The following paper was originally presented at the
Ninth System Administration Conference (LISA '95)
Monterey, California, September 18-22, 1995

Security Administration in an Open Networking Environment

Karen A. Casella
Sun Microsystems, Incorporated

Security Administration in an Open Networking Environment

Karen A. Casella – Sun Microsystems, Incorporated

ABSTRACT

As networking technologies evolve and business needs change, traditionally isolated and secure communication networks are giving way to more open computing environments. Security, network and systems administrators must therefore concern themselves not only with firewall and boundary security, but also with individual system security. Security administration in a large open network is a challenging assignment and requires a combination of auditing, assessment and compliance mechanisms. For very large networks, automation is another variable which is critical to consider in this equation. There are several tools available to assess the security of networks and systems; however, there are few freely available solutions for addressing the problems that these analysis tools detect.

This paper describes the changing network security paradigm and discusses what tools are available for identifying security vulnerabilities in an open network environment. It goes on to state the problem that we faced at Sun and describes the suite of tools that we have designed and implemented as a solution, focusing on the automation of system security assessment and compliance. Finally, *SunSWAT*, the Sun Security Weakness Attack Tool, is introduced and its evolution from a single shell script designed to respond to the results of a network security audit, into a system for improving system security, implementing enterprise security standards and auditing to those standards, is discussed.

Background

The traditional model for network security has been to fortify the boundaries of the network, monitor the perimeter and trust everything and every one inside. For many networks, boundary entry points are limited to a single Internet connection and perhaps a modem or modem pool. The Internet connection is typically made through a carefully controlled firewall complex. Modems are configured for dial-back or one time password schemes. Boundary security is tight and relatively easy to administer and monitor.

Picture a castle with a moat surrounding it with one or two heavily-fortified drawbridges. Everyone within the walls of the castle trusts each other; there are no locks on doors.

Emerging communications technologies, combined with changing business needs, have caused many companies to rethink this model of network security. Some examples follow:

- To keep pace with the competition, many companies are opening up their networks to business partners via the Internet or private network providers.
- Connecting small remote offices using dedicated circuits is not always cost effective, so some companies are investigating alternate networking solutions, including ISDN or the use of private or public networks for data communication.
- Telecommuting is a growing interest for many companies. Employees are beginning to work at home more to allow the company to save money, to keep employees happy and in some states, to comply with clean air regulations.
- Similarly, nomadic computing is taking off. Many employees who travel extensively, especially those in the sales force, carry their work environments on laptop systems, performing their day-to-day business from hotels, airports and phone booths.
- Many companies are finding themselves with a rising contractor workforce, which introduces new challenges for network security and systems administration groups.

These situations and others like them are forcing companies into a more ''open'' security model for their internal networks.

To support this new network paradigm, security must be maintained not only at the firewall and on critical server systems, but also on the individual desktop systems, i.e., every system on the network must be as secure as possible without negatively impacting usage.

Picture a modern day city. Nobody trusts anybody; there are locks on every door.

Herein lies the challenge. How can system support personnel effectively audit and maintain compliance to security standards (which often change) on a large, open network? There are many tools available to systems administrators to assess network and system security, but there has been little work done to assist with the automation of security compliance.

Network Security Tools

Tools that can be used to secure networks and systems generally fall into three basic categories.

Network security auditing tools act on a network of systems, scanning systems from outside of each system, for vulnerabilities that may allow unauthorized access to systems.

System security assessment tools are designed to examine individual system security from within the system by searching for conditions or configuration errors that may cause a system to be vulnerable to attack.

Security compliance tools are used to correct configurations and patch security holes on systems.

Several freely available tools exist that provide for system or network security auditing and assessment; however, the number of non-commercial solutions for actually correcting the security vulnerabilities that they discover is limited.

The following sections describe some of the tools that are available in each of these categories.

Network Security Auditing Tools

AutoHack [1] is a tool for auditing the security of a large TCP/IP network. *AutoHack* attempts to exploit well known vulnerabilities and generates reports based on what it finds. *AutoHack* is currently being used only within Sun Microsystems for network security auditing.

SATAN (System Administrator Tool for Analyzing Networks) [2] is similar to *AutoHack* in that it scans systems connected to the network noting the existence of often exploited security weaknesses. SATAN also investigates the ''web of trust'' in network security and offers a tutorial that explains what problems are discovered and what can be done to address them. SATAN is freely available in the public domain.

ISS (Internet Security Scanner) is also a multilevel security scanner. Since its initial release, ISS has become a commercial product and is no longer freely available.

