Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Securing an Anonymous FTP Server in Solaris 8 with WU-FTPD

An anonymous FTP server can be a valuable asset for Internet sites, allowing them to distribute everything from source code and compiled programs to image files and educational material quickly, easily, and reliably. Unfortunately, anonymous FTP servers "out-of-the-box" are inherently insecure. Insecure FTP servers can cause a site to be vulnerable to malicious users. For example, a FTP server could be turned into a repository of illicit material or could be used to attack other machines on the Internet. Luckily, anony...

Copyright SANS Institute
Author Retains Full Rights

AD
Investigate Insider Threats
Protect your company against corporate IP Theft
Learn More
SpectorSoft

Securing an Anonymous FTP Server in Solaris 8 with WU-FTPD

Mansel Bell

GSEC Practical v1.3

3/30/02

Summary

An anonymous FTP server can be a valuable asset for Internet sites, allowing them to distribute everything from source code and compiled programs to image files and educational material quickly, easily, and reliably. This type of FTP server allows a user to authenticate anonymously with a default username of *anonymous* or *ftp* typically followed by any string as a password. [1]

Unfortunately, anonymous FTP servers "out-of-the-box" are inherently insecure. Insecure FTP servers can cause a site to be vulnerable to malicious users. For example, a FTP server could be turned into a repository of illicit material or could be used to attack other machines on the Internet. Luckily, anonymous FTP servers can be made more secure. It is important to choose a secure FTP server implementation and properly configure it for anonymous FTP in order to protect against Internet threats. [2] A good choice for a secure anonymous FTP server is the wuarchive FTP daemon (WU-FTPD).

This paper will present one method of securing an anonymous FTP server in an UNIX environment. The paper will begin with a brief overview of the FTP protocol as defined in IETF Standard 9, RFC-959, including vulnerabilities in its design. A discussion will then proceed about the advantages and disadvantages of anonymous FTP. Next, a synopsis of anonymous FTP security basics, followed by highlights of the security features of WU-FTPD, will be presented. The paper will then focus on the compilation, installation, and configuration of a secure anonymous WU-FTPD server running on a Solaris 8 platform.

An Overview of FTP

FTP (File Transfer Protocol) is a TCP-based protocol designed to transfer files over any network supporting TCP/IP. It utilizes two separate TCP connections for each session: one for control commands and the other for the data being transferred. FTP provides two different ways to establish the data channel: normal mode and passive mode. Normal mode involves the server opening the data channel to the client, while passive mode involves the client opening the data channel to the server. [1][3][4]

In normal mode, the client allocates two high-order ports (i.e. those with a number above 1024). The client initiates a connection from the first port to port 21 on the server, the standard FTP control channel port. The client then sends a

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

FTP PORT command on the control channel, which tells the server the number of the second high-order port to be used for the data channel. The server then initiates a connection from port 20, the standard FTP data channel port, to the second port of the client. At this point, the FTP session has been established and data transfer can proceed. [1][3][4]

In passive mode, the client again allocates two high-order ports and uses the first port to initiate a connection to port 21 on the FTP server. However, instead of sending a PORT command, the client sends a PASV command. This instructs the server to allocate a high-order port of its own to use for the data connection. The server then responds to the PASV command via the control channel, telling the client the high-order port number to use for the data connection. The client then initiates a connection from its second port to the port sent by the server. At this point, the FTP session has been established and data transfer can proceed. [1][3][4]

A few potential vulnerabilities exist within the FTP protocol itself. The first is that a username and password are the only means provided for client authentication to the server. The client's password is also sent in cleartext to the server, leaving open the possibility of stolen passwords. Additionally, there is no means provided for server authentication to a client. [5] Another potential vulnerability that exists is within the PORT command itself. The PORT command can be used to provide an IP address as well as a port for the server to use to initiate a data channel connection. This provides the client with the ability to direct data connections to any IP address and port, which allows the client to use the server as a type of port scanner, since the server will return information to the client about the connection attempt. [2][6]

