Global Information Assurance Certification Paper

GIAC Security Essentials (GSEC)
Practical Assignment, Version 1.4b, Option 2
Jon M. Budoff

Security and functionality enhancements to Tripwire Academic Source Release (ASR) - A Case Study

Abstract

One element of a "defense in depth" strategy is to have host-based intrusion detection systems in place, also known as "file integrity monitors". Over the last three years, our host-based intrusion detection strategy has changed and improved considerably. This paper gives an overview of where we started, what improvements we have made to the system, and we are today.

We began with Tripwire Academic Source Release (ASR) installed on only a few multi-user machines, each with its own independent configuration file, and with each tripwire database stored on the local machine insecurely. Approximately one year ago, we undertook a project to develop a method for securing the tripwire databases. This project grew into designing an entire framework around the base ASR code that allows us to run the entire process from a central console, to store the databases securely, and to have significantly increased functionality.

Prior to the project initiation, we had made several "incremental" improvements to our monitoring system. This can be considered a "Pre-project" phase, and is briefly described in the "Before snapshot" section. The majority of this paper focuses on the implementation and results of the project, and our current situation.

Before snapshot

NOTE: Although the author of this paper was the primary architect and implementer of the changes described in this paper, I have chosen the convention of using the term "we" throughout the paper. This is because the changes were not made in a vacuum. Other administrators were involved with brainstorming sessions, and have provided valuable feedback throughout this process.

The improvements to our host-based intrusion detection strategy evolved over time. Initially, our host-based intrusion detection system had the following characteristics.

- Tripwire ASR on several multi-user hosts, but the majority of hosts (~40 at the time) unmonitored
- Configuration file unique to each machine

- Only monitoring key OS files
- Database not kept on secured media

During the two years leading up to kickoff of the main project discussed in this paper, we had made a number of incremental improvements in how we used tripwire, many of which were incorporated in the new design or provided a learning experience critical to moving forward with the project. These "pre-project" improvements are summarized below.

Pre-project improvements - summary

- Installed tripwire ASR on all hosts
- Began using central configuration file distributed to each machine
- Began watching files in /local
- Developed simple parsing scripts to allow updating directly from an email or file created from redirecting output or tripwire run.

At this point, we were monitoring all hosts and had made strides in easing the management burden by using a central configuration file, but the database was still residing on the local machines and was therefore insecure.

During snapshot

This project was originally intended as a way of storing the tripwire databases in a secure location. As we began evaluating design options, we decided to go further than the original project charter and remove tripwire from the local machines entirely and run it remotely from a central console (hereafter referred to as the tripwire console), putting the necessary files in place each time tripwire is run and then removing them after the run is complete.

The project had to support two distinct types of functionality. The first was the automated tripwire check which occurs daily and which alerts the administrators when there are changes. Because this is automated, and therefore hands-off, it presented unique challenges. The second functionality was for the administrators to be able to interface with the system and perform the three main functions associated with tripwire; these are running tripwire (aka "integrity checking"), updating the database, and initializing the database.

A number of questions had to be answered; these included:

- How do we put the files in place on the remote machine from the tripwire console?
- How will the automated job run? How will it get the correct privileges to run tripwire correctly?
- Are we opening up new security holes with this new system?
- For the automated check, how do we send the information on changes to the administrators?
- How will the administrators interface with the new system? From where?

- When updating or initializing, how does the updated database get back to the tripwire console?
- How will we be assured that the automated job is actually running correctly? How will the administrators be informed when there are errors?
- How will the system clean up after itself if something goes wrong?

Phase I Design decisions

Since we already had a kerberos infrastructure in place, we made the decision early to take advantage of this. We decided to use kerberos rcp to copy the appropriate files onto each machine from the tripwire console, and then use kerberos rsh to initiate the appropriate processes on the remote machine. Two of the key questions that had to be answered were 1) how to have tripwire run as root on the remote machine; and 2) how to get the updated databases back to the tripwire console. We chose to address these questions differently for the automated process than for the processes the administrators would be doing.