System Security Assessment Tools

COPS (Computer Oracle and Password System) [3] examines a system for a number of known weaknesses and alerts the system administrator to them. COPS does not attempt to correct or exploit any of the potential problems it finds. There is an option to generate a file containing shell commands that attempt to fix a subset of the problems found, but it must be used very carefully.

The **tiger** [4] package of system monitoring scripts is similar to COPS in what it does, but is more extensive in the checks that it performs and is somewhat easier to configure and use.

Security Compliance Tools

The **secure_sun** script by David Safford checks and fixes fourteen common SunOS 4.x security holes. This script does not address Solaris 2.x systems or other operating systems and is limited in its scope with regards to the checks and fixes that it performs.

The Problem

A combination of the three types of tools described in the previous section is required to secure a large, open network.

To identify which security vulnerabilities exist, it is important to regularly audit security by centrally scanning the network and identifying which systems are vulnerable to external threats. Then, each of those systems must be examined to determine which conditions exist that allow for exploitation. Finally, the conditions must be corrected to adequately secure the system.

On a network the size of Sun's (nearly 30,000 hosts), this is no trivial task. Automation of the network security auditing, system security assessment and security compliance functions is critical.

Sun's Solution

Our approach for improving network and system security was to create a suite of tools for auditing, assessment and compliance. *AutoHack* was developed to perform the network security auditing and reporting. We may have been able to use one of the freely available system security assessment tools; however, we still needed a security compliance tool. Since none of the freely available tools matched our requirements very well, we decided to develop a system that combined the system security assessment and compliance functions.

Design Goals

The design and implementation of *AutoHack* are beyond the scope of this paper and are described in a paper by Alec Muffett for the 1995 USENIX Security Symposium [1].

The requirements for a system security assessment and compliance tool are as follows:

- Minimally, the tool must check for the conditions which allow exploitation by *AutoHack*. Additional security configuration checks are desirable.
- The solution must be easy to use and support both interactive and non-interactive use. In interactive, verbose mode, the security assessment and compliance tool can be used for training systems administrators in network and system security. The non-interactive mode is required for automating security administration.
- The tool must function on every version of the Sun Operating System from SunOS 4.0

through Solaris 2.4. There must be no dependencies on languages, tools, or utilities that are not included in the core distribution of the operating system.

- The security assessment and compliance system should ideally be modular, simplifying the modification of existing modules or the addition of new modules.
- Policy based, hands off operation is desirable. The tool must be able to facilitate compliance with enterprise computing security standards.

System Overview

Take One – *AutoHack*

The first release of our system security assessment and compliance solution was a shell script that I wrote one weekend. This quick and dirty prototype, *autohackfix*, was a single interactive script that addressed only those security problems that were identified by *AutoHack*. It was designed to be run interactively by a systems administrator on individual systems and checked for the conditions on a system that would allow *AutoHack* to exploit it. If a condition existed, the systems administrator was given the option to fix the problem automatically or not.

The *autohackfix* script was obviously limited in scope and was replaced within a couple of weeks by *SunSWAT* – the Sun Security Weakness Attack Tool.

Take Two – *SunSWAT*

SunSWAT, in its current release, is a series of scripts and programs designed to fix various security holes in Sun systems running the Solaris operating system (either Solaris 1.x or Solaris 2.x). *SunSWAT* consists of a single driver script and several modules.

Driver Script

The driver script, *sunswat*, is used to determine system configuration, check for prerequisites and call the individual modules. Usage for *SunSWAT* is:

```
# sunswat [-a] [-v] [-h] [-c file]
```

where:

-a is an optional flag that forces the default response for all questions. The default responses may be over-ridden by modifying the *$HOME/.sunswatrc* file or by specifying an

```
echo
echo "Checking rexd ... "
CheckRootOwn $INETD
CheckWorldWritable $INETD
CheckGroupWritable $INETD
if $GREP "^rexd" $INETD > /dev/null; then
        if [ "$VERBOSE" ]; then
                echo
                echo "The rexd daemon is a notoriously insecure service"
                echo "which would allow people worldwide to steal any"
                echo "file on your system.  It should be removed or disabled."
                echo
        fi
        AskYN " Disable rexd?" ${REXD_DISABLE:-"Yes"}
        if [ $? -eq 1 ]; then
                $SED 's/^rexd/#rexd/g' $INETD > $INETD.$$
                $MV $INETD.$$ $INETD
                $CHMOD 644 $INETD
                KillPID inetd
                (echo "$SED 's/^rexd/#rexd/g' $INETD > $INETD.$$") >> $TF
                (echo $MV $INETD.$$ $INETD) >> $TF
                (echo $CHMOD 644 $INETD) >> $TF
                echo "AH0:$INETD:rexd:disabled" >> $AF
        else
                echo "AH4:$INETD:rexd:enabled" >> $AF
        fi
else
        echo "AH0:$INETD:rexd:disabled" >> $AF
fi
```

Figure 1: Sample *autohackfix* Code

alternate configuration file using the **-c** option.