Anonymous FTP

The concept of anonymous FTP was created to allow general, unauthenticated access to information stored in archive sites on the Internet (i.e., those which act as repositories of information). The underlying idea is very simple. A site creates an "anonymous" user account, which can be used to acquire limited access to information stored on the anonymous FTP host. The only authentication required is the default username *anonymous* in conjunction with any string to serve as a password. Originally, no string in particular was required. It became commonplace to either use the word *guest* or an email address as an anonymous password. Normally, anonymous FTP servers do not enforce stringent requirements for password string composition. Plausibility checks are usually limited to confirming the password string contains the character '@'. At best, the password entered is usually just written to a log file, and anonymous access proceeds uninhibited. [1][7]

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Several advantages to providing anonymous FTP exist. First, large amounts of data can easily be made available to the general public as long as RFC959-compliant clients are used (some clients do not support all features of FTP, e.g., passive mode data connections). Second, anonymous FTP server administrators do not have to be concerned with protecting and maintaining authentication information such as system usernames and passwords. This helps to prevent compromise of the server host and network since no system usernames and passwords are used during a FTP session, thereby reducing the possibility of stolen or compromised authentication data. Finally, an additional security mechanism that is inherent to most anonymous FTP server implementations, is that of users being confined to a *chroot* jail when logged into the anonymous server host. A *chroot* jail is an area of a filesystem into which a process can be "jailed" by changing its view of where the root of the filesystem is located. [1][8]

Though anonymous FTP is a useful means of transferring files over TCP/IP-based networks by users at large, there are several pitfalls to providing an anonymous FTP server that must be addressed. First, only data directly pertaining to the operation of the anonymous FTP server should exist on the server host. If malicious users do succeed in breaking free of the anonymous *chroot* jail, there should not be any information of use or interest elsewhere on the host machine. Second, it may be necessary to provide anonymous users with the ability to upload data to an anonymous FTP server. However, this can lead to the problem of malicious users using the server host to store and distribute illicit or unauthorized data. This can result in denial of service attacks (consumption of network bandwidth and disk space), legal liabilities (copyright violations, pornography, etc.), and loss of public confidence and credibility. [1][2] Finally, malicious users can utilize the classic "FTP Bounce Attack" (i.e., misusing the FTP PORT command to cause a FTP server to direct data channel connections to any IP address). This can result in the FTP server being used as a port scanner, as a mechanism for bypassing packet filtering devices, or even as a means to circumvent data export restrictions. [6]

Security Basics for Anonymous FTP Servers

There are several basic steps to follow when setting up a secure anonymous FTP server. First, the most recent version of the FTP daemon being used should be obtained to protect against most if not all of the known vulnerabilities. Next, the directories used for anonymous FTP should have strict access restrictions applied. The anonymous FTP root directory and all of its subdirectories should not be owned by or be included in the same group as the anonymous *ftp* account. It is good practice to make these directories owned by the *root* account and be in the *system* group. It is also prudent to only allow *root* write access to the FTP directory hierarchy. Third, the host system's /etc/passwd and /etc/group files should not be used. Duplicate versions of these files should be placed

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

within the FTP directory structure and should only contain those entries explicitly needed for the operation of the anonymous FTP server. If the system uses an /etc/shadow file, then it, too, should be replicated within the FTP directory structure. All passwords within either the /etc/passwd or /etc/shadow file should be replaced with a "*". Finally, allowing users write-access to an anonymous FTP server is generally not recommended. If this functionality is required, strict limitations upon where anonymous users can upload files must be enforced by configuring filesystem permissions appropriately and by any additional access control features of the FTP daemon itself. A separate disk drive or physical disk partition should also be used to prevent a possible Denial of Service attack. [9]

WU-FTPD as a Secure Implementation

In addition to basic security principles, choosing a secure FTP daemon implementation is a vital aspect of providing anonymous FTP as a service. There are a handful of specialized FTP servers that were designed with an eye towards security. One such implementation, WU-FTPD, is widely used by major and minor Internet sites. WU-FTPD provides useful features for anonymous FTP, which are not included in most "out-of-the-box" FTP daemons. The features that are most beneficial from a security perspective include:

- **Robust logging.** The server can be configured to log uploads, downloads, user commands, the number of accesses by certain classes of users (e.g., anonymous users, etc.)

- **User class definitions.** The server provides the ability to create separate classes of users. Each class is defined based upon the account used to log in to the server or upon the host IP address or subnet from which a client connection originates.

- **Granular access control.** Numerous access control rules can be applied separately to each class of user. For example, one class of anonymous users can be allowed to download, upload, overwrite, and rename files, while another class can be limited to downloads only.

- **Guest user *chroot*ed access.** Classes of guest users can be created which require a specific username and password for authentication and which can be *chroot*ed. This provides specific user access to areas of your server which are not accessible by anonymous users. [1]

In addition to these and other features, WU-FTPD is defined as "freeware" or "Open Source Software", which allows for the custom-compilation of the daemon. This can be beneficial since inherently insecure or unneeded features can be compiled out of the software while only those features that are required can be compiled into the software. The following section describes the steps necessary

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

to compile the latest version of WU-FTPD, on a Solaris 8 platform, using customizations designed for security in an anonymous FTP environment.

Compilation of WU-FTPD for Anonymous FTP Security

Before beginning the process of producing a custom-compiled version of WU-FTPD, a clear outline of the required functionality the anonymous FTP server must possess should exist. For the purpose of this paper, the following guidelines will be used:

- **Anonymous-only FTP.** The WU-FTPD daemon will be configured to support only anonymous FTP users. All other users will be denied access to the FTP server.

- **Download only.** The WU-FTPD daemon will be configured to only allow users to download files. Users will not be allowed write access to any portion of the FTP server.

- **No DNS.** For the purpose of adding performance to an aspect of the post-compilation *chroot* configuration, DNS support will not be included. The main impact is that reverse DNS lookups will not be possible. This is a tradeoff for the added benefit of placing the entire WU-FTPD server within a *chroot* jail.

The most recent version of the WU-FTPD daemon, v2.6.2, can be downloaded from one of the FTP sites at http://www.wu-ftpd.org/mirrors.html. The software comes as a compressed tar file, so you will need to have *gzip* or *uncompress* and the *tar* utility in order to extract the source files. Once the tar file has been uncompressed and the files extracted, the pre-compilation configuration can begin.

The WU-FTPD software uses the autoconf method of configuring and creating UNIX makefiles. This is accomplished by running the *configure* script in the base WU-FTPD directory with the appropriate command-line arguments. The script checks for system compatibility, such as available compilers, required system libraries, etc.. It should be noted that the following procedures were successfully executed on a Solaris 8 operating environment, versions 04/01, 08/01, and 10/01.

The following *configure* script arguments should be used for the above guidelines and a few additional customizations:

- **`--prefix=<TEMP_INSTALL_DIR>`** This defines a temporary install directory in order to avoid actually installing the daemon binary into system directories before additional configuration can take place.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

- **`--enable-anononly`** This limits access to anonymous users only.

- **`--enable-paranoid`** This disallows file overwrites and renaming onto an already existing file.

- **`--disable-virtual`** This disables support for virtual FTP servers, an unnecessary feature for an anonymous FTP site.

- **`--disable-dns`** This disables DNS support (required for reason mentioned above).

- **`--disable-private`** This disables special group access files, an unnecessary feature for an anonymous FTP site.

Once the *configure* script has completed successfully, run *make* and then *make install* in the base WU-FTPD directory in order to compile and link the daemon binary, `in.ftpd`. The binary is then ready to be placed into `<TEMP_INSTALL_DIR>/sbin/`.

Installation of WU-FTPD for Anonymous FTP Security

Preceding the installation of WU-FTPD, the host Solaris machine should be properly hardened. Though a detailed analysis and discussion of securely hardening a UNIX host is beyond the scope of this paper, an efficient and easy tool does exist to harden Solaris 8 platforms. YASSP (Yet Another Solaris Security Package) is a freely available Solaris package supporting Solaris 6, 7, and 8. The core package of the YASSP distribution, SECclean, can be downloaded from http://www.yassp.org/download.html. Once downloaded and uncompressed, the package is easily installed with the *pkgadd* command. Some desirable features of SECclean that aid in the installation and configuration of WU-FTPD is the disabling of the inetd daemon and the tuning of the TCP/IP stack for security and performance.

