Global Information Assurance Certification Paper

Automated Solaris Hardening: An Overview of YASSP

Jim Hurst
June 12, 2001
Version 1.2e

Introduction

A Solaris system administrator faces an immediate and daunting problem. Solaris, while a capable and robust operating system, as delivered by Sun is inherently insecure. In their efforts to make the system easy to install, network, and manage, Sun has included dozens of features and services that pose security risks. This leaves the system open to a long list of vulnerabilities.

This document introduces YASSP, an automated tool for addressing Solaris vulnerabilities. The need for such a tool is examined. Then YASSP is introduced, and the specific changes that YASSP makes are explained. Next the nature and function of several independent packages that are included with YASSP are briefly discussed. The document concludes with an attempt to put YASSP and its capabilities in the larger context of security as an ongoing process. Appendices provide a list of shell scripts and services that YASSP disables by default, and a comparison of representative Solaris configuration files before and after the application of YASSP.

Default Security: Death by a thousand cuts

An administrator responsible for managing dozens of servers and workstations simply cannot address all known vulnerabilities one at a time. Administrators of most sites have developed standardized platforms tailored to the unique needs of their organization. Security is only one of several factors that must be integrated into a "standard" local build, and it is one that is easy to overlook when configuring complex systems.

The process of enhancing operating system security is known as hardening. While an experienced administrator can effectively harden manually, it is vastly more efficient to do this by script. Several hardening tools are available for Solaris, but YASSP is one of the better known and more widely used.

Sun is by no means alone in shipping a default operating system that is not appropriate for internet deployment "out of the box." Many of the problems are historical to the UNIX environment. UNIX traditionally has numerous "well-known" services running on low port numbers that are only rarely used and present both denial-of-service and compromise vulnerabilities. For example, an attacker could craft a packet to the chargen service on server A that pretends to be from the echo service on server B. Chargen sends a string to echo, which sends the string back to chargen, which sends a new string back to echo… The resulting network traffic can quickly bottleneck network connections, and paralyze a pair of servers. The solution is to disable these services on startup.

The problem is that unhardened UNIX is full of such weaknesses. The UNIX operating system was designed and deployed in an age of kinder, gentler networks. The default stance of Solaris (and most other commercial operating systems) is to allow everything, and to assume that unwanted services will then be removed . A modern administrator cannot afford this default-permit stance. Instead, administrators must deploy systems with a default-deny stance: nothing is allowed by default, and permitted services are then turned on.

Enter YASSP

YASSP (Yet Another Solaris Security Package) is a public domain tool designed to address Solaris security. YASSP has been developed by Jean Chounard of the Xerox Palo Alto Research Center, along with an international team of Solaris experts. Chounard and his team are now working with the SANS Institute and other collaborators to make the tool more widely available. The YASSP version current as of this writing is V0 beta#15.

YASSP is designed to be applied to a system BEFORE it goes into production. A system with YASSP applied meets the definition of a bastion host. All unnecessary services are removed. Known vulnerabilities are addressed. Because YASSP makes extensive changes, often to (justly) obscure services and files, inexperienced administrators should use caution. Deciphering the changes YASSP makes so that critical services can be restored is not a productive use of an administrator's time.

A terse summary of the YASSP philosophy is as follows: YASSP will conform as closely as possible to Sun standards and rules. It must install and un-install cleanly. It must be able to run on a minimal install, and must be graceful and tolerant about what it finds on the existing installation. A YASSP system will by default limit nearly all services for security, but it may be modified to allow more services.

The YASSP Installation Process

YASSP is downloadable from http://www.yassp.org as a gzipped or compressed tarball. After download, it is advisable to check the PGP signature, to verify the download is indeed legitimate. To install it, unpack the tarball, which creates a directory "yassp". The following command (which must be run as root) will then install YASSP:

"cd yassp ; ./install.sh"

YASSP will then ask some questions, and run its scripts.

What Happens During YASSP Installation

Rather than a single script or package, YASSP is a bundle of packages. The primary package, SECclean, does the bulk of the reconfiguration and is discussed here. The supporting packages include TCP wrappers, tripwire, SSH, PARCdaily, tocsin, and the GNU packages RCS and gzip. These packages are discussed in the subsequent section.

Package SECclean applies an extensive set of changes to the system. A high-level description of these changes can be found online at: http://www.yassp.org/internal.html. An explanation of the changes SECclean makes at the file level follows.