-v is an optional flag that causes *SunSWAT* to run in verbose mode, explaining why conditions should be changed to enhance security.

-h is an optional flag that displays a help message.

-c **file** can be used to specify an alternate configuration file.

If no options are given, *SunSWAT* is run interactively. The configuration file, *sunswatrc*, can be used to enable or disable the different modules and to change the default response for any or all ‘‘fix this?’’ questions.

After executing each of the individual modules, *SunSWAT* sends a report containing information on what it found, and what it did to correct security problems, back to a central tracking server. The information is processed automatically and a tracking database is updated. This database is used to generate daily progress and escalation reports.

Modules

The *SunSWAT* driver script calls the following modules:

- **autohackfix.sh** repairs the problems identified by *AutoHack*. ‘‘Web of trust’’ checking is done, by investigating the contents of the */.rhosts* and */etc/hosts.equiv* files, optionally removing insecure entries such as ‘‘+’’ or comments. Entries for insecure network services can be commented out of the *inetd* configuration file. Piped mail aliases are verified and optionally disabled. Basic password file checking is done, with comments being removed and accounts with no passwords disabled.

  Figure 1 shows an excerpt from the *autohackfix* module that is used to check and fix the configuration of *rexd* in the */etc/inetd.conf* file. The variable *$TF* represents the name of a trace file that is kept for each *SunSWAT* run. The *$AF* variable is used to represent the name of an audit file which is sent to a central server which is used to track *SunSWAT* usage and update a system status database.
- **fileperms.sh** is used to change the file permissions of system files to a more sane mode (from a security standpoint).
- **ftp.sh** checks for, and optionally disables FTP for root and other non-human users and then proceeds to verify the configuration of anonymous FTP on the system if it is enabled. Ownership and permissions of all anonymous FTP directories and files are verified and optionally corrected.
- **misc.sh** is a script to check and fix several miscellaneous security problems, for example, checking and removing writability of important system directories, checking and modifying the permissions of */etc/utmp*, and searching and modifying permissions of setuid/setgid scripts.

```
#################################################################
# /etc/inetd.conf (or /etc/inet/inetd.conf):
#
# TFTP_DISABLE  Disable tftp completely?
# TFTP_SECURE   Configure tftp in secure mode?
# UUCPD_DISABLE Disable uucpd?
# REXD_DISABLE  Disable rexd?
# SYSTAT_DISABLE        Disable systat?
# NETSTAT_DISABLE       Disable netstat?
# FINGERD_DISABLE=      Disable fingerd?
# RUSERSD_DISABLE=      Disable rusersd?
# SPRAYD_DISABLE=       Disable sprayd?
#
#################################################################
TFTP_DISABLE="No"
TFTP_SECURE="Yes"
UUCPD_DISABLE="Yes"
REXD_DISABLE="Yes"
SYSTAT_DISABLE="No"
NETSTAT_DISABLE="No"
FINGERD_DISABLE="No"
RUSERSD_DISABLE="No"
SPRAYD_DISABLE="No"
```

Figure 2: Excerpt from *SunSWAT* Configuration File

- **nfs.sh** checks the configuration of NFS and modifies the */etc/exports* or */etc/dfs/dfstab* file as needed.
- **patch5x.sh** is a script to apply security patches to Solaris 2.x systems. A list of mandatory security patches is maintained by Sun's Network Security Group and distributed with the *SunSWAT* program.
- **root.sh** checks and fixes root's environment, ensuring that there is no ''.'' in root's path, verifying a sane value for root's umask and checking and optionally modifying the permissions of directories in root's path.

Configuration File

The *SunSWAT* configuration file can be used to predefine directory locations, select which modules to run and how to respond to all questions. The configuration file can always be overridden at run time if *SunSWAT* is invoked in interactive mode.

Figure 2 shows a sample set of rules for *inetd* configuration. The systems administrator can modify the default behavior of *SunSWAT* by following the directions in the configuration file and basically answering a set of ''yes'' and ''no'' questions. Several sample configuration files are provided for the systems administration staff representing different security models. For example, there is a configuration file that provides for adherence to the enterprise desktop system security standard. Others are provided for different production servers and other types of hosts on the network.