To provide an additional layer of security, the entire WU-FTPD server can be installed to work within the confines of a *chroot* jail. A *chroot* jail is a mechanism by which a process is prevented from having access to any part of the filesystem outside of the jail. For instance, if *daemon_xyz* is *chroot*ed to "/chroot/daemon_xyz", then the contents of the directory will appear as "/" to the daemon and nothing outside of the directory will be accessible to it. A good "HOWTO" guide for building a *chroot* jail in a UNIX environment can be found at http://www.linuxdoc.org/HOWTO/Chroot-BIND-HOWTO.html. [8]

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

The process of creating the WU-FTPD *chroot* jail in a Solaris 8 environment begins with choosing a base directory, <BASE_DIR>. The next step is to replicate the required system directories into the <BASE_DIR>:

```
mkdir -p <BASE_DIR>/usr/lib
mkdir -p <BASE_DIR>/usr/platform/SUNW,Ultra-5_10/lib
mkdir -p <BASE_DIR>/usr/share/lib/zoneinfo/US
mkdir -p <BASE_DIR>/etc
mkdir -p <BASE_DIR>/dev
mkdir -p <BASE_DIR>/home
mkdir -p <BASE_DIR>/sbin
mkdir -p <BASE_DIR>/var/run
```

System libraries and additional files needed by the WU-FTPD binary should then be copied into the new *chroot* jail directory structure:

```
cp /usr/lib/libcrypt_i.so.1 <BASE_DIR>/usr/lib
cp /usr/lib/libsocket.so.1 <BASE_DIR>/usr/lib
cp /usr/lib/ld.so.1 <BASE_DIR>/usr/lib
cp /usr/lib/libnsl.so.1 <BASE_DIR>/usr/lib
cp /usr/lib/libc.so.1 <BASE_DIR>/usr/lib
cp /usr/lib/libgen.so.1 <BASE_DIR>/usr/lib
cp /usr/lib/libdl.so.1 <BASE_DIR>/usr/lib
cp /usr/lib/libmp.so.2 <BASE_DIR>/usr/lib
cp /usr/lib/nss_files.so.1 <BASE_DIR>/usr/lib
cp /usr/platform/SUNW,Ultra-5_10/lib/libc_psr.so.1
<BASE_DIR>/usr/platform/SUNW,Ultra-5_10/lib
cp /usr/share/lib/zoneinfo/US/Central
<BASE_DIR>/usr/share/lib/zoneinfo/US/
```

The next step is to replicate the two required device files, /dev/null and /dev/conslog, into the *chroot* jail. The following simple Korn shell script can be used to accomplish this:

```
#!/bin/ksh
#
# Script for making device files
# $1 --> device file name (i.e. null, conslog, etc.)
# $2 --> path to destination directory (i.e. /somebase/dev)
#
ls -l /devices/pseudo/ | grep $1 |
while read -r line ;
do
     result=`echo $line | awk '{print $10}'`
     device=${result##*:}
     if [[ $device = $1 ]] ; then
          major=`echo $line | awk '{print $5'`
          major=${major%,}
```

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

```
        minor=`echo $line | awk '{print $6}'`
        mknod $2/$device c $major $minor
    fi
done
```

The appropriate /etc system files then need to be replicated. When creating the /etc/passwd and /etc/group files, choose user and group ids, as indicated below, that do not already exist on the system. This will add additional security by having the WU-FTPD child processes for each anonymous user session run under a nonexistent user and group id.

```
cat /etc/netconfig | egrep '^(tcp) ' > \
    <BASE_DIR>/etc/netconfig
cat /etc/project | grep default > \
    <BASE_DIR>/etc/project
echo "ftp-data\t20/tcp\nftp\t\t21/tcp" > \
    <BASE_DIR>/etc/services
echo "ftp:x:<FTP_UID>:<USR_GID>::/home:/bin/false" > \
    <BASE_DIR>/etc/passwd
echo "ftpusr:x:<USR_UID>:<USR_GID>::/home:/bin/false" >> \
    <BASE_DIR>/etc/passwd
echo "ftp::<FTPUSR_GID>:\nftpusr::<USR_GID>:" > \
    <BASE_DIR>/etc/group
echo "/bin/false" > <BASE_DIR>/etc/shells
```

The last step of the installation is to copy the WU-FTPD binary from

```
<TEMP_INSTALL_DIR>/sbin/in.ftpd
```

to

```
<BASE_DIR>/sbin/in.ftpd
```

Configuration of WU-FTPD for Anonymous FTP Security

Now that the base installation is complete, it is time to configure the anonymous WU-FTPD server for additional security. The first step is to create a *ftpaccess* configuration file in `<BASE_DIR>/etc/ftpaccess`. The following is an example *ftpaccess* file, supplied with comments, that can be used for a simple, secure anonymous FTP server:

```
# create anonymous class and restrict access to one IP
class  anonymous1  anonymous  999.999.999.999

# limit the number of failed logon attempts
loginfails  2
```

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

```
# set default timeout values
timeout  data  300
timeout  idle  300
timeout  maxidle  300

# set the hostname to something uninformative
hostname  FTPserver

# set the pre-authentication greeting to be uninformative
greeting  terse

# allow selective downloads
noretrieve  *
allow-retrieve  class=anonymous1  /home/anonymous1

# setup chroot()ed home directory for the anonymous class
anonymous-root  /home/anonymous1  anonymous1

# disallow all uploads
upload  /  *  no

# disallow all other configurable commands
compress  no  anonymous*
tar  no  anonymous*
chmod  no anonymous
delete  no anonymous
overwrite  no anonymous
rename  no anonymous
umask  no  anonymous

# make logging verbose and log to syslog
log  commands  anonymous
log  transfers  anonymous outbound
log  security  anonymous
log  syslog
```

The next step is to create a home directory for each anonymous class within the *chroot* jail. As per the example above:

```
mkdir -p <BASE_DIR>/home/anonymous1
```

An additional step in securely configuring the example anonymous WU-FTPD server is to restrict filesystem permissions within the *chroot* jail. All directories and files should be owned by 'root' with the exception of the home directory of the anonymous user class, which should be in the <USR_GID> group and have group write permission:

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

```
chown -R root:system <BASE_DIR>
chmod -R 700 <BASE_DIR>
chown -R <USR_UID>:<USR_GID> <BASE_DIR>/home/*
chmod -R 770 <BASE_DIR>/home/*
```

Once this is complete, the new WU-FTPD server should be started with the following command:

```
chroot <BASE_DIR>/sbin/in-ftpd -a -l -S -W
```

However, in order to add an automated mechanism to initialize the WU-FTPD daemon on system startup and to allow for automatic restart in case of failure, the following simple Korn shell script can be added to the /etc/rc3.d directory:

```
#!/bin/ksh
while :;
do
     chroot <BASE_DIR>/sbin/in.ftpd -a -l -S -W
     sleep 1
     while [[ -n `pgrep -x in.ftpd` ]] ;
     do
          sleep 1
     done
done
```

Conclusion

Securing an anonymous FTP server can be a challenging yet rewarding task. Anonymous FTP servers have several advantages, most notably that of providing quick, easy, and reliable access to archived information. In order to prevent malicious users from victimizing sites, it is crucial that site administrators choose a secure implementation of FTP server. WU-FTPD can be used to achieve this goal. With its robust logging, class-based access control, and easy customization, WU-FTPD meets and exceeds the basic criteria for deploying a secure, anonymous FTP site.

References

[1] Zwicky, Elizabeth D., Simon Cooper, and D. Brent Chapman. Building Internet Firewalls. Sebastopol: O'Reilly & Associates, Inc., 2000

[2] "Anonymous FTP Abuses" CERT Coordination Center. May 4, 2001.

http://www.cert.org/tech_tips/anonymous_ftp_abuses.html

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

[3] Postel, Jon and Joyce Reynolds. "File Transfer Protocol (FTP)", RFC959. USC/Information Sciences Institute, October 1985.

http://www.faqs.org/rfcs/rfc959.html

[4] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading, MA: Addison-Wesley, 1994

[5] Horowitz, M. and S. Lunt. "FTP Security Extensions", RFC2228. Cygnus Solutions (Horowitz), Bellcore (Lunt), October 1997.

http://www.faqs.org/rfcs/rfc2228.html

[6] "Problems With The FTP PORT Command or Why You Don't Want Just Any PORT in a Storm" CERT Coordination Center. February 12, 1999.

http://www.cert.org/tech_tips/ftp_port_attacks.html

[7] Deutsch, Peter, Alan Emtage, and April N. Marine. "How to Use Anonymous FTP", RFC1635. Bunyip Information Systems (Deutsch and Emtage), NASA NAIC (Marine), May 1994.

http://www.faqs.org/rfcs/rfc1635.html

[8] Wunsch, Scott. "Chroot-BIND HOWTO". December 2001.

http://www.linuxdoc.org/HOWTO/Chroot-BIND-HOWTO.html

[9] "Anonymous FTP Configuration Guidelines" CERT Coordination Center. May 4, 2001.

http://www.cert.org/tech_tips/anonymous_ftp_config.html

Useful Websites

YASSP (Yet Another Solaris Security Package) URL: http://www.yassp.org/

WU-FTPD Development Group URL: http://www.wu-ftpd.org/

WU-FTPD Resource Center URL: http://www.landfield.com/wu-ftpd/

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Last Updated: September 4th, 2015

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Crystal City 2015	Crystal City, VAUS	Sep 08, 2015 - Sep 13, 2015	Live Event
SANS Network Security 2015	Las Vegas, NVUS	Sep 12, 2015 - Sep 21, 2015	Live Event
SANS Seoul 2015	Seoul, KR	Sep 14, 2015 - Sep 19, 2015	Live Event
SANS Tallinn 2015	Tallinn, EE	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Baltimore 2015	Baltimore, MDUS	Sep 21, 2015 - Sep 26, 2015	Live Event
Data Breach Investigation Summit & Training	Dallas, TXUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Perth 2015	Perth, AU	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS ICS Amsterdam 2015	Amsterdam, NL	Sep 22, 2015 - Sep 28, 2015	Live Event
SANS Bangalore 2015	Bangalore, IN	Sep 28, 2015 - Oct 17, 2015	Live Event
SANS Seattle 2015	Seattle, WAUS	Oct 05, 2015 - Oct 10, 2015	Live Event
SANS DFIR Prague 2015	Prague, CZ	Oct 05, 2015 - Oct 17, 2015	Live Event
SOS: SANS October Singapore 2015	Singapore, SG	Oct 12, 2015 - Oct 24, 2015	Live Event
SANS Tysons Corner 2015	Tysons Corner, VAUS	Oct 12, 2015 - Oct 17, 2015	Live Event
SANS Gulf Region 2015	Dubai, AE	Oct 17, 2015 - Oct 29, 2015	Live Event
SANS Tokyo Autumn 2015	Tokyo, JP	Oct 19, 2015 - Oct 31, 2015	Live Event
SANS Cyber Defense San Diego 2015	San Diego, CAUS	Oct 19, 2015 - Oct 24, 2015	Live Event
SANS Sydney 2015	Sydney, AU	Nov 09, 2015 - Nov 21, 2015	Live Event
SANS South Florida 2015	Fort Lauderdale, FLUS	Nov 09, 2015 - Nov 14, 2015	Live Event
SANS London 2015	London, GB	Nov 14, 2015 - Nov 23, 2015	Live Event
Pen Test Hackfest Summit & Training	Alexandria, VAUS	Nov 16, 2015 - Nov 23, 2015	Live Event
SANS Hyderabad 2015	Hyderabad, IN	Nov 24, 2015 - Dec 04, 2015	Live Event
SANS Cape Town 2015	Cape Town, ZA	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS San Francisco 2015	San Francisco, CAUS	Nov 30, 2015 - Dec 05, 2015	Live Event
Security Leadership Summit & Training	Dallas, TXUS	Dec 03, 2015 - Dec 10, 2015	Live Event
SANS Milan 2015	OnlineIT	Sep 07, 2015 - Sep 12, 2015	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced