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.

Perimeter Defense-in-Depth: Using Reverse Proxies and other tools to protect our internal assets

The use of a reverse proxy server to provide defense-in-depth security will be discussed in this paper as a practical security solution for protecting our internal assets. There are several tools available that add obscurity to this process. We had a need to disguise our IBM mainframe yet allow access to personal resources. This paper includes a discussion of some of the methods that were successful and some that did not work. With limited resources, we can still provide additional security in our environments.

Copyright SANS Institute
Author Retains Full Rights

AD
Symantec™
Endpoint Protection 12
The next generation of reputation-based security
Download the beta
Symantec.
Confidence in a connected world.

Perimeter Defense-in-Depth: Using Reverse Proxies and other tools to protect our internal assets

Lynda L. Morrison

Feb. 18, 2002

Introduction

The use of a reverse proxy server to provide defense-in-depth security will be discussed in this paper as a practical security solution for protecting our internal assets. There are several tools available that add obscurity to this process. We had a need to disguise our IBM mainframe yet allow access to personal resources. This paper includes a discussion of some of the methods that were successful and some that did not work. With limited resources, we can still provide additional security in our environments.

The Paradigm Shift

As greater demands are placed on our businesses to provide information and to access personal records via the Internet, our internal networks are also experiencing demands. The public is demanding access to collections of data and processes that have previously only been accessible by computer programmers, database specialists, and/or business analysts. In many cases, these processes have been handled by large mainframe computers, which run programs nightly to update user data. In the past decade we have watched our computer centers advance to dynamic databases, for which internal personnel have provided the updates. Now, John Doe wants to be able to access his own information, check if it is correct, update his own information, pay his bills, apply for credit, send in his resume, check his retirement benefits, and basically manage all aspects of his life on line. In order to allow this to happen, businesses are being pushed rapidly into the next paradigm shift. This shift to instantaneous access is forcing our businesses to put their assets "on the line" – the Internet line.

One of these recent "rapid deployments" has been fueled by the IBM mainframe world's newest application, CICS Transaction Server, which takes a CICS transaction window and converts it to an html page ready to be manipulated and presented to the world via the World Wide Web. This is the kind of progress that managers love, since they are able to take the processes that have been developed over the past twenty years and quickly deploy them to the internet world with a brand new look. Behind the scenes, it uses the same CICS Cobol programming. IBM provides another product called the IBM HTTP Server. It is an Apache based server that runs on a variety of platforms including AIX, Linux, Netware, Solaris and Windows 2000. They also have a middle tier product called WebSphere which provides an environment for customized application development of these web sites. This application server environment, running Apache server and using Java, provides defense in depth between the World Wide Web and the mainframe; this is IBM's own middle-tier.

The CICS Transaction Server will run without the aid of IBM's middle tier server, but Transaction Server allows little customization. At our location, we do not have the

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

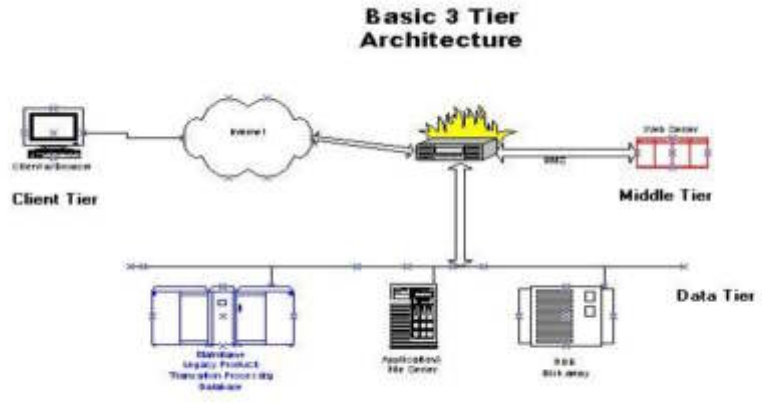
additional WebSphere products. The question that arose for us during this development process was "How do we allow access to our mainframe computer without compromising our security?" There is a misconception in the mainframe world that these large main frame computers are invincible. These machines and their operating systems have not been exposed to the Internet in the same manner that Microsoft has exposed their operating systems. In doing some port maps of the "mainframe" it appears that there are many open portals. In opening one of these portals to the outside world, we could potentially be exposing many portals to the outside world. Therefore, it is our job as "Security Specialists" to find a way to protect them from the big WWW.

