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 Remote Access on AIX 4.3.3 using OpenSSH

The purpose of this paper is to confirm the existence of a security vulnerability relating to the network access of our AIX servers, demonstrate a viable means to overcome it, and to verify that the solution has truly eliminated the exposure. AIX is a robust Unix operating system, but as with many others, out of the box, it is less than optimally configured to prevent unauthorized users from intercepting communications coming or going via its network connections. I offer insight into the implementation of an applicatio...

Copyright SANS Institute
Author Retains Full Rights

AD

Securing Remote Access on AIX 4.3.3 using OpenSSH

ABSTRACT:

The purpose of this paper is to confirm the existence of a security vulnerability relating to the network access of our AIX servers, demonstrate a viable means to overcome it, and to verify that the solution has truly eliminated the exposure.

AIX is a robust Unix operating system, but as with many others, out of the box, it is less than optimally configured to prevent unauthorized users from intercepting communications coming or going via its network connections. I offer insight into the implementation of an application that can be used to greatly increase the confidentiality, integrity and availability of data flowing to and from a machine running AIX 4.3.3. The application is OpenSSH (Secure Shell).

INTRODUCTION:

As in all too many system environments, security in our enterprise *was* a neglected issue. Insecure telnet and ftp access between servers and workstations is the norm. Security, or more correctly the lack of, is pretty much dictated by the vendors that provide server and application configurations aimed at ease of use and ease of support rather than providing secure access that works with and can be integrated into a customer's unique environment.

System access, by a large number of users and the need to ftp many files among servers using manual and automated scripts has always existed. Mostly the standard TCP tools, telnet and ftp, are used to meet these needs.

Being a relatively small company allowed for security-by-obscurity; that is until several shared applications were implemented jointly between ourselves, and another, larger organization. Merging of enterprises meant more people outside of our culture would have the need to access our systems. This injected new unknowns and many more access points into our systems. Security policies that had previously existed, but were haphazardly implemented, were taken out and dusted off with their contents being given much more attention.

As a test to see how easily data shared between our servers could be intercepted and interpreted, we installed a freeware network sniffer, Ethereal[1] on a number of PCs connected on different subnets throughout our enterprise. The resultant traces proved what we suspected. A great deal of confidential data, including passwords, could be seen in plain text. [2]

While the use of switching devices on our network limits the number of servers that can be sniffed from any one access point within our physical server room, a sniffer connected to the network backbone or somewhere on the WAN,

[1] Ethereal is available at http://www.ethereal.com
[2] A screen shot of the trace is available on page 10.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

external to the physical facility, could more readily expose sensitive data to unauthorized viewers.

The article in Computerworld magazine, Network Sniffers by Alan Joch July 23, 2001, gives a good insight into the types and capabilities of network sniffing software [1]. Some are simple, connected to discrete points near a machine. Others are capable of monitoring an entire enterprise from a central location. Their use as a diagnostic tool is invaluable in resolving communication and network issues, but unfortunately, hackers love them too.

While the number of vulnerabilities that exists for any system is large and the effort required to correct them all can be staggering, "the longest journey starts with the first step". Hence, I visit the subject of securing access to my AIX servers with OpenSSH.

The primary goal of this effort is to enhance the security of the traffic between just our servers, including administrative terminals and workstations. Post installation analysis of the OpenSSH packages and its capabilities will determine to what extent it can be implemented further, if any, within our enterprise.

Why implement OpenSSH - A Top Ten List:

10. Bored intellectuals trying to improve their system skills at other people's expense.
9. Professional corporate espionage.
8. That guy sitting beside me.
7. Makes HIPAA[3], happy.
6. It's my password and you can't have it.
5. Disgruntled former employees.
4. Disgruntled *current* employees.
3. IT'S FREE!
2. Admins looking for ways to justify their existence at review time.
1. Why not?

WHAT IT IS:

OpenSSH is a free version of the SSH protocol suite of network connectivity tools. It provides verification of user identity and secure communication across a network [2].

OpenSSH provides user verification by using public and private key pairs. It utilizes a number of authentication methods and can create secure tunnels through a network path.

TCP network tools such as telnet, rlogin, ftp, and others transmit data across networks unencrypted making clear viewing of the data possible. OpenSSH encrypts all communications, including passwords, between two un-trusted hosts over an insecure network. This greatly decreases the likelihood of eavesdropping and network-level attacks.