Initial Implementation

SunSWAT was designed to help solve the problem of securing the sheer volume of systems on Sun's internal network. It is distributed via NFS mounts using Sun's internal software distribution mechanism and is available on every system on Sun's network. Superuser privilege is required to execute *SunSWAT*, since it modifies system files owned by root.

SunSWAT was used in the initial effort to secure all systems on Sun's networks, most frequently executed from the command line by qualified systems administrators.

Systems administrators further automated the use of *SunSWAT* by centrally controlling its execution from a single trusted host within an administrative domain. Configuration files specific to a domain's user requirements are developed and used

```
#!/sbin/sh
#
if [ -f /usr/dist/local/config/pkgs/sunswat/samples/desktop_ens_sunswat_rc ]
then
        SWAT_RC="/usr/dist/local/config/pkgs/sunswat/desktop_ens_sunswat_rc"
elif [ -f /usr/dist/config/pkgs/sunswat/samples/desktop_ens_sunswat_rc ]
then
        SWAT_RC="/usr/dist/config/pkgs/sunswat/samples/desktop_ens_sunswat_rc"
elif [ -f /usr/dist/pkgs/sunswat/samples/sunswatrc.autohackfix ]
then
        SWAT_RC="/usr/dist/pkgs/sunswat/samples/sunswatrc.autohackfix"
else
        exit
fi

if [ -x /usr/dist/pkgs/sunswat/bin/sunswat ]
then
        run_level=`/usr/bin/who -r | /usr/bin/awk '{ print $3 }'`
        if [ "${run_level}" -gt 2 ]
        then
        echo "Executing sunswat"
                ( /usr/dist/pkgs/sunswat/bin/sunswat -a -c ${SWAT_RC} ) &
        else
                echo "Queuing sunswat job to be executed at midnight"
                echo "/usr/dist/pkgs/sunswat/bin/sunswat -a -c ${SWAT_RC}" | \
                        /usr/bin/at -s midnight
        fi
fi
```

Figure 3: *SunSWAT* Startup Script

to control execution of *SunSWAT* from a central location.

Further Implementation

The *SunSWAT* tool and how it is used continues to evolve.

System Security Standardization

SunSWAT has been selected by the Sun computing standards development group as the method to be used to comply with existing and evolving security standards and policies. In the method prescribed by the standard organization, the *sunswat_startup* script, located in */etc/init.d*, is executed during each system reboot. The script, shown in Figure 3, is used to schedule *SunSWAT* to run at midnight following the system boot. If the startup script is started from the command line, at any time other than a system reboot, *SunSWAT* is run immediately.

On existing systems, this *sunswat_startup* script is put into place using either *rcp* or *rdist* from a central administrative trusted host. On new installations and system upgrades, this *sunswat_startup* script is put into place by the Solaris installation process. Most system administration groups at Sun are now using the AutoInstall process for system installations and upgrades and the *sunswat_startup* script installation has been included in the finish scripts of this automated process. For those groups not using AutoInstall, the installation of the *sunswat_startup* script has been included in the system installation procedural documents produced by the computing standards development group.

This model works well for complying with changing security policies. Global security policy changes are made on a centrally located configuration file which all clients reference. Each system is brought into compliance upon its next reboot. If there is some urgency in applying a new security patch, for example, the systems administrators can use the remote execution of *SunSWAT* from a central administrative host method.

System Auditing

Elementary auditing is available with the current version of *SunSWAT*. A standalone script, *swataudit*, can be used to simulate an *AutoHack* attack attempt on the system. This is used by systems administrators to verify the security fixes that they have made at any time without having to wait for the next *AutoHack* sweep. On critical servers and on some desktop systems, *swataudit* is run daily from *cron* which provides for simple auditing and intrusion detection.

More extensive auditing and escalation features are being designed into a later version of *SunSWAT*.

Evolution Of *SunSWAT*

SunSWAT has been used to make modifications on more than 4100 systems on Sun's internal networks and has become widely accepted as the primary method for security policy enforcement. Due to this fact and based on the feedback provided by the systems administrators responsible for applying security measures to systems, *SunSWAT* Version 2 has been designed and is currently being developed and tested. Several new features have been added to *SunSWAT* Version 2.

SunSWAT Version 2 Features

Modularity has been extended. Security checks are grouped logically and many new checks have been added. The functions formerly combined into *autohackfix* have been moved into the other modules and grouped logically by function. The *autohackfix* module still exists (as it is used quite extensively), however, it simply calls the appropriate functions from the other modules.

