/1 and SVR4 is OSF/1's support, in its standard configuration, of symmetrical multiprocessing and "threads."

So how does each operating system handle the various functions in its environment? Let's take a closer look.

THE POWER OF UNIX

Unix had its beginnings in AT&T's development labs over 20 years ago, and has evolved into today's operating system of choice for a large number of platform types and processors. Without going into all of the various evolutions and permutations, it is sufficient to say that Unix has arrived in the '90s as a complete, highly functional OS that has been adapted to the widest range of systems of any operating system.

Unix consists of five major components. These include the operating system kernel, the user command interface, Unix commands and utilities, Unix system services, and the Unix programming interface. Unix is unusual in that the user command interface is not an integral part of the operating system. It is a separate program, loaded like any other, and interchangeable. There are several popular user command systems, among them the "C" shell and the Bourne shell. Another modification is the "X" window, or Motif interface, which gives Unix a graphical element, though these are not as complete as those found in a system designed from the "ground up" for a graphical interface.

AT&T's Unix is a monolithic design, meaning that all of the fundamental portions of the operating system, hardware dependent or otherwise, are interrelated to one another. In terms of adaptive system architecture, this means that Unix is hard to modify for new devices because of the amount of code that has to be changed.

This is an important point. Device support is probably the single most significant problem that any operating system faces in a non-regulated environment. By non-regulated, I mean any situation where hardware comes from a wide variety of vendors, with different system and peripheral types. Unix's origins are in the large-system environment, and Unix is still generally supplied by, and customized to, the hardware vendor.

OSF/1, with its Mach microkernel, has some advantages in this respect. Microkernel architectures operate on a client/server basis, and device support and other functions can operate out of user space. In this format, the system is protected from crashing in the case of an error in application code.

The Unix kernel, the heart or core of the operating system, is responsible for several functions. Among these are process scheduling, device management, disk and file management, system calls, and the user console interface. These general functions exist throughout the kernel in several modules.

The innermost layers interact directly with the system hardware, and perform the basic functions which give Unix its power. These layers are generally written with a fair amount of assembly code, which is non-portable, and means that modifications must be largely written. At this level, the operating system also handles process representation, scheduling, and dispatching. The process scheduler also handles synchronization (Unix is, first and foremost, a time-sharing OS) and interprocess communication.

I/O services are handled by Unix in a unique manner. Each I/O device is established as a special file, and is accessed by the Unix file system. All special files, as opposed to user data files, are kept in the /dev directory. Even a system's main memory is accessed through a special file. Unix file services provide protection for all files in a Unix system, and the kernel provides special protection for main memory and the active disk.

Most readers will be familiar with the layout of the Unix file system because it is the model upon which DOS is based. The standard Unix layout, though, contains several directories that are more or less mandatory to the function of the system. Besides the root directory,/, these include the /dev,/usr, /etc,/bin, and /usrs directories.

I've already mentioned that the /dev (for device) directory holds the special files used for device support. The /usr directory is where the system commands and program files are stored, in much the same fashion as a /dos directory in a DOS system (at least in mine). /etc contains the files which maintain information about the user: who is logged on to the system at any given moment, passwords, etc. /bin is where utility programs are commonly kept. /users may be located under /usr, or may mounted under /, and is where the individual users store their files and/or create their own directories.

Again, Unix is designed to be a time-sharing OS, granting every process a quantum of processor resources in order to maintain equitability of function for each user. In the graphical environment, since Unix does not prioritize the processes associated with running a GUI, as opposed to disk function or I/O, performance at that interface will frequently suffer. It is not uncommon to see individual windows in the "X" environment wait for a redraw as underlying processes, like a disk seek, execute.

One of the major problems is defining a "standard" Unix is that everyone has a version of the operating system ported to their hardware. Unix, in one incarnation or another, is available for every 32-bit and most 16-bit processors ever made. It has been adapted to the realtime environment, supercomputer systems, portables, mini- and mainframe installations, and personal computers.

ENTER NT

While Windows NT is not ported to every processor under the sun, it is available for systems based on Intel's 386-compatible CPUs, the MIPS R4000, and the DEC Alpha. This provides a wide array of system hardware to choose from, more or less something for everyone. And where Unix must have a graphical interface added (in the form of X Windows or Motif), the GUI of Windows NT is an integral part of the OS, identical in appearance to the already popular Windows 3.x.

The core of Windows NT is a microkernel, a similar design concept to Mach, the basis of OSF/1. A microkernel doesn't necessarily mean that the complete operating system is small, though, and in this case it certainly isn't (with all the bells and whistles, the beta release of NT likes 12Mb or more of RAM). What a microkernel does for NT is make the code easily adaptable to any type of 32-bit CPU. NT's kernel sits atop the hardware-dependent portion of the operating system, the HAL, or Hardware Abstraction Layer. The HAL is what makes each variation of underlying hardware appear the same to the kernel.

The HAL handles the system bus, the DMA controller, the memory system, the interrupt controller, and system timers, parceling out resources to the kernel. Microsoft has provided a HAL for Intel and compatible processors, for the MIPS R4000, and for Digital Equipment's Alpha. All of these are for uniprocessor machines. Manufacturers who produce multiprocessor SMP systems, like the Compaq Systempro, provide their own HAL for each platform. The division between HAL and kernel makes this a much easier task than adapting a Unix system to a customized platform.

Like Unix, the Windows NT environment is divided into two separate modes, user and kernel. In the kernel mode space are the system services, a blanket designation for several kernel functions, the kernel itself, and the HAL. The combined kernel and system services are called the NT Executive. While system services include many of the same functions found in the Unix kernel, there are some differences.

The components of the NT Executive are the kernel, the object manager, the security reference monitor, the process manager, the local procedure call facility, the virtual memory manager, and the I/O manager, which has several submodules under the aegis of its control. Each of these modules is isolated via an API-like interface that allows modification or replacement without changes being implemented in adjacent or underlying modules.

Windows NT is as close to an object-oriented operating system as is currently available. Object orientation means that the data that must be manipulated is formatted as a group of "objects" with a defined set of characteristics. This minimizes the amount of software that must be rewritten in order to encompass a change in data type. In an operating system, the "data" that is being manipulated are the components of the system, files, devices, memory, processes, etc. The creation, management, and deletion of these "objects" is handled by NT's object manager.

The process manager is NT's equivalent of Unix's swapper. This module manages the processes and threads of execution that make NT run. But where the Unix swapper runs as a continuous background process, the process manager runs in the context of the executing thread, reducing operating system overhead. Actually, as in OSF/1, the implementation of threads in NT also reduces system overhead, allowing more efficient operation of the OS.

Because NT supports several different operating subsystems (DOS, POSIX, OS/2, Win16, and Win32), the virtual memory manager has a rather complex task in providing protected memory space for all of the simultaneous potential applications that might be running on the system. Indeed, each of the processors currently supported by NT looks at memory in subtly different ways, so that the VM manager has to manipulate data blocks both coming and going.

NT's VM manager is able to map up to 4 gigabytes (232 bytes) of memory space, a not inconsiderable amount. Two gigabytes of this space are earmarked for user processes, while two gigabytes are reserved for kernel processes. In lieu of actually having this much RAM in one's system, the VM manager is able to swap "pages" of memory to disk as the physical memory of the system becomes full. NT supports page sizes from 4K to 64K, but the default is 4K. Because NT is a true multitasking OS, VM manager supports multiple pages associated with multiple threads of execution, protecting memory space while allowing processes shared access.

Windows NT is a secure system, meaning that each user must log on to the system and establish an ID. This system is managed by the security reference monitor. The government's National Computer Security Center has established several levels of security for computers and operating systems, from A1 (highest) to D (lowest). Versions of Unix are represented at about every level, though the standard implementation is generally rated at C1 (Discretionary Security). NT is designed to provide C2 protection (Controlled System Access), a slightly higher level.

The security reference monitor maintains this protection in much the same fashion as Unix security systems, with encrypted passwords, but NT has a generally higher level of granularity than Unix. Greater granularity means that more aspects of more objects are protected, including memory protection, device protection, and auditing of security-related activities. NT is also designed to be easily migrated to the B2 (Structured Protection) level of security, more appropriate for secure government installations or financial institutions.

The user mode of NT, as opposed to the kernel mode, supports the user subsystems, which communicate with the Executive via the Local Procedure Call Facility. The LPC is a highly optimized version of the industry-standard RPC (Remote Procedure Call) found in network environments. The LPC serves as the interface between the user subsystems and the executive, translating messages and coordinating the message traffic.

The messages come from several protected subsystems found in the user mode section of Windows NT. Of these subsystems, the Win32 subsystem is the most significant, for a variety of reasons. The Win32 subsystem controls all output to the system display, and receives all output from the application environments. There are five environment subsystems in which applications may be run. These are Win32, POSIX, OS/2, DOS, and Win16. These environments are call "clients."

The Win32, POSIX, and OS/2 environments are full 32-bit APIs. The DOS and Win16 APIs run in a VDM, or Virtual DOS Machine, which is supported by the Win32 subsystem. There are other subsystems that are not environments--for instance, the security subsystem and networking support subsystems.

To reiterate here, Windows NT is a graphical environment, not a graphical interface running on a character-based system. It looks like plain old Windows, but underneath it is a much different breed. Where Windows is an application for DOS, both Windows and DOS are applications on NT.

One benefit of having all of the client environments run under the Win32 subsystem is that Win32 provides a translation between APIs for the various client environments, making communication between applications possible. It is also another key of the system so that any may be modified, updated, or even removed without affecting the rest of the system.

From the standpoint of migration, the Win16 and DOS subsystems are especially important to the millions of current users of those systems. Both of these subsystems run in a VDM. Because DOS applications are particularly ill-behaved in a multitasking environment, it is the task of the VDM to isolate the DOS application and protect other subsystems while having the DOS app believe that it has control of the environment. DOS applications also handle memory in a different manner than NT's flat 32-bit model, and the Win32 subsystem handles the translation from the segment:offset model to the Windows NT native mode.

THE HARDWARE IT TAKES

Both Unix and Windows NT are high-end operating systems, and the resources required to run them effectively are significant. Unix has an installed base in the workstation environment, where Windows has been conspicuously absent. But the popularity of X Windows and Motif has made it obvious that a graphical interface is a desirable alternative to the command line for almost any user. All this aside, a significant proportion of systems currently in use are not capable of running either system, Windows NT or Unix with X or Motif.

The vast majority of installed systems in the business environment are built around the Intel 8086 processor, with a surprising percentage based on the 80286. After all, the IBM AT-339 was the single most popular computer system ever sold, and there are a fair few left running. But estimates of the number of PC-compatible systems run as high as 75 million, perhaps more, and even the percentage of these that are 386-based or higher far eclipses the number of RISC- or CISC-based workstations installed.

The requirements for running either operating system are remarkably similar. While Unix has been ported to a number of 16-bit processors, it isn't really worth running without a 32-bit engine, preferably a fast one. NT is 32-bit only; indeed, it is only available on three processor types, the 386 and compatibles family, the MIPS R4000, and the DEC Alpha.

Microsoft claims that NT is targeted to run on systems with as little as 8Mb of RAM by the time shrink-wrapped software reaches the shelf, though it certainly hasn't made it there yet. Unix will run quite handily in 8Mb of space, though the results are better if there is more memory. 100Mb is the minimum recommended disk size for either operating system, and again, more is better.

The ideal operating system would be able to run on all of the hardware platforms in the installed base, and would be easily portable to new hardware as it becomes available. Unix has this characteristic, but, unfortunately, it is not a single version of the operating system that accomplishes the task. Instead, every major manufacturer has ported a version of Unix to their systems, and applications are generally not compatible from one version to the next.

TOMORROW'S DESKTOP

Unix has been around for better than 20 years, and every few years some pundit says that Unix is "poised to take off" in terms of popularity. Somehow, though, it still hasn't. SVR4 is as close to unified API as Unix has gotten, yet there is some significant competition in the OSF/1 consortium. Sun Microsystems is the number one provider of workstation-class systems, but IBM is the number one provider of business computers. Of course, the waters are further muddied by IBM's support of OS/2 on its personal computer line, which could have an impact on NT's acceptance. In any case, what really hinders Unix is the lack of "shrink-wrapped" applications for a single standard operating system.

NT provides application compatibility across platform types, and it appears to offer enough additional benefits to justify its use as a primary operating system in just about any environment. One factor in the eventual popularity of NT will be the acceptance, by the power-user crowd, of systems built around the R4000, the Alpha, and Intel's Pentium. Introduction of these systems should push the price point of 486-based platforms somewhat lower than they already are, and this could also boost NT sales.

Microsoft has plans to introduce, in the next couple of years, operating systems that will enhance the position of NT as the patriarch (matriarch?) of a Windows "family" of systems, covering systems from portables to dedicated network, database, and video servers. Microsoft will release, sometime in 1993, Win32s, a version of the Windows NT API that will allow a 32-bit binary to run on a 16-bit Windows 3.1 system, and a new version of DOS, currently called "Chicago," that will (finally) break the 64OK barrier of all previous DOS releases.

Overall, Windows NT holds the better position in the race to become the dominant operating system of the second half of the '90s. Unix is a fine system, but it is carrying the baggage of 20 years of disparate development, while NT is a brand-new system that borrows on the strengths of a number of its predecessors. As this article is published, Windows NT may still not be available to the end user, but if the product is priced well, it should be phenomenally successful.

Company:	Microsoft Corp.
Product:	Microsoft Windows NT

```
Topic:     Operating Systems
           UNIX
           Comparison
           Functional Capabilities
           Software Design


Record#:   13 912 807.
                              *** End ***
```

***** Computer Select, July 1993 : Articles *****

Journal: PC Magazine June 15 1993 v12 n11 p240(4)

Title: Interactive Unix. (from SunSoft Inc.) (Software Review) (one of seven Unix operating systems evaluated in The Beast Turns Beauty: Unix on Intel) (Evaluation)

Author: Linthicum, David S.
Vaughan-Nichols, Steven J.

Abstract: SunSoft Inc's Interactive Unix version 3.0.1 is the oldest of the seven Unix operating systems for Intel processors evaluated in this series, and it is also one of the most solid. A new release is under development despite SunSoft's widely discussed plans to port Solaris to Intel processors. Interactive Unix will be sold to small businesses that need a multiuser, multitasking operating system, while Solaris will be marketed to large corporations needing a distributed computing environment for client/server, mission-critical applications. The operating system is based on the SVR3.2 kernel, so it is not as compatible with applications written for various Unix versions. Interactive Unix costs $495 for two users and $3,195 for unlimited users; a network version of the basic operating system costs $895. Visix Software's Looking Glass 2.01 Motif-based GUI is bundled with the operating system, but this is still not the best choice for users looking for their first Unix package.

Full Text:

Interactive Unix, Version 3.0.1

SunSoft Inc., 2550 Garcia Ave., Mountain View, CA 94043; 800-227-9227, 415-960-3200; fax, 415-336-0362 List price: Basic package for two users, $495; complete package for unlimited users, $3,195. Requires: 386-based PC or better, 4MB RAM (8MB recommended), 120MB hard disk space.

In short: Superior system administration tools, an excellent GUI, and stability are overshadowed by so-so DOS support and lack of non-TCP/IP protocols. Still, this product is good for small, Unix-oriented companies.

The oldest Unix version in this roundup is also one of the most solid. SunSoft's Interactive Unix, Version 3.0.1, has improved with age. And despite SunSoft's porting of Solaris to the x86 platform--a procedure that's still in progress and talked about frequently--Interactive hasn't been laid to rest; a new release is in the works. (For more information on Solaris, see the sidebar "Solaris 2.1: The Rise of a New Sun?")

While SunSoft is readying Solaris to accommodate after large corporations that need a distributed computing environment designed for client/server, mission-critical applications, Interactive remains a solution for small businesses that need a multiuser, multitasking operating system.

Interactive is well-suited for the task. The only drawback is that Interactive's kernel--SVR3.2--is not as compatible with applications written for different flavors of Unix as are the newer SVR4 and SVR4.2

versions of Unix.

You can purchase either a two-user version ($495) or a network version ($895) of the basic operating system. Most users, however, will want to opt for SunSoft's more complete configurations: Application Platform ($985), Network Platform ($1,315), Workstation Platform ($1,535), and Workstation Developer Platform ($2,635). The first of these is designed for users who will be running bread-and-butter character-based

DOS and Unix applications. The network package includes complete TCP/IP and NFS software. The workstation edition adds an excellent GUI system called Easy Windows. Finally, programmers will want to get their hands on the Workstation Developer Platform, which we used for our tests. This included the TCP/IP and NFS networking protocols, the VP/ix DOS emulator, the TEN/PLUS user environment, the software development system, and X Window.

EASY INSTALL AND ROCK-SOLID

Installing a Unix system is not an easy job, but SunSoft has made it as painless as possible for a disk-based system. Though we would prefer to see CD-ROM installation available, even with 3.5-inch floppy disks it took us only an hour to install the entire operating system. You could spend full days installing other versions of Unix. But if you have an unusual setup--say, an IDE hard disk and a SCSI unit--be prepared to get cozy with the manuals. The information you need will probably be in there, but only a Unix systems administrator could love these manuals.

Also, to reap the full benefit of any extra system RAM you might add, you must reconfigure and rebuild the kernel; the other reviewed Unix systems handle reconfiguration more transparently.

Any Unix administrator will appreciate Interactive's one-stop approach to system administration: the character-based sysadm utility. Almost every commonly used system administration task can be accomplished from sysadm; with its clean, complete design, the utility stands out.

Interactive's Unix foundation is SunSoft's port of AT&T's SVR3.2. Although far from the state of the art in Unix research, the operating system has been thoroughly tested and debugged. If you're tired of bleeding all over technology's cutting edge, you will find Interactive's stability a pleasant change of pace. SunSoft hasn't stuck completely to System V Release 3.2 of Unix. Interactive also included some enhancements present in BSD Unix. The most noteworthy of these is sendmail. This mail handler lets you more easily set up multiple-network node connections. Other BSD improvements will be available in Version 4.0 of Interactive. This upgrade will let you time events to within milliseconds instead of seconds, assign files names up to 255 characters in length, and create file names with "aliases," which are symbolic links to other files. These enhancements will let SVR3.2 Unix work with the newer SVR4 and BSD file systems.

STICK TO THE BASICS

Interactive's networking facilities are standard TCP/IP and NFS; you'll find no surprises here. The basic system comes with only standard Unix security; a C2-level module is available separately for $655.