[3] Health Insurance Portability and Accountability Act of 1996

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

OpenSSH uses two protocol versions:

- SSH1 uses RSA authentication
- SSH2 adds DSA authentication, additional encryption (3DES, Blowfish, CAST123 or Arcfour), and integrity (hmac-md5, hmac-sha1).[4]

The OpenSSH suite includes the following programs:

ssh	- Provides a basic rlogin/rsh-like program
sshd	- The ssh daemon
ssh-agent	- The authentication agent that stores private keys
ssh-add	- The tool that adds keys to the ssh-agent
sftp	- A ftp-like program
scp	- A file copy program similar to rcp
ssh-keygen	- A key generation tool
sftp-server	- SFTP server subsystem
ssh-keyscan	- Scans multiple hosts, gathering public keys
ssh-keysign	- The helper program for host based authentication

WHERE TO GET IT:

There are a number of sources for the OpenSSH code. Here are 3 of them:

- Bull's Large Open Source Software Archive for AIX[5]
  Contains precompiled AIX shareware and freeware
- OpenSSH, Portable OpenSSH [6]
  Contains portable OpenSSH code download links
- AIX toolbox for Linux applications[7]
  Contains binary RPMs for Open Source packages available for installation on AIX 4.3.3 & AIX 5L.

IMPLEMENTATION:

I chose to use the Linux version. Having limited experience with Linux I thought it would be beneficial to learn more about AIX's compatibility with open software and to familiarize myself with the RPM tool.

While the files are in RPM format, as opposed to AIX's normal .bff format, the process of installing software using RPM turned out to be straight forward and held no big surprises. Both IBM [3] and RPM [4] websites have ample documentation covering the use of the RPM tool to ensure successful installation of the packages. During implementation, configuration, and learning to use the program the man pages from the OpenSSH web site also proved to be invaluable [5].

OpenSSH presents a number of options for its use. There are several ways to authenticate between client and server, including password, host based,

[4] This version supports the use of either RSA or DSA authentication.
[5] http://www.bullfreeware.com/listaix43.html
[6] http://www.openssh.com/portable.html
[7] http://www-1.ibm.com/servers/aix/products/aixos/linux/download.html

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

keyboard-interactive, and public key. There are also two protocols to choose from as was described earlier. For the purposes of testing OpenSSH as stated above, I have chosen to implement it using public key authentication and the SSH2 protocol. Public key authentication simplifies logging in by not requiring that a password be used, yet still provides secure authentication. The SSH2 protocol uses the stronger DSA (Data Signature Algorithm) authentication method and 3DES (Triple Data Encryption Standard).

In addition to the OpenSSH packages, two additional packages will be required. One is PRNG (Pseudo Random Number Generator). It is used in providing random numbers for DSA key generation. The other is SSL (Secure Socket Layer). It provides encryption of the data stream at the TCP/IP transport layer.

INSTALLATION:

RPM package manager will be required to install the required RPM files.

1. Go to the IBM page, AIX toolbox for Linux applications.
2. Click on the link: Download the AIX installp image for the rpm package manager for POWER.
3. Save rpm.rte to the installation images directory: /usr/sys/inst.images
4. Update the table of contents file by running the command:
   inutoc /usr/sys/inst.images
5. Install RPM using the command:
   installp –qacXgd rpm.rte rpm.rte

NOTE: Regarding the following files: The prerequisite RPM, Pseudo Random Number Generator Daemon, can be found on the AIX toolbox for Linux applications page. All OpenSSH and OpenSSL RPMs are accessed by following the AIX Toolbox Cryptographic Content link on the same page.

These packages are prerequisites for the OpenSSH programs:[8]

- openssl Secure Sockets Layer and cryptography libraries
  - openssl-0.9.6e-2.aix4.3.ppc.rpm
- prngd Pseudo Random Number Generator Daemon
  - prngd-0.9.23-3.aix4.3.ppc.rpm

The required packages for OpenSSH are:

- openssh Open Source Secure Shell
  - openssh-3.4p1-5.aix4.3.ppc.rpm
- openssh-clients OpenSSH Secure Shell protocol clients
  - openssh-clients-3.4p1-5.aix4.3.ppc.rpm
- openssh-server OpenSSH Secure Shell protocol server (sshd)
  - openssh-server3.4.p1-5.aix4.3.ppc.rpm

[8] The software listed is current as of the time this paper was written. More recent versions of rpm and OpenSSH are posted frequently. Check for interdependencies prior to installing or upgrading the software.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

6. Down load the required files and install using the RPM command.[9]
   rpm –i –v [file name]

CONFIGURATION:
To aid in understanding the program, here is a listing of important program files, their locations, and a brief description of each.