New modules and modules which have changed significantly are as follows:

- **trust.sh** handles the ''Web of Trust'' checking formerly done by the *autohackfix* module and extends it to include checking users' *.rhosts* files.
- **inetd.sh** is used to check for insecure services or configurations in */etc/inetd.conf* or in the */etc/inetd/inetd.conf* file.
- **mail.sh** performs all sendmail patching and all checking and repairing in the */etc/mail/aliases* file.
- **nis.sh** checks NIS maps for insecure configurations and notifies the appropriate support group for resolution. No automation of the fixes is done at this time.
- **patch4.x** has been added to install mandatory security patches on SunOS 4.x systems.
- **passwd.sh** has been extended significantly to perform several new checks and fixes within each of the files: */etc/passwd*, */etc/shadow*, and */etc/group*.

The **driver script** is being replaced by a C program that controls all of the modules. Checksums for each of the modules are to be coded into the driver program so that the user can be assured that modules have not been tampered with.

The **user interface** has been modified so that the modules to be run can be specified on the command line. For example, in the new scheme, the following command:

```
# sunswat -a autohackfix passwd
```

would cause *SunSWAT* to execute in automatic mode, calling only the *autohackfix* and *passwd* modules. If *SunSWAT* is called without the **-a** switch, a menu is displayed which allows the user to select which modules to execute.

The scheme used for patching systems has been extended and redesigned. Dependence on a single patch distribution server has been removed and the entire process is now much more distributed. A new module for applying patches to SunOS 4.x systems has also been added.

An *undo* feature is being developed which allows the user to restore the system to the state that it was just prior to running *SunSWAT*.

Future Plans

As new exploits become known, *SunSWAT* will be modified to correct for them. Similarly, as new security features get added to the operating system, *SunSWAT* will change to keep pace.

SunSWAT will be extended to be used as a general auditing tool, with the ability to generate either machine readable or human readable audit reports. Additional functionality in the areas of intrusion detection and integrity checking is currently being researched. We are considering developing a graphical user interface, possibly using a WWW browser, to facilitate configuration and centralized operation.

We are also designing a new tracking and reporting scheme, using a commercial relational database, which will allow us to implement more sophisticated escalation procedures. This database will be tied into the system registration and human resources databases to provide information on system ownership and management chain. The escalation process is simple – the first time a system appears on the list, the system owner will receive notification that the system is out of compliance with corporate security policies. If the system is identified by *AutoHack* as having the same types of security weaknesses the following month, notification is sent to the system owner's manager. On the third consecutive violation, notification is sent to an operating company information resources executive. After this final notice is issued and the system is still in violation of the security policies, it is to be removed from the network.

Conclusions

Security administration in large open networks is a very challenging task and it is imperative to automate as much as possible. Most of the tools available today focus on the identification of security vulnerabilities, not the correction of conditions and configurations that cause a system to be vulnerable. A tool that we designed to address security weaknesses identified by network security audits was easily extended to facilitate standardization and adherence to Sun internal security and system configuration policies.

Availability

SunSWAT is only available for use within Sun Microsystems Incorporated.

Acknowledgments

Thanks to all members of the Network Security Group at Sun for providing ideas for modules and code review. Special thanks to Casper Dik for providing much of the code for the file permissions module and to both Casper and Alec Muffett for developing system exploit code which has been adapted for *SunSWAT*. Many thanks to Tom Jollands for developing the AutoInstallation and *sunswat_startup* scripts and for spear-heading the system standardization efforts. Thanks also go to Karen Doby for encouraging me to write this paper and for her review of it.

Author Information

Karen Casella is a Network Security Engineer for the Network Security Group at Sun Microsystems Incorporated in Mountain View, CA. She has taken the roundabout path to security engineering, beginning her professional career as a mechanical engineer, migrating into systems administration and finally making the move into the exciting field of network security. Reach her via U.S. Mail at Sun Microsystems, Inc.; 2550 Garcia Avenue, Mountain View, CA. Reach her via electronic mail at karen.casella@eng.sun.com.

References

[1] Muffett, Alec, ''WAN Hacking with *AutoHack*: Auditing Security Behind the Firewall'', Proceedings of the 5th USENIX Security Symposium, June, 1995.

[2] Farmer, Daniel and Venema, Wietse, SATAN (Security Administrator Tool for Analyzing Networks) documentation, April, 1995.

[3] Farmer, Daniel and Spafford, Eugene H., ''The COPS Security Checker System'', Proceedings of the Summer 1990 USENIX Conference, pp. 165-170, June 1990.

[4] Safford, David R., Schales, Douglas Lee and Hess, David K., ''The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment'', Proceedings of the Fourth USENIX Security Symposium, 1993.

[5] Garfinkel, Simson and Spafford, Gene, Practical UNIX Security, O'Reilly & Associates, Inc., 1991.