YASSP! A Tool for Improving Solaris Security

Jean Chouanard – Xerox – Palo Alto Research Center

ABSTRACT

This paper presents YASSP, Yet Another Secure Solaris Package, a set of tools developed to help a systems administrator to secure a Solaris host. It explains the internal mechanism used by YASSP to implement these security changes so that a systems administrator can either use this tool in its current form and localize it, or modify the package source to match his needs.

YASSP is composed of the SECclean package and a set of optional packages providing common useful security related tools. The SECclean internal mechanism used to modify the existing operating system is implemented through the Solaris package mechanism and provides a full un-installation procedure. It has required the development of a specific library of shell functions, to ease file manipulations.

The YASSP project provides to the community an easy path to secure a Solaris host. It has taught us a lot about Solaris internals and package manipulation. YASSP is still young, and ready to be enhanced.

Introduction

The project YASSP was started back in 1997 to fulfill the need for installing exposed Solaris servers. For such servers, Sun's Solaris default installation has the following weakness:

- As standardly installed, Solaris is running many network services. This violates the principle that a machine should provide only those services that its owner intends to provide.
- As standardly installed, all network services are available to any host that requests them, violating the principle of "offer services only to those you intend to use them"
- As standardly installed, system logging is inconsistent.
- As standardly installed, the system does not issue security-warning banner messages to people who attempt to use it.

YASSP's current version was created at Xerox Palo Alto Research Center, as a configurable tool to secure a Solaris host. A more complete history of the project can be found in later in this paper.

YASSP can be used on Solaris 2.6, 7 or 8, sparc or Intel architecture to enhance the security of an end-user workstation or a server.

Philosophy

Before looking at YASSP internals, let us understand the concepts used to build YASSP on. These concepts had driven YASSP implementation, had created their own problems and provided solutions, but have stayed constant over the existence of YASSP which has made it very stable.

- It must play by Sun's rules and not fight Sun: we follow as close as we can Sun standard and rules.
- YASSP tries to be as closely integrated to Solaris as it can. It was built and distributed as a Solaris package. YASSP will follow package rules and Sun standards when they exist. When standard does not exist, YASSP will try to accommodate the different choices the systems administrator may have made.
- It can be installed and un-installed without destruction and restores the same context as before. Particular attention was given to these points, to be sure that the systems administrator will never lose some files or modifications done after YASSP installation to any YASSP configuration files. The goal here is double: be sure that any changes done by YASSP can be undone, but also be sure that any changes done after will not be lost in case of un-installation. That is true for all components of YASSP.
- It should run on a minimal installation, like the core choice of the Solaris installation, or any other.
  This is one of the most important, and the most limiting rules, as it directly drove choices we made in the tools and language used in YASSP implementation. Except for a few binaries, the core of YASSP is written in Bourne shell, uses sed and [n]awk.
- Additional scripts, not part of YASSP but referred to as examples on the web documentation may use tools not included in the core install, like Perl.
- Secure and closed by default...
- ... but can be opened up after the installation or modified to be localized.
- YASSP default behavior is to be secure. For each configuration choice YASSP offers, it will by default install the one the YASSP team of developers had consider the most secure.

- Most of these configuration choices are easily reviewed and modified after YASSP install.
- Using a three-layer implementation: from the security issue we are trying to solve, we generate a list of modifications that need to happen; this list is implemented by the package installation (File replaced, file edited, file created...)

Being built as a Sun package, and following Sun rules, this tool cares about the Solaris package database, corrects it when it is wrong, and keeps it accurate over time.

Result of YASSP Installation

After installing SECclean, most of your network services will be turned off and most of the processes started by default under Solaris will not run anymore. Your server will be hardened, perhaps too much. That is the SECclean default: if you do not modify its configuration, or if its configuration does not exist, it will be closed and secure. That fits pretty well for installing an exposed server, but not an end user workstation.

SECclean modification to your system can be summarized as:

- Deleting unwanted files (/etc/auto_home, /etc/auto_master, /etc/dfs/dfstab, /var/spool/cron/crontabs/adm, /var/spool/cron/crontabs/lp, and /var/spool/cron/crontabs/uucp).
- Controlling most of the startup scripts through /etc/yassp.conf.
- Changing default environment setting to make your system more secure (umask, password length, PATH) and make some of these variables available through /etc/yassp.conf (/etc/profile, /etc/default/login, /etc/default/su, /etc/default/passwd, /etc/skel/local.cshrc, /etc/skel/local.profile, /etc/.login, /etc/rmmount.conf) and setting default umask for startup script (/etc/init.d/umask.sh).
- Enabling banner files (/etc/motd, /etc/issue, /etc/ftp-banner).
- Controlling access to the system and to specific commands (/etc/pam.conf, /etc/default/login, /.rhosts, /etc/hosts.equiv, /etc/cron.d/at.allow, /etc/cron.d/cron.allow, /etc/default/sys-suspend, /etc/ftpusers, /usr/dt/config/Xaccess, and /etc/dt/config/Xaccess).
- Simplifying network startup, but keeping backward compatibility through /etc/yassp.conf (/etc/init.d/inetsvc, /etc/init.d/inetinit, and /etc/init.d/network).
- More control on RPC services (keyserv and rpcbind with TCP wrapper) [15] (/etc/init.d/rpc).
- Turning off available network services (/etc/inet/inetd.conf).
- System parameters tuning, attempting to prevent and log stack-smashing attacks (/etc/system).
- Network stack performances and security tuning [16] (/etc/init.d/nettune and /etc/default/inetinit).
- Adds few common services definitions (/etc/inet/services).
- Establishes standard syslog configuration (/etc/syslog.conf).
- Miscellaneous and logging (/etc/shells, /var/adm/loginlog, /etc/norouter, and /etc/inittab).
- Package database correction and filemode changes to make them more secure (clean-up and fix-modes).