System Programs[10]

/usr/sbin	sshd
/usr/bin	ssh, ssh-add, ssh-agent, ssh-keygen, ssh-keyscan, scp, sftp

Server Files

/etc/ssh

shosts.equiv	used with .rhost authentication	
sshd_config	server global default configuration	
ssh_config	client global default configuration	
ssh_host_key	host private key	(SSH1)
ssh_host_key.pub	host public key	(SSH1)
ssh_host_dsa_key	host private dsa key	(SSH2)
ssh_host_dsa_key.pub	host public dsa key	(SSH2)
ssh_host_rsa_key	host private rsa key	(SSH2)
ssh_host_rsa_key.pub	host public rsa key	(SSH2)
moduli	Diffie-Hellman groups used for the key exchange methods	

Client Files

$HOME/.ssh

config	client specific configuration	
authorized_keys	remote client public keys	(SSH1)
authorized_keys2	remote client public keys	(SSH2)
id_dsa	client private dsa key	(SSH2)
id_dsa.pub	client public dsa key	(SSH2)
id_rsa	client private rsa key	(SSH2)
id_rsa.pub	client public rsa key	(SSH2)
identity	client private key	(SSH1)
identity.pub	client public key	(SSH1)
known_hosts	public host keys of all know hosts	

[9] The installation of RPM creates a directory path that may be used to store the download files. It is: /usr/opt/freeware/src/packages/RPMS/

[10] See page 3 for a description of the programs.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Configuration Files

/etc/ssh/sshd_config

This file contains the system-wide options for the server. The system defaults are listed as commented (#) lines. These parameters may be changed and the lines uncommented or, alternatively, replicated with new values and added as a group elsewhere in the file. Making a copy of the unmodified file is recommended before altering the original.

Parameters that I chose to change are:

- Login grace time from 600 to 60
- Permit root login from yes to no
- Banner from /some/path to /etc/ssh/ssh_banner

I preferred to shorten the login grace time feeling that 600 seconds (10 minutes) was too long. No direct root logins are allowed. Users should "su" to root so a record of users switching to root can be logged in the /var/adm/sulog file. I added the hostname to the banner file so that it will display some sort of meaningful information. This will give you some feedback on whether or not you have connected to the correct host (this can help diagnose connection problems).

Other parameters that are prime candidates to be considered for addition are: StrictHostKeyChecking, AllowUsers, AllowGroups, DenyUsers, DenyGroups, Ports, ListenAddress, SyslogFacility, and LogLevel [6].

$HOME/.ssh/config

Create this file to enable per user configuration preferences. Parameter settings in it will be honored over those in the globally effective /etc/ssh/config file. File permissions must be set to read/write exclusively[11] for the owner before ssh will use it's contents.

Parameters that I chose to change showing their new value:

- Protocol 2
- StrictHostKeyChecking no
- PreferredAuthentications publickey
- user [*username*] add this non-existing parameter

The protocol parameter mandates the use of SSH2. Setting StrictHostKeyChecking to no, enables the adding of host keys to the $HOME/.ssh/known_hosts file. Adding a user name via the user parameter negates the need to specify a user name when running the ssh command.

[11] chmod 600 /$HOME/.ssh/config

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Creating Keys

Public and private keys are used to authenticate between client and server. A description of the process, as stated in the man pages, follows:

> When the user logs in, the ssh program tells the server which key pair it would like to use for authentication. The server checks if this key is permitted, and if so, sends the user a challenge, a random number, encrypted by the user's public key. The challenge can only be decrypted using the proper private key. The user's client then decrypts the challenge using the private key, proving that he/she knows the private key but without disclosing it to the server.

To generate the keys, the ssh-keygen program is used. The command, **ssh-keygen -t dsa**, will create a public and a private DSA key in the default file locations, $HOME/.ssh/id_dsa.pub and $HOME/.ssh/id_dsa, respectively. The program will also ask if a passphrase is to be used. A phrase up to 30 characters long may be used or it may be empty to indicate no passphrase.

In order for the authentication mechanism to work, the server must know the public key. To make the key known, copy the public key into the remote host's $HOME/.ssh/authorized_keys2 file.

USAGE:

The two main programs, ssh and sftp, work well in replacing their TCP counterparts. Once the associated files are configured, using the two programs is little different than running a telnet or ftp session.

ssh

1. To log in using the user name you declared in the client config file, on the command line, enter: **ssh [*hostname*]**
2. If all goes well you will be presented with the prompt:
   <**Enter passphrase for key '/home/*username*/.ssh/id_dsa'**:>
3. Enter the passphrase that you chose during key generation.
4. A shell is started on the remote machine and you are presented with a command line prompt. All communications between client and server hosts will take place though an encrypted, secure channel.

sftp

1. From the client command line enter: **sftp [*hostname*]**
2. You will be presented with the prompt:
   <**Enter passphrase for key '/home/*username*/.ssh/id_dsa'**:>
3. You are presented with an **sftp>** prompt. Notice that no option to log in as a specific user is given. You are logged in as the user name specified in the client config file.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

ssh-agent

In short, ssh-agent will keep you from having to enter your passphrase. However, the man pages give a more thorough description.

> ssh-agent is a program to hold private keys used for public key authentication (RSA, DSA). The idea is that ssh-agent is started in the beginning of a login session, and all other windows or programs are started as clients to the ssh-agent program. Through use of environment variables the agent can be located and automatically used for authentication when logging in to other machines using ssh.

ssh-add

ssh-add adds your DSA identity to the authentication agent.

An example of starting the ssh-agent, exporting the environment variables, adding the user identity, and logging into the remote hosts:

```
[username@client-host] /home/username $ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-tjk21020/agent.21020; export SSH_AUTH_SOCK;
SSH_AGENT_PID=14838; export SSH_AGENT_PID;
echo Agent pid 14838;

[username@client-host] /home/username $SSH_AUTH_SOCK=/tmp/ssh-tjk21020/agent.21020;
export SSH_AUTH_SOCK;

[username@client-host] /home/username $ssh-add
Enter passphrase for /home/username/.ssh/id_rsa: [ passphrase ]
Identity added: /home/username/.ssh/id_rsa (/home/username/.ssh/id_rsa)
Identity added: /home/username/.ssh/id_dsa (/home/username/.ssh/id_dsa)
Identity added: /home/username/.ssh/identity (username@client-host)

[username@client-host] /home/username $ssh remote-host

*********************************************************************
*
*  Welcome to AIX Version 4.3
*
*********************************************************************

[username@remote-host] /home/username $
```

Creating a Secure Channel

To create a secure channel from the client-host port 2223[12] to the remote-host 23 (the telnet port), use the command: **ssh *remote-host* -L 2223:*remote-host*:23** This keeps a channel open as long as the originating process is alive, meaning the terminal session that it is running on cannot be closed without closing the channel also. A different terminal session(s) will have to be used to utilize the channel.

[12] This is an arbitrary port number, but is higher than 1024 . Ports 1 through 1024 are only usable by root.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

An example of a telnet session to localhost on port 2223 while the secure channel is open resulting in a secure telnet session to remote-host.[13]

```
[username@client-host] /home/username $telnet localhost 2223
Trying...
Connected to loopback.
Escape character is '^]'.

telnet (remote-host)

AIX Version 4
(C) Copyrights by IBM and by others 1982, 1996.
login: username
username's Password:
*******************************************************************
*
*  Welcome to AIX Version 4.3!
*
*******************************************************************
Last unsuccessful login: Tue Apr 16 12:21:56 EDT 2002 on /dev/pts/1 from
0050057889a7.bogus.net
Last login: Mon Jan 13 12:40:54 EST 2003 on /dev/pts/1 from 00080489382c2.bogus.net

[username@remote-host] /home/username $
```

To further enhance running a secure channel the –N flag can be used to only forward ports. No remote commands will or can be run. Additionally the ssh program can be run in the background. This means the terminal session that started the channel may now be used for other purposes also.

ssh remote-host -N -L 2223:remote-host:23 &

OBSERVATIONS:

This version of OpenSSH does not create or append any files that would make it start automatically at system boot time. Manual editing of "rc" files and/or /etc/inittab has to be performed for this to occur. The freeware version from Bull handles these tasks for you [7].

As with other open source applications, an extra level of administration time and effort may be required to implement and maintain this product vs. a commercial version of secure shell.