A Solaris box is largely configured in the /etc directory, so it's no surprise that YASSP does most of its work here. The files /etc/cron.d/at.allow and /etc/cron.d/cron.allow control who is allowed to schedule jobs on the system. YASSP creates them if they do not exist, allowing only the root user access. The files /etc/rhosts.equiv and /etc/.rhosts are used to establish trust relationships with other systems. YASSP creates them and makes sure they are empty, so that no other systems are trusted by default. The file /etc/notrouter, which prevents a system from forwarding IP packets (and thus being a potential gateway for an attacker), is created. YASSP also adds banners that warn potential intruders the system is off-limits. These files include

/etc/default/telnet, /etc/issue, /etc/motd, and /etc/ftp-banner (banners warning off the unauthorized are the cyber equivalent of no-trespassing signs, and their lack is considered bad form as well as a legal risk). The /etc/ftpd file uses an empty banner (to prevent displaying OS version, which might be useful in footprinting), and also changes the umask (the bit-field which specifying file permissions) to 077, which denies all access to all users except the owner.

Several new startup files are added. /etc/yassp.conf is the YASSP configuration file. This file sets environment values, and provides a single, central location for managing the changes that YASSP makes. Yassp.conf is a shell script that can be sourced by other shell scripts, or searched to retrieve specific values. The first part of yassp.conf is a set of environment values used to manage the startup files. The SECclean package modifies other startup files to conditionally exit if the corresponding shell variable in yassp.conf is not set to "YES". The default installation of YASSP sets all these variables to "NO", thereby disabling the services. YASSP also aggregates some groups of system variables here to allow turning on groups of startup files (ie, services) with a single variable.

The second part of yassp.conf contains shell variables used by other SECclean scripts. These variables control settings such as default umasks, locked out accounts, the execution of internet services, and the degree of security applied to TCP.

The list of initialization files modified for conditional execution is extensive. All of these files are found in the directory /etc/init.d, and an annotated version of the list is given in Appendix A. The list is instructive, because it shows the breadth of YASSP's approach to the UNIX security problem: lots of service and configuration changes are applied. Many sysadmins will be unfamiliar with some of these services, but all pose some unique security risk. Changing these init files to shut down extraneous services is one of the most important benefits of YASSP. Appendix B gives provides more detail as to exactly how YASSP changes these files, with code examples of the differences between before-YASSP and after-YASSP versions of three representative configuration files.

Inetd is a general purpose server that manages numerous network services, and is configured from the /etc/inetd directory. Four files in /etc/inetd use yassp.conf to configure internet services. umask.sh is used to reset the umask to 077 (or other value as set in /etc/yassp.conf) whenever run level is changed. This is accomplished by making it the target of symbolic links from the file S00umask.sh in /etc/rc[0123S].d The file /etc/init.d/nettune is the target of the symbolic link /etc/rcS.d/S31nettune, which adjusts the default TCP parameters. The pair of files /etc/init.d/inetinit and /etc/init.d/inetsvc derive from yassp.conf the value of environment variable $SUNSTARTUP. If it is set to "YES" they do not change the original behavior, but otherwise they configure the internet services to a minimal configuration: set TCP ISS generation, set the default router and domain name, reset netmasks and broadcast addresses for all interfaces, run inetd (iff so specified in yassp.conf via the RUNINETD variable) and run the named daemon if DNS is configured.

YASSP modifies numerous Solaris configuration files by sed script. The originals of some of these files are saved to the /yassp.bk directory before being replaced with YASSP's more security aware versions. The /etc/services file has numerous useful services added, including ssh (the secure shell, discussed below) and the one-time password authentication services SecurID, tacacs, and radius. /etc/system is modified to try to prevent and log stack-smashing attacks (also known as buffer overflow, a vulnerability found in numerous variants across many applications) by adding the lines "set noexec_user_stack = 1" and "set noexec_user_stack_log=1". Blocking this multi-headed threat in a single location is YASSP at its best (on the down side, this may

interfere with some compilers). /etc/rmmount.conf is altered to disallow mounting suid programs. /etc/inetd.conf is modified by commenting out everything (including such favorites as ftp and telnet). /etc/pam.conf is set to disallow rhosts authentication. The file /etc/passwd locks out several system default logins. The file /etc/ftpusers details a list of users not allowed to use FTP; YASSP extends the list if it does not exist.