Tripwire must run as root on the remote machine to have the correct privileges to access all directories. For the automated process, we chose to utilize sudo, in conjunction with kerberos, to allow this to happen. A new user, twuser, was created on each machine, including the tripwire console. Along with this, a new kerberos instance was created for this user. On the tripwire console, the automated job is run via the tripwire user's crontab. In the tripwire user's home directory, there is a "stashed" keytab, which allows the cron job to get a ticket without having to provide a password. The keytab is encrypted and is readable only by the tripwire user via the appropriate kerberos command. After the appropriate files are put in place on the remote machine, the tripwire binary is run on the remote machine using sudo, so that it has the appropriate permissions. This is possible because the sudoers file on the remote machine contains a line allowing this user to run this command without a password. Of course, the user can run only this file via sudo without a password (specified with the absolute pathname), and the tripwire binary itself has been renamed to something innocuous.

Other options considered for the automated job were:

- Option 1: Running the automated tripwire job out of root's crontab on the tripwire console and utilizing stashed root kerberos credentials. This would be advantageous because we would not have to create a new kerberos instance and put a new user account on all of the hosts. Having a current root ticket would give the appropriate privileges for tripwire to run correctly on the remote machines. We deemed this unacceptable because if the tripwire console was ever compromised and someone was to gain a root ticket via the stashed credentials, they would then have root access on all of our machines.
- Option 2: Running the automated job out of the tripwire user's crontab on the tripwire console and utilizing stashed credentials for the tripwire user's

kerberos instance. Instead of using sudo on the remote hosts, the kerberos instance for the tripwire user would be listed in the /.k5login of all remote hosts. This would give the user the ability to run as root on all machines. This was also considered unacceptable for security reasons.

For the administrators' interaction, we chose a different solution. Since all of the administrators have root kerberos instances, we decided to keep things simple and let the administrators run tripwire via root kerberos tickets. This meant that an administrator would get a root ticket and then initiate the tripwire functions as root, eliminating any permission problems.

Operational considerations

Originally, we tried to mimic the existing behavior of tripwire from the administrator's perspective. This meant continuing to use a standard command-line interface. The changes to the automated job were effectively transparent to the administrators; changes detected by the automated job were emailed to the administrators, as they were with the old system. For the manual administrator interface, administrators were accustomed to running tripwire locally on each machine by changing to the tripwire directory and then typing './bin/tripwire', './bin/tripwire -update ...', or './bin/tripwire -init' to perform the various functions of running, updating, and initializing tripwire, respectively.

With the new system, administrators could still run tripwire from the local hosts, albeit in a slightly different manner. Scripts were created which allowed the administrators to run tripwire, and to update and initialize the tripwire database from the local machine. These scripts lived in the ./bin directory of the tripwire user's home directory. These scripts initiated a connection to the tripwire console, which in turn then initiated the appropriate connections back to the machine. In addition to performing the "standard" tripwire functions, the update script incorporated parsing code developed earlier, so that an administrator could specify an "update file" containing all of the entries to be updated in the database.

In addition to having the capability of running tripwire from the local machine, administrators had the option of logging onto the tripwire console and performing all functions from there. This provided significant benefits when updating multiple machines. In lieu of logging into each machine individually and getting a root ticket on each machine, the administrator could log into the tripwire console, get a single root ticket, and update all of the machines from there.

Phase II Design Changes

In any project, there are lessons learned after the initial implementation. This project was no exception. Some of the key problems/learnings were:

- The administrators were able to run tripwire either from the local machines or from the tripwire console, which caused confusion.
- The ability to run tripwire from either the local machine or the tripwire console added unnecessary overhead and complexity to the system.
- Our part-time employees were unable to update tripwire for their own changes. This is because they use sudo to get superuser privileges and do not have a root kerberos instance.
- The system required four connections between the tripwire console and the remote machine each time tripwire was run.
- Many of the scripts contained duplicate code and functionality. When a change needed to be made, the same change had to be made across the run, update, and init scripts.

To address the major issues, we decided to move towards a pure "console-based" approach. As part of this transition, we also redesigned the system to minimize the number of connections needed each time tripwire was run. At a later date, much of the code base was consolidated to eliminate duplication and ease maintenance.

After snapshot

Summary

The improvements we have made to our host-based intrusion detection system are summarized below.

Host-based IDS improvements - summary

- Infrastructure developed around Tripwire ASR.
- Tripwire for ~70 hosts is run from a central console.
- Tripwire files/database no longer live on local machines.
- Single configuration file is used for all hosts; @@ifhost statements used to define files to watch on a per-machine basis.
- Databases live in a secure location on the tripwire console.
- Increased functionality with easy, familiar interface.
- Ability to parse "update files".
- Ability to easily update multiple hosts at the same time.
- Ability to easily determine which hosts have been updated.
- Duplicate copies of all databases are archived to separate secure host.
- Reports from the automated run are archived on the tripwire console.
- There is no sign of tripwire on any client machine (remote host).
- Operates with two distinct connections to each client.

Throughout the design of this system, we paid particular attention to security issues. Below is the summary of the security precautions taken in the design of the new system.

Security precautions

- The stashed keytab is only on the tripwire console machine, and is only readable by the tripwire user.
- The tripwire user has no special privileges.
- The tripwire user uses sudo to actually run tripwire on each host; this is the only command this user can run via sudo.
- The tripwire binary has been renamed to something innocuous so that "tripwire" never shows up in a process list.
- All tripwire files and directories are owned by and only readable by the tripwire user. The tripwire user uses sudo to run the tripwire binary with the necessary privileges.
- The tripwire user has a non-breakable password on each machine (*NP* in /etc/shadow).
- All communication between the tripwire console and the remote machines is encrypted, via kerberos 'rcp -x' and 'rsh -x'.
- The tripwire console is a highly secure machine, with minimal ports open, and accounts only for the administrators.
- The administrators can log into the tripwire console only from their desktop machines; this is accomplished by wrapping (via tcp wrappers) the login service.

Operational details

The system operates via a number of perl and shell scripts that work together, in conjunction with an overlying kerberos and sudo infrastructure.

Automated tripwire check

The automated check is run out of the crontab for the tripwire user. The cron job uses a shell script which does the following:

- Gets a kerberos ticket for the tripwire user via the stashed credentials.
- Runs sequentially through a list of hosts to run the tripwire check. This list is obtained in real-time by parsing the databases that are stored on the tripwire console. This eliminates the need to maintain a separate list of hosts. Renaming a database to ".DISABLED_database" disables the tripwire check for that host.
- For each host, the tripwire database is archived to another secure host.
- For each host, three files are copied over to the remote machine via a kerberos rcp connection. These are a tar file containing the tripwire directory structure, binary, and configuration file; the database for that host; and a "run script". An kerberos rsh session then executes the run script, which does several things:
    - Expands the tarfile.

    - Moves the database into the correct place in the directory structure.
    - Initiates the tripwire binary in "integrity check" mode.
    - The output from the tripwire run is passed back to the initiating script on the tripwire console via the rsh session; if changes are reported, these changes are emailed to the administrators. The script also archives the report for each host, whether changes are reported or not.
    - After the tripwire run is completed, the script on the remote host deletes all files and directories that were put in place.
- The kerberos ticket is destroyed after tripwire has been run on all hosts.

Administrator interface

Administrators log into the tripwire console via as the tripwire user via encrypted kerberos rlogin. When they first log in, they are presented with the output from the "today" script, which tells which hosts have been updated that day, and which hosts reported changes during the last automated check but have yet to be updated. An example of the output from the today script can be seen in Appendix A. Aliases are provided for the run, update, and initialize functions of tripwire, and administrators run tripwire using the following syntax:

- run -h <host>
- init -h <host>
- update -h <host> <entries to update>
  -or-
  update -h <host> -f <updatefile>

An update file can be generated in one of several ways. It can be an email message from the automated run that is saved in a location that the tripwire console has access to (e.g. an nfs filesystem mounted on both machines), or copied over via kerberos rcp; or it can be the redirected output from a tripwire run, e.g. "run -h host1 > host1.twout". The scripts are designed to ignore any non-relevant lines such as email headers, etc. in the update file.

As presented above, there are three ways in which the administrators can use tripwire - run (integrity check), update, and initialize. In reality, each of the three commands is an alias to the same script. For instance, run is an alias to 'tripwire.pl -h'. How each of these functions is detailed below.

The "run" function works very similarly to the automated check. The script does the following:

- Gets a kerberos ticket for the tripwire user via the stashed credentials.
- Copies three files over to the remote host specified. These are a tar file containing the tripwire directory structure, binary, and configuration file; the database for the remote host; and a "run script".
- Executes the run script on the remote host via an encrypted kerberos rsh session then; the run script does several things:

    - Expands the tarfile
    - Moves the database into the correct place in the directory structure.
    - Initiates the tripwire binary in "integrity check" mode.
    - The output from the tripwire run is passed back to standard output on the tripwire console via the rsh session. If the administrator desires, he can redirect the output to a file which can then be used to update tripwire.
    - After the tripwire run is completed, the script on the remote host deletes all files and directories that were put in place.
- After the run is complete, the kerberos ticket is destroyed.

The update and initialize functions work similarly to the run function, with two notable differences.

- For the update function, the script on the tripwire console must be able to pass along the arguments for the entries to be updated or pass along an update file, and the script running on the remote machine must be able to properly parse the arguments or update file.
- For both the update and initialize functions, the updated database must be copied back to the tripwire console. This is accomplished by using forwardable kerberos tickets, which give the script running on the remote host the ability to copy the database back using kerberos rcp.

Examples of output from the run, update, and initialize functions can be seen in Appendix A.

Other features of the system

There are also a number of other features of the system, some of which are "behind the scenes". These include the following.

Error checking: The scripts have error checking to check for the conditions listed below. If the command syntax is incorrect, an error message is given along with a usage statement. If the command syntax is correct, but another error is present, the usage statement is generally not given. These checks are completed before the script gets a kerberos ticket or starts copying any files to the host. Some of the conditions checked are:

- Make sure a host is specified.
- Make sure a database exists for the specified host.
- Make sure there are no arguments specified for running or init'ing.
- If an update file is specified, do several sanity checks, including: Does the file exist?, Is it readable? Is it of the correct type?
- Make sure that either an update file is specified, or arguments on the command -line, but not both.

Examples of error statements can be seen in Appendix A.

Lockfiles: Whenever tripwire is initiated in any mode, a "lockfile" is touched which corresponds to the host being run. The tripwire script looks for the

existence of this lockfile prior to initiating a run. If the script sees a lockfile for the host specified, the administrator is informed that someone else is already running tripwire for this host. The script removes the lockfile once the run is completed.

Kerberos tickets: The kerberos ticket obtained when the administrators run tripwire from the tripwire console is tied to the process id. This allows multiple administrators to run tripwire from the tripwire console without interfering with each other.

Archiving: The database for each remote host is stored on the tripwire console, and is also archived from the central console to another secure host. This archive occurs at the beginning of the automated tripwire run. Thirty days worth of databases are kept for all hosts in this archive. The reports from the automated run for each host are archived on the tripwire console; thirty days worth of reports are kept for each host.

Modifying the configuration file: The administrator can modify the configuration file through any text editor. Once it has been modified, the administrator runs an "update_tarfile" script to put the new configuration file in place in the tarfile that gets copied to the hosts.

Specifying who reports are sent to: There's a section in the code which runs the automated tripwire check where an administrator can easily change who the report is emailed to. This was set up so that reports for desktop machines only go to the owner of the desktop. This is also useful when building new machines that are not in production yet. The reports for production machines are emailed to all administrators.

Future improvements

There are several areas that can still be improved; these include:

- The today script doesn't tell what specific items were updated; a database may be updated for a change that is different from or only a part of the changes reported by the automated run, but it still shows the db as updated. This can cause confusion with someone thinking that certain entries had been updated when they have not been.
- The today script does not tell who updated a database. This could be useful to know to answer questions about what was updated. Keeping an easily accessible log file for each machine telling what was updated and when could also address the problem discussed above.
- Although it is possible to specify who a report for a specific host is sent to, most administrators will not do this because it is inside a script that they are not familiar with. Breaking this functionality out into an easily understandable configuration file would make this easier to do.
- The scripts do not always handle mid-run aborts well, and may leave tripwire files on the remote host.

Summary

The enhancements we have made to our host-based IDS have provided a number of benefits. We have enhanced the functionality while at the same time improving the user interface and easing maintenance requirements. We have improved security of the system, giving us increased confidence in its ability to alert us to machine compromises or problems. And we have done all of this without losing our previous learning investment with Tripwire ASR, with no capital expense, and relatively minimal development effort.

References

Kochan, Stephen & Patrick Wood, UNIX Shell Programming. Hayden Books, copyright 1990, ISBN 0-672-48448-X.

Christiansen, Tom & Nathan Torkington, Perl Cookbook. O'Reilly & Associates, Inc., copyright 1998, ISBN 1-56592-243-3.

Wall, Larry et. al., Programming Perl. O'Reilly & Associates, Inc., copyright 1996, ISBN 1-56592-149-6.

"Tripwire for Servers, Assures the Integrity of Your Data", Can be found at http://www.tripwire.com/files/literature/product_info/Tripwire_for_Servers.pdf.

"Tripwire Open Source, Linux Edition FAQ", Tripwire Open Source project, can be found at http://www.tripwire.org/qanda/faq.php

"Installing, configuring, and using Tripwire® to verify the integrity of directories and files on systems running Solaris 2.x", Carnegie Mellon Software Engineering Institute, can be found at http://www.cert.org/securityimprovement/implementations/i002.02.html.

Appendix A

Examples of output

Output from 'today' script:

```
twuser@twcon $ today

The following databases have been updated today:

 size       update time     database
 -------    ------------    ------------------------------
 1326797    Dec 26 09:33    tw.db_host1.********.edu
 1165804    Dec 26 09:33    tw.db_host2.********.edu
 1005175    Dec 26 09:34    tw.db_host3
 1100837    Dec 26 09:34    tw.db_host4.********.edu

----------
The following hosts reported changes during the last automated
tripwire check and still need to have their databases updated:

   host5
   host6
   host7
   host8

----------
```

Tripwire run:

```
twuser@twcon $ run -h host1
**************************************************************
getting ticket for twuser@********.EDU
**************************************************************
**********
copying files to host host1
**********
initiating run script on host1
This rsh session is using DES encryption for all data transmissions.
### Phase 1:   Reading configuration file
### Phase 2:   Generating file list
### Phase 3:   Creating file information database
### Phase 4:   Searching for inconsistencies
###
###                     Total files scanned:            10884
###                           Files added:              0
###                           Files deleted:            0
###                           Files changed:            10863
###
###                   After applying rules:
###                           Changes discarded:        10861
###                           Changes remaining:        2
###
changed: drwxr-xr-x root          512 Dec 23 22:01:39 2002 /etc/cluster
changed: drwxrwxrwt root         1987 Dec 26 15:40:12 2002 /tmp
### Phase 5:   Generating observed/expected pairs for changed files
###
### Attr        Observed (what it is)         Expected (what it should be)
### =========== ============================= =============================
/etc/cluster
      st_mtime: Mon Dec 23 22:01:39 2002      Tue Nov 26 17:34:12 2002
      st_ctime: Mon Dec 23 22:01:39 2002      Tue Nov 26 17:34:12 2002

/tmp
        st_ino: 9188213                       9800994

**************************************************************
```

```
destroying ticket for twuser@********.EDU
****************************************************************
```

Tripwire update with entries specified on command line:

```
twuser@twcon $ update -h host2 /etc/inetd.conf /usr/bin/ps
****************************************************************
getting ticket for twuser@********.EDU
****************************************************************
**********
copying files to host host2
**********
initiating update script on host2
This rsh session is using DES encryption for all data transmissions.
### Phase 1:   Reading configuration file
### Phase 2:   Generating file list
Updating: update file: /etc/inetd.conf
Updating: update file: /usr/bin/ps
### Phase 3:   Updating file information database
###
### Old database file will be moved to `tw.db_host2.********.edu.old'
###            in ./databases.
###
### Updated database will be stored in './databases/tw.db_host2.********.edu'
###            (Tripwire expects it to be moved to '/home/twuser/integrity/databases'.)
###
****************************************************************
destroying ticket for twuser@********.EDU
****************************************************************
```

Tripwire update with update file specified:

```
twuser@twcon $ update -h host3 -f host3.twout
****************************************************************
getting ticket for twuser@********.EDU
****************************************************************
**********
copying files to host host3
**********
initiating update script on host3
This rsh session is using DES encryption for all data transmissions.
### Phase 1:   Reading configuration file
### Phase 2:   Generating file list
Updating: delete file: /etc/rem_name_to_major
Updating: update file: /etc/path_to_inst
Updating: update file: /etc/path_to_inst.old
### Phase 3:   Updating file information database
###
### Old database file will be moved to `tw.db_host3.********.edu.old'
###            in ./databases.
###
### Updated database will be stored in './databases/tw.db_host3.********.edu'
###            (Tripwire expects it to be moved to '/home/twuser/integrity/databases'.)
###
**********
please manually delete updatefile /tmp/twuser/host3.twout
****************************************************************
destroying ticket for twuser@********.EDU
****************************************************************
```

Tripwire database initialization:

```
twuser@twcon $ init -h host4
****************************************************************
getting ticket for twuser@********.EDU
****************************************************************
**********
copying files to host host4
**********
```

```
initiating init script on host4
This rsh session is using DES encryption for all data transmissions.
### Phase 1:   Reading configuration file
### Phase 2:   Generating file list
### Phase 3:   Creating file information database
###
### Warning:   Database file placed in ./databases/tw.db_host4.********.edu.
###
###            Make sure to move this file file and the
configuration
###            to secure media!
###
###            (Tripwire expects to find it in '/home/twuser/integrity/databases'.)
***************************************************************
destroying ticket for twuser@********.EDU
***************************************************************
```

Email from automated tripwire run:

```
From: T-Integrity <twuser@twcon.********.edu>
To: sys-admins@********.edu
Subject: Tripwire -- host9.********.edu

Warning!  Tripwire output has changed on host9.********.edu!
Current tripwire output follows:

changed: -rw------- root         7617 Dec 30 16:37:26 2002 /etc/dfs/dfstab
changed: -rw------- root         4159 Dec 31 10:10:52 2002 /etc/dfs/sharetab
changed: -r--r--r-- root         4897 Dec 30 16:25:42 2002 /etc/inet/hosts
### Attr        Observed (what it is)       Expected (what it should be)
### =========== =========================== ============================
/etc/dfs/dfstab
       st_size: 7617                        12831
      st_mtime: Mon Dec 30 16:37:26 2002    Wed Dec  4 09:50:37 2002
      st_ctime: Mon Dec 30 16:37:26 2002    Wed Dec  4 09:50:37 2002
    md5 (sig1): 1v.VPTpNiCHluraIfePI3I      0soicFAIwk7uM5MNvw33B9
 snefru (sig2): 3TSghYNd3U3UTSN11fuZyT      1riUoxKuQBYkuGHpQnfazQ

/etc/dfs/sharetab
       st_mode: 100600                      100644
        st_ino: 64964                       64791
        st_gid: 1                           0

/etc/inet/hosts
       st_size: 4897                        643
      st_mtime: Mon Dec 30 16:25:42 2002    Tue Jul 16 07:21:10 2002
      st_ctime: Mon Dec 30 16:25:42 2002    Tue Jul 16 07:21:10 2002
    md5 (sig1): 0Y:Y43bpSo6zLy2VfwVf:W      0TGLsFhz5qYfH6CuoFibzu
 snefru (sig2): 0bYdDEQlh:8SrGCQI5RPbP      2k8exEg1IqdMuCcpC9tEDR
```

Examples of error checking in script:

```
twuser@twcon $ run
***********************************************************
ERROR - no host specified to update!
***********************************************************
usage:  /home/twuser/bin/tripwire.pl  -r|-u|-i  -h  host  [-f  updatefile]-or-[file(s)  to
update]
        -r run tripwire for specified host
        -u update tripwire for specified host
        -i initialize tripwire for specified host
        -f updatefile:  parse updatefile (saved tw output)

NOTE:   run     is an alias for /home/twuser/bin/tripwire.pl -r
        update  is an alias for /home/twuser/bin/tripwire.pl -u
        init    is an alias for /home/twuser/bin/tripwire.pl -i
```

```
twuser@twcon $ update -h host2 -f tmp/host2.twout
************************************************************
ERROR - please specify absolute path to updatefile
************************************************************

twuser@twcon $ update -h host4 -f /tmp/host4.twout
************************************************************
ERROR - updatefile specified does not exist
************************************************************

twuser@twcon $ update -h host5 -f /tmp/aaa
************************************************************
ERROR - /tmp/aaa does not appear to be an update file that is
      parseable
************************************************************
```

Last Updated: October 24th, 2017

Upcoming Training

San Diego Fall 2017 - SEC401: Security Essentials Bootcamp Style	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, United Arab Emirates	Nov 04, 2017 - Nov 16, 2017	Live Event
Community SANS Vancouver SEC401^	Vancouver, BC	Nov 06, 2017 - Nov 11, 2017	Community SANS
Community SANS Colorado Springs SEC401~	Colorado Springs, CO	Nov 06, 2017 - Nov 11, 2017	Community SANS
SANS Miami 2017	Miami, FL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Sydney 2017	Sydney, Australia	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS San Francisco Winter 2017	San Francisco, CA	Nov 27, 2017 - Dec 02, 2017	Live Event
SANS London November 2017	London, United Kingdom	Nov 27, 2017 - Dec 02, 2017	Live Event
Community SANS St. Louis SEC401	St Louis, MO	Nov 27, 2017 - Dec 02, 2017	Community SANS
Community SANS Portland SEC401	Portland, OR	Nov 27, 2017 - Dec 02, 2017	Community SANS
SANS Khobar 2017	Khobar, Saudi Arabia	Dec 02, 2017 - Dec 07, 2017	Live Event
SANS Munich December 2017	Munich, Germany	Dec 04, 2017 - Dec 09, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Dec 04, 2017 - Dec 09, 2017	Community SANS
SANS Austin Winter 2017	Austin, TX	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Bangalore 2017	Bangalore, India	Dec 11, 2017 - Dec 16, 2017	Live Event
SANS vLive - SEC401: Security Essentials Bootcamp Style	SEC401 - 201712,	Dec 11, 2017 - Jan 24, 2018	vLive
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Cyber Defense Initiative 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Dec 14, 2017 - Dec 19, 2017	vLive
Community SANS Nashville SEC401^	Nashville, TN	Jan 08, 2018 - Jan 13, 2018	Community SANS
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
Community SANS Hawaii SEC401	Honolulu, HI	Jan 08, 2018 - Jan 13, 2018	Community SANS
Mentor Session - SEC401	Memphis, TN	Jan 09, 2018 - Mar 13, 2018	Mentor
Northern VA Winter - Reston 2018	Reston, VA	Jan 15, 2018 - Jan 20, 2018	Live Event
SANS Amsterdam January 2018	Amsterdam, Netherlands	Jan 15, 2018 - Jan 20, 2018	Live Event
Mentor Session - SEC401	Minneapolis, MN	Jan 16, 2018 - Feb 27, 2018	Mentor
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
Las Vegas 2018 - SEC401: Security Essentials Bootcamp Style	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	vLive
SANS Miami 2018	Miami, FL	Jan 29, 2018 - Feb 03, 2018	Live Event