How do we begin to do this? Defense-in-depth is the first step in this process. Somehow we must hide this main frame computer behind some layers of security that hopefully will make a less attractive target.

The Firewall

The first line of defense is of course the firewall. The firewall "opens" a portal that allows the world into our internal network. Now, how far into our network do we want to allow the world? We have a large mainframe computer with massive data archives that contain in depth information about our customers in multiple databases. We also have a variety of other systems besides the mainframe to protect. Like other businesses that have come into the information age, we also have a large variety of servers, databases, personal workstations, network equipment, telephone equipment, and other miscellaneous devices that reach far beyond just the mainframe! If we allow access to our internal network for the mainframe, we are also allowing potential access to our many other resources and assets.

Ideally, we would like to limit Internet access only to a DMZ zone, which is a network connected to the firewall, but isolated from the internal network. From a security prospective, we do not want to put our mainframe computer on our DMZ. The following is an example of a basic 3 tier architecture. The idea is to separate the presentation layer from the application and data logic.



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

The Firewall is implemented so that the Internet users and the internal network users can talk to the servers on the DMZ, but the Internet users cannot talk to the internal network. The servers on the DMZ may be mail forwarding servers, web servers, or web proxying servers. Often these servers will access the internal servers for the client browser, giving the added "defense-in-depth". In addition, the Firewall may implement NAT or Network Address Translation, which hides the addresses of the internal servers from the WWW by mapping each server to a different address on the internal network.

The Middle Tier Servers

Our next decision involved how to implement a presentation layer on the DMZ for the mainframe web server. When I needed a quick proxy server, I had installed an apache web proxy onto a Windows 2000 machine and was able to proxy a limited number of users through it to a test system inside another firewall. A regular proxy server's function is to pass a group of user requests from a client machines through a single server to the requested URL or web server. The URL may vary, but for a firewall implementation, it makes it possible to set up a single rule for one IP address to pass through the firewall instead of all of the users requesting access to the URL or web server.

Why couldn't this be done in reverse? The term – Reverse Proxy – is abundant in the research on perimeter defense. That means that instead of the users being set up to go through one proxy machine, the web site is set up to be proxied for the users. Additional research on this subject was definitely in order.

A middle tier is often used for initial http requests. Sometimes it processes requests and passes those requests to an application server for further processing. In this case, the reverse proxy is used to manage the request by passing the request through to another server. Any exploits that may include malformed URL strings will be stopped at the proxy, as they will not be passed through. Also, the only computer program actually communicating with the main frame computer is the proxy program on the reverse proxy server.

Apache Web Server

Since an Apache binary running on a Windows 2000 machine worked as a proxy to the mainframe, perhaps it would work as a proxy for the mainframe. Additional research and testing was needed to determine whether or not this same kind of Apache application could be used as a reverse proxy. A hardened Microsoft Window's 2000 server was loaded with the Apache server application, including the mod_proxy module. It was configured to run as a service, and configured to use proxy port 80. It was tested on an internal server, using the ProxyPass directive to proxy an entire site from one machine to another - which worked. I was hopeful that this method might be used to proxy the Mainframe server.