Several files in /etc/default are enhanced during the YASSP install. /etc/default/sys_suspend is set to block execution of sys-suspend to all users except root. /etc/default/login changes the default PATH and SUPATH variables, and sets the umask. /etc/default/passwd is modified to require at least 8 character passwords, and /etc/default/inetinit is modified to require RFC 1948 TCP sequence number generation (used to block attacks based on sequence number guessing). Files /usr/dt/config/Xaccess and /etc/dt/config/Xaccess are altered to allow XDMCP connections to only the local host. Finally, the files /etc/skel/local.profile and /etc/skel/local.cshrc, which set the default login environments for Korn shell and C shell environments, respectively, are modified to provide minimal configurations.

Remote procedure calls are configured in the file /etc/init.d/rpc. The RPC services are both widely used and a serious security risk, so they are treated as a special case. /etc/init.d/rpc is one of the files whose conditional execution is controlled by yassp.conf. Further, Sun's version of rpcbind is replaced by Wietse Venema's version if requested.

A pair of binaries are installed outside of /etc as well. /opt/local/bin/md5 provides the md5 digital signature (checksum) program from OpenSSH. /usr/sbin/noshell provides the noshell program from the Titan project, which is used to log attempted accesses to locked accounts.

The list of issues addressed by SECclean shows the scope of YASSP. Dozens of files are removed, modified, or replaced, affecting every aspect of system operation. Most systems will never require more than a handful of the removed services. But, as discussed in the next section, some modification after the install should be expected, because the YASSP install removes all but the most critical services. The reason FTP and Telnet are so popular is that they meet users needs. Turning on required services that YASSP disabled by default during the install is the next step.

Post Install

The job is not done after the YASSP install script has run. Now the administrator must add back any services they wish to run on this system that YASSP has removed. For example, if FTP is required on the system, the administator must edit /etc/yassp.conf and /etc/inetd.conf. A more thorough coverage of the issues involved here is given on the YASSP website.

User accounts have been locked in file /etc/passwd. If any of these accounts are to be active, they must be reactivated now by editing this file. Cron has been made accessible to root only, and any old cron files have been replaced (they can be recovered from the /yassp.bk directory). SSH is locked down so that no access is available. To grant access, hosts must be added to hosts.allow. Inetd services have been shut down.

Supporting Roles: The Other YASSP Packages

What SECclean accomplishes is the lockdown of the system on bootup: services are removed, access is controlled, and numerous vulnerabilities are addressed. To assist in maintaining a secure system, YASSP brings with it a sophisticated toolkit of other packages. These packages

are TCP wrappers, tripwire, SSH, PARCdaily, tocsin, and the GNU packages RCS and zip. The last pair are standard utilities to manage revision control and create compressed archives, used by package PARCdaily.

TCP wrappers is a technique to limit access to services based on filtering incoming connections by IP address. The YASSP version of TCP wrappers is WVtcpd (where WV stands for Wietse Venema, the author). Solaris communicates with other computers using the TCP/IP protocol. While some popular services like FTP and Telnet have their own ports and daemons running continuously, many of the lesser used services are managed by the inetd daemon. This program monitors many ports and initializes a particular service when it is needed. The idea of TCP wrappers is that rather than an incoming connection talking directly to the controlling daemon of a service, it first talks to the TCP wrapper daemon, who validates the connection and verifies that it should indeed be allowed access the service being requested. Two configuration files, hosts.allow and hosts.deny, control what hosts should be allowed to connect. TCP wrappers is not a complete security solution, but it is one more layer of the defense-in-depth required in the enterprise.

The Tripwire package is used for integrity checking. The version of tripwire bundled with YASSP is a public domain tool from the Purdue Research Foundation of Purdue University. Tripwire takes "snapshots" of an existing (and presumably, known good) system configuration, saving it as a reference. Later, it compares the existing configuration with the reference. This is one of the only ways to detect malware such as trojans, which makes Tripwire an invaluable part of the security specialists' toolkit. There is also a commercial version of tripwire available from Tripwire, Inc.

The down side of Tripwire is that it takes care and feeding. Tuning is required to reduce the number of false positives, that is, reports of altered files that do not represent security problems. Also, when patches are installed, the old tripwire configuration becomes obsolete, and a new reference must be created.

Another handy auxiliary package is OPENssh. SSH, or the secure shell, is a program for logging into and executing commands on a remote system. It is intended as a secure replacement for telnet, rsh, and rlogin. SSH provides encrypted communications between two endpoints, so that the problem of running an unencrypted session across an insecure channel is addressed. SSH can also forward X11 connections, and arbitrary TCP/IP ports.

Many users don't realize that passwords are sent in cleartext across the internet in many protocols, including FTP, Telnet, and rlogin. SSH addresses this problem by assuring the passwords, if sent at all, are sent in encrypted form. OPENssh offers several methods of authentication, but for Version 2 (which is preferred), public key using the DSA algorithm is the first choice. If the public key authentication fails, the encrypted password is sent.

PARCdaily is a package used to set up a daily cron job that performs a set of basic maintenance functions, including rotating log files, backing up key system files, and doing a minimal check of filesystems and packages. PARCdaily comments out all existing lines in the file /usr/lib/newsyslog, because it assumes management of the system log files. This is a straightforward tool. More information about PARCdaily can be found on the YASSP site. The source is available there for review.

PARCdaily uses a pair of tools from the GNU toolkit, namely gzip (a compression and archiving utility) and RCS (the revision control system, including the programs ci for checking in and co for checking out). More on these packages can be found at http://www.gnu.org.

A recent addition to the list of YASSP packages (and one that is not currently installed by default) is tocsin, developed by Doug Hughes of Auburn University. Tocsin is a lightweight intrusion detection system. It puts a network interface into promiscuous mode, and monitors and logs port scans (note that this will only see traffic to the local node on switched networks). Scanning networks for vulnerable systems is one of the preliminaries to an attack, so tocsin adds one more piece to the security puzzle that YASSP addresses.

Finding Expert Answers – The YASSP Mailing List

Administering Solaris is a complex process. Even experts have questions they are unable to answer. The response of the computing community to this has been the development of forums, mailing lists, and discussion groups where questions can be put to the community at large. YASSP has its own mailing list, which can be joined by sending an email to the address secure-sol-request@parc.xerox.com, with a subject line of subscribe. A public archive also exists on The TheoryGroup website. These resources offer useful technical support to both new and experienced users.

Fitting YASSP into the Security Process

YASSP does not exist in a vacuum. As shown above, it consists of an ensemble of interlocking parts. SECclean rewires the system configuration to a default-deny stance, removing many services. Tripwire and PARCdaily promote monitoring, backup, and intrusion detection. SSH provides encrypted remote access. WVtcpd provides a degree of security to TCP connections. Tocsin logs attempted scans. The GNU utilities ease comparison and management of logs.

None of this is enough.

YASSP converts a system with a default install into a bastion host. While this greatly enhances the security of the system, the administrator must *use* the tools YASSP provides to be effective. The tripwire and daily logs must be monitored, and suspicious results investigated. Relevant mailing lists and websites must be monitored for new vulnerabilities, and the appropriate patches from vendors obtained and applied. All of this takes time and effort, on an ongoing basis.

Good security is a process. YASSP can greatly aid this process by closing hordes of known vulnerabilities, but it cannot do the work of maintaining security. There is no substitute for administrators knowing their systems, and monitoring them closely. Effort spent in this direction will be time well spent.

With that caveat, YASSP represents a state-of-the art approach to automated hardening. It provides administrators a convenient and effective way to take advantage of collected security expertise. YASSP offers Solaris users a solid foundation upon which to build and maintain secure systems.

References

Beale, Jay "Tripwire - The Only Way to Really Know." July 11, 2000 URL: http://www.securityportal.com/topnews/tripwire20000711.html. (June 8, 2001).

Boran, Sean "Hardening Solaris with YASSP." March 8, 2001 URL: http://www.boran.com/security/sp/Solaris_hardening3.html. (May 30, 2001).

Boran, Sean "Comparison of Solaris Hardening Scripts." November 24, 2000 URL: http://www.boran.com/security/sp/comparison1.html. (May 30, 2001).

Chounard, Jean "YASSP – Yet Another Solaris Security Package." Version 1.61. November 19, 2000. URL: http://www.yassp.org (June 1, 2001).

Chounard, Jean "Post installation steps." November 15, 2000. URL: http://www.yassp.org (June 5, 2001).