A more detailed and up-to-date explanation of what is actually done can be found on the web, at: http://www.yassp.org/internal.html .

On top of SECclean, YASSP will provide integrity check (tripwire) [4], TCP and RPC controls (tcpd and rpcbind with tcp_wrappers) [15], encrypted channel (ssh) and others log rotation and version control of system files (PARCdaily, GNUrcs, and GNUgzip).

An example of how a system looks after the reboot can be found on Appendix 1.

First, as Sun packages have an important place in YASSP implementation, let us explain what they are and how they work.

Sun Packages Overview

In the Solaris world, application software or system components are delivered in units called packages[1]. A package is a collection of files and directories required for implementing a new function, and is usually designed and built by the developer of this new function (new applications, system enhancement) after completing the development of the code.

The components of a package fall into two categories: package objects which are the application files to be installed, and control files which control how, where, and if the package is installed.

Control Files

The control files are divided into two categories: information files, and installation scripts. Some of these control files are required and some are optional, installation scripts are always optional. The installation script must be composed of Bourne shell commands.

Installation script performs customized actions during the installation of your package, especially procedure scripts defining actions that occur at particular points during package installation and removal, and the class action script, which defines a set of actions to be performed on a group of objects. Four procedure scripts are predefined: preinstall, postinstall, preremove, and postremove.

[1]This part does not intend to be a package tutorial, but rather a short introduction to provide the reader enough knowledge about them to understand YASSP internals [13].

Package Objects

These are the components that make up the application. They can be: files (executable or data), directories, named pipes, links or devices.

Package objects are described in the prototype files, one line per single deliverable object. An object description includes its location (pathname, relative or absolute), attributes (owner, group and permission mod), class and file type.

Object classes allow a series of actions to be performed on a group of package objects on installation or removal. The sed's predefined class provides a method for using sed instructions to edit files upon package installation and removal. See Sun's documentation for more details about object classes.

Package Delivery, Installation, Registration and Checking

At the installation of a package, all of its objects are registered in /var/sadm/install/contents. Information stored in this file is either static information found in the package's information files (package name, file type of the object, attributes, class), or dynamic information generated at the installation by the pkgadd command, like the date, the size and the checksum (*sum*(1)) of the object, which provides a good integrity check but a poor authenticity check.

The accuracy and the integrity of the various objects of an installed package can be verified using the *pkgchk*(1) command. The '-n' option, which tells pkgchk not to check volatile or editable files, should be used in a post-installation check. Checks are done on the existence of each object, their recorded attributes, their size, and checksum.

Package information can be updated manually anytime using the *installf*(1M) and *removef*(1M) command.

Packages are the recommended way to deliver Solaris applications. Solaris packages provide tools and methods to install new objects, and modify or delete existing ones through either the action class or the installation script. They also provide a basic mechanism to save files modified and will record all installed files, including their attributes and a basic checksum.

Packages are not perfect but exist on every Solaris system.

SECclean Use of the Package Mechanism

SECclean is the core of YASSP, implementing most of the security tuning. SECclean is built as a package, using the rules we explained in the last section. To implement this security tuning, SECclean will modify the existing Solaris system by adding, deleting, replacing or modifying various files.

Files can be sorted into different groups: created, deleted, replaced, and modified files and startup scripts.

New files, deleted files and replaced files are three categories of files easily understood. Sometimes, it is more prudent to modify an existing file instead of replacing it with our version, as they may be different on each system.

Startup files, or Run Control Scripts as Sun documentation named them, are Bourne shell scripts to control run level changes. Each run level has an associated rc script located in the /sbin directory: rc[0-6S].

Run control scripts are located in the /etc/init.d directory with links from the corresponding run control scripts in the /etc/rc[0-6S].d directories. Due to their specific functions, and requirements, startup files are handled as special cases by SECclean.

Since nothing can be simple, exceptions will exist to this classification. SECclean is compatible with three versions of Solaris, and files needed to be replaced may be dependent of the version. Other files we want to modify will need special care, as these modifications can be more complex. Binary files installed by SECclean need to be handled depending of the architecture of the target. Last, some files may need to be modified only if they exist.

File Management Under SECclean

As explained in the Philosophy section, SECclean must be un-installable, and the state of your system after a SECclean installation and un-installation should be the same as before. All the information associated with a file (its contents, its attributes (owner, group and permission mode) and also the package information related to this file) must be extracted and saved at SECclean installation, so that the removal step will be able to restore it. These backup and restoration procedures have to be created, as packages provide you a guideline on how it should be done, but it is up to the package developer to implement such features. A Bourne shell[2] library named cleanlib.sh provides functions to help deal with package registration, files backup and restoration. To perform any file operations, SECclean uses cleanlib.sh functions, as package operations provided by Solaris are very primitive.

