Interested in learning more about security?

SANS Institute InfoSec Reading Room

Securing Solaris Servers Using Host-based Firewalls

This paper will cover the addition of security to several Solaris servers through the use of host-based firewall software. The servers reside on an unsecured university network. I will attempt to detail the choices that were made in the selection of the firewall software as well as its installation and configuration. I will conclude with a comparison of the vulnerability assessments from before and after. First, I'll describe the topology of the network, and then I'll describe the various servers and their roles on the...

Securing Solaris Servers Using Host-based Firewalls

William Kirt Karl

GSEC Assignment version 1.4b option 2

Abstract:

This paper will cover the addition of security to several Solaris servers through the use of host-based firewall software. The servers reside on an unsecured university network. I will attempt to detail the choices that were made in the selection of the firewall software as well as its installation and configuration. I will conclude with a comparison of the vulnerability assessments from before and after.

Before:

First, I'll describe the topology of the network, and then I'll describe the various servers and their roles on the network. These two pieces of information will set the stage for a discussion of the current security stance of the computers to be protected. I'll talk about the threats, risks, and vulnerabilities that face each machine. Next will be a discussion of the reasons for implementing a firewall setup. Finally, the details of the setup as well as before and after assessment of the servers will be presented.

The local area network (LAN) is composed of 2 class B address ranges. The LAN is located at a major university with more than 48,000 students. The physical network is maintained by a central unit on campus (Data Communications Operations or DCO), but servers on the network are the responsibility of individual departments and organizational units. There is no appreciable filtering done at the border of the LAN. The separation between the duties of system administrators and network engineers has led to a situation in which network security is difficult and expensive to implement.

The three servers discussed in this paper are part of the information technology (IT) department a branch campus. They perform the various functions as described below: The first server, WWW, is a web server that hosts several websites for the campus. It runs on a Sun Enterprise 250 (E250) with Solaris 7 as its operating system (OS). The second server is GIS, which also runs on a E250 with Solaris 7. This server runs a number of geographic information systems (GIS) applications as well as hosting print services for a group of Linux clients. The final server described in this paper is Folio, Sun Ultra 10 running Solaris 8. It hosts a web application that allows students to maintain a portfolio of projects representing their work as they progress through the curriculum of their program. Each of these servers is located on a logically distinct subnet of the branch campus LAN.

Each one of the servers mentioned above is protected from the network by following many of the recommendations of the security community. The measures that were taken were based largely on the SANS document "Solaris Security: Step-by-Step"[1]. Precautions also included regular backups and system patches. Unfortunately, security

[1] Pomeranz

is only a small part of many job responsibilities. The initial goal was to increase the overall security of the servers as efficiently as possible. That goal is what led to this practical.

Though efforts were made to secure the machines, it is still necessary to look at their vulnerabilities, the level of threat that they face, and consequently the risks that are posed to them. Toward the end of this section, I'll examine the different attacks that could be launched against them and why the attacks represent risk.

The systems may be vulnerable in a number of ways. First are vulnerabilities found in the OS. These vulnerabilities become greater as it becomes easier to identify the OS of a machine. Next are vulnerabilities in the necessary services the computers are providing (primary services). HTTP has been a common target for hackers to attack[2], but it is a necessary evil for Folio and WWW. Though X windows is running over secure shell (SSH), it still requires remote procedure that call (RPC) be enabled in order to use the desktop environment provided by the GIS. Even though RPC is not a primary service provide by GIS it is still a vulnerability.

An unprotected university network is a dangerous place[3]. Such networks are defined by the academic freedom of the people associated with the university and an openness to the world in general. A free exchange of knowledge makes it very difficult to prevent outside traffic from entering the LAN. Unfettered traffic from across the world makes university networks a prime target for hackers everywhere. The policy of academic freedom extends down to the rules regarding what students and professors can and can't install on their computers. This further complicates protecting servers since all of that software adds additional threat vectors that could bypass a border firewall. With so many people pursing so many different interests on the internet, it is very difficult for DCO to adequately protect the LAN. The security of particular machines is then left up to individual system administrators. The lack of protection at the border of the university is a very large source of threat to the computers.

After looking at the threats and vulnerabilities, you can see that each of the servers faces a fair amount of risk. The risk can be alleviated by: 1. Limiting which networks have access to particular ports on each server and, 2. by completely shutting off access to vulnerable secondary services that are necessary but don't need to be directly accessible. One of the ways to add this type of security is to add a firewall. In this case, a firewall will add a layer of security to the servers, making them much less susceptible to attack.

Why is this necessary? What sorts of attacks could be launched against the servers? Each server is susceptible to attack in its own way, but all are important. The following are some of the attacks that could be suffered by each of the servers. Availability: WWW, as the face of the campus to prospective students, must be accessible most if

[2] Cox

[3] Lemos

not all of the time. The other two servers need to be available for students and researchers to use when necessary. Integrity: GIS contains research data that can be irreplaceable and WWW needs to show an accurate representation of the campus. Students also need to know that years worth of work will be kept secure. Confidentiality: Folio contains not just students work but also other information about them, including ID numbers.

During:

In the previous section I laid out the basic state of affairs with regard to the security of 3 Sun Solaris servers, WWW, Folio, and GIS. This section will initially deal with why firewalls are important. Then, I'll discuss what decisions led to the application of host-based firewalls on each server, and also how a decision on a particular firewall was reached. After that discussion, I will show how each firewall was configured and the results of before and after testing.

First I'll discuss why I saw a need for firewalls at all. As seen in the opening section, the computers were relatively well hardened via traditional techniques as time permitted. It was also demonstrated that they exist on a very hostile network, and that they perform functions that need to be assured of confidentiality, integrity, and availability. One of the guiding principles I have learned through my study of SANS course material is the need for "defense in depth"[4]. As part of that strategy, it is often recommended that firewalls be put in place to protect computers on a network.

Firewalls protect in a number of ways: they allow system administrators to control what services specific hosts or subnets have access to; they can completely block any services that should not be reachable from the network; finally, they can also prevent information from flowing out from the LAN to the internet. All of these functions help to mitigate the threat posed by a hostile network.

After deciding that a firewall was necessary, I began defining criteria with which to pick a solution. It was also necessary to start gathering information about the various solutions that were available to me.

The criteria that I decided on were as follows:

- First, it needed to be cheap. The cheaper, the better, with free being the ultimate goal. The university budget does not allow for many additional expenses. If I could set up an inexpensive solution, it may be possible to convince management of the need to implement a more costly but better suited solution later.
- The firewall had to be accessible by my department. Without direct control, it can be difficult to find help when it's needed. It is also easier to mold the firewall to the specific application if you control both the firewall and the hosts that it's protecting.

[4] Huston

- The firewall needed to be easy to configure and maintain. The resources of the department were extremely taxed and simplicity and ease of setup would be crucial. A simpler setup would also mean less chance of configuration errors.

  > "Simplicity is a security strategy for two reasons. First, keeping things simple makes them easier to understand; if you don't understand something, you can't really know whether or not it's secure. Second, complexity provides nooks and crannies for all sorts of things to hide in;…"[5]

- An obvious final criterion would be the need for any firewall software to run on the appropriate platform.

After establishing this list of criteria by which I would judge each candidate, I began exploring solutions to the problem. The first solution to present itself was a dedicated hardware appliance set up and maintained by DCO. The second was dedicated hardware running firewall software set up by our department. Lastly, there were several host-based solutions that could be applied to the servers themselves. Below, I'll examine the pros and cons of each solution and explain how I arrived at my final decision.

First I considered a dedicated firewall appliance that could be provided for and maintained by the network engineers who run the campus infrastructure. Their choice of appliance was a Nokia unit running IPSO with Firewall One installed (http://www.nokia.com/securitysolutions/platforms/index.html). This particular appliance has received good reviews. Dedicated hardware tuned to a specific purpose makes these firewalls very capable performers on a LAN. The units themselves are expensive, and there is an ongoing fee for licensing and an additional fee for management by DCO. The cost would be prohibitive in this instance. Because of cost considerations, it would be necessary to cluster the servers together behind a single firewall appliance. Moving the servers adds time, effort, and complexity to the project. Also, because these firewalls are maintained by a different group of people, it is more difficult to control the types of rule sets that are implemented. That lack of control by the department is also difficult to overcome. The negative factors outweighed the positives for the use of these appliances.

The next solution to be considered was firewalling the servers using a dedicated, general purpose computer with a packet filtering firewall installed. This would alleviate some of the negatives associated with the Nokia solution, while maintaining many of the positives. The firewall would be under the department's control to configure as necessary. This would also make ongoing maintenance easier, as it would involve fewer people. It would be less costly than the other appliances, and wouldn't have any recurring investment to consider [such as subscription costs, etc.]. There are some downsides to consider with this possibility. The prime concern is once again the cost of this installation. Setting up one box would be over our budget, let alone setting up one

[5] Zwicky, et al

for each server. Moving all of the servers to be protected by a single firewall introduced some additional concerns. One concern was that as the number of computers behind the firewall grows, the rule sets protecting them become more complex, harder to manage, and harder to test. The choke-point firewall can also limit bandwidth and, in the event of a failure, leave all three servers without connectivity.

The final solution I considered was to have the servers protect themselves using host-based packet filters. This was the most cost effective solution (which I will discuss below). Also, host-based packet filters have simpler rule sets that are easier to configure and test. There is no need to rearrange the network to accommodate a choke-point firewall. Because there is no single restrictive point of access to the LAN, the bandwidth of the protected network is not affected overall. Host-based firewalls also mean that there is no single point of failure. In addition, because I can control exactly what's installed on the server, these firewalls are less likely to be circumvented by other threat vectors[6].

Though host-based firewalls have a lot to recommend them, they also have a few drawbacks. In this instance, application of a host-based firewall meant configuring 3 separate setups, one for each server. That wasn't as bad as it sounds since the three servers were fairly similar in the services that they provide. A second drawback is that a host-based solution relies on the processing power of the host. This may slow down a heavily used server.

Now that I'd decided on a host-based solution, which one should be implemented? Again, the crucial considerations were ease of use and cost. An added bonus would be the general applicability of the firewall to more than one platform. After doing some research I considered several options. Many of them could be dismissed because they won't work on a Solaris machine (i.e. iptables and packet filter). Some were dismissed on the basis of price. One such utility was Sunscreen firewall software. It has a license that would cost upwards of $15,000. The free version, Sunscreen Lite, is not listed as being compatible with Solaris 7. Even if it operated on all of the servers, Sunscreen Lite is still a fairly complicated piece of software that is not at all easy to set up. The field was finally narrowed to IP Filter.

IP Filter met each one of my original criteria. It was inexpensive (free). IP Filter (IPF) is freely available from coombs.anu.edu.au/~avalon/, and precompiled binaries are available at http://www.maraudingpirates.org/ipfilter/. It was relatively easy to use. Its rule sets are straightforward and are written in almost plain English. IPF is also stateful which allows the software to keep track of active connections. Statefulness eliminates the need for complex or convoluted rules to allow outbound server connections to continue. If a dedicated piece of equipment were to be set up in the future, it would be possible to quickly come up with the rule set based on previous knowledge and experience. IPF also allows for logging.

[6] Bellovin

After deciding to use IPF, it was necessary to download, install, and configure the software on each server. I'll present the download and installation procedure, followed by the general guidelines for the configurations, and then a discussion of the configuration options specific to each of the three servers will be presented. This will be concluded with a description of the testing that was done on each host and a comparison of the before and after results.

Precompiled 64-bit binaries are available for Solaris 7 and 8 from www.maraudingpirates.org/ipfilter. I chose the package that was appropriate to my application and downloaded it to a working directory. I uncompressed the package, used pkgadd to install the software, and rebooted the system to allow the changes to take effect. What follows is an example of the configuration:

```
gunzip ipf-3.4.28-Sol8-sparc-64bit.pkg.gz
pkgadd -d ./ipf-3.4.28-Sol8-sparc-64bit.pkg
shutdown –y –i 6 –g 120
```

When the system comes back up it should have a copy of ipfilter installed and running. If this doesn't work, you can download the source and compile it yourself from scratch. Refer to the source distribution for more information on that version of the installation process.

After successfully downloading and installing the software, it is time to start configuration. One of the most helpful resources that I've found on ipf configuration is located at: www.obfuscation.org/ipf/ipf-howto.html[7]. Much of this section is derived from information found in that howto. I will give a very brief description of how to write rules as most of that information can be found elsewhere. What follows is a rule that allows an outside user to make a connection to the ssh service on one of the machines. Each server would have a similar rule to allow for secure remote logins.

```
pass in quick on hme0 proto tcp from any to <server's IP address> port
= 22 keep state
```

The fist word "pass" allows packets matching the rest of the rule to move through the firewall. "in" indicates that this rule is working on packets coming into the system. "quick" means that if the packet matches this rule, it should not be compared to any other rules in the rule set. If "quick" were not present, the packet would continue to be compared to each of the rules in the config file. The last one that matched would determine the fate of the packet (e.g. blocked, passed, logged, etc.). "on" indicates on which interface the rule is working (in this case the first Ethernet interface, hme0). "proto" is the type of protocol the packet has to be (tcp, upd, icmp) and is followed by the selection. "from" indicates the ip address that the packet is originating from. "to" is the destination of the packet, in this case the server that ipf is resident on. "port" is used to define the port that the packet is destined for. In this case that is port 22 because that is where the ssh service is running. Finally, "keep state" sets up the state table to track any connections matching this rule.

[7] Conoboy & Fichtner

Next, I'll discuss the other rules common to all of the servers. First is the default deny rules. These should appear at the end of the config file. This ensures that a packet that doesn't match any other rule doesn't go any further. The rules look like this:

```
block in on hme0
block out log on hme0
```

The first rule blocks all traffic coming into the first Ethernet interface by default. The second rule blocks all outgoing traffic that is not explicitly defined in rules earlier in the configuration. This rule also logs all blocked traffic. The logging has a two fold advantage. First, it lets you know if the server is trying to contact the outside world in ways that you didn't intend (e.g. there is a Trojan trying to open a back door). Secondly, the logging will be valuable in determining if the firewall is the cause of a service being inaccessible. If you find that the logs are being filled up by services that you know exist but can't turn off because of dependency issues, special rules can be created to block but not log them. (e.g. block out on hme0 proto up … a rule to not allow rpc out.).

The remainder of the rules are going to be specific to each host. I'll go through them one at a time starting with WWW.

```
pass in quick on hme0 proto tcp from any to <IP Address of WWW> port =
80 keep state
```

This rule allows incoming connections from anywhere to the http service running on the box.

Next is Folio. Folio hosts a web application so it will have a similar rule to WWW. Folio should only be accessible from university subnets. Though it isn't hard to spoof IP addresses, it still makes it a bit harder for an attacker to get in. Here is the rule:

```
pass in quick on hme0 proto tcp from <university networks> to <IP
Address of Folio> port = 80 keep state
```

Notice the substitution of the ip address range for the university instead of the keyword "any". The address range can be specified in CIDR format.

The final set of rules is for GIS. Recall that GIS serves X Windows connections, and is a print server for a number of Linux hosts. This will call for only a few additional rules. The X Windows connections are taken care of because X is tunneled over SSH. Therefore the first common rule that was defined takes care of X as well. The only other rules to institute are the ones that allows printing.

```
Pass in quick on hme0 proto tcp from <university networks> to <IP
Address of GIS> port = 515 keep state
Pass out quick on hme0 proto tcp from <GIS's ip address> to <IP Address
of each printer> keep state
```

That takes care of the specific rules for each server. This completely does away with other ports that may be open but not necessary for the outside network. It also effectively limits certain connections to specific subnets. At this point you can add rules for ip ranges that should never be seen on the internet. See www.obfuscation.org/ipf/ipf-howto.html - TOC_56 for a complete list of all the unroutable address that can be blocked. This isn't strictly necessary but it adds another layer of security.

The next step was to make sure that the servers were still providing the services necessary. This was accomplished by attaching to each service both from within the university network and outside. The results were as expected: WWW allowed SSH and http requests; GIS allowed X Windows and printing from the LAN, but not from outside of it; and Folio only allowed http requests that originated from the university network.

After all of the rules are put into place, the servers should now be able to accept connections only to services that are intended to be left open only from machines that are permitted to have access to those services.

None of these configurations are worthwhile unless they have been tested. In order to test each box and provide a before and after snapshot, I used several tools, including: nmap[8], SuperScan from Foundstone, and ISS. The scans were conducted from inside the University LAN. The results were encouraging.

Before firewall implementation, ISS showed that between the 3 servers, there were 2 high risk vulnerabilities, 10 medium risk vulnerabilities, and 1 low risk vulnerability. These included vulnerable secondary services such as rpc. After firewall implementation, there were 4 medium risk vulnerabilities and 1 low risk vulnerability.

Nmap was used to scan the hosts with the following options –sS (stealth scan) –O (host fingerprinting) –P0 (don't ping) –v (verbose output) –p 1-65535 (scan every port in the range). Before application of the packet filters nmap showed a combined total of 17 open ports and correctly identified all three operating systems. After the firewall rules were in place, nmap only detected 5 open ports total (3 ssh, 2 http) and only correctly identified one operating system.

SuperScan showed less complete results when compared with nmap. It identified only 15 open ports before implementation and 5 after. SuperScan was run using the options to scan unresponsive pings (needed because after the firewall setup, the hosts don't return icmp requests), and scan "every port in the list" (1-65535). It still only showed the ports open that were intended to be open.

The fact that these computers were already in a production environment precluded harsher techniques and launching direct attacks. A natural next step from this research would be to set up a test machine similar to the others and attempt full scale cracks. Unfortunately both time and money are not currently available to do that.

[8] Fyodor

After:

The addition of firewalls to each of the servers helped to enhance their security in a couple of ways. First, each server was left with only those ports that are absolutely necessary open to the internet. This has the effect of limiting the possible attacks on the servers to only those ports / services that are supposed to be running. A remote attacker can only attack what he can get to across the network. By limiting the possible attacks the system administrator can concentrate precious time and energy on only a few purposely exposed services. Second, connections to the exposed services can be further limited to only subnets / machines that need access. There is no sense in exposing an interdepartmental web server to the whole internet. This further limits the exposure of a particular service to potential attacks by a much smaller group of machines. That means that an attacker will either have to spoof their IP address or they will first have to compromise another machine to begin even launching an attack on the server.

This security strategy also resolves the problem of having secondary services that are needed by the local machine exposed to the internet. By adding a firewall the local machine still has access to any services it needs (such as MySQL on port 3306) but those same services are not available to the outside world. Each additional service constitutes another possible vulnerability and increases the overall risk to the system.

Though the plan outlined in this paper decreases risks to the machines, it does not eliminate them. First and foremost, it is still necessary to be diligent in checking for security vulnerabilities in the services that were purposely left open. Also, there is still the need for defense in depth. That includes general OS and program patching, chrooting exposed services, logging (one of the most important), and ongoing vulnerability testing. The only way to completely eliminate exposure to the internet is to unplug the network cable. Unfortunately, that makes the machines useless.

The plan is also limited to addressing network based attackers. Physical security is still important. Also limiting who has legitimate access to the servers is critical. A lot of damage can still be done by someone who is designated as having legitimate access.

Impact:

The initial goal of this practical was to add an additional layer of security to each of three UNIX servers running various flavors of Solaris. The additional security was necessary to mitigate the risk associated with existence on an unprotected university network. Based on many recommendations by the security community, it was decided to firewall the servers to prevent unauthorized access to university resources. A number of firewall solutions were weighed against one another based on specific criteria set out ahead of time. After a host-based solution was decided on, the decision was further narrowed to a particular implementation. A set of firewall rules was then implemented with the goal

of limiting outside access in mind. Each server had a set of rules specific to its function. Each host was then tested using a variety of vulnerability scanners. The tests produced favorable results accomplishing the overall goal of the practical.

The overall impact of this practical was to help not only secure the machines but also to make the network a safer place. Preventing the network compromise of the servers will prevent their use in further attacks.

Though there are a number of topics that were not discussed in this practical, they are no less important. These include such things as reevaluation of the security concerns facing the network, fine-tuning the firewall rules to improve performance, and staying vigilant with respect to security concerns that arise from the services that are still available on the network.

In conclusion, the use of host-based firewalls enables system administrators to secure machines without disrupting their network configuration. These firewalls also allow for easier rule maintenance and testing without the shortcomings of traditional chokepoint firewalls.

References:

1. Englund, Martin. “Securing Systems with Host-Based Firewalls – Implemented with Sunscreen Lite 3.1 Software.” Sun BluePrints Online, 09/01/01. URL: http://www.sun.com/blueprints/0901/sunscreenlite.pdf (05/20/02).

2. Russel, Rusty. “Linux 2.4 Packet Filtering HOWTO.” Version 1.2. 02/19/02. URL: http://netfilter.samba.org/documentation/HOWTO//packet-filtering-HOWTO.html (05/20/02).

3. Dibowitz, Phil. “IP Filter FAQ.” 05/18/02. URL: http://home.earthlink.net/~jaymzh666/ipf/index.html (05/20/02).

4. Fyodor. “nmap man page.” URL: http://www.insecure.org/nmap/nmap_manpage.html (07/08/02)

5. “Internet Scanner Technical Overview.” 12/02. URL: http://documents.iss.net/whitepapers/IS_TechOverview.pdf (07/08/02).

6. Zwicky, Elizabeth, et al. Building Internet Firewalls (2nd Edition). Sebastopol: O’Reilly & Associates, Inc., 2000.

7. Lemos, Robert. “University Systems a Haven for Hackers.” 05/02/02. URL: http://news.com.com/2100-1001-898084.html (06/27/02).

8. Pomeranz, Hal ed. “Solaris Security Step by Step, Version 2.0.” Sans Institute. URL: http://www.sans.org/newlook/publications/solaris_toc.htm (07/26/02)

9. Cox, Mark. “Overview of security vulnerabilities in Apache httpd 1.3.” 11/16/01. URL: http://www.apacheweek.com/features/security-13 (07/08/02)

10. Conoboy, Brendan and Fichtner, Eric. “IP Filter Based Firewalls HOWTO” 03/1/02. URL: http://www.obfuscation.org/ipf/ipf-howto.html (07/27/02)

11. Burgiss, Hal. “Security Quick-Start HOWTO for Red Hat Linux” 07/21/02. URL: http://www.tldp.org/HOWTO/Security-Quickstart-Redhat-HOWTO/index.html (07/27/02)

12. Bellovin, Steven. “Distributed Firewalls” 11/01/99. URL: http://www.research.att.com/~smb/papers/distfw.pdf (07/20/02)

13. Sidel, Scott. “Test Center-New Products Tested in Real-World Environments” 09/01/2001. URL: http://www.infosecuritymag.com/articles/september01/departments_products1.shtml (07/15/02)

14. Huston, Brent. “A Higher View of Defense in Depth” 02/20/2002. URL: http://www.itworld.com/nl/security_strat/02202002/ (09/31/02)

Last Updated: December 17th, 2017

Upcoming SANS Training

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

SANS Security East 2018	New Orleans, LAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Amsterdam January 2018	Amsterdam, NL	Jan 15, 2018 - Jan 20, 2018	Live Event
Northern VA Winter - Reston 2018	Reston, VAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SEC599: Defeat Advanced Adversaries	San Francisco, CAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SANS Dubai 2018	Dubai, AE	Jan 27, 2018 - Feb 01, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NVUS	Jan 28, 2018 - Feb 02, 2018	Live Event
Cyber Threat Intelligence Summit & Training 2018	Bethesda, MDUS	Jan 29, 2018 - Feb 05, 2018	Live Event
SANS Miami 2018	Miami, FLUS	Jan 29, 2018 - Feb 03, 2018	Live Event
SANS Scottsdale 2018	Scottsdale, AZUS	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS London February 2018	London, GB	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Southern California- Anaheim 2018	Anaheim, CAUS	Feb 12, 2018 - Feb 17, 2018	Live Event
SANS Secure India 2018	Bangalore, IN	Feb 12, 2018 - Feb 17, 2018	Live Event
SANS Dallas 2018	Dallas, TXUS	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Brussels February 2018	Brussels, BE	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Secure Japan 2018	Tokyo, JP	Feb 19, 2018 - Mar 03, 2018	Live Event
Cloud Security Summit & Training 2018	San Diego, CAUS	Feb 19, 2018 - Feb 26, 2018	Live Event
SANS New York City Winter 2018	New York, NYUS	Feb 26, 2018 - Mar 03, 2018	Live Event
CyberThreat Summit 2018	London, GB	Feb 27, 2018 - Feb 28, 2018	Live Event
SANS London March 2018	London, GB	Mar 05, 2018 - Mar 10, 2018	Live Event
SANS San Francisco Spring 2018	San Francisco, CAUS	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS Secure Osaka 2018	Osaka, JP	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS Secure Singapore 2018	Singapore, SG	Mar 12, 2018 - Mar 24, 2018	Live Event
SANS Paris March 2018	Paris, FR	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS SEC460: Enterprise Threat Beta	OnlineCAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced