Global Information Assurance Certification Paper

Aspects of biological evolution
and their implications for Unix computer security.

Introduction

To those versed in both evolutionary theory and the functionality and structure of modern operating systems analogies between the two are obvious. Both biological organisms and operating systems are entities that have evolved their complexity over time. While changes in structure and physiology of organisms are driven by processes of natural selection (Darwin, 1859), development of an operating system is driven by the need to improve efficiencies, add functionality, and integrate new technologies into core functionalities (Tanenbaum & Woodhull, 1997). During this process and with advances in technology it is inevitable that software which once supported necessary functionalities will cease to be critical or even useful for the vast majority of users. This change in status could be based on a number of factors ranging from simple technological obsolescence to changes in the environment in which the OS operates that makes its use no longer safe or prudent.

Although parallels exist with biological organisms a fundamental difference is apparent. For example selective pressures could cause a physiological processes to change and become more efficient. During this process these selective pressures would drive the complete replacement of the "old system" with the new. Therefore, antiquated processes and systems are naturally removed. The same cannot be said for operating systems, while additions to an OS occur rapidly removals of previously added components appear rare. Because a program or suite of programs have an established presence in the OS and the belief that it may be of use to some small percentage of users antiquated legacy programs remain a component of the general distribution.

Numerous programs exist in a distribution of any modern operating system that either fit this description today or soon will. This paper contains a partial listing of this software and a discussion of its history and associated security issues.

UUCP

Created in the late 70's the UUCP suite was designed to facilitate movement of email and Usenet News between computers over telephone lines (2). Throughout the early 80's it continued to be an important system to allow for movement of mail and News between sites world-wide. Because of advances in technology and the development of the Internet other software and protocols assumed the task previously performed by this software suite and its initial reason for existence ended in the late 80's (4). However, it remains part of modern versions of the Unix OS.

Today there is no standard UUCP distrbution. Besides the initial implementation (2) over the program's history three different versions have been developed: 1) “Version 2 UUCP” appeared in 1977; 2) “HoneyDanBer UUCP”, aka HDB UUCP, was developed in 1983; and, 3) Taylor UUCP, its most recent version 1.06.1 was released in 1995 (1, 2). These package contains many programs including uucico, uusched, uuxqt, uux, uucp, uustat, uulog, uuname, uuto, uupick, and cu (2).

Security issues associated with UUCP include bad or missing passwords (5), misconfiguration of key files that allow for root access (6), and buffer overflows that allow privileged system access (7, 8). One of the programs in UUCP, cu – "Call another Unix system", has been the subject of several buffer overflow exploits (7). Additionally, if the program does not function properly it can allow a user to take control of a modem that is attached to the system possibly even allowing for the reconfiguring of the device to permit a user to dial any number chosen thus making external connections to external unauthorized systems. This reconfiguration could even extend to setting the modem on a system to answer incoming call thus creating the possibility of not only compromising the system carrying the modem but also any associated system on a local network (Garfinkle & Spafford, 1996).

NIS

Network Information Systems, developed in the early1980's by Sun Microsystems, allowed for the management of a large number of machines via configuration files on a single system (Garfinkle & Spafford, 1996). NIS is actually a distributed database utilizing a number of daemons including ypbind, ypserv, ypxrfd, and rpc.nisd to share password files, group files, host tables, plus other files over a network (Garfinkle & Spafford, 1996; Nemeth, et. al., 2001). Its value came from the fact that important files, such as the password files only needed to be maintained on one system.

Unfortunately, NIS was never designed to be a secure system. Its vulnerabilities are associated with the fact that the program controls access to user accounts. An attacker's ability to penetrate a client is based on his/her ability to fool the server. Once a server believes you have an account the clients will give you access. Additionally, NIS can provide great amounts of information about a system. Anyone who can guess or has the NIS domain name for a system can get its encrypted passwords, hostnames, usernames, and ip addresses (11; 12; Garfinkle & Spafford, 1996).

Finger

Finger is a utility that allows people to query a system and get information about a user (16). The main direct vulnerability associated with the program is buffer overflows (10). This appears to be rare (13, 14, 15) however the real danger lies in the proper functioning of the program. It can provide a great deal of information to an attacker such as user ids and periods of activity on a system thus allowing for exploits based on social engineering (17, 19).

Telnet & Ftp

Telnet, i.e. TErminaL NETwork protocol (20, 21), and ftp, i.e. File Transfer Protocol (22), are two of the older applications on the Internet. Telnet allows a user to logon to a remote system via a TCP/IP network while ftp allows a user to transfer files across the same network. Both control access to system resources by requiring a user to enter their user id and password to login. However the design of these programs is inherently insecure since the user's password is transmitted across the network as clear text. With the advent of packet sniffers, programs that are able to monitor and intercept traffic on a TCP/IP network, it became possible for a user on one system to monitor traffic and intercept passwords of users on systems connected to the same subnet (Garfinkle & Spafford, 1996).

Besides issues associated with transmission of passwords as clear text numerous security issues have been discovered over the years that exploit problems in the daemons associated with telnet, telnetd, and ftp, ftpd (23, 24, 25, 26, 27). These vulnerabilities have allowed many systems to be compromised. Subsequently these systems can be used to launch attacks on other systems remotely.

R utilities (rcp, rdist, rsh, & rlogin)

These commands allow for easy access of resources on one machine from another. Rlogin is a telnet replacement that allows a user to quickly login from one system to another with the command "rlogin *machine_name*"; rsh allows a user to remotely run a program on a system with the command "rsh *machine-name:/pathTo/file*"; rcp allows a user to copy a file from one system to another with the command "rcp */pathTo/source_file machine_name:/pathTo/destination_file*"; and rdist which allows a whole series of files to be copied from one system to another. All these utilities function without use of a password.

The control mechanism which regulates this access is either the .rhost or hosts.equiv file. The .rhost file can either contain a list of machine-user name pairs, one per line, of systems and users that have access to the machine or simply a line containing the machine name. In the first case the rights that these users have access to are based on the position of the .rhost file in the directory structure. For example, if a .rhost file in the home directory of "jill" on the machine "chilili" contains the line "zuzax jake" the user "jake" on the machine "zuzax" can use the rutilities to login, copy, or execute files on "chilili" with the privileges of "jill" (Freeland, McKay, & Parkinson, 2001). However, in the second case, that of only a machine name, all users on that system would have access. In this example if the .rhost file contained the line "zuzax" all users on zuzax would have access to chilili as jill. On some systems the structure of the hosts.equiv file can be the same as the .rhost file while on others only machine names are allowed. Here the use of a user name would allow that user to log on to any account on the system. Using the present example if the hosts.equiv file of chilili contained the line "zuzax jake" jake could log on to any account on cilili.

Of special note here are the massive security issues associated with .rhosts files in the root home directory and "+" signs in the hosts.equiv file (Garfinkle & Spafford, 1996). A .rhosts file in the root's home directory would allow for listed users on a listed system or worse yet in the case of just listing the machine name, any user on the listed system to run any of the R utility programs as root. Also dangerous is a "+" sign the hosts.equiv file. Here permission to access system resources are essentially granted to all and sundry.

Security concerns associated with the use of the R utilities revolve around the easy, almost automatic, access they provide to a user or program on a remote system. Besides allowing hackers to move freely between systems these utilities have also allowed software driven Internet worms to spread among machines (9, 25).

Conclusions

While some of the mentioned programs are clearly antiquated and could be removed from the general distribution of a modern OS without notice of most users others are still actively used today even thought problems associated with their continued use are obvious to all. During future evolution of the OS Unix vendors will have to deal with the conflicting needs for stability and backward compatibility of applications versus the ever increasing need for security. Certainly market forces will drive this process.

It is interesting to speculate how the continued presence of these ”historic” programs in a modern OS will be managed as knowledge of their use fades. Here it is interesting to note that a casual examination of several recent books on administering Unix systems no longer give significant coverage to UUCP (for example see Nemeth, et al. 2001; Freeland, McKay, & Parkinson, 2001; Frisch, 1995). While the R utilities are still discussed they are generally dismissed and administrators are simply advised to turn them off. One wonders if this knowledge will go the way of UUCP in the next edition of the same books.

References

Internet Resources

(1) http://rpmfind.net/linux/mdw/LDP/nag/node147.html UUCP History
(2) http://www.ensta.fr/internet/unix/communications/Taylor_UUCP.html Taylor UUCP
(3) http://www.uucp.org/history/index.shtml The UUCP Project: History
(4) http://www.stargate.com/history.html Stargate History
(5) http://xforce.iss.net/static/1012.php Unix default login and password
(6) http://xforce.iss.net/static/554.php Incorrectly configured UUCP
(7) http://security-archive.merton.ox.ac.uk/bugtraq-200101/0310.html Buffer overflow in Solaris cu
(8) http://security-archive.merton.ox.ac.uk/bugtraq-200101/0328.html Buffer overflow in HPUX cu
(9) http://linuxtoday.com/stories/4408.html Report of Linux X86 worm found in the wild.
(10) http://www.cis.ohio-state.edu/Services/rfc/rfc-text/rfc1135.txt The Helminthiasis of the Internet
(11) http://www.scrye.com/~kevin/lsh/april-99/old/Security-HOWTO.info-3 NIS security Information.
(12) http://linux.cska.net/howto/Security-HOWTO Linux Security HOWTO
(13) http://security-archive.merton.ox.ac.uk/bugtraq-199805/0195.html Hpux finger security issue
(14) http://security-archive.merton.ox.ac.uk/bugtraq-199805/0240.html Hpux finger security issue
(15) http://www.csclub.stthomas.edu/~bugtraq/1998/msg00919.html HPUX finger security issue
(16) http://aa11.cjb.net/sun_managers/2000/08/msg00207.html Sun Managers Summary, Finger Security
(17) http://www.cis.ohio-state.edu/Services/rfc/rfc-text/rfc1309.txt Technical Overview of Directory Services
(18) http://www.cis.ohio-state.edu/Services/rfc/rfc-text/rfc0385.txt Comments on the File Transfer Protocol (RFC 354)
(19) http://www.seas.rochester.edu:8080/CNG/docs/Security/node9.html Definition of Social Engineering
(20) http://www.urec.cnrs.fr/cours/Applis/general/tsld014.htm Telnet RFC 854
(21) http://info.internet.isi.edu/in-notes/rfc/files/rfc854.txt Text of RFC 854
(22) http://www.wu-ftpd.org/rfc/ Listing of RFC's for ftp
(23) http://www.cert.org/incident_notes/IN-2000-09.html Systems Compromised Through a Vulnerability in the IRIX telnet daemon.
(24) http://securityportal.com/topnews/caldera20000313.html Caldera Linux Advisory: Security problem in telnetd.
(25) http://securityportal.com/cover/coverstory20000814.html Stupid, Stupid Protocols: Telnet, FTP, rsh/rcp/rlogin
(26) http://www.cert.org/advisories/CA-1991-11.html CERT® Advisory CA-1991-11 ULTRIX LAT/Telnet Gateway Vulnerability
(27) http://www.cert.org/advisories/CA-1989-03.html CERT® Advisory CA-1989-03 Telnet Breakin Warning.

Books

Darwin, C. 1859. The Origin of Species: By Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life. Grammercy, Inc. ISBN: 0-51712-320-7

Freeland, C., D. McKay. and K. Parkinson. 2001. Solaris for Managers and Administrators, third edition. OnWord Press. ISBN: 0-7668-2137-4

Frisch, A. 1995. Essential System Administration, second edition. O'Reilly & Associates, Inc. ISBN: 1-56592-127-5

Garfinkle, S., and G. Spafford. 1996. Practical Unix & Internet Security, second edition. O'Reilly & Associates, Inc. ISBN: 1-56592-148-8

Gregory, P. H. 2000. Solaris Security. Sun Microsystem Press, A Prentice Hall Title. ISBN: 0-13-096053-5

Nemeth, E., G. Snyder, S. Seebas, and T. R. Hein. 2001. UNIX System Administration Handbook, third edition. Prentice Hall, Inc. ISBN: 0-13-020601-6

Tanenbaum, A. S., and A. S. Woodhull. 1997. Operating Systems: Design and Implementation, second edition. Prentice Hall, Inc. ISBM: 0-13-638677-6