The VP/ix DOS emulator runs well, within limits. While VP/ix can emulate

LIM EMS 4.0, it can't supply you with extended memory. Ordinary DOS character-based programs will run smoothly, but you can forget about anything more elaborate, such as Microsoft Windows 3.1. (Windows 3.0 is supported in Real mode only.) SVR3.2 is a Unix first and doesn't play as nicely with the DOS world as do the other operating systems.

On the other hand, since Interactive is compatible with SCO Xenix applications, you may not miss DOS. For office work under Unix, SunSoft supplies TEN/PLUS, which combines an easy-to-use file manager with a word processor and an optional e-mail interface. While not the equal of top-flight apps that do the same things, it is easy to use and functional enough for most.

For programmers, Interactive comes with two C compliers, a full-fledged context-sensitive editor, and a source-level debugger. Source code is provided for X Window System 11 Release 5, and you also get access to the Motif toolkit.

CALLING ALL SMALL BUSINESSES

Users who want to use a GUI, not write one, will enjoy Visix Software's Looking Glass 2.01, Interactive's bundled GUI desktop manager. Looking Glass's Motif-based user interface, ease of customization, and speed gives us what we want from a GUI: a way of getting more work from the machine without any hassles.

Interactive's documentation is plentiful, and we found the customer-support technicians (available for the price of a call) to be friendly and well-informed.

Interactive may not be as fancy as some of the other systems in this roundup, but then none of the others could run with as little memory and with just a 386SX. Popular Unix and DOS business software ran flawlessly. Interactive is a stable, tried, and tested product that makes an ideal backbone operating system for a small business. While it doesn't have the bells and whistles of UnixWare or SCO Open Desktop, users who are already comfortable with Unix may find it the ideal platform for their homes and offices. But if you're not already sold on Unix, this might not be the place to begin.

[TABULAR DATA OMITTED]

Type:	software review
	evaluation
Company:	SunSoft Inc.
Product:	Interactive Unix 3.0.1
Topic:	Evaluation
	Unix-like operating systems

Record#: 13 762 026.

*** End ***

Journal: Corporate Computing May 1993 v2 n5 p121(1)

Title: In defense of Unix security. (Special Report on Security: Weak Links)

Author: Ruber, Peter

Abstract: Most users see Unix as providing poor security, but its security is comparable to other systems, and new security products for Unix are coming out in 1993. Products are also emerging for a wider variety of Unix implementations than ever before. Unix security is increasingly important as Unix becomes more popular for distributed systems. Demax Software's SecureMax 3.0 software meets the National Security Agency's (NSA) C2 requirements. The package protects against break-ins, checks passwords and detects Trojan horses and worms. Woodside Technologies' Fortress software also detects Trojan horses and worms, as well as viruses. In addition, the package inoculates files and allows users to test possible passwords. Computer Associates' CA-Unicenter and Tivoli Systems' Tivoli Management Environment are large-scale security systems. CA-Unicenter monitors system performance and access attempts. Tivoli Management Environment provides complete management capabilities and will be incorporated in part into the Open Software Foundation's OSF/1 operating system.

Full Text:

Unix security has a bad rap, fueled in large part by the infamous "Morris" Internet worm. With one program and a small dictionary of possible passwords, one individual managed to disable tens of thousands of computers across the country. Is Unix really less secure than other major operating systems? Probably not, according to the experts. But it certainly is better known.

"Why do you think there are so few MVS accesses compared to Unix or DEC?" asks James Kates, a security consultant with Janus Associates of Stamford, Connecticut. "It's not because MVS is particularly secure. But just think about what's on campus. Our schools are training grounds for hackers." Today, though, Unix security tools are available that will stop most trespassers. In fact, 1993 promises to be a watershed year for Unix security. Not only are important big-iron products migrating to Unix this year (CA-Unicenter particularly comes to mind), but new security products will proliferate on a greater variety of Unix platforms, including Sun's Solaris, HP/UX, DEC's Ultrix, and IBM's AIX. With Unix increasingly seen as a major component in distributed corporate systems, such tools are a must for managers who are serious about downsizing.

Tightening the Grip

While Raychem Corp. doesn't think it has had any security breaches, it prefers not to wait for the inevitable. "Although our existing security was tight, we wanted to add virus protection, password production, and other security features," says John Denis, a systems administrator for

Raychem in Raleigh, North Carolina. So the company installed Fortress on its Sun servers and Sparcstations. Available from Woodside Technologies of Sunnyvale, California, Fortress automatically detects Trojan horses, worms, and viruses; inoculates files; looks for vulnerable passwords; and even lets users test potential passwords for security weaknesses. "It relieved me from having to check each user account [for security vulnerabilities]," says Denis. (See "Invasion-Proof Unix," April, page 47.)

Like Raychem, Chase Manhattan prefers to play it safe. Its data center in Brooklyn, New York, uses SecureMax 3.0 from Demax Software of San Mateo, California, for two Sun 690 servers. "We have about $800 billion in financial production flowing through this system each year," says Tama Herman, second vice president for the data center. The software, which complies with the National Security Agency's C2 minimum security requirements, automatically checks passwords, protects files against break-ins, and detects worms and Trojan horses. And because help screens can sometimes be too helpful, it also analyzes GUIs.

Chase Manhattan doesn't simply assume SecureMax is working properly; a senior system administrator periodically tests it by attempting to break into the protected network from the outside.

"So far he hasn't had much luck," Herman says happily. "SecureMax isn't hospitable to his efforts, and always reports him."

Fortress and SecureMax are examples of add-in packages designed specifically for security. Other vendors are taking a whole-enchilada approach to Unix management, offering all of the management functions needed to make Unix work like the big boys. Two such products are CA-Unicenter and the Tivoli Management Environment.

Initially available only in HP/UX and Solaris versions, Computer Associates' CA-Unicenter for Unix schedules work, monitors system performance, and tracks problems. It also provides an audit trail of access attempts; controls user sign-ons by date, time, and location; and validates passwords. Although the system is still too new for managers to offer much in the way of concrete impressions, Corporate Computing has placed CA-Unicenter on its short list of top business tools (see "Best Buys for 1993," January, page 133), and gave it high marks in its First Look section (see "Unix's Missing Link," February, page 43).

Like CA-Unicenter, Tivoli Systems' package provides an entire suite of Unix management tools. But more than that, some of the package's capabilities will find their way into the Open Software Foundation's OSF/1 operating system. That in itself represents a major advantage for companies anticipating a new Unix that could become de rigueur for cross-platform application development.

"Since Tivoli is the core technology of the OSF Distributed Management Environment, we felt it was our best bet to ensure that our corporate needs will be met," says Casey Canby, data center manager for Northwest National Gas of Portland, Oregon. "Besides, there wasn't anything else on the market to support our hardware environment," says Canby, who is responsible for a mix of Sun, Hewlett-Packard, and IBM RS/6000 servers and workstations.

Until now, Unix managers simply haven't had many choices. This year they will. As a result, Unix may emerge from university laboratories to assume a new and more secure place in the constellation of enterprise IS tools.

```
---------------------------------------------------------------------------
Company:   Woodside Technologies Inc.
           Demax Software Inc.
           Computer Associates International Inc.
           TIVOLI Systems Inc.
Product:   Fortress
           SecureMax 3.0
           CA-Unicenter for UNIX
           Tivoli Management Environment
Topic:     UNIX
           Security
           Trends
           Outlook
           User Ratings
           Security Software
           Management of EDP
           Software Packages
           Network Management Software


Record#:   13 746 825.
                              *** End ***
```

Journal: UNIX Review April 1993 v11 n4 p42(6)

Title: Easy UNIX security. (techniques, public-domain tools for enhancing security on UNIX operating system platforms) (Tutorial)

Author: Dichter, Carl

Abstract: Tools and techniques for enhancing UNIX operating system security are discussed. Most UNIX systems have a file containing encrypted versions of user passwords but can be 'cracked' by using a dictionary and rules to guess passwords. Shadow passwords are an effective defense; the encrypted passwords are kept in a different file which can be read only by the system manager. A public-domain replacement for the standard UNIX password program is available, and users should be prevented from choosing bad passwords. Machines connected to the Internet face such dangers as 'worm' and 'rabbit' programs spread by electronic mail. Some sites disable the 'finger' program, which locates users at other sites, to prevent infection, but a more effective and less intrusive measure is a 'firewall' that acts as a secure gateway to the outside world. Anonymous FTP (file transfer protocol) must be properly configured so that crackers cannot get at confidential system files. Managers should also ensure that they do not have versions of system programs containing well-known bugs crackers can exploit.

Full Text:

Security should be important to everyone. Many articles, including some in UNIX REVIEW (February 1988 and November 1989), illustrate The potential cost of a security breach may not mited to the intellectual property of a company's data. Here at Motorola ASIC, for example, our customers have confidential information that we are required to protect.

This article discusses inexpensive, reasonable, effective security measures that do not impede productivity. Since security is important, and it is now easy to improve, why wait any longer? On sites with any sensitive data, all accounts need to be secure-- even those with trivial information. Any account can give a potential "cracker" a foothold to get further into your system, your network, the rest of the world.

To emphasize this point, I asked permission to audit a site of another Fortune 500 company. I was given a normal user account. It took me less than two hours to download, compile, and run Crack, a public-domain tool to guess passwords. More than half the 200 users had easy passwords that Crack could guess. Several of these users were managers who had personnel information in their directories. Almost everyone had sensitive information, such as schedules, specifications, or code.

Password security. In most UNIX systems, the /etc/passwd file contains an encrypted version of the user's password. This encrypted version cannot be decrypted to form the original password. The only way to know whether a given string is the correct password is to encrypt the string and compare it with the user's encrypted password in /etc/passwd,

/etc/shadow, or yppasswd (YP/NIS).

For added security, the UNIX password system adds some extra information (called the "salt") to the password. The UNIX crypt(3) function takes an entered password and the salt and creates an encrypted version (containing the original salt as the first two characters). Also, to prevent someone from trying out millions of passwords, most UNIX implementations of crypt have a built-in delay loop.

crypt's encryption scheme purposely differs from the standard Data Encryption Scheme (DES) so that it is harder to implement in hardware. This feature is intended to keep crackers from making a "computer-age siege engine."

Crack works by using a dictionary and rules to make password guesses. It tries out these guesses with its own (or UFC) crypt using all salts found in the password file. Crack's built-in crypt, and the optional UFC crypt, are optimized for speed, and they don't have any intentional delays.

One effective defense against this sort of attack is shadow passwords. In a system with shadow passwords, the encrypted passwords are not kept in/etc/passwd. They are kept in /etc/shadow, which should be readable only by root.

Although shadow passwords are standard on many new systems (including SVR4), your site may not be so lucky. Fortunately, at least one implementation is available--appropriately called "Shadow." Unfortunately, it does not yet support NIS/YP. However, some versions of NIS (NIS+) support password shadowing.

As Larry Wall and Randal Swartz point out in Programming Peri,(1) cracking the password file to find bad passwords is not enough--it is better to prevent people from choosing bad passwords. Programming Peri provides a replacement for the password program. (See perl_passwd in the Tool Sources section at the end of this article.)

An improved version of the Perl passwd is also available. This improved version handles YP/NIS (yppasswd), and it uses the dictionaries and rules of Crack to determine password quality (see perl_passwd2 in the Tool Sources section).

Some strict password programs, such as the first Peri passwd mentioned, replace the normal password program and operate as a wrapper around passwd and yppasswd.

Other strict passwd programs are available (see passwd+ and npasswd in the Tool Sources section). Some of these programs also support password aging and prevent reuse of old passwords.

Better passwords do not have to be especially long. My six-character password is not guessed by the Crack program, even when the Crack dictionaries are loaded with information about me. However, some of the strict password programs do not allow short passwords, even if they are cryptic.

Network security. If your machine, or network, is connected directly to the Internet or a dialin modem, you face many dangers: mail, uucp, and rcp can be used to spread "worms," "rabbits," or other nasties ("malware"). For example, a bug in the earlier BSD fingerd (finger daemon) was the point of infection for the famous "Internet worm." Even a bugfree finger helps crackers get information to crack an account or confidential information about a project schedule and organization (from .plan and .project).

To protect these Internet-connected systems, administrators at some sites disable the finger daemon. But this measure has costs, too..You lose the basic use of finger--to help users at different sites find out how to contact each other. You also lose finger as a user's security tool. When run without arguments, finger provides information about others who are logged in, what their full name is, where they are logged in from, and so on. Also, users can run finger on a dormant account to see whether it shows an overly recent last login.

A more effective measure, and one that impinges less on productivity, is a "firewall." A firewall is a system that acts as a secure gateway to the outside world. If all Internet access is done through a firewall, outsiders cannot finger your employees or telnet into your network, but inside employees do not lose this ability. For more information, see Bill Cheswick's "The Design of a Secure Internet Gateway,"(2) David Curry's UNIX System Security: A Guide for Users and System Administrators,(3) and Simson Garfinkel and Gene Spafford's Practical UNIX Security.(4)

If your system has a modem, disable incoming use if possible (by disabling auto-answer and making sure the port is not set up with a getty). If the system must be used for incoming connections, consider adding a security box, such as SecureID from Security Dynamics (Cambridge, MA), to better authenticate users. Authentication requires users to prove that they are who they claim to be. The Security Dynamics system provides a credit-card-sized device for each remote user. This card displays an additional password, which, when used with a personal identification number (PIN), grants the employee access through the box. After access is granted, the user goes through the normal routine: port selection if you have a multiplexer, and then the UNIX login/password mechanism.

Other security dangers are present if you allow anonymous FTP, Trivial FTP (tftp), rlogin, and other network services. Unless anonymous FTP is properly configured, a cracker can easily get/etc/passwd or other files that can provide information to break in. Although tftp is intended to be a safer form of FTP used for booting diskless nodes, some early versions can be used to get/etc/passwd or other files.

Proper configuration is also the key to preventing users from accessing your system via remote shell (rsh, riogin) or remote copy (rcp). For example, the /etc/hosts.equiv file must contain only trusted hosts and must not be writeable by any user except root. Additional protection can be done on a per-user basis with a $HOME/.rhosts file.

Authentication is important in many types of communication, as we'll explore later. Authentication is also an issue with rlogin, rsh, and rcp. Kerberos is one way to address the problem, and it is included in some packages, such as OSF's Distributed Computing Environment (DCE). A public-domain Kerberos (see "Tool Sources" section) is available, and you can get commercial support from Cygnus Support (Mountain View, CA). Cygnus Support has fixes and workarounds for some of these problems and limitations. For more information on Kerberos, see "Authentication of

Unknown Entities on an Unsecure Network of Untrusted Workstations.''(5) Don't be intimidated by the long title--the paper is only a page long.

In Practical UNIX Security, Spafiord and Garfinkel point out that Kerberos is only for single-user systems; it has serious security problems otherwise. Spafiord has also told me that it may not work in some heterogeneous environments. (Also see "Who Are You?" UNIX REVIEW, November 1992.)

Another protection from misuse of services based on TCP/IP (such as tftp, exec, ftp, rsh, telnet, riogin, finger) is provided in the TCP wrappers written by Wietse Venema. Much more information on network security is contained in Practical UNIX Security(1) and Managing NFS and NISf ther security tools. When it comes to security, UNIX is not all that bad, although it does have terrible defaults. The default user's umask may be set to create files with wide-open permissions. System directories and files may have wrong permissions, all hosts may be trusted, and all ports may have a getty (login prompt). These bad defaults and inadvertently opened "holes" (by users and administrators) can make security on UNIX seem a herculean task.

Many tools, both commercial and in the public domain, make security more convenient. In this article, we're concentrating on the freeware. Despite the extensiveness of these packages and their low cost, they are surprisingly easy to install and use.

Another resource is the Computer Emergency Response Team (CERT) set up by DARPA. CERT has two especially useful functions: It responds to and distributes reports of security problems, and it helps collect and distribute security tools. CERT can be reached at (412) 268-7090 or cert [@]cert.sei. cmu.edu.

In addition to the weaknesses of the UNIX password system, the bugs in some network software, and sendmail (which has been fixed on most systems), other types of operating-system bugs can be exploited by crackers. A tool called crashme (by George Carrette) finds ways to crash an operating system by executing random strings of data as if they were programs. This causes exceptions and traps that should be properly handled by the operating system-- without crashing or corrupting the operating system environment. When the operating system does not properly handle these exceptions, a bug exists.

At the least, crackers could exploit these bugs to crash your system. However, they can do much worse. For example, Gordon Irlam (then of Adelaide University) discovered a way to get the system to grant him super-user status. This method was based on his knowledge of operating-system bugs he had found with crashme and other methods. Since Irlam is not a cracker, he reported this problem to CERq and the operating system vendor.

Computer Oracle and Password System (COPS) is a collection of more than a dozen security tools that together handle many areas of UNIX security. The5 were developed at Purdue University, under the direction of Gene Spafiord. The tools are implement. ed in two versions: one in C, the other in Peri.

A Readme file for COPS states that COPS performs the following checks:

* File, directory, and device permissions or modes.

* Poor passwords.

* Content, format, and security of password and group files.

* The programs and files run in/etc/rc* and cron(tab) files.

* Existence of root SUID files, their writeability, and whether or not they are shell scripts.

* A CRC check against important binaries or key files to report any changes therein.

* Writeability of users' home directories and start-up files (.profile, .cshrc).

* Anonymous ftp setup.

* Unrestricted tftp, decode alias in sendmail, SUID uudecode problems, hidden shells inside inetd - * conf, or rexd running in inetd.conf.

* Miscellaneous root checks, such as current directory in the search path, a "+" in/etc/host.equiv, unrestricted NFS mounts, and ensuring root is in /etc/ftpusers.

* Dates of CERT advisories va. key files. This feature checks the dates that various bugs and security holes were reported by CERT against the actual date on the file in question. A positive result doesn't always mean a bug was found, but it is a good indication that you should look at the advisory and file for further clues. A negative result, obviously, does not mean that your software has no holes, merely that it has been modified in some way (perhaps only touched) since the advisory was sent out.

To prevent security problems, make sure you are using a recent version of TFTP and that both FTP and TFTP are properly configured.(1)

COPS uses the Kuang expert system, which takes a set of rules and tries to determine whether your system can be compromised (for a more complete list of all of the checks, look at the file release.notes or cops.report; for more on Kuang, have a look at kuang.man).

According to its authors, Gene Kim and Gene Spafiord, Tripwire is a file-integrity checker, a utility that compares a designated set of files against information stored in a previously generated database. Any differences are flagged and logged, and a user can be notified through mail. When run against system files on a regular basis, any changes in critical system files will be spotted, and appropriate damage control measures can be taken immediately. Tripwire has been potted to most UNIX variants (including BSD, System V, OSF/i, and Mach).

Department of Defense organizations can get Network Security Monitor (NSM), a suite of routines that monitor and analyze network traffic, to determine whether an intruder is present. NSM captures Ethernet traffic, identifies new connections between host machines attached to the network, matches 1 strings in the connection data, and permits either live or off-line monitoring of intruders.

Secure mail. Security also is important with electronic mail. Two major issues must be addressed to ensure secure electronic mail:

* The message should be seen only by its intended recipient (authentication of recipient and privacy of message).

* The senders should be who they claim to be. When you consider the important applications of e-mail, it is surprising that the second item is so often overlooked. Many mail packages are appearing on the market and in the public domain to meet these needs. A popular technique is public-key encryption.

To help implement Internet Privacy-Enhanced Mail (PEM), RSA Data Security provides a free cryptographic toolkit called RSAREF. This toolkit supports RSA encryption and key generation as defined by RSA Laboratories' Public-Key Cryptography Standards (PKCS), MD2 and MD5 message digests, and Data Encryption Standard (DES) in cipher-block chaining mode. One implementation based on this toolkit, RIPEM from Mark Riotdan, is also available for free. This implementation operates with your mailer (for example, sendmail).

Other security issues. These security programs do not automatically give you a secure system.

Three other things are needed: proper system administration, physical security, and user participation. John Kinyon, Motorola's Manager of Network Security, provides these tips:

* Know where your programs are coming from and what they are supposed to do. This tracking includes shell scripts, PostScript files, spreadsheet macros, and other bits of executables for various applications, as well as binaries and operating-system updates.

* Make backups before installing software.

* Install the software in an execute-only mode.

* Put the executable files in a file system that is mounted as read-only.

* Install a tool that looks for suspicious changes in critical files. Public domain and commercial programs exist for this purpose, such as Tripwire.

* Install change-password programs that ensure thai obvious passwords are not used.

Dan O'Neill of Cadence Design Systems suggests:

* Limit the number of people with root access: You can use Tom Christiansen's op program to distribute limited root access on specific machines.

* Many administrators find it valuable to control administrative files (/etc/passwd,/etc/hosts, and others) under a configuration management system (RCS or SCCS).

* Auto-logout helps when people forget to log off.

A screen lock can be valuable because you can leave your programs running (even in foreground). However, some X-window versions can also be a security hole: If your X-window server is remote and it dies (power outage, reset), your workstation may be left as an unlocked, logged-in, console.

When a program or script needs to run as another user (or group), there is a right way and a wrong way to allow this. It is wrong to allow the system to grant user and group ID based on the suid/sgid bits in the file permissions. On most systems, this mechanism can be spoofed. It should be disabled in the kernel, if possible, or disabled by an option to mount(l) when file systems are mounted. The right way is to have the script or program call a set user or group ID mechanism. In C, you can use system calls (setuid, setgid). For Peri, a program called suidscript is provided to make a C wrapper for Peri programs that sets the user or group ID. Some newer shells provide setuid/setgid-type calls.

Careful administration can find security holes before they become breaches. At one company, a system administrator found that a computer vendor had set the FI function key to the root password--on all 12 terminals.

Extra care can also find the most unusual program behavior. Jerry Whelan of Bradley University tracked an unauthorized network incursion down to an unusual source: At another company, a person evaluating some commercial software did not realize that the software went into a "global network autodiscovery mode."

Your other job: Security. In addition to good system administration, good security requires proper account administration by users. Users need to be trained about the dangers of poor security and their own responsibility for good security.

Consider users who leave passwords next to the terminal: Until your users understand the importance of good security, and their responsibility for it, your system cannot be secure. Users must be informed of the dangers of bad security and the techniques of good security. For example, among the techniques used by sophisticated crackers, "social engineering" has evolved: the art of fooling users into giving them passwords or other information. A classic scam (I did! it in 1971) was a fake login screen: Users type in their login and password, the program records it, and then logs them in. Users don't know it, but their passwords are no longer a secret.

With proper user training, an explicit policy on security, and regular use of the utilities described here, significant improvements in security can be attained. .

Carl Dichter is a staff engineer/scientist at Motorola's ASIC Division. He is coauthoring Software Engineering with Peri, due soon from Prentice Hall. He can be reached at dichter[@] ureview.com

References

1. Wall, Larry, and Randal Swartz. Programming Peri. Sebastopol, CA: O'Reilly & Associates, 1992.

2. Cheswick, Bill. "The Design of a Secure Internet Gateway." research.att.com:/dist/Secure-Internet-Gateway. ps.

3. Curry, David (with Brian W. Kernighan, consulting editor). UNIX

System Security: A Guide for Users and System Administrators. Reading, MA: Addison-Wesley, 1992.

4. Garfinkel, Simson, and Gene Spafiord. Practical UNIX Security. Sebastopol, CA: O'Reilly & Associates, 1991.

5. Neuman, Clifford, and Jennifer G. Steiner. "Authentication of Unknown Entities on an Unsecure Network of Untrusted Workstations," USENIX UNIX Security Workshop Proceedings, August 1988.

6. Stern, Hal. Managing NFS and NIS. Sebastopol, CA: O'Reilly & Associates, 1991.

Recommended Reading

* Cheswick, Bill. "An Evening With Berferd: in which a Cracker is Lured, Endured and Studied." Available from: research.att.com: /dist/internet--Security/berferd.ps

* Spafiord, Gene. "The Internet Worm Program: An Analysis," ACM Computer Communications Review, 19(1), January 1989, pp. 17-57. (Also issued as Purdue Technical Report CSD-TR-823.)

* Proceedings of the USENIX Security Workshops and Symposia, USENIX Association, Berkeley, CA. office[@]usenix.org, (510) 528-8649.

Tool Sources

Note: Anonymous FTP locations are given as "site: filename"

* COPS: V1.04, available at these locations: cert.sei.cmu.edu: pub/cops, archive.cis.ohio-state.edu: pub/cops, or ftp.uu.net: pub/security/cops_1O4.tar. Z.

* Crack/UFC: Crack v. 4.If and UFC Patchlevel 1. This is available from the comp.sources. misc directory at any major Usenet archive or: usenet/comp.sources.misc/volume26/crack/part*Z.

* Kerberos: Public domain version, support available from Cygnus Support at ftp.uu.net:networking/athena/kerberos *.

* npasswd: (Clyde Hoover's "official" version), available at ftp.cc. utexas.edu: pub/npasswd/npasswd.tar. Z.

* NSM: Available to Department of Defense organizations only, from Tim Grance at the Air Force Cryptologic Support Center (AFCSC) at Kelly AFB, TX.

* Passwd+: "alpha version, update 3" (beta version due soon), available at dartmouth.edu: pub/passwd+.tar. Z or ftp.uu.net: pub/security/passwd+.tar. Z.

* perl--passwd, contained in the examples for Programming Peri, ftp.uu.net: published/oreilly/nutshell/perl/perl.tar. Z.

* perl_passwd2, which handles YP/NIS, is available from info. mcs.anl.gov:/pub/systems/papers/perl_passwd.ps.

* Securelib, available from eecs.nwu.edu: pub/securelib.tar or

ftp.uu.net: pub/security/securelib. tar. Z.

* Shadow, available from the comp.sources.misc directory at any major
Usenet archive (see entry for Crack).

* TCP Wrappers, available from cert.sei.cmu.edu: pub/network
_tools/tcp_wrapper.shar, ftp.win. tue.nl: pub/security/log_tcp .shar.
Z, or ftp.uu.net: tcp_wrappers.tar. Z.

* Tripwire, available from ftp.cs.purdue.edu: pub/spar/COAST /Tripwire.

To find other tools, join the cert-tools mailing list. Send e-mail to
cert-tools-request[@]cert.org.

Acknowledgements

Thanks to all the security gurus who helped me and to Michele Vening,
documentation engineer and security buff.

Type:	tutorial
Topic:	Data Security
	Operating Systems
	Public Software
	Tutorial
	Network Management
	UNIX
	Security Software

Record#:   13 497 724.

*** End ***

```
*****  Computer Select, July 1993  : Articles *****

Journal:    UNIX Review  April 1993 v11 n4 p109(1)
* Full Text COPYRIGHT Miller Freeman Inc. 1993.
----------------------------------------------------------------------------
Title:     Computer Security Institute. (hosting annual UNIX Security
           Conference) (Recent Releases) (Brief Article)


----------------------------------------------------------------------------
Full Text:

If this issue's coverage of UNIX security leaves you hankering for even
more information, consider attending the Second Annual UNIX Security
Conference hosted by the Computer Security Institute, Nov. 11-12, 1993.
Held in conjunction with CSI's 20th Annual Computer Security Conference
at the Anaheim Hilton in Anaheim, CA, the conference will feature
discussions by technical experts of UNIX security basics, TCP/IP
security, intrusion detection, authentication, advanced UNIX security
administration, business resumption planning, and other topics.

Contact Computer Security Institute, 600 Harrison St., San Francisco, CA
94107.  Tel.: (415) 905-2626, fax: (415) 905-2218.
----------------------------------------------------------------------------
Type:      brief article
Topic:     Conferences and Meetings
           Data Security
           Seminars
           Security Software
           Computer Security Institute
           Trade and Professional Associations


Record#:   13 498 340.
                              *** End ***
```

```
*****  Computer Select, July 1993  : Articles *****

Journal:    Digital News & Review  Dec 7 1992 v9 n23 p34(1)
* Full Text COPYRIGHT Cahners Publishing Associates LP 1992.
--------------------------------------------------------------------------
Title:     Anti-virus protection for Unix. (Woodside Technologies'
           Fortress security program) (Brief Article) (Product
           Announcement)


--------------------------------------------------------------------------
Full Text:
```

Woodside Technologies has introduced Fortress, a comprehensive graphical user interface-based security program with anti-virus protection for Unix. Unix users and system administrators can investigate and monitor system security risks. The Unix security tool kit features four security modules that run from a central menu. Users can check hazard areas including Trojan horses, worms, viruses, and weak passwords by clicking the appropriate icon from the main menu. Available now for Sun 4 and Sparcstation hardware, the SunOS 4.1+ operating system, and Open Windows versions 2.0 and 3.0, it is priced from $495 with quantity discounts. Woodside Technologies is located at 474 Potrero Ave., Sunnyvale, CA 94086, (408) 733-9503.

```
CIRCLE NO.  341
--------------------------------------------------------------------------
Type:      brief article
           product announcement
Company:   Woodside Technologies Inc.
Product:   Fortress
Topic:     Product Introduction
           Security Software


Record#:   13 217 282.
                              *** End ***
```

***** Computer Select, July 1993 : Articles *****

Journal: UNIX Review Nov 1992 v10 n11 p46(4)

Title: Who are you? (Kerberos system's authentication processes)

Author: McNutt, Dinah

Abstract: MIT's Kerberos system provides a unique solution to the authentication problem found in distributed computing environments. Authentication involves the systematic restriction and availability of a system to users depending on whether users have legitimate access rights. The traditional methods require passwords and user name log-ons. Kerberos assumes a distributed environment. The environment encompasses workstations that function in insecure locations and with untrustworthy software, as well as servers that exist in secure machine rooms with potentially untrustworthy software.The environment also consist of Kerberos key distribution machines operating in secure areas with trusted software. The Kerberos system must be physically secure and requires modification of all network services. The system can provide protection from external attacks and accounts for internal usage and access control.

Full Text:

A distributed computing environment has two main security issues: authentication and authorization. Authentication is the process of verifying that users are who they claim to be. Authorization involves determining whether a user has legitimate access to a resource and allowing the owner of a resource to define who has what type of access to the resource. This article focuses on the problem of authentication and describes Kerberos, which was developed at MIT as part of Project Athena, as a solution to authenticating users.

The problem. Traditionally, authentication is done with a user name and password combination that is verified against the entries in the system password file at login time. If a user has root access, it is possible for that user to masquerade as another by using the su command as the root user. Another problem is maintaining consistent copies of the password file (and propagating changes) on all of the systems. Sun's NIS offers a reasonable solution for small networks, but it is not a viable solution for a large number of users and workstations or for heterogeneity.

Even with reasonable schemes for managing password files, accessing resources on remote hosts requires either using a trusted host (or user) mechanism or providing user names and passwords for accessing remote services. These passwords are sent in the clear over untrusty networks.

UNIX has the concept of trusted hosts, where one host trusts another host. Users from the trusted host do not have to enter a password to access resources on the other host, as long as the user has an account on both machines. For instance, if host zempo trusts host rockytop, then user joe can remotely log in to host zempo from rockytop without entering a user name and password. Trusted hosts are a very powerful feature, and they make it easy for users to access network resources casually. UNIX also has the concept of a trusted user, which is similar to a trusted host, except a user from a specific host can be trusted and access your account without entering a user name and password on your host.

The negative side of trusted hosts is that you must implicitly trust the users and hosts on the systems you define as trusted. Not only must you trust them, but you must trust your network not to allow hosts to masquerade as each other.

If you do not define trusted hosts for your system or if you want to access your system from untrusted hosts, you must provide a user name and password that will be sent in clear text over the network. This makes your system vulnerable to anyone eavesdropping on your network. A solution to this problem is to encrypt the passwords, but this does not prevent someone from capturing the network packets containing the encrypted password and replaying the packets as if they were coming from you (and your system).

Kerberos solves authentication and enables authorization. Therefore, Kerberos provides a reliable mechanism for determining whether or not I am really the user dinah on the host rockytop. The applications must use the authentication information to make decisions about authorization. As I mentioned earlier, both authentication and authorization are essential for a secure environment, but without reliable authentication, any authorization scheme can be compromised.

What is Kerberos? Kerberos assumes a distributed environment where you have workstations in unsecure locations with untrusted software, servers in moderately secure machine rooms with potentially untrusted software, and Kerberos key distribution machines in secure areas with trusted software.

Kerberos uses a third-party authentication scheme in which each Kerberos client trusts the Kerberos server to identify the other clients on the network. In addition, information is time-stamped so that if an eavesdropper captures information (and can decipher it), the usefulness of the information will expire within a relatively short period of time (usually measured in hours).

Design goals of Kerberos are that:

* No clear text passwords be sent over the network

* No clear text passwords be stored on servers

* The exposure of client and server keys be minimal

* Compromises only affect the current session

* Authentication lifetime be minimal

* Authentication always be transparent during normal use

* Changes require minimal modifications to existing network applications.

When users request access to a Kerberized service their identities must be established. Users do this by presenting a ticket to the server providing the service along with the authenticator that proves the ticket was not stolen. This process involves three steps:

* Obtaining the credentials required to request access to other services

* Requesting authentication for a specific service (such as NFS)

* Presenting the credentials to the server offering the specified service.

Two types of credentials are used: tickets and authenticators. Both use private key encryptions, but they are encrypted using different keys. A ticket is used to transmit the identity of an individual to a server securely and to gain a temporary session key. The authenticator contains additional information to ensure that the client presenting the ticket and the client to whom the ticket was issued are the same.

A ticket contains the name of the server, the name of the client, the IP address of the client, a time stamp, a lifetime, and a random session key. This information is encrypted using the Data Encryption Standard (DES) and the private key of the server for which the ticket will be used. The ticket may be used by the client to access that particular server until the ticket expires. Then, another ticket must be obtained. When the server decrypts the ticket, it should find its own service name in the ticket. If it does, it compares the name of the client in the ticket to the client in the authenticator. If they match, it knows the client is who it claims to be.

An authenticator contains the name of the client, the IP address of the client, and a time stamp. An authenticator may only be used once; a client must create new authenticators at the moment of service requests.

Using Kerberos. Let's look at what happens when a user logs on to a system that has been modified to use Kerberos. Once the user name has been entered, the system sends a request to the authentication server that contains the user's name and the name of a service called the ticket-granting service (TGS). (I will use client name instead of user name because the mechanism is the same for applications requesting Kerberos authentication as it is for users. See the references at the end of this article for more information on Kerberos naming conventions.) If the authentication server has an entry for that user, it generates a random session key that will be used between the client and the ticket-granting server. It also generates a ticket that can be used by the client to request access to services on the network. This ticket contains the client's name, the name of the ticket-granting server, the current time, the lifetime of the ticket (this is configurable, but normally between 8 and 24 hours), the client's IP address, and the random session key. This ticket is encrypted with a key that is only known to the ticket-granting server and the authentication server. (Note that the client has no knowledge of the key or the contents of the ticket.)

This ticket is sent back to the client (along with a copy of the random session key) in encrypted form, using a key that is derived from the client's Kerberos password by applying an encryption algorithm to the password.

Next, the user is prompted for a password, which is converted to a DES key and used to decrypt the response from the authentication server. Note that the decryption is performed on the local system, and the user's password is not sent out over the network. The decrypied ticket-granting ticket and the session key are stored for future use (until the lifetime of the ticket expires). The next step is to gain access to a service.

The client must obtain a separate ticket for each service it wishes to use. Tickets must be obtained from the tgs; to access the tgs, you must first have a ticket*granting ticket. This is why we automatically received a ticket-granting ticket at login time. The request to the tgs contains the name of the server for which a ticket is requested, the ticket-granting ticket (which is encrypted using a password only known to the authentication service and the TGS), and an authenticator (remember that a client can build an authenticator as often as required). If the request is valid, the TGS generates a new random session key to be used between the client and the requested server. It also builds a ticket for the server that contains the client's name, the server's name, the current time, the client's IP address, and the new session key. The lifetime of the new ticket is either the remaining life of the ticket-granting ticket or the default for the server, whichever is smaller. The ticket is encrypted with the server's private Kerberos key, which is unknown to the client. The new ticket is sent back with the session key and is encrypted using the session key that was part of the ticket-granting ticket. This process prevents the user from having to re-enter the password.

Finally, the client requests access to the service by sending the newly acquired ticket and an authenticator to the server. The server decrypts the ticket using its private Kerberos key. Then, using the session key obtained from the decrypied ticket, it verities the information in the authenticator, the client name, the IP address, and the current time. If everything matches, the request to access the service is granted.

A summary of this process is shown in Figure 1. It seems complicated, but keep in mind that the user only had to enter the password one time.

Issues. One of the main problems in using Kerberos is that each network service must be modified or Kerberized. Not only do you require the source code for each application, but you must spend the effort to modify the necessary programs.

Your Kerberos server must be a physically secure system dedicated to providing the Kerberos service. This cost can be prohibitive for some sites. However, as the price of UNIX systems continues to drop, visualizing a Kerberos server that costs less than $5,000 for a large site is not too difficult. Small sites could use an even less-expensive solution. Keep in mind that access to services is denied if the Kerberos server is unavailable, so you should plan for redundant servers.

As stated previously, Kerberos is a solution to network security. It does not protect you if your systems are compromised, but it does make you less susceptible to external attacks by providing internal accountability and access control.

If a server or client password is obtained, it is possible for an eavesdropper on the network to use that password to decrypt tickets. Plainly put, sharing passwords is just as outrageously stupid as ever; don't do it and don't tolerate it.

Getting Kerberos. For a copy of the Kerberos source code and papers on Kerberos, contact MIT Software Center, W32-300, 20 Carlton St., Cambridge, MA 02139, (617) 253-7686. These files are also available via anonymous ftp from athenadist. mit.edu.

Dan Geer and Jon Rochlis teach a tutorial at USENIX, Interop, and other conferences. For information, contact the USENIX office at (714) 588-8649 or office@usenix.org.

Dinah McNutt is a customer support consultant at Tivoli Systems Inc. Tivoli, based in Austin, TX, specializes in distributed system management solutions for UNIX environments. She can be reached at dinah@ tivoli.com.

References

* Champine, George A., Dan Geer, and William N. Ruh. "Project Athena as a Distributed Computer System," Computer, September 1990, pp. 40-51.

* Steiner, Jennifer G., Clifford Neuman, and Jeffrey I. Schiller. "Kerberos: An Authentication Service for Open Network Systems," Usenix Conference Proceedings, (Winter 1988).

* Garfinkel, Simson, and Gene Spafiord. Practical UNIX Security. Sebastopol, CA: O'Reilly and Associates Inc., 1991.

* Russell, Deborah, and G.T. Gangemi. Computer Security Basics. Sebastopol, CA: O'Reilly and Associates Inc., 1991.

* Champine, George A. MIT Project Athena: A Model for Distributed Campus Computing. Digital Press, 1991.

* Geer, Dan, and Jon Rochlis. "Network Security: The Kerberos Approach," USENIX Summer 1992 Technical Conference (tutorial notes).

* Geer, Dan, Jon Rochlis, and Jennifer G. Steiner. "Security Issues in II* N%

Topic: Data Security
Security Systems
Access Rights
Passwords
Massachusetts Institute of Technology
Source Code

Record#: 12 663 598.

*** End ***

```
*****  Computer Select, July 1993  : Articles *****

Journal:    UNIX World  Oct 1992 v9 n10 p73(3).
--------------------------------------------------------------------------
Title:     New hope for UNIX security? (set of standards to improve
           security on UNIX operating system)(includes related articles
           on where to obtain more information and comparison of UNIX
           system security features)
Author:    Farrow, Rik


Abstract:  A new set of standards emerging from the federal government
           and Bell Communications Research Inc (Bellcore) promises to
           improve UNIX operating system security.  'Minimum Security
           Functionality Requirements for Multi-User Operating Systems'
           (MSFR), a standard developed at the National Institute of
           Standards and Technology (NIST) to cater to both vendors and
           buyers in the commercial market, was created by Bellcore, the
           National Security Agency and a variety of industry
           participants.  MSFR can be used as a guideline for upgrading
           UNIX to meet the NSA's 'Orange Book' requirements.  The lack
           of security in UNIX has kept many commercial users from
           adopting the system, and some doubt that even the new
           standards will be widely accepted because the commercial
           sector is reluctant to have the government involved and
           because vendors do not want to create or adopt standards that
           are not accepted in the user community.  MSFR's success or
           failure will depend partly on NIST's ability to promote the
           standard and partly on the efforts of commercial supporters.
--------------------------------------------------------------------------
Topic:     UNIX
           Data Security
           Management of EDP
           Standards
           Software Design
           Operating Systems
           United States. National Institute of Standards and Technology
           United States. National Security Agency


Record#:   12 677 869.
                              *** End ***
```

***** Computer Select, July 1993 : Articles *****

Journal: Software Magazine August 1992 v12 n11 p71(5)

Title: Open security an oxymoron?: Unix security features, tools available, but users must set and follow procedures. (includes related articles on OSF moving toward independence and Orange Book security) (Open Systems)

Author: Johnson, Jim
           Feit, Sidnie

Abstract: Security concerns are increasing as Unix finds increased use in corporate networks. The majority of MIS professionals do not believe Unix is as secure as proprietary systems. Organizations must overcome the perception that Unix is a security risk and reduce user resistance to security procedures in order for open systems to work in the corporate environment. User security functions should include identification and authentication, access control, accountability, audit trails, object reuse, accuracy, reliability of service and data exchange. The Massachusetts Institute of Technology (MIT) developed the Kerberos encryptographics system to avoid network eavesdropping. A variety of security systems are briefly described.

Full Text:

Unix security features, tools available, but users must set and follow procedures

ABSTRACT   MIS professionals worry that adding "open" systems such as Unix to the corporate network will leave the organization vulnerable to security breaches. But experts contend that Unix's bad reputation comes more from cultural resistance to security than functional deficiencies. The Open Software Foundation's DCE implementation of MIT's Kerberos encryption system helps foil network eavesdroppers.

As Unix gains a stronger foothold in corporate computing networks, security concerns surface. The academic and research community, where Unix grew up, strongly advocated open systems that allowed the free exchange of information and easy access for users. Not so the corporate community, where data is a competitive asset that MIS protects.

Given the cultural differences between these two communities, can Unix serve both? Or are "open systems" and "secure systems" mutually exclusive? Experts maintain that open systems can work in the corporate environment, but organizations must overcome two obstacles: the perception that Unix is a security risk, and user resistance to security procedures.

Only 11% of MIS professionals believe that Unix is as secure as proprietary systems, according to the Demands Assessment Requirement Tracking Service, provided by the Standish Group, Hyannis, Mass. One reason for this negative perception is cultural, said Laurence Brown, director of programs and operations at Unix International, Parsippany, N.J. "Unix systems grew up in a programming community, a research community and an academic community in which the interest was in being able to share."

In those early days of Unix, "open" meant something different than it does today. Small bands of users, "liberated" from the glass house, had easy access to every resource. Every user was a privileged user, and they could logon at any time. "In some of the universities where Unix was heavily used, security was viewed as interfering with intellectual freedom," said Alfred Spector, president of Transarc Corp., Pittsburgh.

Another reason for Unix's reputation among MIS professionals is its highly publicized security breaches.

In 1988, the Internet Worm, a self-replicating program created by Robert T. Morris, Jr. at Cornell University, Cornell, N.Y., affected approximately 6,000 Unix-based systems attached to the government and academic networks of Internet, Arpanet and Milnet. As it turned out, however, security holes were not part of the Unix operating system, but part of a mail program and information server.

Another incident occurred at the Lawrence Berkeley Lab, Berkeley, Calif., where a group of West German hackers broke into the lab's Unix network and the supposedly secure Department of Defense Milnet network. The hackers, who were selling information to the now defunct Soviet intelligence agency, the KGB, had wandered around unencumbered in the military industrial complex of computers for almost a year before being apprehended.

Today's Unix systems are continually improving on security. The newest versions of Unix System V from Unix System Laboratories (USL), Inc., Summit, N.J., and the OSF/1 operating system, based on technology from the Open Software Foundation (OSF), Cambridge, Mass., both include features that satisfy the requirements of the U.S. government's security provisions. The OSF's Distributed Computing Environment (DCE) has many built-in security features and includes application development tools that simplify the construction of secure applications.

SECURITY HOLES

Getting users to establish and follow security procedures can be difficult, however. In fact, the lack of attention to security issues by users may be the real security hole, many contend.

Because users have neglected security in the past, few have a solid understanding of what is involved in creating an overall secure environment. Tightening security also has repercussions, such as less access to data, more time spent for security checks and more computer cycles used.

"Our large customers are worried about security, but I'm beginning to see a pattern," said Roy Proudman, Ctos software product marketing manager at Unisys Corp., Blue Bell, Pa. "They wheel in a security expert. He makes demands. He gets in the way of operation. He becomes unpopular. He gets shoved aside, and security disappears.

"The security expert understands. He's paranoid about what serious, knowledgeable crackers can do, but the business operations people don't see it as a risk," Proudman added.

In addition to being costly, implementing security can be hard on users and systems administrators. Security systems tend to be complicated to plan and implement.

Vendors are starting to address the complexity problem, however. "Within our Top End product, we have implemented an interactive graphical utility," said Randy Smerik, production development department manager at NCR Corp., Dayton, Ohio, the network computing resource division of AT&T. "This graphical utility is list box-oriented so that you can select users, attributes and services by pointing and clicking."

Smerik added, though, that "you don't want to make security too easy. It's sort of a mindset change we have to go through; when you have good security, things aren't convenient." He explained that this lack of convenience does not mean giving up a quality, user-friendly interface, but it does mean more restrictions.

Implementing security does not have to be complicated, or inconvenient, said Ralph Hakim, chief information officer at Insurers World, Boston. The company uses the security features of the NetWare network operating system from Novell, Inc., Provo, Utah. "The simplest mechanism and most overlooked security feature is maintaining passwords," he maintained. "The security management must make users change passwords on a regular basis." He offered guidelines for preventing security breaches:

* Require unique user ID codes;

* Require users to identify themselves before performing an action;

* Keep the identity of all active users current;

* Disable user identification codes that have not been used for a period of time (60 to 90 days);

* Temporarily disable user ID codes during extended employee leave; and

* Limit multiple logon sessions by a single user.

Jim Crear, manager of information technology at Instrumentation Laboratories, Lexington, Mass., believes that password management is a major key to security. "Our users are required to change their passwords every 30 days." Crear said that the seven-digit alphanumeric passwords are encrypted. The firm has a security officer in addition to a system administrator.

"He is the only person who can change a user's password," said Crear. The user notifies the security officer when he or she changes a password. The security officer does not have access to the encrypted password file, however; the user must make the change. "Therefore, if he [the security officer] were to change a user's password, the user would point to him in cases of violation," he added. The firm is using Security/3000 from Vesoft, Inc., Los Angeles, as well as security features of the MPE operating system.

EIGHT SECURITY FUNCTIONS

In January 1992, the National Institute for Standards and Technology (Nist) in Gaithersburg, Md., drafted "Minimum Security Functional Requirements for Multi-user Operating Systems," which describes security requirements to users. The eight security functions are as follows:

Identification and Authentication. The traditional method for user identification and authentication is via a user ID and password. Since this information often serves as the basis for many other security operations, it is vital that the system protect user IDs and passwords from unauthorized access or modification.

Nist also commends the value of alternate authentication mechanisms, such as smart cards and biometrics such as hand geometry or thumb prints. "Fundamentally, you want to be assured that only authorized users can access your network," said Jim Geary, vice president of marketing at Cambridge, Mass.-based Security Dynamics, a leading supplier of smart card technology. "Therefore, we believe that the most important segment within the whole model is user authentication."

Although methods such as smart cards, cryptographic-based validation and biometrics are becoming more common, password systems are the most used authentication mechanism for system access.

Access Control. Users must be prevented from gaining unauthorized access to information or resources. The system should be able to control access down to single user. Increasingly, computers are addressing this requirement by supporting access control lists, which define who can run an application and who can create, read or write a file.

"Today's microcomputers not only have enormous capacity to house critical information and software assets, but they also represent the window into the total enterprise computing environment," said John Worthen, president of the Rocky Hill, Conn.-based Pyramid Development Corp., which develops and markets security and security management tools for PCs and workstations. "They must be protected via access control systems from inadvertent loss of information, malicious hacker destruction or compromise of data and virus disruption," he added.

Another mechanism is privilege control. Privilege control lets the system assign a user the privileges necessary to accomplish the task at hand, and no more. Sets of privileges can be bundled together to define functional job responsibilities.

Features such as access control lists and privilege control are important to industries such as health care, where sharing vital information is critical, but so is protecting the confidentiality of the patient.

"The main benefit of open systems architecture is the level of information sharing it creates throughout all components of an integrated information system," said Wendy Roloff, director of corporate marketing at CHC, Inc. in Houston, a supplier of health care information systems. CHC developed security features in-house.

"This is particularly true in the health care environment, because certain data in patient records are confidential, and not everyone who has access to the system should have knowledge of sensitive patient information," Roloff added. "A physician or nurse may have information privileges that are not shared by a pharmacist or lab technician."

Accountability. The system must be able to link the work done on behalf of a user with the user's identity.

Audit Trails. The system needs a way to find out whether security violations have occurred and, if so, what information or other resources were compromised. The usual mechanism is to run audits on a selected set of events such as logins, file updates and program initializations.

Object Reuse. The system should ensure that resources can be reused while preserving security. A user should not be able to read any kind of data left in memory by another user, or read the contents of documents that are in a shared printer queue.

Accuracy. The system should protect against unauthorized or undesired modification of data. This includes modifications to the system. For example, viruses should not be allowed to lodge themselves within systems by hiding inside legitimate system or end-user programs.

"This is one of the hardest areas to protect. The people who are launching the space shuttle care more about the unauthorized modification of data than who is listening to their commands," said Dr. Paul Karger, senior technical consultant, computer security, at the OSF. Karger, while working for Digital Equipment Corp., Maynard, Mass., was also reviewer of the Orange Book, the U.S. Department of Defense's Trusted Computer Systems Evaluation Criteria, published in December 1985. "This is the part of security that needs the most work," he added.

One solution is computing and saving an encrypted check sum, or summation of digits used to verify the integrity of data, for every file in a system. The check sum is recalculated when a file is accessed. If the new value does not match the old, it indicates tampering. Of course, the check sum program and the file holding the check sum must also be protected.

Reliability of Service. The system needs to protect itself against intentional or accidental monopolization by any user. No user should be able to commandeer 100% of the CPU or disk space, or interfere with time-critical operations.

Data Exchange. The system must promote the secure transmission of data over communications channels.

Implementing the above functions may be difficult and costly. "First, security has to be easy for programmers. It has to be relatively straight-forward for the developer of an application to put security into it," said Transarc's Spector. "Without the services of, for example, DCE security, it's a real pain in the neck to go off and add security to a program. A lot of programmers have never done it."

For successful implementation, security must also be easy to administer. One of the problems in heterogeneous systems is that every sub-system may have its own security model. So, when an organization adds a new user to a system, it might be necessary to add that user five times into five subsystems. That is inconvenient for administrators and users who might then have five separate passwords.

"The third thing is that while users expect, and will tolerate, some degradation in performance relating to security, it can't be catastrophic," said Spector. "For example, a lot of users might like to have end-to-end encryption across a network, where the client processor encrypts its data before sending it out to the server [which] then decrypts. That may be too expensive today. So, users may not use network privacy on an end-to-end basis for that reason until DES [Data

Encryption Standard] chips are common in every machine."

Spector said that other aspects of security are affordable. "Authentication is relatively inexpensive, as are authorization checks and auditing," he said. End-to-end encryption is cheaper when done in hardware. IBM,DEC and Xerox offer some fast end-to-end encryption systems.

TOOL VARIETY

Security in a networked environment is trickier than security for a single system. For example, an eavesdropper may have found a way to watch the network traffic, easily spotting user IDs and passwords.

The Athena project at the Massachusetts Institute of Technology (MIT), Cambridge, Mass., developed the Kerberos encryptographics system to foil eavesdroppers. Kerberos is named after the multiheaded dog that guards the gates of Hades. All the heads are alert at all times, and stored in a very safe place. Each head interacts with the user, with various checks and balances for stopping users from breaking into services they have no right to use. The OSF's DCE incorporates an implementation of Kerberos.

Third-party software vendors offer a variety of Unix security/auditing software products for the many flavors of Unix. These include: SecureMax-Network from Demax Software, Inc., in San Mateo, Calif., a security management package that analyzes remote node security, makes recommendations and monitors changes. SecureMax supports SunOS from Sun Microsystems, Mountain View, Calif.

Unix Guardian, from Datalynx, Inc., San Diego, Calif., is account profile and access security software that provides password control, access control with time windows and automatic logoff. It supports SunOS and DG/UX from Data General, Westborough, Mass.

The SafeWord Unix-Safe security system from Enigma Logic, Inc., Concord, Calif., verifies user identity through passwords, and allows access through a handheld password generator. The software supports Unix System V from USL;SCO Unix and Xenix from The Santa Cruz Operation, Santa Cruz, Calif.; Ultrix from Digital; and IBM's AIX.

From Information Security Corp., Deerfield, Ill., Secret-Agent is a menu-driven public-key encryption utility, which includes digital signatures, message authentication, Ascii conversion, data compression and file erasure utilities. It supports SCO Unix and Sun's SunOS.

Security Audit from SunSoft, Inc., Mountain View, Calif., supports Unix System V by detecting security violations, specifying corrective actions and suggesting procedures to reduce security problems.

A menu-driven security administration system, Usecure from Systems Center, Inc., Reston, Va., supports Sun Sparcstation/SunOS; DEC Ultrix; HP-UX from Hewlett-Packard Co., in Cupertino, Calif.; USL's Unix System V; and SCO Unix.

Supporting Unix System V, UniSec from UniSecure Systems, Inc., Newport, R.I., prevents unauthorized access and unauthorized file changes.

CMW + from SecureWare, Inc., Atlanta, is a trusted X Window-based multilevel workstation which binds on SecureWare's SMP+. It provides mandatory access control, information labels, discretionary access control, identification and authentication. It supports SCO Open Desktop and Apple Macintosh A/UX from Apple Computer, Cupertino, Calif.

Sentinel from Intelligent Software Products, Inc., Lynbrook, N.Y., protects modems and terminals from unauthorized entry. It supports SCO Unix and Xenix, Sun's SunOS and USL's Unix System V.

SoftLock from X-Lock Corp., Alpharetta, Ga., prevents access to the hard disk when the computer is booted from a floppy diskette. It supports Unix System V.

In addition, ACMW 4.0 from Addamax Corp., Champaign, Ill., is a security-modified version of AT&T's SRV4.0 Version 3.0.

'BAD GUYS' GETTING SMARTER

Admittedly, life was simpler when our computer systems were unencumbered by security mechanisms such as encryptographics systems and access control software.

When computers lived in the glass house, the mainframe was easy to secure: Combination door locks into the computer room limited access to data, and backup tapes were secured in a vault. However, the environment has changed. Data has grown more valuable as organizations increasingly rely on their computer systems. And users require more external computer services. Life is much more complex.

"Not only is life more complex, the bad guys are getting smarter," said the OSF's Karger.

"Hackers, or crackers, have a lot of tools to break into your system and steal your valuable data. You can go to your local mall bookstore and buy a book on how to become a cracker. Therefore, the good guys need to pay attention to network security issues and protect themselves."

Topic: Networks
Security
UNIX
Open Systems
Security Software
Methods

Record#: 12 552 265.

*** End ***

```
*****  Computer Select, July 1993  : Articles *****

Journal:    UNIX Review  August 1992 v10 n8 p53(1)
* Full Text COPYRIGHT Miller Freeman Publications 1992.
-------------------------------------------------------------------------
Title:     Practical UNIX Security. (Off the Shelf: Everything Else)
           (Brief Article)


-------------------------------------------------------------------------
Full Text:

For larger or networked systems, I recommend Practical UNIX Security.  It
is authoritative, detailed, and practical; it could keep you out of some
trouble.
-------------------------------------------------------------------------
Type:      brief article
Topic:     Books
           Review


Record#:   12 420 817.
                              *** End ***
```

Journal: UNIX Review August 1992 v10 n8 p53(1)

Title: UNIX System Security Handbook. (Off the Shelf: Everything Else) (Brief Article)

Full Text:

Bellcore's UNIX System Security Handbook is an annotated checklist for tightening up UNIX security. It could be used safely by most novice administrators and would improve security. The book is overpriced at $49, however. Bellcore should have O'Reilly trim the over-abundant white space, add some detail where needed, and issue it as a Nutshell Handbook. Bellcore's order department is at Room 4D110, 290 W. Mt. Pleasant Ave., Livingston, NJ 07039. Its telephone numbers are (800) 527-1080 (voice)and (201) 740-6882 (fax).

Type: brief article
Topic: Books
           Review

Record#:   12 420 969.

*** End ***

```
*****  Computer Select, December 1992  : Articles *****

Journal:    UNIX World  August 1992 v9 n8 p85(2).
--------------------------------------------------------------------------
Title:     Auditing UNIX security, effortlessly: Securemax/UX automates
           checking files, permissions, and ownership for baseline
           security. (Software Review) (Demax Software Inc.'s security
           software, version 1.1) (Evaluation)
Author:    Farrow, Rik


Abstract:  Demax Software Inc's $495 Securemax/UX 1.1 security software
           is a useful program for security administration on Sun
           Microsystem workstations that will find most of the problems
           that occur in a normal environment.  Securemax/UX is easy to
           install, which involves loading the software from a floppy
           disk into a temporary directory; an installation script does
           the rest.  The software focuses on five areas of system
           security.  System Security checks system files, Network
           examines weaknesses in UUCP security, Accounts checks the
           password file for correctness of user accounts, Passwords
           looks for missing passwords and File Systems checks all local
           file systems for ownership of files.  Securemax/UX is not
           perfect.  For example, the correction script includes no way
           to reverse corrections, and it did not find a change in the
           start-up file.  It is easy to use, however, and its report and
           correction tool can improve default security without requiring
           an experienced security expert.

--------------------------------------------------------------------------
Type:      software review
           evaluation
Company:   Demax Software Inc.
Product:   SecureMax/UX 1.1
Topic:     Evaluation
           Security Software


Record#:   12 455 874.
                               *** End ***
```

Journal: LAN Computing May 1992 v3 n5 p37(2)

Title: Oxymoron obliteration; UNIX and security - Systems Center's USECURE resolves the contradiction in terms that characterizes UNIX systems today. (Software Review) (Evaluation)

Author: Miller, David M.

Abstract: USECURE 3.0.3 from Systems Center's Unitech Software Inc provides easy-to-use, consistent security for Unix systems. There are two components to the program: UPASS, for password and user management; and USHELL, which replaces the Bourne shell and provides a high degree of control over user activities. USECURE 3.0.3 encrypts its data files and requires the insertion of an encryption key for use. Passwords can be user-defined or automatically generated. The program replaces some traditional Unix security control files and programs and modifies users' security information. It is important to follow installation procedures closely. The documentation is clear and understandable.

Full Text:

If the words "UNIX" and "security" are never in your mind at the same time, you're not alone. Until recently, the two terms were seldom mentioned in the same breath. But as UNIX gains acceptance as a commercial operating system, security is one area that vendors will quickly improve. Enter USECURE from Systems Center's Unitech Software division. USECURE adds an extra level of consistent, easy-to-use security to many UNIX systems. We ran USECURE version 3.0.3 on our HP 9000/834 running HP-UX version 7.0.

INSTALLATION

Systems Center products are managed with a package called UCONTROL, which is installed first. UCONTROL provides a menu-driven method of installing, upgrading and removing Systems Center products.

Installation is nearly automatic. During installation, you're prompted to enter the serial number and activation codes necessary to validate UCONTROL and any other System Center package you've purchased.

Because USECURE replaces some of the traditional UNIX security control files and programs, and because it can modify users' security information, it is important to strictly follow the installation checklist and to read the release notes and other USECURE documentation before proceeding with the installation.

Some installation tasks are worth explaining because they highlight USECURE features. For example, you're asked to specify the login name of the system security officer. The SSO has full security rights and is the only person authorized to make systemwide changes. However, the SSO does not have to be root, so an extra measure of security is realized here.

USECURE encrypts its data files. You are required to enter an encryption key for USECURE to use. Passwords can be user-defined or automatically generated. You determine your preference initially at installation.

CONFIGURATION

USECURE comprises of two components: UPASS, for password and user management and USHELL-- the really restricted shell.

We configured UPASS first. UPASS provides a menu system in which to do this, but you also can edit the UPASS main configuration file manually, once you know the names and possible settings of each parameter.

Figure 1 shows the systemwide parameter configuration screen, accessed from UPASS' primary menu. System settings determine the locations of various UPASS files and the messages users will receive related to their passwords.

You also establish those users who can help you maintain system security. Persons designated as UPASS operators will receive system security messages, but they can do little else. UPASS administrators can perform security related duties for other users, but they cannot affect the security of the SSO or other administrators, nor can they change the system configuration.

Other password related items you might consider are placing a limit on the number of times a user can guess his password, what the minimum and maximum password lengths should be, how long a password may exist before the user must change it and the use of separate dial-up passwords. User settings such as the default shell and default TERM variable value can be set up in this menu also.

Control can be exercised on the user level also. The User configuration menu lets you determine if particular users can have dial-up access, what their password lifetime is, when they get reminded to change their password, what ports they are allowed to use, specific settings for some environment variables such as PATH, the initial umask for files, their login shell, and what days and hours they are allowed access. User specific settings override systemwide settings where there is overlap.

Individual ports can be secured. The Port menu allows you to determine ports that can be used for dial up, the number of failed login attempts allowed, how long the user must wait before trying to log in again and how long the user has to log in. You can even restrict a port to specific users and restrict the hours that the port is available for logins.

UPASS' primary menu Password Management option provides specific control over users' passwords. The SSO can change a password, lock/unlock it to restrict/allow access or re-initialize the password to force a user to enter a new password immediately upon his next login.

Individual users are managed from the User Management primary menu option. In addition to modifying, deleting or acquiring the status of existing users, SSOs can enter typical new user information such as their user ID, group, home directory and login shell. A comments field is included to enter information such as the user's department, phone number, and so on.

Login groups are cared for via the Group Management primary menu option.

You can modify and delete user groups as well as create new ones.

SHE SELLS USHELL

USHELL (ush) is a replacement for the Bourne shell and provides a high level of control over what your users can do on your system.

Figure 2 is USHELL's systemwide configuration screen that establishes parameters for all USHELL class users. These parameters establish how detailed an activity log you need for audits, whether or not you force users to follow a secondary login procedure, and whether or not you want USHELL to check commands entered by the user (including internal shell commands such as cd and pwd) against an allowed command list.

Other features you can configure are the maximum number of minutes a user can stay logged on or have an idle terminal before being automatically logged off. Input and output redirection and the use of pipes can be restricted also.

If you configure USHELL to check each command entered by the user, USHELL compares the command against a systemwide, groupwide and/or individual "allowed commands list" you can establish.

Here, you establish the name of the command the user will enter, the specific directory in which the command resides, the specific file that will be run, and whether or not the user can invoke the command with switches and parameters. The last column lists users who are not allowed to use this command.

Further tailoring is possible. You can specify up to five profile command files that each USHELL user will run upon logging in. You also can establish a list of specific terminals that a user will be allowed to log in to. For example, you could restrict a user's dial-up access to a specific port.

Users become USHELL users when they are assigned /bin/ush as their login shell. This can be done through the UPASS/USHELL menu system or from the command line. In the event you are not using UPASS (UPASS and USHELL can be used independently), you can simply edit your /etc/passwd file to modify the user's password record.

Security officers have access to detailed audit trails of each user's activity. Log files are maintained on a user-by-user basis. A new log is created each time a user logs into the system. USHELL's menu allows the security officer to print, display and delete log files.

While you'll do a fair amount of work getting things configured, the user notices nothing unless they try to access things they shouldn't. From a user's perspective, nothing has changed.

DOCUMENTATION

UCONTROL's documentation is separate from System Center's other packages, such as USECURE. I appreciated the inclusion of typical configuration scenarios to use as guides in configuring my own system. The documentation is clear and understandable.

UNIX and security. Businesses banking in UNIX systems can't allow this contradiction in terms to exist any longer. In addition to knowing your users, a package like USECURE can give you the peace of mind needed to tackle your real MIS challenges while allowing users to freely do their jobs.

Type:	software review evaluation
Company:	Systems Center Inc. Unitech Software Inc.
Product:	USECURE 3.0.3
Topic:	Data Security Software Packages Evaluation
Record#:	12 182 986.

*** End ***

***** Computer Select, December 1992 : Articles *****

Journal: HP Professional April 1992 v6 n4 p58(4)

Title: Oxymoron obliteration. (Unitech Software Inc.'s USECURE 3.0.3 for UNIX systems) (Software Review) (From the Lab) (Evaluation)

Author: Miller, David B.

Abstract: Unitech Software Inc's Usecure data security software provides security to UNIX systems that most users never believed would be possible. The tested version of Usecure runs on an HP 9000/834 with HP-UX 7.0. Installation is very simple; Usercure prompts the user to enter serial numbers and activation codes to validate the software. Usecure replaces the usual UNIX security files; the user needs to follow a checklist provided and read the documentation before installation. UPASS and USHELL are the two main components of Usecure, and configuration of these establishes the security procedures for the system. USHELL replaces the Bourne shell and gives the manager control over which users have access to what on the system. All decisions about security and access are made when Usecure is set up and can be customized to particular needs. The documentation for Usecure is clear and well-written. Pricing ranges from $1,000 to $4,000.

Full Text:

If the words "UNIX" and "security" are never in your mind at the same time, you're not alone. Until recently, the two terms were seldom mentioned in the same breath. But as UNIX gains acceptance as a commercial operating system, security is one area which vendors will quickly improve.

Enter USECURE from Systems Center's Unitech Software division. USECURE adds an extra level of consistent, easy-to-use security to many UNIX systems. We ran USECURE version 3.0.3 on our HP 9000/834 running HP-UX Version 7.0.

Installation

Systems Center products are managed with a package called UCONTROL, which is installed first. UCONTROL provides a menu-driven method of installing, upgrading and removing Systems Center products.

Installation is nearly automatic. During installation, you're prompted to enter the serial number and activation codes necessary to validate UCONTROL and any other System Center package you've purchased.

Because USECURE replaces some of the traditional UNIX security control files and programs, and because it can modify users' security information, it is important to strictly follow the installation checklist and to read the release notes and other USECURE documentation before proceeding with the installation.

Some installation tasks are worth explaining because they highlight

USECURE features. For example, you're asked to specify the login name of the system security officer (SSO). The SSO has full security rights and is the only person authorized to make system wide changes. However, the SSO does not have to be root, so an extra measure of security is realized here.

USECURE encrypts its datafiles. You are required to enter an encryption key for USECURE to use. Passwords can be user defined or automatically generated. You determine your preference initially at installation.

Configuration

USECURE is comprised of two components: UPASS, for password and user management and USHELL -- The Really Restricted Shell.

We configured UPASS first. UPASS provides a menu system in which to do this, but you also can edit the UPASS main configuration file manually, once you know the names and possible settings of each parameter.

Figure 1 shows the system wide parameter configuration screen, accessed from UPASS's primary menu. System settings determine the locations of various UPASS files and the messages users will receive related to their passwords.

You also establish those users who can help you maintain system security. Persons designated as UPASS operators receive system security messages, but they can do little else. UPASS administrators can perform security related duties for other users, but they cannot affect the security of the SSO or other administrators, nor can they change the system configuration.

Other password related items you might consider are placing a limit on the number of times a user can guess his password, what the minimum and maximum password lengths should be, how long a password may exist before the user must change it, and the use of separate dialup passwords. User settings such as the default shell and default TERM variable value can be set up in this menu also.

Control can be exercised on the user level also. The User configuration menu lets you determine if particular users can have dialup access, what their password lifetime is, when they get reminded to change their password, what ports they are allowed to use, specific settings for some environment variables such as PATH, the initial umask for files, their login shell and what days and hours they are allowed access.

Individual ports can be secured. The Port menu allows you to determine ports that can be used for dialup, the number of failed login attempts allowed, how long the user must wait before trying to login again and how long the user has to login. You can even restrict a port to specific users and restrict the hours that the port is available for logins.

Reports you can generate from UPASS's menu system include a system log, password change report, a report of logins and who performed them, and an su listing. Other report formats provided by UPASS include listings of successful and unsuccessful logins and successful and unsuccessful pwd attempts.

UPASS's primary menu Password Management option provides specific control over users' passwords. The SSO can change a password, lock/unlock it to restrict/allow access or reinitialize it to force a user to enter a new password immediately upon his next login.

Individual users are managed from the User Management primary menu option. In addition to modifying, deleting or acquiring the status of existing users, SSOs can enter typical new user information such as user IDs, group, home directory and login shell. A comments field is included to enter information such as the user's department, phone number, etc.

Login groups are cared for via the Group Management primary menu option. You can modify and delete user groups as well as create new ones.

USHELL Sea Shells

USHELL (ush) is a replacement for the Bourne shell and provides a high level of control over what your users can do on your system.

Figure 2 is USHELL's System Wide configuration screen that establishes parameters for all USHELL class users. These parameters establish how detailed an activity log you need for audits, whether or not you force users to follow a secondary login procedure and whether or not you want USHELL to check commands (including internal shell commands such as cd and pwd) against an allowed command list.

Other features you can configure are the maximum number of minutes a user can stay logged on or have an idle terminal before being automatically logged off. Input and output redirection and the use of pipes can be restricted also.

If you configure USHELL to check each command entered by the user, USHELL compares the command against a system-wide, group-wide and/or individual allowed commands list.

Here, you establish the name of the command the user will enter, the specific directory in which the command resides, the specific file that will be run and whether or not the user can invoke the command with switches and parameters. The last column lists users who are not allowed to use this command.

Further tailoring is possible. You can specify up to five profile command files that each USHELL user will run upon logging in. You also can establish a list of specific terminals that a user will be allowed to login to. For example, you could restrict a user's dialup access to a specific port.

Users become USHELL users when they are assigned/bin/ush as their login shell. This can be done through the UPASS/USHELL menu system or from the command line. In the event you are not using UPASS (UPASS and USHELL can be used independently), you can simply edit your/etc/passwd file to modify the user's password record.

Security officers have access to detailed audit trails of each user's activity. Log files are maintained on a user-by-user basis. A new log is created each time a user logs into the system. USHELL's menu allows the security officer to print, display and delete log files.

While you'll do a fairr amount of work getting things configured, the user notices nothing unless they try to access things they shouldn't. From a user's perspective, nothing has changed.

Documentation

UCONTROL's documentation is separate from System Center's other packages, such as USECURE. I appreciated the inclusion of typical configuration scenarios to use as guides in configuring my own system. The documentation is clear and understandable.

UNIX and security. Businesses banking in UNIX systems can't allow this contradiction in terms to exist any longer. In addition to knowing your users, a package like USECURE can give you the peace of mind needed to tackle your real MIS challenges while allowing users to freely do their jobs.

Type:	software review
	evaluation
Company:	Unitech Software Inc.
Product:	Usecure
Topic:	Data Security
	Evaluation
	Security Software

Record#: 12 079 742.

*** End ***

```
*****  Computer Select, December 1992  : Articles *****

Journal:    UNIX World  April 1992 v9 n4 p59(3).
--------------------------------------------------------------------------
Title:     Improve your security. (developing a UNIX system security
           policy)(includes a related article describing security
           resources)
Author:    Farrow, Rik


Abstract:  Security experts claim that companies have more to fear from
           internal security breaches than external hackers; they insist
           that a computer security program begin with measures intended
           for employees.  UNIX systems are distributed by nature, which
           makes security planning more difficult, but there are many
           simple steps managers can take to make their UNIX systems more
           secure.  A security policy that clearly states the rules of
           computer use is the first step, after which an audit of the
           system's hardware and software is conducted to indicate
           security weak spots.  The weakest link in a computer system is
           often users; the security policy must specify penalties for
           breaking the rules as well as listing the rules themselves.
           The security audit will provide a status report on system
           security by examining file and directory permissions and
           ownership.  The trend toward smaller computers and components
           makes physical security more difficult; system access controls
           and UNIX security features are also discussed.

--------------------------------------------------------------------------
Topic:     UNIX
           Data Security
           Strategic Planning
           Management of EDP
           MIS
           Access Controls
           User Behavior
           Security Systems
           Methods
           Training of Employees


Record#:   12 021 154.
                              *** End ***
```

***** Computer Select, December 1992 : Articles *****

Journal: DEC Professional March 1992 v11 n3 p94(4)

Title: Basic tools for UNIX security. (ULTRIX operating system features for data security)(column) (UNIX) (Technical)

Author: Bourne, Philip E.

Abstract: ULTRIX, DEC's UNIX-like operating system, offers users tools to prevent unauthorized access to data. Version 4.2 carries a C2 security rating, substantially higher than the earlier versions which carried a D rating. Users can expect this rating to be even higher in the future when Open Desktop, the replacement system for ULTRIX, is implemented. Open Desktop is based on the OSF/1 operating system and will be able to be configured as a B1 system. The C1 rating of the current release of ULTRIX includes audit and password control functions. Users should carefully consider whether or not they need to audit their systems, since auditing both requires effort and uses disk space. Accounting is a powerful but poorly documented method for detecting break-ins. Using accounting functions can give managers both the identification and activity of an intruder.

Full Text:

This month we explore some basic but often overlooked features that can assist you in detecting break-ins to BSD-based UNIX systems, including ULTRIX. Great strides have been made in making UNIX more secure. Early versions of BSD UNIX had a D security rating, whereas ULTRIX V4.2 (and VMS, for that matter) has a C2 rating. The closer to A, and the higher the number, the more secure the system.

ULTRIX users can expect even higher levels of security in the future. The OSF/1 operating system, which is the foundation of Open Desktop and is slated to replace ULTRIX on RISC platforms, can be configured with simple BSD UNIX security features. Alternatively, it can be configured as a B1 system, with B2 and B3 features such as least privilege, trusted management facility, and access control lists (ACL). ACLs, the fine control over which users and which groups of users can access an individual file, come as no surprise to managers of VMS systems. However, they are new to UNIX.

Least privilege and the trusted management facility do away with the all-privilege-or-no-privilege concept of root versus user access. A graduation of privileges means, for example, that you can define operator-level accounts that permit read and write access to selected filesystems for making 0 level dumps but that do not permit the operator to have other privileges reserved solely for the root account, for example, the ability to edit the password file, /etc/passwd. Future releases of OSF/1 are expected to reach a complete B3 rating, implying, among other things, more sophisticated auditing.

With security features such as auditing and improved password control (part of the C2 rating), you can easily overlook some of the basic commands and utilities, which have been around since the early versions of UNIX, for detecting break-ins. Let's look at some of these tools, concentrating on those that receive scant or no coverage in the ULTRIX documentation.

Auditing

Should you audit? How you answer this question reveals a great deal about your security policy. Auditing requires effort and understanding on your part and consumes system resources, notably disk space. Whether auditing is justified depends on bow security-conscious you need to be. To understand what is required to audit an ULTRIX system, refer to Figure 1. It provides a useful preamble to the ULTRIX Security Guide for System Administrators, which is part of the ULTRIX documentation and is rich in detail but poor in putting the components of the auditing subsystem in perspective.

The audit tool is powerful. However, it must be administered closely, and it requires a significant amount of free disk space and some CPU and memory use. Human nature dictates that many of us will not consider auditing our systems until it is too late, that is, until a break-in is already suspected that requires detailed interrogation. If this occurs, how do you make the best of a bad situation? You resort to the standard BSD UNIX tools.

Figure 2 summarizes some of the events that you can monitor using basic BSD UNIX commands for signs of security violations. Your most powerful counterweapons are the accounting logs, if you happen to be running the BSD accounting program /etc/acct for billing or other purposes.

Accounting

Accounting is poorly described in the ULTRIX documentation, but several books cover BSD accounting in detail. UNIX System Administration Handbook by Evi Nemeth, et al. (PrenticeHall, 1989), and Essential System Administration by Aeleen Frisch (O'reilly & Associates, 1991) are excellent sources of information. The ULTRIX accounting software comes as an optional software subset loaded with the etc/setld command. Once loaded, accounting is toggled on and off, usually at boot time from the file etc/rc.local, with the command /etc/accton.

If a specific accounting file is not specified, each time a process terminates, the kernel writes a record to the default file, /usr/adm/acct - a rich source of information in looking for potential security violations. As you can imagine, this file grows quickly and usually is summarized daily. The command /etc/sa (summarize accounting) serves this purpose and is a tool to search for security violations. Figure 3 illustrates the use of sa in conjunction with grep for interrogating the accounting file in search of an intruder.

If you have some notion that a particular account is being used to infiltrate the system, you can interrogate the accounting records belonging specifically to that account. In the first example, sa -m summarizes the information in the accounting files, providing a single record for each user. The files interrogated are the accounting file, /usr/ adm/acct, and the accounting summary file, usr/adm/savacct (produced with sa -s). This output is piped to grep, which displays the record corresponding to our login name intruder. This display includes the total number of processes activated by our potential intruder, the total CPU time in minutes, the total number of input and output operations, and the average memory use in kilobytes per second for all these processes.

This is a useful start, but it gives no indication of what the intruder was actually doing while he was logged on. The next step is to use the command sa -u, which summarizes the accounting file by user identification (UID) for each user, followed by command name and the amount of CPU and memory used by this command. By interrogating the password file /etc/passwd, we have determined that the UID of the potential intruder is 103. Then, using grep, we have filtered the output of the accounting summary to contain just those records for user 103. This tells us that the potential intruder used tcsh, the T" shell, which is an enhanced C" shell.

Could our intruder have changed this program in some way? Before we worry about that question, we need to consider plugging the gap. We could of course modify the file /etc/passwd to prevent the intruder from accessing the account, but he may have access to other accounts, in which case an we have done is alert him. Figure 4 illustrates some other standard BSD UNIX commands that are at our disposal in our quest to thwart the intruder.

First, the command ac -d intruder provides a daily summary of connect time for the login name intruder, and it may help to establish a pattern of when the intruder accesses the system. Second, the command last -d intruder shows, in reverse chronological order, when the intruder logged in and, perhaps more important, from where he logged in. In this example, the intruder accessed the system via a network connection to a pseudodevice (/dev/ttyp*), always from the host named cuhhmd. If intrusion is always from a single host, you may wish to solicit the help of the administrator of that host to establish whether this is the source of the intrusion or whether that host was infiltrated from yet another host. Also, note that the potential intruder used ftp to copy a file to or from your computer.

The third example, cat usr/adm/ shutdownlog, summarizes when the system was last shut down and by whom. This may not be important for central servers where administrators are likely to notice shutdowns, but client shutdowns and reboots are easily missed. In a worst-case scenario, the intruder could have been rebooting the client system to use a modified kernel, which could potentially cause problems on other systems. Of course, to reboot the system, the intruder has already gained root access to the system, so you are in serious trouble already. At the very least, a change to all passwords is required, and to be sure, the system may need to be either restored from dump tapes known to predate any intrusion or rebuilt from the original distribution tapes.

In the fourth example, lastcomm sendmail intruder ttyp1 illustrates the resolution with which a user can be interrogated. When and for how long the user intruder accessed the command sendmail from the pseudodevice /dev/ ttyp1 is displayed. Accessing sendmail implies that the intruder used the mail program.

Finally, the command cat /usr/ adm/sulog displays the log reporting use of the su (become superuser) command. This would readily indicate whether the intruder has access to the root password and, because failed attempts are also reported, any attempt at guessing the password.

Instrusions should be dealt with by passing the information either to the

people responsible in your organization or directly to the Computer
Emergency Response Team, based in Pittsburgh. n

```
---------------------------------------------------------------------------
Type:       column
            technical
Company:    Digital Equipment Corp.
Product:    Ultrix
Topic:      UNIX-Like Operating Systems
            Data Security
            Software Design


Record#:    11 935 742.
                              *** End ***
```

```
*****  Computer Select, December 1992  : Articles *****

Journal:    DEC User  March 1992 p20(3).
-------------------------------------------------------------------------
Title:     Sense of security. (UNIX operating system)
Author:    Vincent, Jo


Abstract:  Some analysts argue that the VMS operating system is more
           secure than Unix.  Others would defend Unix, claiming that
           security features are available for the product, but it is up
           to the user to request and implement them.  The current
           version of Unix, System V version 4.1 ES (Enhanced Security)
           is currently undergoing certification to the B2 security
           level.  A recent report on Unix security by SRI International
           indicates that account security, network and file network
           security are the three areas most users are concerned about.
           The increase in distributed computing within corporations has
           led to more problems for systems administrators, who are
           concerned with unauthorized access to company files.  One of
           the main problems with Unix is that the security device that
           accompanies the system comes switched-off.  The user must
           therefore learn how to activate the feature before security
           can be implemented.

-------------------------------------------------------------------------
Product:   VMS
Topic:     Access Controls
           UNIX
           Data security
           Operating systems
           Comparison


Record#:   12 039 934.
                               *** End ***
```

***** Computer Select, December 1992 : Articles *****

Journal: UNIX Review March 1992 v10 n3 p38(10)

Title: Security on UNIX systems: how to build security into your open systems and measure its effectiveness. (includes related articles on the National Computer Security Center and its security level designations, typical security features, and an example of a covert channel) (Tutorial)

Author: Bunch, Steve

Abstract: Traditionally, UNIX systems have been used in environments where users have implicit trust in their co-workers and, as a result, UNIX security has been only fair. With the migration of UNIX into more vulnerable environments, steps have been taken to improve the security, or trustedness, of UNIX systems. Both the National Institute of Standards (NIST) and the National Computer Security Center (NCSC) are involved in the development and evaluation of security of computer systems. The NCSC publishes the Orange Book, a compilation of security requirements aimed at government computer systems but also applicable in large part to commercial open systems. The NIST is currently at work on a set of specifications that combines those in the Orange Book and specifications established in Europe. Vendors may or may not choose to formally evaluate systems according to NIST or NCSC guidelines. Often there is a trade-off between the level of trustedness and the number of available features on a system; it is up to the customer to decide whether features or security is to be given priority in a purchase decision.

Full Text:

The security of UNIX systems has traditionally been considered only fairly good-and only if they are well-administered. As UNIX systems are used in more commercial contexts, the need to rely on system security is growing. UNIX systems are being used to hold information that is as valuable as the contents of bank vaults, but how do we know if the systems are worthy of that level of trust? When is "pretty good" not good enough? What is better? How do you tell how good it is?

This article is an overview of the base security technology being incorporated into commercial UNIX systems. It is meant to help you understand the technology better, so you can cut through the marketing hype and ask vendors questions that will get you the answers you need for your application.

I use the words "secure/security" and "secure system" interchangably with "trusted/trustedness" and "trusted system." The preferred terminology in many environments, especially where the word "security" appears to promise too much, is "trusted." No distinction is intended here.

Computer security technology. Most computer users think computer security makes it harder to get their jobs done. That is often going to be true, generally because of potentially harmful security weaknesses in the way people are accustomed to working. In a bank, the vault combination is not given to every cashier, but in a computer installation, many employees usually have access to the computer. With the quality of protection provided by most computer systems today, giving most employees access is like giving them the keys to the vault (along with a key to the front door so they can come in on weekends). Users have to accept the fact that adopting security technology will require a change in the way they do things.

The UNIX system was designed to be used in environments where users don't think they have any real secrets from each other and trust their co-workers implicitly. Computer security is based on the premise that either or both of those assumptions could be false. Most uses of UNIX systems in commercial environments would be considered dangerous if the system did not provide good protection against malicious insiders, since such insiders are by far the biggest threat to any business.

The threats. One job of a computer system is to guard data from specific kinds of threats. In a cooperative environment, people often assume the only threat is carelessness. The security features tend to protect you from mistakes, such as deleting an important system file. However, in most environments, the severe threats come from misplaced trust: people who have legitimate access but abuse their access privileges. Outsiders who can casually or maliciously access your computers and its data also pose a threat. Often an outsider gains access via user carelessness: telling the "repairman" who calls on the telephone everything they type when they log in (including the password) so that an imaginary problem can be "diagnosed." The defense a system must provide varies with the specific threat. Some threats are easy to defend against; some are not. Some must by necessity be handled administratively rather than with system software.

Some typical threats include:

* Reading a password "hidden" on a cheat sheet in the top desk drawer

* Lending someone your login

* Reading data over someone's shoulder

* Changing or accessing data left unprotected on a system

* Changing data the user is authorized to change, but should not

* Using a terminal left logged in overnight or at lunch to access data

* Stealing floppy disks with back-up data on them

Overly permissive default protections on newly created files

* Incorrect administration of user information or system protections.

Defense vs. Commercial Usage. In the past, computer security was mainly of interest to the Department of Defense, which has highly sensitive information to protect. In recent years, privacy and other legislation has forced other government agencies to worry much more about security than they used to. Banks and other commercial computer users are also being forced to worry about computer security as they pass around large amounts of money, stocks, and options, with paper trails that are sluggish by comparison. The formerly esoteric realm of computer security is coming into the mainstream.

Historically, government and commercial organizations have implemented the absolute legal minimum computer security controls. The implementation of stricter controls costs more than the potential losses. This cost/benefit ratio is now moving in the direction of stricter security.

Secrets vs. Data Integrity. Two distinct aspects of accessing computer-stored information are reading it and writing (or modifying) it. Denying users access to some information-keeping secrets-is different from preventing them from creating new information. The difference between these types of access resembles the difference between the way a security agency feels about the names of its informants in a foreign government and the way a banker feels about the bank account balances stored in computer systems. Protecting information from disclosure has historically been extremely important to the defense community, which has secrets to protect but does not as often depend on the integrity of any individual copy of that information. Integrity has been more important to the commercial world, which manages money and other valuables using computers that contain the master version of the information but does not need to protect that information from disclosure as strongly (since little true harm typically would come from it). Of course, both types of users care about both kinds of protection, and in a computerized world with increasingly strong information-protection laws, both types of security are important.

Features and Assurances. The distinction between features and assurances is useful in understanding trusted systems. A feature is a visible security facility in the system that users must be aware of, such as protection mechanisms on files or passwords when they log in (see sidebar). Assurances are mechanisms and activities that make it possible to trust the system security features to operate correctly. These may or may not be visible to users.

To make a system as trustworthy as possible, vendors usually build and run exhaustive security test suites that test all the security features of the system. They might write formally proven security models; they might perform penetration testing in which experts attempt to break into the system; they might keep track of the person who originated each individual line of code in the system-just in case the programmer turns out to be careless or malicious. The goal of these techniques and specific (usually not user-oriented) security features is to uncover security flaws before the system reaches users and to discover and prevent security failures after it reaches the field. Such assurances increase users' confidence that the system and its security features are implemented correctly and will work as intended. In the Orange Book, the NCSC (discussed later) has explained the minimum level of assurance it requires from a system at various security levels.

This feature/assurance distinction is important to understand. Features can be provided to perform some security-related function, but in a system full of serious bugs, such features may provide no protection whatsoever. Similarly, a system with many assurances may be highly secure when used according to its designers' intent, but if the system does not provide the right features for its end use, people may find the system hard to use and circumvent the protection features. The engineering of a trusted computer system is not just the inclusion of some new system calls and commands. It requires carefully analyzing a customer's needs and designing features that satisfy those needs, combined with the appropriate assurances to meet the intended use of the system.

NIST and the NCSC. Two U.S. government agencies are heavily involved in the specification and evaluation of the security of computer systems. The National Institute for Science and Technology (NIST) is responsible for computer security in the private sector. The National Computer Security Center (NCSC), a part of the National Security Agency (NSA), has responsibility for evaluating the security potential of systems that might be sold to the U.S. government.

The NCSC has published a guide commonly known as the Orange Book (because of its orange cover)-its actual name is the "Trusted Computer System Evaluation Criteria" or TCSEC. In the Orange Book, levels of trustedness are defined, ranging from a low of D to a high of Al (see sidebar). D is essentially no security protection (for example, an IBM PC). The somewhat esoteric B3 and Al ratings require that the system be designed from the beginning to be evaluated at those levels. The Orange Book was originally written to satisfy the security needs of the U.S. government, not commercial users. For example, it has few specific requirements for integrity features. It also uses a security model based on hierarchical and nonhierarchical classification of the sensitivity of information (based on the DOD classification system: Top Secret, Secret, and so on), which does not exactly match what most companies use as a model.

However, the majority of the Orange Book requirements do apply to commercial use. The UNIX system itself provides many additional important features, and system vendors who want to sell Unix-based systems in the commercial marketplace have been striving to provide features over and above the minimum mandated for the target evaluation level. The result is that even though the Orange Book is being used beyond its original intent, it is proving to be a solid framework on which to build commercial systems. It also provides a framework for evaluating many areas of functionality and assurance that all trusted systems share. Perhaps its most valuable role has been to serve as a common yardstick against which different systems can be measured.

To increase the level of security offered by U.S. computer vendors, the U.S. government several years ago mandated that all computer systems bought by the federal government must be evaluated at least at the C2 level by 1992. This policy may not be applied absolutely in 1992, but many vendors are still striving to have C2 level systems in that timeframe.

Security standards. Most western governments, and several organizations within the U.S. government besides the NCSC and NIST, have produced their own security documents. These documents variously include or reference the Orange Book, offer similar information in a different form, or offer alternate security requirements based on different assumptions or end applications. Few, if any, genuine incompatibilities exist in these different requirements, but the union of all the sets of requirements is a very large set.

NIST is currently working on a unified standard that addresses the primary European requirements, the Orange Book requirements, and commercial requirements. A question that has not really been answered adequately is whether all the security features and requirements discussed in these standards meet real-world needs. Since systems implementing them are just beginning to appear, we are entering a period of discovery in which we will be getting the answer. Early results indicate that some kinks may need to be worked out.

Computer vendors implementing security functionality in their systems are generally using common interfaces in an attempt to satisfy the existing specifications they feel are necessary for their markets as well as their nongovernment customers who need better security. Unfortunately, different vendors have, in some cases, targeted different customers for security-enhanced systems, and they have often invented different interfaces to satisfy these and their commercial needs.

Application writers and users would clearly like to see common interfaces among vendors, and similar if not identical concepts being presented by the system and its documentation. The industry needs interface standards to reduce this interface divergence and to create an environment in which independent software vendors (ISVs) can produce applications that use security features and are portable among different vendors' systems with minimal effort. No security-interface standards have been established, although there have been several working papers and many proposals. This situation will be changing over the next few years as standards are completed and implemented.

Several efforts to create standards for UNIX security features are underway. The first was the /usr/group Subcommittee on Security, formed in 1986. This group eventually passed the baton to the POSIX P1003.6 group, which is currently proposing a security extension to POSIX. Unlike the P1003.1 group, which was able to adopt pre-existing technology, P1003.6 was forced to invent new functionality. It has taken a number of years to reach this stage, but the P1003.6 draft standard is now out for balloting. However, since it is operating so close to the leading edge of technology, resolution and formal adoption is likely to take a while. Because of the rapid commercial deployment of security features, the P1003.6 committee may be able to adopt more de facto practice in the future, more like P1003.1 operated, rather than inventing so much.

Although X/Open produced a position paper on auditing that was adopted as a starting point by P1003.6, it has not recently been active in security. It is now returning to this area, however.

As previously mentioned, NIST is addressing security technology in a set of unified security requirements, but this is not an interface definition and is not available yet.

Porting bases for security technology. The most important influences in the security area are likely to be de facto standards in the form of porting bases. Several technology vendors have been selling security functionality add-on kits in the past, and most of the large vendors have had security efforts of their own in progress for many years. However, most UNIX system vendors today have aligned with OSF or USL as the base for their ongoing UNIX system technology and source base. USL's System V Release 4.1 Extended Security (ES) and OSF's OSF/1 system will be major porting bases for building UNIX systems for the next few years. Both of these systems include security features. The 4.1ES system is being formally evaluated by the NCSC against its B2 requirements on an AT&T reference platform. Vendors using this porting base will need to perform an evaluation on their own platform.

The OSF/I system is targeted at the B1 security level, although OSF itself is not performing an NCSC evaluation on a reference implementation of OSF/L. OSF's member companies, often in conjunction with the supplier of the OSF/I security technology to OSF, are performing such evaluations on their own products. The same technology has also been licensed by several vendors to incorporate into non-OSF/1, nonSvr4.1ES-based systems. For the purposes of this discussion, such systems will generally have the same properties as we describe here for OSF/1.

USL's source base has the more stringent assurance requirements of the B2 goal, and the actual porting base is being evaluated by the NCSC at that level. These factors give USL's source base an advantage from the security standpoint. However, by porting either USL's or OSF's source base, a system vendor obtains a reasonable set of security features and an idea of the level to which the system could be evaluated. Both of these systems can be configured to provide C2- or B-level feature sets, allowing vendors to tailor their own products (based on these source bases) to different customer requirements. Note that the use of common porting bases by a large number of system vendors has a subtle catch. While the vendors will all benefit from the efforts of the source-base suppliers to remove security flaws from the system, the vendors will all suffer from any flaws that escape into the source base. A set of well-known security bugs in BSD-based UNIX systems was exploited by the infamous Internet Worm that a few years ago infected many systems from several vendors. Source-base providers have a serious responsibility to promptly inform their customers (the system vendors) of security flaws and to provide fixes. They must try to keep those flaws closely held until vendors have had an adequate chance to propagate fixes to their field installations, but this is not always possible.

Later in the 1990s, the effects of de jure standards will be felt by all vendors and source-base providers. Both OSF and USL have agreed to become compliant with the POSIX 1003.6 standard when it is finalized, and both are doing everything they can to anticipate it. While the details of implementation and interface in the P1003.6 standard differ from these systems, the concepts are compatible. Source-level interface compatibility with P1003.6 will provide a level of functionality sufficient to implement many security-aware applications.

Application porting and ISV issues. Will all these security features break existing programs? Probably not as often as you might fear, but there may be big differences in what users can conveniently do. Nonprivileged programs that do not make unwarranted assumptions will seldom break because of security mechanisms-unless the user tries to do something that is illegal on a system with higher security.

On the other hand, some security mechanisms are meant to prevent information propagation that people take for granted. While the programs may work, they may become less useful. For example, the mail feature of an office-automation program may send mail just fine, but recipients might have to log into the system at the same security level as the sender or they would never find out that mail has arrived. If they cannot log in at that level, they may never get the mail. Programs may encounter new error situations, but generally the error codes for failures caused by these new features are compatible with old ones.

Some programs must perform operations that have become privileged on more secure systems, however, which can lead to fairly glaring problems. For example, an office-automation package that is accustomed to being able to write directly into a printer spool directory may find itself unable to do so and may no longer print. A program that temporarily assumes the identity of the super-user to abort an operation will abruptly lose that capability. Such programs must be recoded to work in ways consistent with a higher degree of protection. They are failing for a reason, generally because their implementation has a security flaw. ISVs and users who write privileged software will have to make changes to work within the new security framework of trusted systems. Most applications that previously worked without special privileges will continue to work on a trusted system, possibly with some new security-imposed restrictions, as previously noted.

Some security features, particularly Mandatory Access Controls (MAC) (see sidebar), will be a force-fit into some environments and applications. For example, a database package cannot be adapted simply to deal correctly with sensitivity levels, which may make their use unwieldy for so cations. Or, the administrative roles implemented in the system may cut across department boundaries. Many companies do not categorize the sensitivity of their information at all, and many that do ("company proprietary," "internal use only") have a fairly large error margin. ("Sure, I know it says 'company private,' but he's a good customer. Go ahead and give him a copy."). The computer's enforcement of these security policies will be absolute, but people aren't used to such strictness. Either the pre-existing practice must change to match the features, or the features must adapt to meet the practice. Feedback to vendors when this sort of situation occurs is critical to develop efficient security features and company security policies.

ISVS and users who want to take advantage of new vendor-provided security features must learn new concepts. These concepts will generally be portable from vendor to vendor, but the interfaces offered by vendors cluster into a small number of sets depending on the source of the vendors' security technology. All interfaces will be migrating to future standards like P1003.6 as they develop.

To protect their investment in security additions, ISVS and programmers should write their programs using abstracted primitives and interfaces of their own definition, then write a software layer to translate that to the vendor-provided set of interfaces. Later, when porting the software to a different system, only the small interface layer need be rewritten. The bulk of the application can remain unchanged. Programmers should study the features of several vendors' systems before deciding what common subset of functionality they should use in their applications. Programmers should target their applications to work with a specific feature set that corresponds roughly to the Orange Book levels (probably the lowest usable one), since most vendors are making it possible for end users to configure their system to different levels. Assuming all vendors will provide functionality similar to that provided in the P1003.6 draft, for example, is reasonably safe.

NCSC levels and evaluation. The NCSC is currently performing evaluations of all higher-security systems (B2 and above) and will likely continue to do so. However, although the NCSC is also performing all evaluations at lower levels, it is likely that this responsibility eventually will pass to NIST. NIST's proposed approach is that the system evaluation for certification to be done via the National Voluntary Laboratory Accreditation Program, in which the actual evaluation is performed by an accredited laboratory (as is now done for POSIX certification). Evaluations at the C2 and BI levels are significantly less stringent and thorough than at the B2 level. C2 and B1 evaluations also have simpler requirements for maintaining the rating of the system as it evolves (essentially, the vendors evaluate the security relevance of their changes). As a result, most systems on the market with security enhancements will be at these levels. The levels above B2 are currently the realm of security professionals, and we'll ignore them here.

A particular vendor will decide-generally dictated by its customer base and market-whether to formally evaluate a system (either with the NCSC or NIST) as compliant with formal security requirements. This choice is based on the vendor's customers' knowledge about security, what firm security requirements are written into typical bids the vendor responds to, and whether the vendor perceives an adequate return on investment in security measures.

What level is enough? It's hard sometimes to decide what level of system is needed for a given commercial environment. The NCSC has a published guideline to help decide for the government environment, but it is not applicable in commercial applications. There are two issues: feature content and confidence level in the quality of the system implementation (related to the assurances used in its production). A system claiming "B3 features" or "B1 features" is saying nothing whatsoever about the level at which the system can be evaluated, just that it has all the features required at that level. Similarly, a system might have all the feature content mandated for an A1 system, but be completely insecure because of bugs. Both facets matter.

The weakest link in security of most commercial UNIX systems will be the users and system administration, so features that help users and administrators avoid mistakes are potentially the most valuable. Also valuable are features that help keep honest people honest by improving the chances that if they aren't, they'll be caught (and letting them know the feature is in use). By these criteria, the most critical security features (auditing, good password checking, system integrity, documentation) are present at the C2 level; this is, therefore, the minimum working feature set. The "trusted path" to the system and the tighter control over system integrity, privilege, and system administration required at the B2 level are extremely useful in a commercial setting.

Access-Control Lists, an access-control mechanism that probably best matches the way companies control information today, first becomes a mandatory feature at the B3 level. The most visible feature added at the B levels, Mandatory Access Control modeled on the controls used for DOD classified information, is not as directly applicable as ACLs to implementing existing security practices in most commercial environments. Further, it is much less convenient in daily use. However, because of its power, it will be desirable in some situations despite its disadvantages. ACLs tend to be a better model for commercial usage because commercial users generally want to protect information by establishing the list of people who can see or change it. However, in the DOD world where MAC originated, information has inherent properties that determine what its sensitivity is, and individuals have related properties that determine what they can or cannot see. Most vendors supply features beyond the minimum set required for a target evaluation level, since they want their system to be attractive to a wide variety of users. For example, many commercial systems will include all the above features, even in a system targeted for the B1 level of trust.

The issue that really matters is the true difficulty of violating the system's security. The most important sources of confidence in a system's trustability are good design, the smallest possible number of features and interfaces, good specifications, good documentation, and good testing. A well-thought-out system is less likely to have subtle flaws. A system that is small can be more completely analyzed and will have fewer feature cross-product interactions that could have security repurcussions. A well-documented system is much easier to understand and less likely to be compromised through accidental misuse. A system whose security features have been exhaustively tested, perhaps by experts attempting to penetrate them, is less likely to contain errors that can be exploited by a nonexpert.

A system whose security has been examined by an outside body like the NCSC will tend to be more completely examined. The NCSC evaluation criteria reflect how thoroughly all such aspects of the system must be inspected to achieve a given level, with the higher levels corresponding to stronger requirements. Therefore, a successful evaluation on the NCSC scale is a good way for a nonexpert to judge. Higher levels correspond to more careful examination of each element that makes the system more likely to withstand attack. Given that few vendors have evaluated systems today. this is admittedly not a good enough answer, but it will improve with time.

Some system vendors will have done a much better job than necessary for a given rating and in fact have systems more worthy of trust than formally evaluated systems of higher ratings. However, vendors with the security expertise to achieve this quality are rare. Today, buyers may need to learn enough about the vendor and the process the vendor used to produce its system so they can judge the likelihood of the system to correctly implement its security features. Since many potential purchasers of trusted systems need to learn about security anyway, this evaluation could be made into an excellent education opportunity.

Later 1990s software technology. The technology of building secure systems has not changed

In much in the last ten years. The leading edge of the technology is advancing, the number of "worked examples" has increased, and the folklore available to system designers has become more extensive, but in the trenches the work that's done to secure a system has not undergone any revolutions. The number of practitioners outside the DOD has risen, however, so more people are beginning to be aware of trusted system technology. The most important progress in security technology in recent years is its wider availability on commercial systems.

Practical experience and theory show that the most important things that can be done to improve the security of UNIX systems are to reduce the size of the part of the system that must be highly privileged, and to implement the system on top of a well-structured starting point. These concepts are related, and they coincide with two of the major goals that have generated so much interest in micro-kernels in recent years.

Considerable effort is being put into micro-kernels like the Mach operating system, created at CarnegieMellon University with DOD funding, and the Chorus system, built with private funding by Chorus Systems in France. Micro-kernels are a promising way to reduce the size of the privileged part of the operating system and hence simplify the job of providing a higher level of trust. But a lot of research has yet to be performed before highly trustable systems based on this approach to security technology are available commercially. The basic technology being employed, a security kernel, is around 15 years old.

It will not be another 15 years before products come out. Commercial time-sharing systems based on micro-kernels will become available in the 1992-1993 timeframe, but general-purpose time-sharing UNIX systems that exploit the modularity and size of an underlying micro-kernel to achieve higher than B1 ratings will be several years behind that. Research prototypes will appear sooner, but they will not be ready for prime time yet.

Technology advances and security. As previously mentioned, software technology for securing operating systems has not changed markedly in the last few years. However, in those same years, hardware has been leaping ahead in speed and capacity. This has had some effects on the overall security of systems. As hardware gets faster, systems become more susceptible to certain kinds of security problems.

For example, covert channels become more insidious and hard to fix. A covert channel is an information leak that uses side effects to transfer information rather than normal data transfer channels. The speed of such channels, when based on a physical device like a disk or the CPU, scales at least linearly with the speed of the device being used. Similar and difficult-to-solve problems are introduced by shared-memory multiprocessors. This area is ripe for research and discovery.

If all the dead bodies of failed secure UNIX projects are any indication, securing the UNIX system is a hard problem. Given that all UNIX systems must have some degree of security in the future, this is actually a worrisome point. The high-jump bar is being raised, and an increasing share of the time spent in any UNIX system is spent doing activities required for security. This isn't necessarily bad, however, as many of those activities raise the overall quality of the product.

Some of the extra effort that goes into a secure UNIX system will go into supporting and building the additional security features. However, the majority of the needed features will be made available from the porting bases. An area of added value that vendors can address will be features required for specific commercial uses, such features needed for niche markets.

Assurances. It can take a lot of effort to give customers enough confidence in the protection provided by a system that they are willing to bet the company on it. Vendors can undertake many engineering activities during their development activity to help provide that assurance:

* Careful configuration control. Careful configuration control is straightforward to accomplish and a really good idea, anyway.

* Penetration testing. Penetration testing, that is, attempts by experts to defeat the system security features, is not especially hard and can be reasonably effective. It can be time-consuming, and true experts at cracking systems are rare. Since it is not an exhaustive procedure, not much real confidence should be expected from this unless it has gone on for a long period of time with no successes.

* Security testing. Security testing is detailed testing of all

functionality with respect to a tightly written interface specification, including trying all error and illegal cases. It can be exhaustive and exhausting, but it is very important.

* Informal security policies. A clear statement of the security policy the system is to uphold is a crucial starting point for the implementation or installation of a secure system. When a secure system is being built, an informal statement of the intended operation of the security mechanisms of the system is a powerful way to communicate to developers, documentors, and users what the goal of the system is and how it is to accomplish that goal. Such policies should be among the first things written when the development or adoption of a secure system is contemplated.

Formal security policies. Formal security policies help clarify exactly what the system is supposed to do. Like a good requirements specification, this kind of documentation is useful for everything from writing user documentation to making penetration tests.

Proofs of correctness. Proofs of correctness of specifications are feasible today but are expensive and not often used. Proofs of correctness of code are beginning to work for some special cases. This is perhaps the ultimate goal in any system: to be able to mathematically prove that the system does what it's supposed to. This is an active research area.

The previously mentioned examples were listed roughly in increasing order of difficulty. In this short list, every item represents an activity that raises the overall quality of the system being built, independent of security.

In the process of doing the above, vendors are forced to control their product better and will necessarily find and eliminate bugs (not all of which will be security bugs). One pragmatic problem with this is that customers demand visible features, and vendors tend to use up their resources trying to supply them. Efforts that do not deliver new features are often not financially justifiable. Vendors must balance the need for features with the desire to provide a product that is truly trustable.

Many of the NCSC's assurance requirements are really directed at making sure vendors use a good development methodology, do adequate testing, control their product, and use the best available technology for building their programs. If the NCSC guidelines did not force vendors to do these things, vendors often would not do them.

It is a vendor's decision whether to spend development resources on features or on achieving higher levels of trust. It is a customer's decision whether to buy a system that offers features at the cost of trust, or vice versa.

The 1990s will see an increase in the degree of security that systems must offer, and a large number of UNIX systems with security features will become commercially available. This is being driven by the requirements of customers. As UNIX systems move into wider commercial and government usage, customer requirements for features and higher levels of trust will become more stringent.

The feature set needed by commercial users in performing their functions will not be a perfect match for some of the security features vendors are now implementing. The biggest challenge for vendors and users for the next few years will be to develop better models of commercial security and to invent and implement features that directly implement those models.

Until now, vendors and agencies like the NCSC have been in control of the definition and implementation processes because they have had the job and the expertise to do so. After the current generation of systems reaches the field and commercial users have a chance to try it out, they may develop a better idea of what they really need, and they will have developed the expertise to communicate those needs back to vendors.

Steve Bunch is the chief scientist at the Motorola Computer Group's Urbana Design Center in Urbana, Ill. Motorola's Urbana Design Center is responsible for UNIX SVR4 for the Computer Group. Bunch has been involved in security since 1976, was involved in the first Ncsc-certified UNIX system (Gould's UTX/325), and was one of the instigators of the joint Usl-Motorola-umdahl project that produced UNIX SVR4.1 ES. He may be contacted at srb@aurbana.mcd.mot.com.

A Covert Channel Example

Here's a simplified example of two cooperating processes, which shouldn't be able to communicate, exploiting a covert channel. (The hard Problems you'll encounter in making t s work well are glossed over or ignored.)

Assume a system is running two processes, A and B. A is forbidden by the MAC rules to send information to B using any of the system's normal data transfer methods, although they can both read system files at a lower security level than either. A and B will collude to send data using the "time of last read" property of a low-security file they can both read.

A and B initialize by opening the shared file and start their activity at a pre-arranged time.

A sends a 1 to B by reading from the file at some time within a given one-second interval. This updates the time of last read to that second. A sends a 0 to B by not reading the file in a particular second.

B monitors the time of last read of the file by checking its status once each second. If the file has been read during that second, then the time stamp will so indicate, and a 1 has been detected. If not, a 0 has been detected.

This simple example transfers data at the rate of one bit per second. It seems slow, but at that rate, a significant memo can be leaked from A to B in a few hours. Depending on the memo, that might be a serious security problem. This channel actually works and is capable of much faster operation on many UNIX systems. This may be the best-known covert channel in the UNIX system, and specific provision was made in the POSIX 1003.1 standard to permit it to be dramatically slowed by vendors without violating the standard. Better-secured UNIX systems have slowed the channel substantially, and the best-secured ones will announce or log, as it is happening, that this channel is being exploited.

NCSC And its Levels

The National Computer Security Center (NCSC) was established in 1982.

its charter includes the responsibility to establish and maintain ...technical standards and criteria for the security evaluation of trusted computer systems that can be incorporated readily into the Department of Defense component life-cycle process ..... As part of this charter, the NCSC produced a document known as the "Department of Defense Trusted Computer System Evaluation Criteria" (TCSEC). This document, also known as the Orange Book because of the color of its cover, defines a series of four major divisions of protection (D, C, B, A) each with certain security-relevant characteristics. These four divisions are further subdivided into more precise levels of trustworthiness. The chart below summarizes the defined levels and their key properties.

The NCSC also publishes a series of technical guides that give guidance on specific topics such as vendor information, configuration management, trusted facilities management, auditing, and so on. This set of guides is collectively known as the "Rainbow Series." The NCSC can be contacted at: National Computer Security Center, ATTN: C11, 9800 Savage Rd., Ft. George G. Meade, MD 20755-6000,

The table below lists the Ncsc-defined levels in decreasing order of trustworthiness and securityrelated feature content. The table shows the name of the level and the key features added when going up to that level from the lower ones. Note that each level contains all features from the lower levels. The Orange Book contains many details describing the level of testing, design documentation, formality of security policy, covert channel analysis and bandwidth, and so on, required at each level. This chart concentrates on those features that are highly visible to the end user. The detail is visible indirectly, in the level of trust the user can place in the system.

The NCSC maintains an Evaluated Products List, which describes the products that have been evaluated by the Center to satisfy the requirements at a specific level and offers additional summaries of in-progress evaluations. The NCSC can be contacted for further information. To date, Unix-based systems have been evaluated at levels from C2 to 62. Commercial security-enhanced products, like the IBM mainframe application Top Secret, have typically been evaluated at the C2 level. Some specialized systems have been evaluated as high as the Al level. The C1 level has not proven interesting in practice for most applications and is not really used for general-purpose systems. Since potential buyers who ask for specific levels typically specify at least C2, this is not surprising.

Typical Security Features

The following list describes the major security features described in the Orange Book and present in trusted systems at various levels of trust. Most features below are present to a greater or lesser degree in all current security-enhanced UNIX systems.

Identification and Authentication (I&A). This part of the system checks your password when you log in, perhaps ages that password so you must change it as often as your administrator thinks you should, and so on. Some I&A systems can be easily extended by end users to include additional devices such as magnetic card readers.

Auditing. The auditing function makes an indelible record of the security-relevant events that occur on a system. This includes users logging in and out, accessing data files, printing documents, changing a password, and so on. Auditing is probably the single most important feature for commercial use of the UNIX system. it's a feature that keeps honest people honest (because it records their actions and makes it likely that they'll be caught) and helps catch dishonest ones. The basic functions of auditing are fairly common among implementations, but considerable difference exists in the details. Audit trail analysis tools, overhead cost to system performance, and audit file size are key areas where differences arise.

Discretionary Access Controls (DAC). DAC is access controls that an owner can place on a file, such as, 1 owner can read and write, others can only read." In UNIX, the well-known mode bits or file-protect bits on a file serve this function. A much more powerful and useful feature for commercial use, Access Control Lists or ACLS, has been borrowed from the Multics system. (In fact, the UNIX file-access mode concept is a simplified version of Multics' ACLs, so this amounts to a return to UNIX'S roots.) ACLs permit specifying the access permitted to each member of an almost arbitrarily long list of individual users, not just to owner, group, and others. For example, Joe, the owner of the payroll file, could state that he has read-write permissions on the file; all other members of the payroll department and two specific people in the personnel department have read-only permission; everyone else has no access. Such an access control list can be summarized as:

Access	ACL Entry
rw	joe.payroll
r	pete.personnel
r	fran.personnel
r	*.payroll
-	*.*

As implemented in most UNIX systems, an ACL can be attached to any file or directory (and most other objects with mode bits).

Trusted Path. The Trusted Path feature provides a reserved keyboard sequence (or in some cases, turning off or unplugging the terminal) that is always intercepted by the system, and always puts users into a conversation with the system. With this feature, users can walk up to a terminal, enter the reserved sequence, and be absolutely sure they won't be spoofed into typing a password by a program left running on the terminal.

Administrative Least Privilege. This general area of functionality contains several concepts and admits to many implementations. The basic problem is that the UNIX system's super-user is simply too powerful for safety. So, some systems implement less-privileged administrative roles. A user operating in such a role has just enough privilege to perform some administrative task and no more. For example, a system might define a printer administrator, with privileges to delete any job from any print queue, reconfigure printers, move jobs from one print queue to another, and so on. Unlike the super-user, someone acting in the role of printer administrator would not be able to read another person's mail or modify system files unrelated to printing.

Mandatory Access Controls MAG). Mandatory access control mechanisms are controls that assign a sensitivity level to all information in a system and assign a clearance level to all users. The term mandatory refers to the fact that the protection on a given piece of information is not solely under the control of the owner of the information but is enforced absolutely by the system as additional restrictions above and beyond the user's DAC rules. The levels can be hierarchical (in a strict lower-to-higher order in which someone cleared for a level is cleared for anything lower) , non-hierarchical, or a combination. MAC is a feature mandated by the Orange Book at the B and A levels. The actual mechanisms are modeled after the DOD classified information model, but they have applicability in non-classified environments. This mechanism has two goals: users can only see information at levels for which they are cleared, and users cannot give information to someone who is not cleared to see it.

The MAC mechanism operates at all times, on all accesses to all data objects in the system, and may deny access that other mechanisms (such as DAC) would otherwise permit. This can reduce mistakes. For example, a commercial system could be set up to operate with four strictly hierarchical classification levels, called, say, "Executive Only".."Management," "Employees," and "Public." The mandatory access control mechanism would make it necessary for an executive who had created a memo with the classification "Executive Only" to take a specific action to downgrade the file before someone with a lower clearance could access it. This might be used, for example, to help protect intermediate drafts of a memo from being read by accident.

Type:	tutorial
Topic:	UNIX
	Data Security
	Networks
	Standards
	Open Systems
	Tutorial
Record#:	11 936 668.

*** End ***

Journal: Government Computer News Jan 20 1992 v11 n2 p49(1)

Title: National Guard sites use off-the-shelf Unix security. (security software)

Author: Schwartz, Karen D.

Abstract: The Army National Guard reports they are using off-the-shelf Unix security software applications because officials do not believe the Unix operating system is completely secure. Systems Center Inc's Ucontrol application is used in 55 sites. The package consists of four modular products which help the National Guard to secure its personnel, financial and logistics information in a database at each of the 55 sites. Ucontrol is an attractive option because it provides excellent tape library and backup functions, as well as security features. It helps to pinpoint critical data and maintains an on-line audit trail, as well as automatically verifying backed-up data. The National Guard did not experience security problems before using Ucontrol, but the software was purchased to guard against potential problems.

Full Text:

The Army National Guard has standardized off-the-shelf Unix security software because officials said they do not believe the operating system and hardware provide complete security.

Fifty-five sites, including each of the 50 states, Washington, Guam, the Virgin Islands, Puerto Rico and headquarters in Alexandria, Va., use Ucontrol from Systems Center Inc. of Reston, Va. The packagae is made up of four modular Unix security software products.

The half-million-member military reserve organization uses Uncontrol to help secure its financial, personnel and logistics data stored in a database at each site. Oracle RDBMS from Oracle Corp. is the database program.

Each site has from one to our Unisys 5000 minicomputers running Unix System V. Each location has one Uncontrol site license. The minis total 118 at the 55 locations, connected through the Defense Data Network, said Ed Byrne, technical director for the Chief of the Army National Guard.

Unix is not as efficient as Ucontrol in some areas of security, Byrne said. The Unisys "doesn't offer us much in the way of security or efficient tape backups and library functions," he said.

Guard officials chose Ucontrol primarily for its security features and backup and tape library functions, Byrne said. Ubackup is Ucontrol's module for tape and library backups. Ubackup records the status and backup location of all files and directories while identifying critical information. It maintains an on-line audit trail and automatically verifies backed-up data.

Guard units use Ubackup's tape library function for backing up daily and weekly production at each location. Headquarters does a collective backup each month, Byrne said.

"The basic Unix operating system doesn't give you a library function," Byrne said. "Every time you back up, you have to dump a tape to see what is on it. This provides a listing of what is on the tape, which can save time if you are looking for something specific."

Usecure is Ucontrol's module for password administration, with audit trails, terminal restrictions, superuser limitations, automated password administration and per user, per system or user-generated password capabilities.

Usecure gives the Guard "an element of security that wasn't there before," Byrne said.

Uqueue is the print spooler and queue manager. It lets the user reshuffle jobs within a queue, transfer jobs between queues, assign devices to queues and assign more than one device to a queue. It also supports batch job processing and multiple queries while maintaining an audit trail of print activities.

While some units choose to use Uqueue, others do not, Byrne said.

Upass is Ucontrol's optional module for added security. It layers over existing log-in procedures.

It stores passwords as one-way encrypted data and manages log-in attempts on a port-by-port basis.

The Army National Guard chose the optional Upass module for an extra level of security, Byrne said.

"Access through the network is easy to achieve," he said. "Along with the database password sign-on read-write authority, it also gives us a password validation that a person has to go through to get access to information. That is important to us for protection from viruses."

At the time the Guard began looking for additional Unix security, it ran a notice in the Commerce Business Daily, but Ucontrol was the only off-the-shelf package offered, Byrne said. "We had lots of offers from people that wanted to customize for us," he said, "but we were committed to a commercial solution."

"Before Ucontrol we really didn't have anything for security," Byrne said. "We didn't experience any difficulty but there was a potential for problems. It isn't so much that we run sensitive information, but it is just a common-sense approach to data processing."

Company:	Systems Center Inc.
Product:	Ucontrol
Topic:	United States. National Guard
	Security Software
	UNIX
	Data Security

Record#:   11 897 493.

*** End ***

***** Computer Select, December 1992 : Articles *****

Journal: HP Professional Jan 1992 v6 n1 p40(4)

Title: UNIX security comes of age; market forces are reversing UNIX's once-dubious reputation for security. (Cover Story)

Author: Caccamo, Wayne

Abstract: The manner in which the Unix environment developed has made it particularly vulnerable in the past to security breaches. Unix was devised as a development environment and earned its initial popularity in the academic and government markets where users were accustomed to developing their own security measures. Because Unix normally lacks the security tools that are common with proprietary systems, a system administrator must either develop the tools in-house or obtain them from a third-party vendor. Vendors are beginning to address such areas as password administration, access control and auditing for user accountability. Such developments are market-driven, both by demand from the government sector and through the efforts of both the Open Software Foundation and UNIX International. User-friendly interfaces for the administration of system and network security are beginning to appear, and HP's OpenView is emerging as an industry standard for the management of mixed Unix and non-Unix systems.

Full Text:

The future of UNIX system security is bright, but it wasn't always that way. UNIX was born a development environment, it grew up, went to school and became ery popular within the academic community. There UNIX acquired some radical values, didn't always practice "safe sex" hence the viruses - and obtained a dubious reputation for security. Nevertheless, UNIX graduated and entered the business world.

At first, the young UNIX didn't have the management and administration tools necessary to make it with Fortune 500 companies. Now UNIX has gone back for its M.B.A. concentrating on commercial security - an important credential in large corporations.

From rags to riches, this is the story of how UNIX acquired a shabby reputation for security and why that reputation is changing. Indeed, market forces are shaping the future for a more secure UNIX, without sacrificing any of its well-known talents for portability, interoperability and emerging leadership in performance and usability.

UNIX'S rather weak reputation for security is rooted in its original orientation as an development environment rather than a business system. Engineers typically ensure that they have access to any file or resource and, therefore, disable any security features that might interfere with their ability to get work done quickly.

Subsequently, these "wide-open" systems often would be delivered to customers unmodified. Today, reputable UNIX vendors deliver systems in more restrictive configurations and include instructions on how to install systems securely.

UNIX Goes To School

AS NEWS OF THIS SIMPLE and elegant development environment spread throughout the academic community, beginning of openness and information sharing was fostered. Universities began to acquire the UNIX source code and books were published providing source code to the UNIX kernel - the very blueprint of the security mechanism.

With the keys to UNIX systems widely distributed, systems were highly vulnerable to security breaches.

The UNIX kernel has changed so much since these documents were published in the 1970s, they no longer represent a significant threat. And although information on UNIX source code is still publicly available, the source code is now legally protected.

Nevertheless, the policy of openness and information sharing among the technical and academic communities persisted, contributing to the explosive growth and popularity of UNIX systems and, consequently, networks, hackers and even crackers. As a result, both the virtues of this environment and its flaws, especially with respect to security, have come to light.

Most experts agree that UNIX isn't necessarily any less secure than those proprietary operating systems it might be cohabitating with or replacing. There are just as many problems with other operating systems, but vendors and users of those proprietary systems usually prefer not to discuss such flaws. For instance, it is cheaper for a bank to absorb a $1 million loss resulting from an MVS mainframe break-in than to bear the cost of negative publicity.

More recently, well-publicized virus or worm attacks on UNIX systems have exploited some combination of well-known bugs, poorly written superuser programs, careless system administration, or the introduction of free software from public bulletin boards or user special interest groups.

For example, the estimated $89 million loss resulting from the Internet break-in of 1988 has done much to refuel concerns with UNIX security. Unfortunately, most people assume that all such incidents are a direct result of security vulnerabilities intrinsic to UNIX. But responsibility for most intrusions should not be attributed to the nature of UNIX, but to the nature of an open access network that happens to involve UNIX systems.

In general, putting a computer on a network automatically makes it less secure, regardless of whether the nodes on the network are running UNIX or some proprietary operating system. Ultimately, the only way to protect the flow of data over the network from electronic eavesdroppers or network "spoofers" is to physically secure and isolate it from the rest of the world. Nevertheless, somehow UNIX has been pronounced guilty by association with such electronic misadventures.

UNIX Enters The Business World

DESPITE SOME BAD PUBLICITY, the sphere of UNIX'S acceptance grew out of the academic and government sectors and into the commercial marketplace. Initially, however, UNIX lacked the required pre-packaged security facilities - the academic and government users had grown accustomed to developing these features in-house.

It comes as no surprise then that more than 90 percent of all UNIX security problems are caused by mistakes made by users and system administrators, not by viruses and crackers.

Perhaps more than any other operating system, UNIX security can be either made or broken by the system administrator. A seasoned UNIX administrator can employ a host of tools which, in the appropriate combinations, can accomplish almost any job. However, unlike proprietary systems, these tools traditionally are not pre-arranged for the administrator.

Rather, they must either be developed in-house or be purchased from a third party. And, because UNIX'S orientation in the past was that of a development rather than production environment, few such tools have been available.

UNIX Goes Back For Its MBA

UNIX'S SUCCESS IN THE marketplace has attracted a host of developers of sophisticated system administration packages providing security audits, password management and monitoring facilities.

Vendors of UNIX systems have enhanced their offerings with user-friendly interfaces for administering system and network security. In addition, HP provides an emerging industry standard interface for managing networks of UNIX and NON-UNIX systems called HP Openview.

In an attempt to meet the needs of ever-expanding target markets, security enhancements and functionality are being built into the UNIX operating environment. Most vendors are bolstering UNIX security in the areas of password administration, access control, and auditing for increased user accountability. Thus, in many areas including security, UNIX functionality has evolved to the point that it is on a par with or better than proprietary environments.

Clearly, the need for enhanced UNIX security is being addressed. Ultimately, however, no level of operating system security is a substitute for establishing a security policy document that considers the particular needs of an installation. The primary responsibility for complying with and enforcing policy falls on the user and administrator community and not with the operating system - UNIX or otherwise.

UNIX'S Career Outlook

THERE ARE TWO market forces driving continuous improvement to UNIX system security. First, affinity for UNIX in the defense-related market is pushing vendors to provide the highest levels of security to meet that market's rigorous requisites.

Second, both the Open Software Foundation (OSF) and UNIX International (UI) are trumpeting enhanced security as a predominant design and strategic theme.

The defense-related marketplace, including the U.S. Department of Defense (DOD) and its suppliers, represent the leading-edge of security needs, at least with respect to data confidentiality. The DOD, already the largest UNIX user in the world, has taken a strategic turn toward

UNIX in the past few years.

Vendors are scrambling to satisfy the more stringent DOD security requirements defined by the National Computer Security Center's (NCSC) Trusted Computer System Evaluation or "Orange Book" criteria. The NCSC evaluates operating systems and assigns various ratings or levels of security including: Cl, C2, Bl, B2, B3 and Al.

Currently, DOD purchasers are requesting systems that provide a BI level of security or higher. HP now offers Bl level security for HP-UX. Called HP-UX BLS, this operating system addresses the more complex multilevel security needs typical in the defense-related community. The security enhancements to HP-UX found in HP-LTX BLS outpace the requirements of commercial customers. Some of the security features provided by the B-level product most likely will be required by commercial customers in the future.

The second important driving force is the competitive environment. Both OSF and Ul are focusing on improved security as a central design theme. Both plan to offer B-level security. Moreover, OSF is planning to address even higher levels of security (i.e., B3, Al) by implementing a more modular Mach-based kernel design. Such a redesign and implementation of the kernel is considered necessary to address the system architecture criterion specified by the highest levels of security.

Another important OSF security initiative addresses the special threats in the area of network authentication and access control associated with the client-server paradigm of computing. OSF'S Distributed Computing Environment (DCE), which will be implemented on both HP-UX and OSF versions of UNIX from HP, will provide network authentication services, an area not covered in the Orange Book.

Because OSF is building in core functionality, such as security from the ground up rather than retrofitting the operating system, users will not have to sacrifice system usability, performance or compatibility in order to take advantage of new security features.

UNIX has matured in many areas important to commercial users, and as a result has a promising future in business environments. Look for industry-leading versions of the operating system, such as HP-UX, to distance UNIX from proprietary alternatives in providing secure, open computing. -Wayne Caccamo is an HP product manager, HP-UX Security, based in Cupertino, CA.

Type:	cover story
Topic:	Operating Systems
	Data Security
	Access Controls
	Enhancements
	UNIX

Record#: 11 868 777.

*** End ***

```
*****  Computer Select, December 1992  : Articles *****

Journal:    Software Magazine  Jan 1992 v12 n1 p106(1)
* Full Text COPYRIGHT Sentry Publishing Company Inc. 1992.
--------------------------------------------------------------------------
Title:     SecurityCheck. (Raxco Software Inc.'s security software)(New
           Products: Minicomputers: Systems Support)(brief article)
           (Product Announcement)


--------------------------------------------------------------------------
Full Text:

From Raxco Software, Rockville, Md., the SecurityCheck package is a
security assessment software solution for Unix systems.

SecurityCheck has a Motif-style graphical user interface that provides an
overall view of a system's security.  The product looks at various areas
of Unix security, including user passwords, file system security, network
facilities setup and integrity and end-user file security.  These checks
also look for potential viruses, worms and Trojan horses.

A user can define batch procedures that will run automatically on a daily
basis.  SecurityCheck also automatically generates shell scripts to
correct problems and upgrade security to a desired level.

The product supports Digital's Ultrix and Sun Microsystems' SunOS
operating systems.  License fees range from $495 for a single-user system
to $6,995 for a 50-node workstation network.


--------------------------------------------------------------------------
Type:      brief article
           product announcement
Company:   Raxco Software Inc.
Product:   SecurityCheck
Topic:     Product Introduction
           Security Software


Record#:   11 732 270.
                              *** End ***
```