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

On the IBM mainframe side, they set up a test region and converted their legacy CICS forms to their new web look. A URL could be typed into our browser that was something like

http://10.10.10.10:3000/cics/firstdirectory/seconddirectory/program

and several dynamic pages were generated, including a logon screen. To proxy this site, the following statements were added to the Apache configuration, the httpd.conf file. The proxy module had to be made available, and it had to allow access through port 3000:

ProxyPass / http://10.10.10.10:3000/cics/firstdirectory/seconddirectory/program
ProxyPassReverse / http://10.10.10.10:3000/cics/firstdirectory/seconddirectory/program

The programmers needed to use "relative addressing" in their programs in order for the true location to be hidden. At one point, the IP address of the mainframe application actually came back to the browser request, even though it had been directed through the reverse proxy server. It appeared that the proxy was only acting as a redirect to the web site. Hard coded IP addresses had been inadvertently used in the html code. This problem became more evident when we actually began using a DNS that pointed to the reverse proxy server. Using the relative addressing allowed the internal server to virtually disappear behind the proxy address.

Next, the programmers wanted to test with two different "regions" – one they would call test and one they would call development. For this, they set up two different service ports on the mainframe – such as 3000 and 3001. The following changes were made to the httpd.conf file:

ProxyPass /td/ http://10.10.10.10:3000/cics/firstdirectory/seconddirectory/program
ProxyPass /tp/ http://10.10.10.10:3001/cics/firstdirectory/seconddirectory/program

ProxyPassReverse /td/ http://10.10.10.10:3000/cics/firstdirectory/seconddirectory/program
ProxyPassReverse /tp/ http://10.10.10.10:3001/cics/firstdirectory/seconddirectory/program

This did not work. Other suggestions were found indicating either the need to set up two different virtual hosts, or use the mod_rewrite module to change the directory structure of the URL. The mod_rewrite seemed the better alternative. After including the mod_rewrite module in the Apache build, the following directives were added:

RewriteEngine On
RewriteLog logs/rewrite.log # this is a very useful tool
RewriteLogLevel 9 # this is good for troubleshooting – later set it to 3 or less
ProxyRequests On
AllowCONNECT 3000

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

RewriteRule / http://10.10.10.10:3000/cics/firstdirectory/seconddirectory/program

This essentially redirected the browser to the new URL. It also needed to return through the proxy. So, to the end of this command was added the following flag: [P]. The command looked like this:

RewriteRule / http://10.10.10.10:3000/cics/firstdirectory/seconddirectory/program [P]

This forced the rewrite rule to send it on to the proxy module and so that the request is also proxied.

This method worked for one site, but what about two different sites? After additional testing, and discovery, the directive was changed similar to the one above but with a little more mapping information:

RewriteRule ^/td/(.*) http://w.x.y.z/td:3000 [P] and
RewriteRule ^/tp/(.*) http://w.x.y.z/tp:3001 [P]

The rewrite log generated the following for the request http://10.10.3.10/td/ :

10.10.3.15 - - [18/Feb/2002:11:26:16 -0800] [10.10.3.10/sid#515888][rid#5c7a00/initial] (2) init rewrite engine with requested uri /td/
10.10.3.15 - - [18/Feb/2002:11:26:16 -0800] [10.10.3.10/sid#515888][rid#5c7a00/initial] (3) applying pattern '^/td/(.*)' to uri '/td/'
10.10.3.15 - - [18/Feb/2002:11:26:16 -0800] [10.10.3.10/sid#515888][rid#5c7a00/initial] (2) rewrite /td/ -> http://10.10.10.10:3000/
10.10.3.15 - - [18/Feb/2002:11:26:16 -0800] [10.10.3.10/sid#515888][rid#5c7a00/initial] (2) forcing proxy-throughput with http://10.10.10.10:3000/
10.10.3.15- - [18/Feb/2002:11:26:16 -0800] [10.10.3.10/sid#515888][rid#5c7a00/initial] (1) go-ahead with proxy request proxy:http://10.10.10.10:3000/ [OK]

Another problem discovered in this process was that the mainframe server had not been set up with a default home page, which Apache looks for as index.html. Since the program that was running served up dynamic content, the URL page never changes, but the content does. Apache was specifically looking for index.html – and never found it. It was determined that the mainframe application had the capability to set up a home page using the index.html. They were able to set that up so that our programmers were able to insert another program that contained only a link to the correct program page that needed to run.

SSL

To add another twist to the equation, these transactions needed to be transmitted over a secure socket layer. Initially, our IBM systems folks reported that they would have SSL running for us just shortly before we would go live with our project. Their initial SSL offering was with 56 bit encryption, which was unacceptable. They went back to the

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

drawing board and discovered that with some patches, and some additional micro-code, they could have 168 bit encryption installed. They set up a test area for us with the 56 bit encryption and continued to work on what was needed for the more advanced encryption.

The mainframe site now needed to be proxied through the Apache Server using the SSL connection. Since Apache could be used as a proxy to a secure site, using port 443, it was necessary to make some changes to find out of it would work using a different port number. After configuring the proxy to listen on a new port, it did work. Therefore, it seemed logical that it should be able to do the reverse proxy, by using the rewrite rule and passing it to the proxy module. The commands used were as follows:

```
RewriteCond %{SERVER_PORT}      ^443$
RewriteRule ^/(.*)              https://10.10.10.10:3000/$1 [P]
```

The rewrite log shows the following:

```
init rewrite engine with requested uri /
10.10.3.15 - - [18/Feb/2002:14:03 -0800] [10.10.3.10//sid#515888][rid#5c79c0/initial] (3) applying pattern '^/(.*)' to uri '/'
10.10.3.15 - - [18/Feb/2002:14:14:03 -0800] [10.10.3.10//sid#515888][rid#5c79c0/initial] (4) RewriteCond: input='443' pattern='^443$' => matched
10.10.3.15 - - [18/Feb/2002:14:14:03 -0800] [10.10.3.10//sid#515888][rid#5c79c0/initial] (2) rewrite / -> https://10.10.10.10:3000/
10.10.3.15 - - [18/Feb/2002:14:14:03 -0800] [10.10.3.10//sid#515888][rid#5c79c0/initial] (2) forcing proxy-throughput with https://10.10.10.10:3000/
10.10.3.15 - - [18/Feb/2002:14:14:03 -0800] [10.10.3.10//sid#515888][rid#5c79c0/initial] (1) go-ahead with proxy request proxy:https://10.10.10.10:3000/ [OK]
```

The access log shows

```
10.10.3.15 - - [18/Feb/2002:14:14:03 -0800] "€L__" 403 -
```

Rewrite rules can become very complex. They have the ability to utilize the power of regular expressions. It is possible to use a rewrite condition and compare in incoming URL string to a regular expression. There is a quick introduction to regular expressions at http://www.boutell.com/wusage/7.1/patterns.html. An example that he uses is url /oldpath* /newpath\1 where the * is used like a wild card.

I attempted various other combinations of the above command and continued to research whether or not this would work. The only successful reference to this approach indicated that the session had to be decrypted and re-encrypted at the proxy server. Richard Goerwitz outlined an application for secure socket layer using just such an approach at Brown University. They describe their solution as a reverse or pass-through proxy and use a Kerberos key server for authentication.

CYGWIN

The Apache environment used was on a Windows 2000 machine with the pre-compiled Apache binary, Version 1.3. Other suggestions hinged around the binary installation for

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

Apache, and proposed a new compilation of Apache with the modules that were needed. In order to compile the Unix code in a Windows 2000 environment, some kind of a Unix shell was needed. The Cygwin environment was suggested as a robust and extensible Unix emulation layer program.

The Cygwin program is a Windows 32 bit Dynamic-Linked Library (DLL) which provides a UNIX environment for the porting of UNIX applications. It was first developed by Cygnus Solutions in 1995. Cygnus chose to make Cygwin available under the terms of the GNU public license. The original intent was to compile Apache in the Cygwin environment and to test the differences between the binary installation and a fresh compile with only the needed modules.

The Cygwin environment uses a BASH shell. Its development environment includes all of the tools necessary to compile the latest Apache version. It has all of the GNU tools and libraries, including the GNU C/C++ compiler, assembler, linker and debugger. Several additional software packages have already been successfully ported to the Win32 environment. Apache is one of those ports. Ssh, PERL5, GNU inetutils and other utilities have also been ported to the Win32 environment.

Apache is configured and installed in the same way that it is on any Unix server. Apache running in the Cygwin environment is considered a better transition from Unix systems to Windows systems for HTTP services. There is a detailed document on how to compile Apache in a Cygwin environment from apache.org.

At this point, I needed a proof of concept. Additional research indicated that TCPPROXY or SQUID were workable as simple passthrough proxies that wouldn't be cluttered by Apache's complexity. SQUID is available with the download from http://www.cygwin.com and is a full featured Web Proxy Cache. It is also a free open-source product. BSDProxy is another freely available TCP Proxy. TCPPROXY is the one that I choose to use for proof of concept. It has less complexity in that it truly was just a passthrough or transparent proxy. It is from http://www.quietsche-entchen.de/software/tcpproxy.html. All of these TCP proxies claim to proxy, or pass through, any TCP protocol.

TCPPROXY

TCPPROXY was easily downloaded and compiled in the Cygwin environment. The only file missing for the compile was a wait.h file which needs to be in the /usr/include directory. It should just be a one line file, which says #include <sys/wait.h>.

TCPPROXY is a transparent proxy server that provides a passthrough for TCP protocols. It basically forwards incoming connections from one machine to another machine.

The configuration file includes the following:

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

```
timeout 300 # this can be used for tuning
port 443  # port on which I accept connections

interface 10.10.3.10  # my physical interface
server nameofinside-server:3000
writefile /somedirectory/proxylog          # my log file for server/client communication
```

Since a DNS connection is a potential security risk, a host table for name resolution for nameofinside-server was used. The outside DNS entry for the mainframe points to the proxy server; the proxy server points to the inside address.

That's it. It's clean; it's simple. It works. Note that a log file must be set up within the configuration or it will not log communications.

In a Windows environment, the process can be run as a service. Cygwin has a utility called cygrunsrv which is a program that allows the administrator to install or remove a service from the Windows' system registry, or to start and stop a previously defined service. The Cygwin User's Guide is an excellent resource about how Cygwin works with the Window's environment.

In a production environment, care should be taken to harden the Windows system prior to installing Cygwin. The SANS Reading Room has some great documents on how to go about doing just that. See http://rr.sans.org/win2000/standalone.php.

After the development was complete, and ready to be placed into production, there were a few more things that need to be addressed. The UNIX environment itself needs to be hardened. This hardening is dependent on what is running in the environment. In general, it is advisable to remove all development tools from the Cygwin environment. The passwd file is used for authentication, which is exported from the Windows environment using the mkpasswd command. The mkpasswd command has several flags that allow you to control which accounts are included in the file. For example, mkpasswd –l > /etc/passwd will export all of the local accounts to the passwd file. Once this is exported, it is static, but it can be edited. If accounts are needed, or groups, for specific access rights, it is necessary to run the mkpasswd directive again. If the 'ntsec' feature (nt security) is switched on, Unix style file permissions are used for security permissions. Environmental variables can be added or removed from the root directory file called cygwin.bat, which runs when the Cygwin shell start up. Ssh is one of the tools available when Cygwin is downloaded. It is wise to download it and use it. It is the most current port of OPEN-SSH and it includes the sftp daemon. It installs easily as a service.

SUMMARY

Defense-in-depth involves using a plethora of layers. Our newest paradigm shift to instant access is forcing us to move quickly. We can do this and still insist on providing

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

protection for our internal assets. We need to take some time to evaluate how we can improve our security through obscurity. At each layer, we must also take time to insist that it be as secure as possible.

The firewall is our first layer of defense; the proxy may be the second; using NAT, a third; obscuring the type of system adds another layer of defense. By running Cygwin or a Unix shell on a Windows 2000 machine, both of which have been hardened for maximum security, we have made it a little more difficult to get information about our system. It is no longer an easy target. It will take a little more effort on the part of a hacker to determine what is actually going on with the machine. We still need to be aware of possible buffer overflows and patches that may need to be applied to prevent such overflows. We need to be sure that we have logging enabled and that we monitor our logs. We need to take the time to watch our systems, becoming familiar with their operations so that when something out of the ordinary occurs we are able to recognize that it has occurred. There are many ways to create obscurity – this is just one example.

One of the drawbacks of using a proxy server is that it could become a single use machine. It is possible to configure virtual hosts on one machine, for most proxy servers. That may be a way to allow multiple accesses through one machine. In that way, each URL has a unique IP address, or DNS entry, associated with it, and a single destination to continue on its journey.

References:

Anderson, Todd, "Basic Steps to Hardening a Standalone Windows 2000 Installation", March 21, 2001, http://rr.sans.org/win2000/standalone.php

"Apache HTTP Server Versions 1.3, Apache Module mod_proxy"
http://httpd.apache.org/docs/mod/mod_proxy.html

"Apache HTTP Server Versions 1.3, Module mod_rewrite URL Rewriting Engine"
http://httpd.apache.org/docs/mod/mod_proxy.html

"Apacheweek", Issue 280, 1st February 1, 2002
http://www.apacheweek.com/issues/02-02-01

Bailey, Dave, "BSDProxy - A kevent-driven TCP Proxy for FreeBSD", April, 2001
http://sydney.daveb.net/bsdproxy

Bartley, Richard, "Hacking the DMZ" ,2001,
http://www.xinetica.com/tech_explained/general/hacking_dmz/wp_hacking_dmz.htm

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

Berk, Vincent; Bates, Marion, "Modified Reverse Proxy Website Vulnerability Test Results", September 10, 2001, http://www.ists.dartmouth.edu/IRIA/projects/jeanne/labtest.pdf

Engelschal, Ralf S, "Apache 1.3 URL Rewriting Guide", December, 1997, http://httpd.apache.org/docs/misc/rewriteguide.html

Engelschall, Ralf; Reiber, Christian, "URL Manipulation with Apache", 1996, http://www.heise.de/ix/artikel/E/1996/12/149/

Ghaffar, Atif, "Using Apache ProxyPass to access servers behind a Masquarading host", March, 2000, http://www.linuxfocus.org/English/March2000/article147.html

Goerwitz, Richard "Pass-Through Proxying as a Solution to the Off-Site Web-Access Problem", http://www.stg.brown.edu/pub/proxydoc/report.shtml

"IBM WebSphere Application Server", February, 2002, http://www-3.ibm.com/software/webservers/appserv

"IBM CICS Transaction Server, Version 2.2", February, 2002, http://www-3.ibm.com/software/ts/cics/v2/index.html

Kuenz, Don, "Using Apache as a Proxy Server", May 19, 2000, Inside the Internet, http://www.zdnet.com/devhead/stories/articles/0,4413,2573047,00.html

Noer, Geoffrey J, "Cygwin: A Free Win32 Porting Layer for UNIX Applications", August 4, 1999, http://sources.redhat.com/cygwin/usenix-98/cygwin.html

Repa, Jessica, "eBUSINESS SYSTEM REQUIREMENTS" http://www.bizforum.org/whitepapers/webridge.htm

"TCPPROXY(1)", http://www.quietsche-entchen.de/manpages/tcpproxy-1.html

"Using Apache with Cygwin", Apache HTTP Server, http://httpd.apache.org/docs/cygwin.html

Vinschen, Corinna; Faylor, Christopher; Delorie, DJ; Humblet, Pierre; Noer, Geoffrey, "Cygwin User's Guide" , 2001 http://www.cygwin.com/cygwin-ug-net/cygwin-ug-net.html

"Wusage 7.1 Manual, Patterns and Regular Expressions", 2000, Boutell.Com, Inc., http://www.boutell.com/wusage/7.1/patterns.html

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

Last Updated: May 26th, 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**