Chounard, Jean "YASSP –Daily Cronjob." Version 1.10. November 13, 2000. URL: http://www.yassp.org/daily.html. (June 8, 2001).

Chounard, Jean "daily." Version 1.32. November 30, 2000. URL: http://www.yassp.org/src/PARCdaily/daily.html. (June 8, 2001).

Farmer, Dan "Titan." URL:http://www.trouble.org/titan/lisa-paper.html. (June 14, 2001).

Frisch, Aileen. Essential System Administration. Sebastopol, California. O'Reilly & Associates, Inc. 1995.

Gregory, Peter. Solaris Security. Upper Saddle River, New Jersey: Prentice Hall, 2000.

Hahnke, Petra "Trick Hackers With TCP Wrappers." Performance Computing. September 1997. URL: http://www.performancecomputing.com/archives/articles/1997/september/9709f1bi.shtml.

Lammle, Todd. Cisco Certified Network Associate. San Francisco, California. Sybex, Inc. 2000.

Nipp, Scott "Linux Security: TCP-Wrappers?" May 4, 2000. URL: http://www.linux.com/newsitem.phtml?sid=93&aid=8518. (June 4, 2000)

OpenBSD "OpenSSH." Version 1.111 June 1, 2001. URL: http://www.openssh.com. (June 13, 2001).

Scambray, Joel, Stuart McClure, and George Kurtz. Hacking Exposed (2nd Edition). Berkeley, CA. McGraw-Hill. 2000.

TheoryGroup "The YASSP Development Mailing List." URL: http://www.theorygroup.com/Archive/YASSP/. (June 7, 2001).

Appendix A: Initialization Files modified for conditional execution by YASSP

This is a list of the files that are conditionally executed (that is, may or may not be run, with the default being not), depending on the settings in /etc/yassp.conf. See the third example in Appendix B for an illustration of how this is done.

Filename	Function
apache	apache http server configuration
asppp	managers asynchronous point-to-point protocol (PPP)
autofs	managing automount daemons
autoinstall	install script that is part of the Jumpstart procedure
cacheos	configure the Cache file system
cacheos.finish	"
cachefs.daemon	"
cachefs.root	"
devfsadm	configuration of the /dev directory
devlinks	configuration of the /dev directory
dhcp	configuration of the DHCP server
dhcpagent	configuration of the DHCP client daemon
dtlogin	auto starts dtlogin window after a multi-user boot
dmi	Desktop Management Interface service provider config
init.dmi	DMI service provider config (Solaris 8)
inetinit	manage phase two of TCP startup configuration
inetsvc	manage phase three of TCP startup/configuration
ldap.client	configuration and startup of ldap daemon
llc2	controls execution of llc2 software interface to physical LAN
lp	manages the lp print service
ncakmod	configuration of Network Cache and Accelerator Kernel module
ncalogd	configuration of Network Cache and Accelerator logging
networks	defines known network numbers
nfs.client	managing client side of network file system remote mount capabilities
nfs.server	managing nfs remote disk mount capabilities
nscd	managing the name service cache daemon
power	configuration of the power management system
PRESERVE	specifies location of files being edited to /usr/preserve
rpc	manages remote procedure calls
slpd	configuration of Service Location Protocol Daemon
snmpdx	configure SNMP master agent
init.snmpdx	SNMP master agent configuration
spc	manages spooling as part of the print subsystem
sysid.sys	system config script invoking sysidsys, sysidroot, and sysidpm
sysid.net	final network config script invoking sysidnet
xntpd	managing network time protocol service daemon
utmpd	the utmp daemon clean up after terminated processes
uucp	manages the Unix-to-Unix copy protocol
volmgt	managing volume management, that is, disks, tapes & other peripherals
webstart	multiple CD Install reboot script
init.webm	CIM boot manager

Appendix B: Comparison of Representative Files Before and After YASSP installation

This appendix compare certain Solaris configuration files before and after the YASSP installation. YASSP uses a disciplined and predictable approach to modifying files. When files are saved, they are stored in the /yassp.bk/Before_<timestamp> directory, where <timestamp> is a string representing the time when YASSP was installed, for example: "2001.05.24-16.37.09".