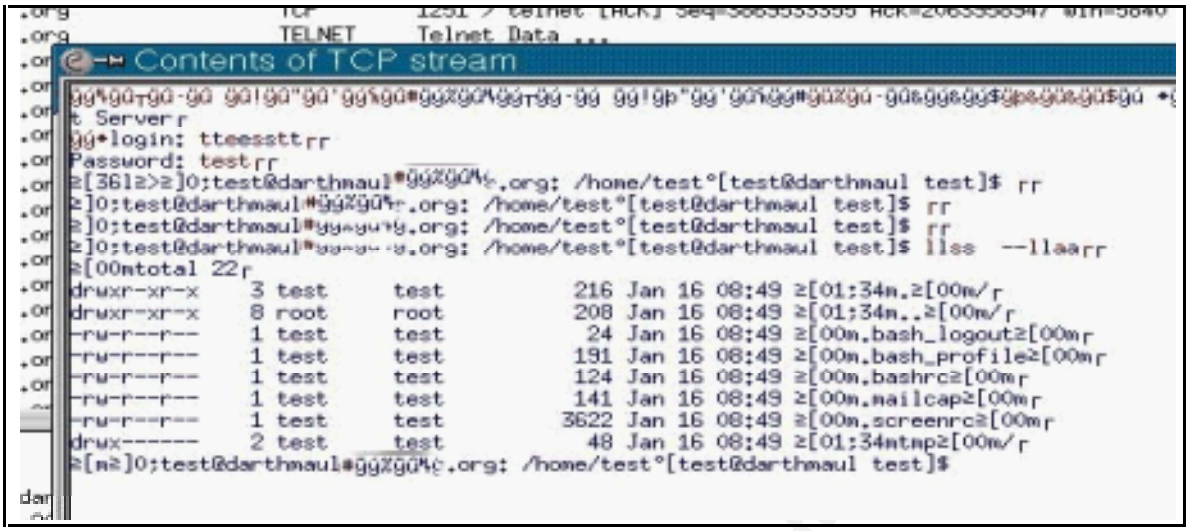
CONCLUSION:

The goal of implementing OpenSSH on our servers has been met. OpenSSH was installed on all AIX (and other vendor's) servers and administrator's workstations. File transfer scripts were converted from ftp to sftp and continue to work normally. Traces of network traffic during client authentication, login, and normal terminal usage displays only characters that are garbled and unintelligible.

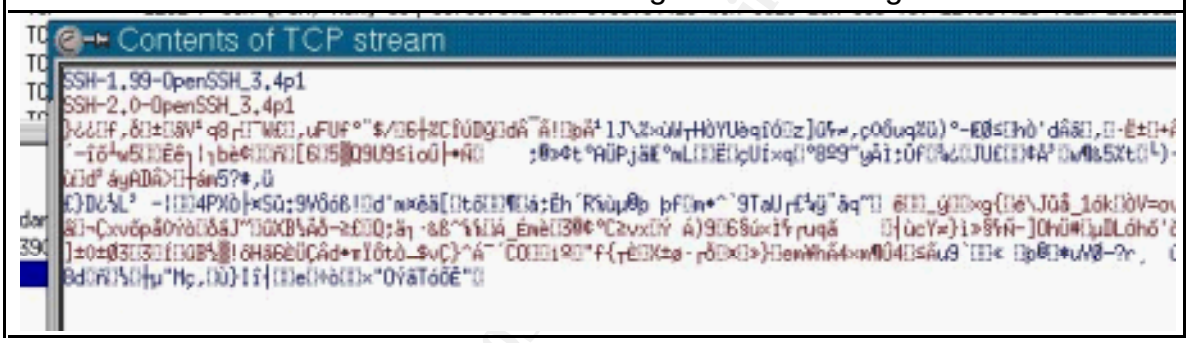
Ethereal trace of a telnet log in and file listing

[13] The login herald is AIX *standard issue* for this demonstration example. Normally it would display a security related message regarding authorized access and usage of the system.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.



Ethereal trace of a ssh log in and file listing



The challenge that lies ahead is to work with our vendors to incorporate secure shell access to their applications across our enterprise. The biggest limiting factor is the ability of their supported terminal emulator packages to support secure shell. Migrating to versions of the emulators that do provide secure shell would be the easiest resolution. Replacing those that do not support secure shell with other products would likely be harder and more costly.

A more desirable scenario, however, would be to implement secure tunneling from our Windows based workstations to our servers at the operating system level. Ideally this will be done transparently to the users and application interfaces. Providing a generic secure tunnel to servers that is transparent to *all* of our application interfaces would seem to be a better solution than addressing each of the interfaces individually.

A brief investigation of commercial tunneling products shows promise in making this task doable. Entunnel[14] uses the open Secure Shell protocol and provides SSH2 data tunneling (port forwarding) services in a Windows environment when connected to an SSH2 server. Other products will have to be investigated in the future.

14 Entunnel is a trademark of VanDyke Software, Inc.
http://www.vandyke.com/products/entunnel/

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

In summary, OpenSSH has enhanced the security of our enterprise and helped to meet our security policy mandates.

© SANS Institute 2003, Author retains full rights

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

REFERENCES:

[1] Joch, Alan, "Network Sniffers" Computerworld magazine 23 Jul. 2001 URL: http://www.nettest.com/pdf/Computerworld2_0701.pdf (14 Jan. 2003)

[2] OpenSSH, "Features", v 1.17. 5 Nov. 2002. URL: http://www.openssh.com/features.html (11 Jan. 2003)

[3] IBM Corp., "AIX toolbox for Linux applications". URL: http://www-1.ibm.com/servers/aix/products/aixos/linux/altlic.html (14 Jan. 2003)

[4] Barnes, Donnie. "RPM at Idle" v 3.0 Red Hat, Inc. 3 Nov, 1999 URL: http://www.rpm.org/RPM-HOWTO (14 Jan. 2003)

[5]OpenSSH, "Manual Pages" v 1.10 Dec. 2002 URL: http://www.openssh.com/manual.html (14 Jan. 2003)

[6] Farazdel, Abbas Gentry, Marc Kerouanton, Bruno and Khor, Chune Keat. "Additional AIX Security Tools on IBM "e"server pSeries, IBM RS/6000, and SP/Cluster". IBM Corp., December 2000. pg 75 & 76

[7] Gresham, Austin H., "Securing AIX5L, Version 5.1 on an RS/6000 E30" Jul. 2001 URL: http://www.giac.org/practical/Austin_Gresham_GCUX.doc (14 Jan. 2003)

© SANS Institute 2003, Author retains full rights

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Last Updated: May 24th, 2013

Upcoming SANS Training

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

SANS Malaysia @ MCMC 2013	Cyberjaya, MY	Jun 03, 2013 - Jun 08, 2013	Live Event
SANS Pen Test Berlin 2013	Berlin, DE	Jun 03, 2013 - Jun 08, 2013	Live Event
Industial Control Systems Security Training - Houston	Houston, TXUS	Jun 10, 2013 - Jun 15, 2013	Live Event
Security Impact of IPv6 Summit 2013	Washington, DCUS	Jun 14, 2013 - Jun 16, 2013	Live Event
SANSFIRE 2013	Washington, DCUS	Jun 14, 2013 - Jun 22, 2013	Live Event
SANS Canberra 2013	Canberra, AU	Jul 01, 2013 - Jul 13, 2013	Live Event
Digital Forensics & Incident Response Summit 2013	Austin, TXUS	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS London Summer 2013	London, GB	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS Rocky Mountain 2013	Denver, COUS	Jul 14, 2013 - Jul 20, 2013	Live Event
SANS Mumbai 2013	Mumbai, IN	Jul 22, 2013 - Jul 27, 2013	Live Event
SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification	Washington, DCUS	Jul 22, 2013 - Jul 26, 2013	Live Event
SANS San Francisco 2013	San Francisco, CAUS	Jul 29, 2013 - Aug 03, 2013	Live Event
SANS SEC 560: Network Penetration Testing @ Bangalore 2013	Bangalore, IN	Aug 05, 2013 - Aug 10, 2013	Live Event
SANS Boston 2013	Boston, MAUS	Aug 05, 2013 - Aug 10, 2013	Live Event
Critical Security Controls Summit	Washington, DCUS	Aug 12, 2013 - Aug 18, 2013	Live Event
Industrial Control Systems Security Training - DC	Washington, DCUS	Aug 12, 2013 - Aug 16, 2013	Live Event
SANS Thailand 2013	Bangkok, TH	Aug 19, 2013 - Aug 31, 2013	Live Event
SANS Virginia Beach 2013	Virginia Beach, VAUS	Aug 19, 2013 - Aug 30, 2013	Live Event
Mobile Device Security Summit 2013	OnlineCAUS	May 30, 2013 - Jun 06, 2013	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced