Global Information Assurance Certification Paper

Centralized UNIX System Monitoring Using SSH and MRTG

Aaron J. Wilson
v1.4b Practical for GIAC GSEC Certification Attempt
September 07, 2002

Abstract

There are many system and network monitoring utilities available today. Most of these products rely on the Simple Network Management Protocol (SNMP) to deliver the statistics to a centralized network management station (NMS). I was able to find very few documents that suggest using secure copy (scp) with Public Key authentication from the Secure Shell (SSH) suite as a more secure alternative to SNMP. **[1]** What documentation I was able to locate did not exactly discuss how to accomplish the task end-to-end. This paper discusses the example of an OpenBSD web server that needs to be monitored by an OpenBSD NMS using the Multi Router Traffic Grapher (MRTG). MRTG is a time-tested and flexible monitoring tool. **[2]** The procedures in this paper could easily be adapted to gather other system resources and to support other operating systems and other monitoring tools.

SNMP Versus SSH

SNMP has some security features such as community strings. According to snmp.com, SNMPv3 will even support encryption and data integrity checks. **[3]** This is good news, but it will probably take some time for vendors to support SNMPv3. It is likely that when SNMPv3 is supported it will include backwards compatibility for SNMPv1 and SNMPv2 for their popularity. This is ideal from an integration standpoint, but can be challenging to secure since the legacy protocol support will probably be enabled by default, potentially bringing all of their insecurities with them. It was discovered by the Oulu University Secure Programming Group that SNMPv1 mishandles request and trap messages that "may allow unauthorized privileged access, denial-of-service attacks, or cause unstable behavior". **[4]** Similarly, there is a history of security problems associated with SNMP daemons. Many of these are vendor-specific and can lead to unauthorized remote root access, such as Sun Microsystems' buffer overflow in mibiisa and the format string vulnerability in snmpdx discovered by Entercept's Ricochet Team. **[5]**

SSH has had its own share of security problems, especially in late 2002, such as the vulnerability found by IIS with the OpenSSH "challenge-response" authentication mechanism. **[7]** There has even been a trojaned OpenSSH distribution! **[8]** Even in light of these troubles, the public key authentication and

strong encryption make SSH the preferred choice for high security installations.

The primary functions of an enterprise NMS are to:

- Monitor system health (using SNMP read)
- Collect historical trends and statistics (using SNMP read)
- Maintain and configure systems (using SNMP write)
- Serve as an alerting station for failures and other alarm conditions (using SNMP traps)

There are many (usually very expensive) commercial NMS systems that perform all of these tasks well with SNMP. This paper does not advocate the carte blanch replacement of SNMP with SSH on your network. Rather, it proposes that several specific read-only functions of a common NMS can be achieved using SSH. Specifically, this paper describes how an SSH-equipped NMS can:

- Monitor system health
- Collect historical trends and statistics

The goal is to disable SNMP on your monitored systems, not necessarily to protect the monitoring statistics. Some organizations utilize SSH with public key authentication for system administration and to deliver log files and other data to a centralized server. Why not take advantage of the architecture and use SSH to deliver monitoring data as well?

As with everything in information technology, higher security comes with the price of less functionality. The reader will need to decide which protocol works best on their network. If you are using 1000+ devices on a metropolitan area network, centralized configuration management is an obvious requirement and will most likely require SNMP. Could SSH maintain and configure systems, and provide alerting? It is certainly possible, but that exercise will be left to the reader.

If your organization isn't collecting any historical performance data, it should be seriously considered. Monitoring system statistics can prevent outages due to high utilization and help forecast growth. Performance monitoring and trending can even help identify security-related events such as a distributed denial of service attack or unauthorized use of systems. What is that 6Mbps network spike between 3AM and 4AM all about?

Overall Game Plan

Virtually any statistic that can be gathered via a non-interactive query on the command line can be collected and reported by MRTG. For this example, we want to know the historical NIC utilization of our OpenBSD web server. Any experienced UNIX engineer knows this information can easily be obtained using

the netstat command. We will be using Bourne Shell as the primary scripting language. The same results could be obtained using Perl or another language, but Bourne Shell is still the most widely supported scripting language by default.

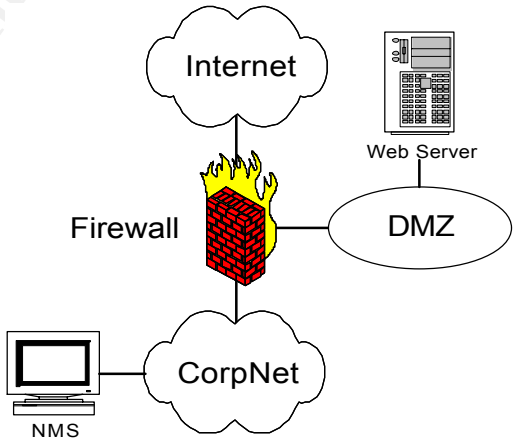
These are the main steps to achieve the goal:

1) Set up SSH public key authentication trust in the appropriate direction.
2) Create a "Host Script" that runs on the monitored system. This script will build a second script (the "NMS Script") every five minutes via cron. The Host Script will collect system data and write it to a file in script format. In other words, the output of the Host Script is actually the NMS script.
3) Configure the NMS to pull the Host Script from the web server every five minutes via cron.
4) Install and configure MRTG on the NMS and schedule the collection process every five minutes via cron. This will retrieve and process the statistics from the NMS script that was pulled from the web server.
5) Install and configure Apache on the NMS to serve the MRTG web pages.

About SSH Public Key Authentication Trust Relationships

SSH has the ability to create trust relationships between systems using RSA or DSA public keys for SSH protocol version 2. This means that a user could log in from the trusted host to the trusting host using public key authentication instead of using password authentication. This is needed so that scp can operate on the trusted system in a non-interactive (scripted) mode without requiring a password.

Consider the simple architecture of an organization with a permanent Internet connection, a demilitarized zone (DMZ) with a web server, and an internal network with a NMS as pictured below.



Trust relationships should be made from least trusted to most trusted systems. Using the example company above, this means that our web server can trust the NMS but the NMS should not trust the web server. In other words, the trusted system in this example is the NMS. The trusting system is the web server. Direction of trust is critical to proper security. It is possible that our web server could be breached when the next remote root exploit for Apache is released. If the trust relationship was from the web server to the NMS, the attacker could simply SSH to our NMS using public key authentication. Then not only was the DMZ breached, but the internal corporate network as well! We will be pulling (not pushing) data from the web server to the NMS.

Step One – Setting up SSH Public Key Authentication

We will be building the SSH trust such that the web server is the trusting system and the NMS is the trusted system. In other areas of the network, it may make more sense to push the data from the monitored device to the NMS (i.e. automated authentication to the firewall is probably NOT a good idea).

Web Server Procedures Part I

1) Log in to the web server as a non-root user (You do use "sudo" for accountability don't you?).
2) Use sudo to create a non-root "mrtg" user and create a good strong password.

```
nmsbox:home {16} sudo useradd –m -c "mrtg stats" –s /bin/sh -d
/home/mrtg mrtg
nmsbox:home {17} sudo passwd mrtg
Changing local password for mrtg.
New password:
Retype new password:
```

3) If the /home/mrtg/.ssh directory doesn't exist, create it and confirm proper permissions (protect your private key).

```
nmsbox:home {18}  mkdir /home/mrtg/.ssh
nmsbox:home {20}  chmod 700 /home/mrtg/.ssh
```

NMS Procedures

4) Log in to the NMS as a non-root user.
5) Create a non-root "mrtg" user and create a password.

```
nmsbox:home {16} sudo useradd –m -c "mrtg stats" –s /bin/sh -d /home/mrtg
mrtg
nmsbox:home {17} sudo passwd mrtg
Changing local password for mrtg.
New password:
Retype new password:
```

6) After entering an appropriate strong password, substitute to the mrtg user.

```
nmsbox:home {18} su - mrtg
Password:
$
```

7) Make a /home/mrtg/.ssh directory, then generate a DSA public/private key pair.

```
$ mkdir /home/mrtg/.ssh
$ cd /home/mrtg/.ssh
$ ssh-keygen –b 1024 –t dsa –N "" –f /home/mrtg/.ssh/id_dsa
Generating public/private dsa key pair.
Your identification has been saved in /home/mrtg/.ssh/id_dsa.
Your public key has been saved in /home/mrtg/.ssh/id_dsa.pub.
The key fingerprint is:
1c:f7:7c:d4:8d:0c:98:49:42:f4:e9:e3:d6:44:81:cb mrtg@nmsbox
```

8) Copy the public key to the /home/mrtg/.ssh directory web server. Use a unique name for the destination file in case you need to support multiple systems.

```
$ scp id_dsa.pub mrtg@webserver:/home/mrtg/.ssh/nmsbox.dsa.pub
mrtg@webserver's password:
nmsbox.dsa.pub       100% |*****************************|   601
00:00
```

Web Server Procedures Part II

9) Back on the Web Server, change to the /home/mrtg/.ssh directory and append the content of nmsbox.dsa.pub to the authorized_keys file to establish the NMS as a trusted system.

```
$ cd /home/mrtg/.ssh
$ cat nmsbox.dsa.pub >> authorized_keys
```

10) Lastly, test the public key authentication by logging in from the NMS to the Web Server. A common reason this would be unsuccessful is that your global ssh_config file on the NMS prefers SSH protocol version 1 over version 2. It is assumed that both your SSH client and SSH server can support and prefer protocol version 2. If not, then it is

time to upgrade!

```
$ ssh webserver
Last login: Sat Sep  7 15:23:35 2002 from 192.168.0.35
OpenBSD 3.1-stable (GENERIC) #0: Tue Aug 27 16:32:19 PDT 2002

$ hostname -s; id
webserver
uid=1001(mrtg) gid=10(users) groups=10(users)
$ exit
```

Step Two – The Host Script

MRTG expects to receive data in the following format:

```
first measurement
second measurement
system uptime
system hostname
```

The Host Script will need to be written such that the execution of its output (the NMS Script) yields output in this format. Below is an example of a hostscript.sh script that takes the measurements of a single network interface with one IP address on an OpenBSD web server. This script could easily be modified to include support for more IP addresses and other NICs, other measurement points such as CPU and memory utilization, disk space, Apache statistics, DNS usage statistics (ndc stats) or anything else you can get from the command line.

1) Log in to the web server as the mrtg user.
2) Create a scripts directory in mrtg's home directory and give permissions only to the mrtg user. We are creating a script that will be run on another system, and we want to make sure that an attacker couldn't easily replace our script with a malicious script.

```
$ mkdir /home/mrtg/scripts
$ chmod 700 /home/mrtg/scripts
```

3) Using your favorite text editor, create /home/mrtg/scripts/hostscript.sh that consists of the following content (a "case" statement is included to allow for future support of other statistics as well as to make troubleshooting easier):

```
#!/bin/sh

# this Host Script runs on the monitored system
# its output is the NMS script

# get some global parameters
HOSTNAME=`/bin/hostname -s`
UPTIME=`/usr/bin/uptime | /usr/bin/awk -F, '{print $1}' | /usr/bin/awk '{print $3 " "
$4}'`
NMSSCRIPT="/mrtg/scripts/$HOSTNAME.nmsscript.sh"
# only one NIC and one IP is assumed here
NIC=`/sbin/ifconfig -a | /usr/bin/grep BROADCAST | /usr/bin/tail -1 /usr/bin/awk '{print $1}' awk -F: '{print
$1}'`
IP=`/sbin/ifconfig -a | /usr/bin/grep broadcast | /usr/bin/awk '{print
$2}'`

# gather the needed statistics
BYTES_OUT=`/usr/bin/netstat -nbI $NIC | /usr/bin/grep "$IP" | /usr/bin/awk '{print $6}'`
BYTES_IN=`/usr/bin/netstat -nbI $NIC | /usr/bin/grep "$IP" | /usr/bin/tail -1 | /usr/bin/awk
'{print $5}'`


# build the NMS script with the gathered data
touch $NMSSCRIPT
chmod 700 $NMSSCRIPT
echo "#!/bin/sh" > $NMSSCRIPT
echo "case \"\$1\" in" >> $NMSSCRIPT
echo "  cpu\)" >> $NMSSCRIPT
echo "    echo $BYTES_OUT" >> $NMSSCRIPT
echo "    echo $BYTES_IN" >> $NMSSCRIPT
echo "    echo \"$UPTIME\"" >> $NMSSCRIPT
echo "    echo \"$HOSTNAME\"" >> $NMSSCRIPT
echo "    ;;" >> $NMSSCRIPT
echo "  *)" >> $NMSSCRIPT
echo "    echo \"usage: \`baseline \$0\` nic \"" >> $NMSSCRIPT
echo "    ;;" >> $NMSSCRIPT
echo "esac" >> $NMSSCRIPT
```

4) Ensure the proper permissions on the hostscript.sh file. Our directory should protect its contents, but this is a good idea in case you make changes in the future like moving the script to another directory.

```
$ chmod 700 /home/mrtg/scripts/hostscript.sh
```

5) Test the script to make sure you get the expected output.
6) Configure cron to run the hostscript.sh file every 5 minutes using "crontab –e" (make sure you are the mrtg user).

```
*/5 * * * * /home/mrtg/scripts/hostscript.sh > /dev/null 2>&1
```

7) Check at the 5-minute mark to make sure your NMS Script is getting

generated by cron. On the monitored host, the output of the hostscript.sh script should be /home/mrtg/scripts/webserver.nmsscript.sh and should look similar to below:

```
#!/bin/sh
case "$1" in
  nic)
    echo "1807702164"
    echo "1025031518"
    echo "10 days"
    echo "webserver"
    ;;
  *)
    echo "Usage: `basename $0` nic"
    ;;
esac
```

When pulled down and executed by the NMS, the script above will produce an output similar to below:

```
1807702164
1025031518
10 days
webserver
```

Step Three – Pull the NMS Script from the Web Server to the NMS

A short script is needed to automate the pulling of the NMS script from the web server to the NMS.

NMS Procedures

1) Log in to the NMS as user mrtg.
2) Make a place to keep all of the Host Scripts from your monitored hosts and give it appropriate permissions. The scripts will be overwritten every time a new one is pulled down, so you don't need to worry too much about space required.

```
$ mkdir /home/mrtg/scripts
$ chmod 700 /home/mrtg/scripts
```

3) Using your favorite text editor, create /home/mrtg/pull.sh. The contents should look similar to the example below:

```
#!/bin/sh

# this script pulls Host Scripts from our monitored boxes

/usr/bin/scp mrtg@webserver:/home/mrtg/scripts/webserver.nmsscript.sh
/home/mrtg/scripts/
```

4) Make sure that this script has the proper permissions because it will be called by cron.

```
$ chmod 700 /home/mrtg/scripts/pull.sh
```

5) Run the script to make sure it behaves as you would expect, then check the /home/mrtg/scripts directory for the webserver.nmsscript.sh file.
6) Now that it works, schedule it to occur every five minutes. Make sure you are the mrtg user. You will need to stagger the cron jobs so that you give the monitored host enough time to generate the NMS script filled with the monitoring statistics. You may need to further adjust the run time of the pull.sh script if you are collecting a lot of data on the monitored system. It is assumed that both your NMS and the web server are synced to the same NTP source. Using 'crontab –e', add the following entry.

```
1,6,11,16,21,26,31,36,41,46,51,56 * * * * /home/mrtg/scripts/pull.sh > /dev/null
2>&1
```

Step Four – Installation and Configuration of MRTG

The complexity of this step will depend primarily on your operating system. On an OpenBSD system with the ports distribution, installation doesn't require much user interaction. However, MRTG requires some dependencies and those dependencies have dependencies such as gd, libpng, freetype, gettext, ttf, libiconv, and gmake. **[9]** The time to download and compile MRTG and its dependencies will vary depending on your Internet connection and the speed of your system.

1) Log in with your normal non-root user ID.
2) Change directories to the /usr/ports/net/mrtg directory and install the product.

```
nmsbox:mrtg {1} cd /usr/ports/net/mrtg
nmsbox:mrtg {2} sudo make install
```

3) The name and location of the default MRTG configuration file may vary. If installed from the OpenBSD ports tree, the location is /etc/mrtg.cfg. So

far we've been able to keep our files related to MRTG in the /home/mrtg directory. This makes backups of our configuration easy. In pursuit of that goal, we will create a brand new mrtg.cfg file in the /home/mrtg directory and use the /etc/mrtg.cfg file as syntax reference.

4) Add the appropriate content to the /home/mrtg/mrtg.cfg file. Another good source for syntax is the MRTG Documentation Pack page. **[10]** A configuration file consistent with the example used in this paper might look like:

```
WorkDir: /home/mrtg/web
Options[_]: growright,bits

Target[webserver-nic]: `/home/mrtg/scripts/webserver.nmsscript.sh nic`
# We have a 100Mbps NIC
MaxBytes[webserver-nic]: 100000000
Title[webserver-nic]: Internal NIC
YLegend[webserver-nic]: Bits per Second
ShortLegend[webserver-nic]: b/s
Legend1[webserver-nic]: Incoming Traffic in Bits per Second
Legend2[webserver-nic]: Outgoing Traffic in Bits per Second
Legend3[webserver-nic]: Maximal 5 Minute Incoming Traffic
Legend4[webserver-nic]: Maximal 5 Minute Outgoing Traffic
LegendI[webserver-nic]:  In:
LegendO[webserver-nic]:  Out:
PageTop[webserver-nic]: <H1>Traffic Analysis for Internal NIC
 </H1>
 <TABLE>
   <TR><TD>System:</TD><TD> Web Server </TD></TR>
   <TR><TD>Max Speed:</TD>
       <TD>100MBytes/s (Fast Ethernet)</TD></TR>
  </TABLE>
```

5) Note in the mrtg.cfg file above that the working directory is /home/mrtg/web. We will later need to configure Apache to serve the files in this directory as web pages. The /home/mrtg/web directory should be readable and executable by at least the Apache user (usually www), and full permissions to the mrtg user. This means that now the mrtg process doesn't have to be called by root and can be run by the mrtg user. This is good because scripts running from cron's root tab require extraordinary precautions (not that we won't necessarily be just as careful with mrtg's crontab).

```
$ mkdir /home/mrtg/web
$ chmod 755 /home/mrtg/web
```

6) As mentioned in the "RUNNING" section of the UNIX MRTG Installation Guide, you will need to run the mrtg program twice to generate the initial files. **[9]** The "WARNING" messages below are normal.

```
$ /usr/local/bin/mrtg /home/mrtg/mrtg.cfg
Rateup WARNING: /usr/local/bin/rateup could not read the primary log file for webserver-
nic
Rateup WARNING: /usr/local/bin/rateup The backup log file for webserver-nic was invalid
as well
Rateup WARNING: /usr/local/bin/rateup Can't remove webserver-nic.old updating
log file
Rateup WARNING: /usr/local/bin/rateup Can't rename webserver-nic.log to webserver-nic.old updating
log file
$ /usr/local/bin/mrtg /home/mrtg/mrtg.cfg
Rateup WARNING: /usr/local/bin/rateup Can't remove webserver-nic.old updating
log file
```

7) Add a crontab entry for mrtg to run every five minutes. You will want to make this run after the pull.sh script.

```
2,7,12,17,22,27,32,37,42,47,52,57 * * * * /home/mrtg/scripts/pull.sh > /dev/null
2>&1
```

Step Five – Configuring Apache

The installation of Apache can be daunting to an unseasoned engineer. I highly recommend reading as much as you can about what compilation options you need. A good place to start is at Apache's documentation project. **[11]** Especially pay attention to the security concerns that evolve around a web server. OpenBSD ships with a version 1.x of Apache in the core operating system. It is enabled at boot time by changing httpd_flags=NO to httpd_flags="" in the /etc/rc.conf file. You can also use httpd_flags="-DSSL" to enable SSL encryption. Enabling SSL will not be discussed in this document. To reiterate, the primary goal is to keep SNMP off of your critical UNIX systems, not necessarily to protect the monitoring and trending data. If you plan on monitoring sensitive data, or you feel that disclosure of monitoring data is a significant security concern, consider the use of SSL and requiring login access to your NMS web server.

The step-by-step installation of Apache on other operating systems will not be presented in this paper as it is deemed out of scope. However, configuring Apache to work with our new MRTG directory is relevant.

1) Log in to the NMS with your normal non-root account.
2) Edit /etc/rc.conf and change httpd_flags=NO to httpd_flags="" to enable Apache at boot time.
3) Create a symbolic link in your Document Root directory that points to the /home/mrtg/web directory. On an OpenBSD system, /var/www/htdocs is

the default Document Root directory.

```
nmsbox:htdocs {99} sudo ln -s /home/mrtg/web /var/www/htdocs
```

4) Start the web server.

```
nmsbox:htdocs {100} sudo apachectl start
```

5) Lastly, open your web browser and point it to http://nmsbox/web/webserver-nic.html and you should see the monitored network interface charts! The MRTG home page has some sample sites using MRTG where you can compare your results. **[12]**

Conclusion

Sites that hold security as a high priority most likely already use SSH. The SCP utility from the SSH protocol suite coupled with public key authentication can perform many of the read-only functions that SNMP provides in the network monitoring arena. With more research, SSH could also potentially replace the write functions of SNMP. This process could also be ported to Windows systems and other devices that support SSH/SCP and shell or batch scripting. Using SSH could yield greater overall security because the SNMP service could be disabled on monitored systems, and that is one less thing to worry about patching.

References

[1] Buckholz, Zachary. "Monitoring System Performance with MRTG." Daemonnews. 7 Sep 2002.
URL: http://www.daemonnews.org/200001/sysperf.html (Jan. 2000).

[2] Oetiker, Tobias and Rand, Dave. "The Multi Router Traffic Grapher." Version 2.9.22. 7 Sep 2002.
URL: http://people.ee.ethz.ch/~oetiker/webtools/mrtg/

[3] Butterworth, Jim. "SNMPv3 Goes Full Standard." SNMP Research. 7 Sep 2002.
URL: http://www.snmp.com/news/snmpv3_fullstandard.html (8 April 2002).

[4] CERT®. "CERT® Advisory CA-2002-03 Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol (SNMP)." 7 Sep

2002.
URL: http://www.cert.org/advisories/CA-2002-03.html (28 Aug 2002).

[5] Entercept™ Security Technologies. "SEA SNMP – Buffer Overflow and Format String Vulnerabilities in Sun Solaris." 7 Sep 2002.
URL: http://online.securityfocus.com/advisories/4174 (4 Jun 2002).

[6] Internet Security Systems, Inc. X-Force. "OpenSSH Remote Challenge Vulnerability." Bulletin #00219. 7 Sep 2002.
URL: http://openssh.org/txt/iss.adv (26 Jun 2002).

[7] OpenBSD. "OpenSSH Security Advisory (adv.trojan)." 7 Sep 2002.
URL: http://openssh.org/txt/trojan.adv (1 Aug 2002).

[8] OpenBSD. "OpenBSD Ports & Packages." Version 3.1. 7 Sep 2002.
URL: http://openbsd.org/ports.html

[9] Oetiker, Tobias and Rand, Dave. "doc/unix-guide." The Multi Router Traffic Grapher. Version 2.9.22. 7 Sep 2002.
URL: http://people.ee.ethz.ch/~oetiker/webtools/mrtg/unix-guide.html

[10] Oetiker, Tobias and Rand, Dave. "MRTG Documentation." The Multi Router Traffic Grapher. Version 2.9.22. 7 Sep 2002.
URL: http://people.ee.ethz.ch/~oetiker/webtools/mrtg/

[11] The Apache Software Foundation. "Documentation Project – The Apache HTTP Server Project." Version 1.3 and 2.0. 7 Sep 2002. URL: http://httpd.apache.org/docs-project/

[12] Oetiker, Tobias and Rand, Dave. "MRTG Users and Uses." The Multi Router Traffic Grapher. Version 2.9.22. 7 Sep 2002.
URL: http://people.ee.ethz.ch/~oetiker/webtools/mrtg/users.html

Last Updated: July 24th, 2017

Upcoming Training

SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
Community SANS Trenton SEC401	Trenton, NJ	Aug 21, 2017 - Aug 26, 2017	Community SANS
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS New York SEC401	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Community SANS Charleston SEC401	Charleston, SC	Oct 02, 2017 - Oct 07, 2017	Community SANS
Mentor Session - SEC401	Arlington, VA	Oct 04, 2017 - Nov 15, 2017	Mentor