The comparison shown use the output of the Solaris "diff" utility. Diff takes two filenames as arguments, and returns the difference of the two files. In the list of differences, lines preceded by the "<" symbol are from the first file (in our case, the after-YASSP file), while line preceded by the ">" symbol are from the second file (in our case, the pre-YASSP file).

The first comparison is /etc/password. This file controls user accounts, and specifies the account's ability to log in. YASSP applies one change: for all accounts besides root, it makes the default shell /usr/sbin/noshell. This blocks the account from logging in, and also attempts to ensure that login attempts by the account are written to the logfiles.

```
# diff passwd /yassp.bk/B*/etc
1,12c1,12
< root:x:0:1:"Root at sapapp3":/:/bin/tcsh
< daemon:x:1:1::/:/usr/sbin/noshell
< bin:x:2:2::/usr/bin:/usr/sbin/noshell
< sys:x:3:3::/:/usr/sbin/noshell
< adm:x:4:4:Admin:/var/adm:/usr/sbin/noshell
< lp:x:71:8:Line Printer Admin:/usr/spool/lp:/usr/sbin/noshell
< uucp:x:5:5:uucp Admin:/usr/lib/uucp:/usr/sbin/noshell
< nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/sbin/noshell
< listen:x:37:4:Network Admin:/usr/net/nls:/usr/sbin/noshell
< nobody:x:60001:60001:Nobody:/:/usr/sbin/noshell
< noaccess:x:60002:60002:No Access User:/:/usr/sbin/noshell
< nobody4:x:65534:65534:SunOS 4.x Nobody:/:/usr/sbin/noshell
---
> root:x:0:1:Super-User:/:/bin/tcsh
> daemon:x:1:1::/:
> bin:x:2:2::/usr/bin:
> sys:x:3:3::/:
> adm:x:4:4:Admin:/var/adm:
> lp:x:71:8:Line Printer Admin:/usr/spool/lp:
> uucp:x:5:5:uucp Admin:/usr/lib/uucp:
> nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
> listen:x:37:4:Network Admin:/usr/net/nls:
> nobody:x:60001:60001:Nobody:/:
> noaccess:x:60002:60002:No Access User:/:
> nobody4:x:65534:65534:SunOS 4.x Nobody:/:
```

The second comparison is of the file motd, the message of the day, which users see upon login. YASSP changes the motd to a virtual "no trespassing" sign. It also removes information that might be useful to an attacker in profiling the system.

```
# diff motd /yassp.bk/B*/etc
1,3c1
<
< This computer system for authorized use only
<
---
> Sun Microsystems Inc. SunOS 5.8      Generic February 2000
```

The final comparison show a typical "conditional execution" change from YASSP. This is how YASSP shuts down services. If the yassp.conf file doesn't contain a value of "YES" for the environment variable the shell script searches for, the shell script exits before starting its designated service.

The third comparison is of the file nfs.server, but the results would be very similar for most, if not all, of the files listed in Appendix A:

```
2,10d1
< # SECclean START
< # ************************************************************************
< # This shell script was modified by SECclean to start only if
< # the shell variable NFSSERVER is set to 'YES' in /etc/yassp.conf
< # ************************************************************************
< if [ -f /etc/yassp.conf ] ; then
<       . /etc/yassp.conf
< fi
< # SECclean END
23,27d13
< # SECclean START
<       if [ "X${NFSSERVER}" != "XYES" ] ; then
<               exit 0
<       fi
< # SECclean END
```

First, the shell script executes script yassp.conf if it exists. Then it checks the value of the variable NFSSERVER. If it is not equal to YES, then shell script exits before invoking whatever services the script controls.

Last Updated: May 23rd, 2017

Upcoming Training

SANS Stockholm 2017	Stockholm, Sweden	May 29, 2017 - Jun 03, 2017	Live Event
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
Community SANS Portland SEC401	Portland, OR	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Minneapolis SEC401	Minneapolis, MN	Jul 10, 2017 - Jul 15, 2017	Community SANS
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor
Community SANS Atlanta SEC401	Atlanta, GA	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Colorado Springs SEC401	Colorado Springs, CO	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
Community SANS Charleston SEC401	Charleston, SC	Jul 24, 2017 - Jul 29, 2017	Community SANS
SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
Community SANS Fort Lauderdale SEC401	Fort Lauderdale, FL	Jul 31, 2017 - Aug 05, 2017	Community SANS
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event