Cleanlib provides the following functions:

- Install_file(): It installs a file by moving it from its prefixed name to its real name. It keeps a copy of the installed file so that any modification to it can be tracked, and registers the file as part of the SECclean package.
- Install_RC_file(): Same function as Install_file() but to install a startup script. It also creates the symbolic links needed under the /etc/rc[0-6S].d directories.
- Backup_user_file(): Copies a file and re-create its path under a backup directory.

[2]Why Bourne shell? These functions will be used in the package installation scripts, which must be written in Bourne shell, and we should be able to use this library on a core installation.

- Save_and_move_file(): Moves the indicated file and its package information into a backup area and then un-registers it from the package database.
- Disable_RC(): Removes links to a startup script. It will backup all package information found.
- Disable_Init(): Searches for any registered link to a startup script and calls Disable_RC() for each of them.
- RCconfized_Init(): Modifies a startup script to control its start through /etc/yassp.conf. Will also update /etc/yassp.conf for each startup script modified.
- Create_RCconf(): Initializes the static header of /etc/yassp.conf.
- DE_RCconfized_Init(): Undoes the modification done by RCconfized_Init() on startup scripts.
- Restore_RC(): Restores a startup script links from the saved information and updates the package database.
- Restore_file(): Restores a file from its backup and updates the package database.
- Init_preremove(): Initializes the value for the backup directory for files modified since YASSP installation.
- Cmp_and_backup_file(): Checks to see if any previously installed files have been modified and backs them up if so.

To delete an existing file, we first save its current package information, move it to the package save directory and un-register it from its original packages. This is done by the Save_and_move_file function.

Replacing a file involves more steps. We must first deliver the new file under a different name, in order to not overwrite the existing one. The convention for SECclean is to install the new file with a name prefixed by 'SECclean_'. Then, we must delete, as explained in the last section, the existing file, move the new file to its real name, un-register the prefixed name from the SECclean package and register the real name in it. Also, as SECclean wants to be able to track any changes to these files after its installation, a copy of the original installed file is saved. These operations are done using the Save_and_move_file and the Install_file functions.

Files that need to be modified are defined as part of the sed class in SECclean prototype. No specific action is required as the associated sed script handles the installation and un-installation phase.

For startup scripts, we wanted to offer the systems administrator the choice of running the different startup script present, and we wanted this choice to be as convenient as possible. The solution chosen in SECclean was to modify the existing script so that it will start only if a shell variable is set to YES in YASSP's configuration file /etc/yassp.conf. The name of this shell variable is based on the name of the script, stripped of all non-alphabetic characters and capitalized. The function RCconfized_Init is used to modify each original startup script.

The Disable_Init function is used to backup and delete any information associated with a startup script. It will, from the name of the startup script under /etc/init.d, find all the existing links to it from any /etc/rc[0-6S].d/ directory, record all these links and the package information associated with them, save the content of the original startup scrip and delete them all. It is used when we need to replace or delete a startup script.

Cleanlib also provides the reverse functions, used at the removal phase of the package to restore the state and the contents of these files.

Exceptions exist: some startup scripts need further modification, some startup scripts need to be replaced by our version but depending on the operating system version, some files need to be installed only if they were not already present.

All these functions keep track of the list of packages they touch, so that when we are done, we know which changes have to be validated.

Other Tasks Performed by SECclean

In addition to the file operations we described, SECclean also needs to do the following tasks:

User Convenient Backup

We found it very useful and convenient to copy the tree of files SECclean will touch and provide it as an informational copy to the systems administrator. This copy is not used by the package backup mechanism, but provides the systems administrator an easy way to see which and how files were changed. The backup is done by default under /yassp.bk (See the 'Putting it all together' section for more details).

Password File Cleanup

The password file will be checked and cleaned up if needed, to lock all non relevant system accounts, change the shell of all locked accounts to a binary which will syslog any use of these accounts before exiting. Password file cleanup is implemented in the clean_passwd script, part of SECclean, and can be rerun anytime after SECclean installation.

Operating System Cleanup

When you install Solaris, even if Solaris components are built as packages, you will find some incoherence if you check the package integrity. These incoherencies reflect either files modified during the install, or files registered incorrectly, or both. Using a shell script, which is very OS dependent, SECclean will correct these incoherencies and leave the package database clean.

After, using Casper Dik fix-modes program for Solaris [3], SECclean secures various files permission and ownership. It does this by removing group and world write permissions of all files/devices/directories listed in /var/sadm/install/contents, with the exception of those pre-listed in fix-modes program.

Putting It All Together

Installation Process

SECclean makes use of the preinstall and postinstall procedure scripts.

Preinstall

The preinstall script first determines under which directory the SECclean component will be installed. Some components of YASSP, which may be installed later, have pathnames hard coded in their binary[3]. On the other hand, systems administrators may want to install their software in another place. If /opt/local exists and if we can change dir to it[4], SECclean uses /opt/local as its default installation path. If we cannot change dir to /opt/local, but if /usr/local exists and we can change dir to it, /opt/local is created as a symbolic link pointing to /usr/local. If none of them exists, /opt/local is created as a directory.

Then, the preinstall script initializes the two shell variables $SECBCK and $CLEANUPDIR, if they are not already defined. $SECBCK is used as the pathname under which the user convenient backup will be done. Its default value is /yassp.bk. $CLEANUPDIR is the path where the scripts and binaries related to the operating system clean-up will be installed. Its default value is /var/sadm/clean-up.

Next, the preinstall script defines four shell variables, corresponding to categories of files. The variable $RCCONF corresponds to an exhaustive list of startup scripts. Startup scripts are always referred to by their name under the /etc/init.d directory. This list is exhaustive, gathering all the startup script names we want to control in all Solaris versions we support. Next variable is $NRC, defining the startup script we want to replace with our own OS dependent version. Three startup scripts are part of this list: inetsvc, inetinit and networks (only on Solaris 8). They are the scripts driving the network setup at the boot time. The last two variables defined are the list of files to be deleted ($SD) and the list of files to be replaced ($SA). All these variables are recorded in a file ($PKGSAVE/.PROC_Init_Var), so that the other scripts used in SECclean installation or removal can share the same lists. For an up-to-date list of the files defined in each of these variables, please consult http:://www.yassp.org.

The user convenient backup is the last thing done by the preinstall: all files SECclean intended to touch will be copied under $SECBCK/Before_‘date +%Y.%m.%d-%H.%M.%S‘ .

[3]An example is the RCS package, compiled with the GNUdiff: RCS binary has the pathname of GNUdiff hard coded. The default pathname we used is /opt/local

[4]/opt/local can be a symbolic link to any existing directory.

Installation

It is at the installation phase that all the package objects defined in SECclean will be installed. Objects which correspond to files we want to replace have been registered and will be installed with a name prefixed by ’SECclean_". For objects dependent on the operating system version (inetsvc for example) or on the architecture of the host (binary), all possible instances were registered and will be installed using the OS version or the architecture as a suffix.

All sed scripts associated with any package objects part of the sed class will be then executed, and the targeted file modified.

Postinstall

The postinstall script is the most complex one.

After reading the PROC_Init_Var file created by the preinstall to learn the value of the different categories of the file, it first disables startup files we will replace later, by calling the Disable_Init function.

Next, all the files listed in RCCONF are modified, using the RCconfized_Init function. For each of them, a sed script is generated and applied to them. From the result of which startup files were present on this system, and from some static information, the yassp.conf file is generated.

All the files we will delete or replace are backed up and moved out using Save_and_move_file. At this step, SECclean is done modifying existing packages, and it can now close and validate all the removals done to the package database.

The next steps deal with SECclean package objects. For files that are dependent on the operating system version or on the machine architecture, the right version is kept and others are deleted and unregistered. SECclean package information is updated and the database is closed. All the files we want to replace, that have been installed using ’SECclean_’ as a prefix in their name, are renamed to their right name. The new inetinit, inetsvc and networks[5] startup scripts are installed and symbolic links are created from the appropriate /etc/rc[012].d directories, and special cases are handled for specific files. SECclean database is then closed again, to validate all these changes.

We are now done with file installation and package database update. Tuning to some newly installed files like the /etc/system is done depending on the operating system version, syslog files used by the newly installed syslog.conf file are touched, and the password file is cleaned up by calling clean_passwd.

The last thing done is the operating system clean up, which includes the package database correction and running fix-modes.

We are done: SECclean is installed.

[5]These startup scripts are treated as a special case: we want them to execute the strict minimum instead of handling all the cases planned by Sun (DHCP for example). The new scripts provided are always based on the original ones. Setting variables in yassp.conf, can restore the original Sun behavior of these scripts.

Removal

The removal phase has two main tasks: restoring the state your workstation as it was before SECclean installation, but also keeping a copy of any files, installed by SECclean, and modified by the systems administrator so that these modifications will not be lost. It will use the preremove and postremove procedure scripts to implement these tasks.

Preremove

After reading the PROC_Init_Var file created by the preinstall to learn the value of the different categories of files, it checks all files replaced by SECclean against the originals, and keeps a backup copy under /var/tmp/SECclean.Backup_$$ if they have been modified. Next, it makes a copy of the cleanlib.sh library, as this file will be removed in the removal phase, and the postinstall needs it.

Removal

All files registered as part of SECclean prototype are deleted. Sed scripts associated to package object of the sed class will be executed to undo any modifications done.

Postremove

After reading the PROC_Init_Var file created by the preinstall to learn the value of the different categories of files, it will unregister from the package database all files registered as part of SECclean in the postinstall script, listed in the $SA and $NRC variables, and close the database.

Next step is to restore the original contents of files SECclean has deleted or replaced, and re-register them in the package database with their original attributes.

The modifications to control the start of the startup scripts are undone, and the package database is closed to validate these changes.

Last, the copy of the cleanlib.sh library is deleted.

How We Tested It

This complex process was validated by installing (and un-installing) YASSP on various hosts, for testing as well as being part of the day-to-day work.

The final state after the installation is tested using tools like titan [8], ISS ... and of course we asked the beta testers to use their expertise.

The SANS team is also very helpful for advice or consensus on proposed changes, or for pointing out existing documentation and tools [2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14].

Particular attention was given to test the package removal function. Testing can be time consuming, especially when the package is broken and corrupts the package database. Use of Vmware virtual machine running Solaris for testing is a big time saver.

YASSP Other Goodies

YASSP also offers to install a set of useful and common tools, built as separate packages. It includes:

- GNUrcs: RCS 5.7 and diff 2.7 [GNU] to ease version control of your files.
- GNUgzip: gzip 1.2.4a [GNU]: file compression.
- PARCdaily: Some daily scripts, logs rotation, backup and RCS for system files... Need GNUgzip and GNUrcs. It is mainly an example to show people what they can do.
- WVtcpd: tcp_wrappers 7.6 + rpcbind 2.1 [Wietse Venema]: tcp wrapper, and Wietse's version of rpcbind compiled with tcp wrapper library, for people who need to run RPC.
- PRFtripw: Tripwire 1.2 [Purdue Research Foundation of Purdue University]: Integrity checking tools
- SSHsdi: version 1.2.30-SDI, build with SecurID support (See ftp://ftp.parc.xerox.com:/pub/jean/sshsdi/README).

All these packages will, when installed, take all the necessary steps to be ready for use. For packages relying on configuration files, some proposed configuration files are provided, and they will be installed as the default configuration file if they were not already present.

YASSP Installation

What Can Go Wrong During the Install

From our experience, nothing should go wrong if it is a fresh Solaris install. For an installation done on an existing system, you may find some conflicts during the package installation with files or directories.

Example of Installation

```
# uncompress yassp.tar.Z
# tar xvf yassp.tar
# cd yassp
# ./install.sh
... check and modify all
      the configuration files ...
# reboot
```

A commented installation log can be found in Appendix 2.

What May Be Broken After?

SECclean and YASSP defaults are to disable (or to 'secure') most of the existing services. If you are planning to install YASSP or SECclean on an existing server, you must know what application(s) are running on it, and what network services and system resources are being used. Your work after the YASSP install will be to re-enable these services, and only these services, to have a system where you know why any process or service is running. YASSP will help you by providing some tools to monitor these changes and re-enable some security features after its installation, but YASSP cannot understand your setup as you can do.

Also, if your system had non-standard applications or configurations, SECclean may have missed some tuning specific to your application.

What To Do After?

Most of YASSP configuration is handled by a single text file /etc/yassp.conf. A man page, *yassp.conf*(4) http://yassp/man/yassp.conf-4.html, describes the different tuning that can be done through this file. A more generic description of what to do after the installation, not focused on SECclean, is maintained at: http://yassp/after.html .

And do not forget to send us feedback about YASSP; we are eager to receive comments, suggestions or just acknowledgement that people are using it.

Where Are We Now?

As of October 10, 2000, beta#12 of YASSP is released on http://www.yassp.org . It has been tested and supports Solaris 2.6, 2.7 and 8, under sparc architecture. Beta#12 is used widely[6], and is the release candidate for v1.0 [17].

YASSP is built on top of two independent components:

- The core package, named SECclean, implementing the OS security modification, which also includes correcting the incoherence left after a standard Solaris install and modifies some filemodes to make the file hierarchy more secure.
- A set of additional packages, provided as an optional install as they represent some commonly used features not implemented by Solaris.

Systems administrators can use these tools choosing either the YASSP install.sh script included on the YASSP tar file, or to run each package or script manually.

The default install.sh script of YASSP will cleanup the Solaris install, modify the filemodes to secure various files and directories, install the core package and propose to install a set of additional packages.

After installing the core package, SECclean, the result is a system quite closed and secure, where most of the services have been turned off, minimal processes are running, and with some security tuning applied at various levels of the operating system, from the network stack to the XDM configuration file.

Future

Still a lot to do, from gathering more feedback, to porting it to new Solaris releases, and improving the web documentation.

[6]Widely is vague, but it is hard to be more precise. Since the YASSP site was announced, the logs show the number of downloads, but unfortunately, we have received very little feedback.

Two main directions can be guessed from the talk in YASSP mailing list:

- Develop new tools to help monitor the system after the initial installation, to check that no modifications happen and be able to re-apply some security settings on-demand.
- Develop a tool to manage some more restrictive file permissions, especially for SUID/SGID binaries, as an option.

Conclusion

The YASSP project has taught us a lot on Solaris internals, both from a security point of view and about the Solaris package mechanism. The complexity added by the way YASSP deals with files and package information has shown its strength as it provides a robust un-installation mechanism and eases a lot YASSP maintenance for any new version of Solaris.

YASSP is used by various organizations, in various contexts from end user workstations to exposed servers. Feedback we received shows us that it is a useful tool that people appreciate. YASSP is young, and will continue to be improved, but we have to be careful to keep it focused on securing a Solaris host, and keep its size reasonable so that we can maintain it

Last, we would like to thank all the people who have participated in the YASSP mailing list or have sent us feedback. Please continue.

References

[1] Bastille Linux, http://bastille-linux.sourceforge.net/ .

[2] Boran, Seán, *Installing a Firewall bastion host: Hardening Solaris* http://www.boran.com/security/sp/solaris_hardening3.html .

[3] Dik, Casper, *Great Solaris tools (fix-modes) and the Solaris 2 FAQ*, http://ftp.wins.uva.nl:/pub/solaris , http://www.science.uva.nl/pub/solaris/solaris2.html .

[4] Kim, Gene and Gene Spafford, *Tripwire: Free version V1.2,* ftp://ftp.cerias.purdue.edu/pub/tools/unix/ids/tripwire .

[5] Noordergraaf, Alex and Keith Watson, *Network Settings for Security,* http://www.sun.com/software/solutions/blueprints/1299/network.pdf .

[6] Noordergraaf, Alex and Keith Watson, *Minimization for Security: A Simple, Reproducible and Secure Application Installation Methodology*, http://www.sun.com/software/solutions/blueprints/1299/minimization.pdf .

[7] Pomeranz, Hal, *Building Bastion Hosts With Solaris: Step by Step*, http://www.deer-run.com/solaris.html .

[8] Powell, Brad, Matt Archibald, and Dan Farmer, *The Titan Project,* http://www.fish.com/titan/ .

[9] *Sabernet Solaris Security Guide,* http://www.sabernet.net/papers/Solaris.html .

[10] *Solaris Security FAQ*, http://www.sunworld.com/common/security-faq.html .

[11] Spitzner, Lance, *White papers*, http://www.enteract.com/~lspitz/papers.html .

[12] Sun, *Solaris 8 System Administration Guide, Volume 1*, http://docs.sun.com/ab2/coll.47.11/SYSADV1/@Ab2TocView?Ab2Lang=C&Ab2Enc=iso-8859-1 .

[13] Sun, *Application Packaging Developer's Guide*, http://docs.sun.com/ab2/coll.45.13/PACKINSTALL/@Ab2TocView?Ab2Lang=C&Ab2Enc=iso-8859-1 .

[14] University of Waterloo, *Security: How to Documents,* http://ist.uwaterloo.ca/security/howto/ .

[15] Venema, Wietse, *Tools and papers*, ftp://ftp.porcupine.org/pub/security/index.html .

[16] Vöckler, Jens-S., *Solaris 2.x – Tuning Your TCP/IP Stack and More*, http://www.rvs.uni-hannover.de/people/voeckler/tune/EN/tune.html .

[17] *YASSP Mailing list archive,* http://www.theorygroup.com/Archive/YASSP/ .

History of the YASSP project

A Long Time Ago . . .

Back in 1997, we faced the task of having to deploy firewall and external servers in various locations around the world. Even if the configuration was standard and well known, we realized quickly that, as the local Unix skills were unknown, we needed to help systems administrators and provide a way to install these servers so that we could be pretty comfortable that the result of this installation was what we expected to see.

We started to write down some guidelines on how to install and then secure these servers. That was perfect for experienced systems administrators, but we quickly realized that something written does not mean that people will read it, and that this guideline was written for people understanding the context of the installation (skilled in network, Unix, Solaris), which was not always the case at these remote sites.

We translated these guidelines to shell script, very primitive, which was supposed to do all the work automatically. It required good Unix skills from the systems administrator running it, was very touchy with the existing state of the server, which had to be a fresh installed OS, and was destructive in the sense that files were replaced without backup. There was no way to undo the damage caused by it, neither was there a way to use it to perform a different task than installing an external server as defined per our organization.

Xerox PARC

In 1999, the author came to Xerox Palo Alto Research Center and was able to allocate more time to this project. The idea was to enhance the existing baseline and enable its use on end-user workstations.

The shell script was ported to the Sun package format, allowing it to be un-installed and integrated into our (PARC) default installation process.

This package grew, including lots of suggestions from its users (mainly Xerox people) and new features or tuning gathered from various sources.

At this time, it provided a way to harden a fresh Solaris installation to be used mainly on an exposed server (most of the services will be turned off), was un-installable if necessary. A public version was available on our external ftp site and was used by various people. Basic documentation existed on the same ftp server.

SANS

With the SANS Institute, we created a team to work on tools for securing Solaris. This team (10 people at the beginning, about 160 now) was open to anyone who committed to spend time testing or reviewing existing tools and providing feedback.

It drove some major changes on the existing tools.

First, the scope was extended: it should still focus on exposed servers, but should be also suitable for end-user workstations.

Also, it was modified to be permissive on the expected state of the operating system and able to be installed on existing install rather than requiring a fresh Solaris install.

Some recommended security tools were added to it, implementing additional features not available in Solaris.

At last, and on top of all the bug corrections and enhancements suggested by this team, the tool was modified to be more verbose, have better documentation and provide the systems administrator an easier way to configure the desired security level.

Of course, some problems were raised and some nice verbal argumentation happened, but life would be too quiet without that.

Why Does It Still Exist?

At the time it was started, there were not many tools available to secure Solaris, but now, various other tools exist (Titan, Bastille-Solaris), and people might wonder why we spend our energy on creating and maintaining a separate tool rather than merging our efforts with existing tools.

The first reason to keep YASSP alive was that it existed, was working, was used by people and was useful to them if we believe the feedback we received at the time.

Also YASSP deals correctly with Solaris packages, and will keep the database clean. This part was unique to YASSP, as well as the unique configuration file it offers.

Last, we found that for some systems administrators, it was less confusing to have the minimum interaction with the script, especially with less Unix experienced systems administrators. These systems administrators, with little or no Unix/Solaris experience, may be required to install Solaris server in a hostile environment. YASSP will do that by default.

Security is a field where having a choice of tools is a plus. More choices provide flexibility and more appropriate solutions.

Appendix 1: System with YASSP installed

```
Rebooting with command: boot
Boot device: disk  File and args:
SunOS Release 5.8 Version Generic 64-bit
Copyright 1983-2000 Sun Microsystems, Inc.  All rights reserved.
configuring IPv4 interfaces: hme0.
Hostname: zeta
Tweaking Solaris TCP/IP: Solaris 7 or above (excellent)
   tweaking separate connection queues
   tweaking against SYN flood symptoms
   tweaking timeouts
   tweaking pMTU discovery interval and common timers
   tweaking misc. parameters
   applying security tweaks...
   tweaking windows, buffers and watermarks
done.
The system is coming up.  Please wait.
checking ufs filesystems
/dev/rdsk/c0t0d0s5: is clean.
Setting netmask of hme0 to 255.255.255.0
syslog service starting.
sshd starting.
The system is ready.
WARNING: To protect the system from unauthorized use and to ensure that
the system is functioning properly, activities on this system are
monitored and recorded and subject to audit. Use of this system is
expressed consent to such monitoring and recording. Any unauthorized
access or use of this Automated Information System is prohibited and
could be subject to criminal and civil penalties.
zeta console login: root
Password:
Last login: Wed Jul 19 19:07:06 on console
This computer system for authorized use only
# ps -eaf
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0 22:49:18 ?        0:15 sched
    root     1     0  0 22:49:19 ?        0:00 /etc/init -
    root     2     0  0 22:49:19 ?        0:00 pageout
    root     3     0  0 22:49:19 ?        0:00 fsflush
    root   212     1  0 22:49:45 console  0:00 -sh
    root   173     1  0 22:49:43 ?        0:00 /usr/sbin/syslogd
    root   185     1  0 22:49:44 ?        0:04 /opt/local/sbin/sshd
    root   227   212  0 22:53:42 console  0:00 ps -eaf
    root   168     1  0 22:49:42 ?        0:00 /usr/sbin/cron
# netstat -an
UDP: IPv4
   Local Address         Remote Address     State
-------------------- -------------------- -------
      *.514                                 Idle
      *.*                                   Unbound
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*                0      0 24576      0 IDLE
      *.22                 *.*                0      0 32768      0 LISTEN
      *.*                  *.*                0      0 32768      0 IDLE

TCP: IPv6
 Local Address     Remote Address  Swind Send-Q Rwind Recv-Q   State      If
--------------- ---------------- ----- ------ ----- ------ ---------- -----
    *.*              *.*              0      0 24576      0 IDLE
# pkgchk -n
```

Appendix 2: Installation example

Commands typed by the systems administrators are in `fixed-width bold`; comments are in *italic roman* typeface.

```
zeta console login: root
# ./install.sh
```

Running YASSP installation script.

```
<...>

Type the package list you want to install or hit return to accept the
default:

 SECclean GNUrcs GNUgzip PARCdaily WVtcpd PRFtripw

<return>
```

We chose to install all packages proposed.

```
Do you want to install SSH (See the SSH-COPYING file for the SSH
license)?: [y|n] (n) y
```

We want SSH.

```
YASSP will install: SECclean GNUrcs GNUgzip PARCdaily WVtcpd PRFtripw SSHsdi
Installing the various package:
==========
SECclean
==========
```

SECclean installation start.
The pre-install runs, initialize some variable and back-up files
it will modify.

```
Using /opt/local as the root dir.
Backing up all files under /yassp.bk/Before_2000.07.19-22.35.53:
<... Long list of files ...>
```

Pre-install is done.
The install runs: files declared in the prototype are installed
silently. Files, part of the sed class in the prototype, are
modify by the associated sed script.

```
Modifying <...>
...
```

The postinstall start. It first reads the variables stored by
the pre-install.

```
The postinstall script is silently running. It may take a while
on slow machine. Just be patient
```

Disabling init files we will replace later.

```
Disabling startup files: inetsvc inetinit network
```

Modifying startup files to be controlled by yassp.conf.

```
Modifying Startup files to use /etc/yassp.conf: <... list of all startup
files modified ...>
```

Creating /etc/yassp.conf, as we know now which startup file were modified.

```
Creating your default /etc/yassp.conf
```

Saving (in the package's save directory) and deleting files.
Some of them will be replaced by SECclean's own version later.

```
Saving files: <... list of files backed up ...>
```

We have unregistered (removef) all the files we deleted from
the package database.
We must now close (removef -f) these open packages.

```
Closing the package we touched <... package name list ...>
```

These are the files that will be replaced by SECclean
version. They have been installed (as part of SECclean's
prototype file) as /path/name/SECclean_{name_of_the_file}, and
are registered under this name as part of SECclean package. We
must first unregistered (removef on SECclean) them.

```
Updating SECclean package DB: <... list of file ...>
```

and close SECclean (removef -f SECclean)

```
Closing SECclean DB
```

Move the files from their SECclean_{name} to {name} and
register them as part of SECclean (installf)

```
Replacing: <... list of files  ...>
```

OS specific startup files: chose the right version.

```
Choosing the right startup files: /etc/init.d/inetsvc
/etc/init.d/inetinit /etc/init.d/network for your OS: Solaris 5.8
```

Replacing them, registering (installf) as part of SECclean package, and creating the sym-link.

```
Replacing Special startup files: /etc/init.d/inetsvc
/etc/init.d/inetinit /etc/init.d/network and creating the symlink
```

Closing (installf -f SECclean) SECclean.

```
Closing again SECclean DB
```

Specific OS tuning: for Solaris 8, no priority_paging

```
Tuning /etc/system to comment out priority_paging
```

Running clean_passwd

```
Cleaning the passwd file...
 Disabling UID 0 account(s):
Disabling system account(s):
        daemon bin sys adm lp uucp nuucp listen nobody noaccess nobody4
 Deleting account(s):
root identity will be changed to "Root at zeta"
password and shadow files saved under /etc/passwd.Old and /etc/shadow.Old
```

Doing the OS cleanup: fix-modes is run first...

```
Doing the OS Clean-up
Running fix-modes  2.6 2000/01/13 14:13:35 casper
fix-modes done, log file under: /var/sadm/clean-up/clean_up.log
```

Then we correct well-known incoherence in the Sun installation

```
clean-up the contents database
cleanup done, log file under: /var/sadm/clean-up/clean_up.log
Running /usr/lib/makewhatis /opt/local/man
```

We are done with SECclean, echo the summary.

```
   ======================================================
   SECclean installation has finished.
   Changes to the file-system  and package database are documented in:
      /var/sadm/clean-up/clean_up.log
   All changed or replaced files are archived in
      /yassp.bk
   If existing, the crontab for the users:
      lp adm uucp root
   has been deleted. Please, re-enable manually the entries needed
   Backup for the crontab files are under:
      /yassp.bk/var/spool/cron/crontabs/
   To finish hardening, this host must be rebooted.
   However, you should first check that /etc/yassp.conf is configured
   to your requirements. See also yassp(1) and yassp.conf(4).
   ======================================================
Installation of <SECclean> was successful.
==========
GNUrcs
==========
Installation of <GNUrcs> was successful.
==========
GNUgzip
==========
Installation of <GNUgzip> was successful.
==========
PARCdaily
==========
Modifying /usr/lib/newsyslog
Installation of <PARCdaily> was successful.
==========
```

```
WVtcpd
==========
```

tcp_wrappers add some example of how to use it in comment in /etc/inetd.conf

```
Modifying /etc/inet/inetd.conf
```

and install some default configuration file if they were not present.

```
Creating /etc/hosts.deny from the distribution file
 *** Please configure it!
Creating /etc/hosts.allow from the distribution file
 *** Please configure it!
Installation of <WVtcpd> was successful.
==========
PRFtripw
==========
```

Default configuration file is created if it was not present.

```
Creating /secure/tripwire/tw.config from the distribution one:
/secure/tripwire/tw.config.Dist
 *** Please configure it!
you may use tripwire now.
Type: "cd /secure/tripwire/; ./tripwire -i 2 -initialise -c tw.config"
    to create a new database,
Use "cd /secure/tripwire/; ./tripwire -q -i 2 -c tw.config"
    to check,
 ***** SAVE YOUR DATABASE IN A SECURE PLACE *****
Installation of <PRFtripw> was successful.
==========
SSHsdi
==========
Modifying /etc/syslog.conf
```

default configurations files are created if they were not present.

```
Creating /etc/ssh_config from the distribution file
 *** Please configure it!
Creating /etc/sshd_config from the distribution file
 *** Please configure it!
```

We generate the keys if they were not present.

```
Initializing random number generator...
<...>
Your public key has been saved in /etc/ssh_host_key.pub
ssh has been installed.
run '/etc/init.d/sshd stop;/etc/init.d/sshd start' and restart syslogd
to use the new binaries/configuration
Installation of <SSHsdi> was successful.
```

YASSP install is done, now recreate the whatis database if it was present.

```
Rebuilding the whatis database
YASSP is installed.
Most of these changes will take action at the next reboot.
**** YOUR WORK IS NOT DONE YET ****
     *) Edit and configure /etc/yassp.conf
     *) Edit and configure /etc/hosts.deny /etc/hosts.allow
     *) Edit and configure /etc/sshd_config /etc/ssh_config
     *) Read http://yassp.parc.xerox.com/after.html
        and the papers linked under http://yassp.parc.xerox.com/ref.html
     *) make any additional changes/software installation
     *) CREATE YOUR tripwire DATABASE AND SAVE IT!!!

 Type:
vi /etc/yassp.conf /etc/hosts.deny /etc/hosts.allow /etc/sshd_config
  /etc/ssh_config ; cd /secure/tripwire; ./tripwire -i 2 -initialise -c
  tw.config; cp /secure/tripwire/databases/tw.db_zeta TO_A_SECURE_PLACE

   ***YOUR feedback*** is important: please send comments or flame to:
```

```
            sansro@sans.org, chouanard@parc.xerox.com
            with "YASSP" in the subject
# reboot
```