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.

Practical Assignment by Alexei Proskura.

Preface:

Please excuse my English it is not even my 4th language.

Detects presented here are from my home and office/colocation networks. In all used detects networks are masked with name.my.net alias. If you will find anything worth restoring correlation please let me know.

NOTE ON ACTIVE TARGETING PLEASE READ.
I was puzzled when I saw "Active Targeting = YES" in quite a lot of detects presented by GIAC students. To me "Active Targeting" is when "Target" is clearly defined. Means the OS is known and the list of potential vulnerabilities is created and attacker is trying to exploit one of them on a given host. I do not consider "Active Targeting" as trolling for a _specific_ service on _any_ computer.
It is quite stupid to target PCAnywhere on my OpenBSD box. Everywhere in my detects I stick to the above definition of active targeting.

Detect 1:

```
[**] IDS152/Ping BSDtype [**]
05/26-14:14:45.247510 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:12863  DF
ID:4268   Seq:0  ECHO

[**] IDS152/Ping BSDtype [**]
05/26-14:14:46.244672 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:12864  DF
ID:4268   Seq:1  ECHO

[**] IDS152/Ping BSDtype [**]
05/26-14:14:47.243559 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:12865  DF
ID:4268   Seq:2  ECHO

[**] IDS152/Ping BSDtype [**]
05/26-14:19:51.096527 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:56539  DF
ID:4782   Seq:0  ECHO

[**] IDS152/Ping BSDtype [**]
05/26-14:19:52.089740 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:56540  DF
ID:4782   Seq:1  ECHO

[**] IDS152/Ping BSDtype [**]
05/26-14:19:53.089125 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:56541  DF
ID:4782   Seq:2  ECHO

[**] IDS152/Ping BSDtype [**]
05/27-14:34:25.509221 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:13375  DF
ID:6038   Seq:0  ECHO
```

© SANS Institute 2000 - 2002, Author retains full rights.

```
[**] IDS152/Ping BSDtype [**]
05/27-14:34:26.504397 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:13376  DF
ID:6038   Seq:1  ECHO

[**] IDS152/Ping BSDtype [**]
05/27-14:34:27.503776 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:13377  DF
ID:6038   Seq:2  ECHO

[**] IDS152/Ping BSDtype [**]
05/27-14:35:52.119168 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:34439  DF
ID:6209   Seq:0  ECHO

[**] IDS152/Ping BSDtype [**]
05/27-14:35:53.110666 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:34440  DF
ID:6209   Seq:1  ECHO

[**] IDS152/Ping BSDtype [**]
05/27-14:35:54.113983 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:34441  DF
ID:6209   Seq:2  ECHO

[**] IDS152/Ping BSDtype [**]
05/27-14:36:18.820235 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:34442  DF
ID:6270   Seq:0  ECHO

[**] IDS152/Ping BSDtype [**]
05/27-14:36:19.812218 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:34443  DF
ID:6270   Seq:1  ECHO

ls [**] IDS152/Ping BSDtype [**]
05/27-14:36:20.817499 195.2.83.226 -> gateway.my.net
ICMP TTL:242 TOS:0x0 ID:34444  DF
ID:6270   Seq:2  ECHO
```

1.1 Source of trace.
gateway.my.net
1.2 Detect was generated by
Snort 1.6 with arachNIDS. Operating system is OpenBSD pre 2.7 snapshot
1.3 Probability the source address is spoofed
Close to 0. The source address is real (they suppose to get echo-replies) and belongs to zenon.net (Russian ISP part of aha.ru)
1.4 Description of attack.
Strange interest to my network from Russian ISP. The pattern insinuate that that was just a network sweep. Broad recon.
1.5 Attack mechanism
This pings appeared at my gateway by groups of three, one ping per second. So we have 5 groups 3 pings each in two days. IP IDs are sequential by groups. But seems like "pinger" host was quite busy between the first and the second attempts. IP ID jumped by approx 44500 in 5 minutes. So was it pinging other hosts with the same speed - probably no (one per second ping and sequential increment of IP IDs should give us increment by approx 300.My quite wild

© SANS Institute 2000 - 2002, Author retains full rights.

guess would be a file transfer.
In 1 day and 20 minutes my guests are back with more interesting pattern. Another big jump in IP IDs between first and second group and sequential IP IDs through second and third group. My gateway drew closer attention?
The starting IP IDs are close to yesterdays - so probably script or tool with fixed starting IP Ids. The TTLs are the same so same host – same route.

1.6 Correlation

I had no visitors from this address as far as my scope goes except this two days.

1.7 Evidence of active targeting

I do not think so. I would assume tidy network sweeping for host discovery.

1.8 Severity

critically=5 (gateway)
lethality=0 (recon, no attack yet)
system coutermeasures=3 (older system, always busy, hard to patch)
network countermeasures=5 (firewall dropping ICMP)
SEVERITY= -3

1.9 Defensive recommendations

Patch gateway, run security check on it.

1.10 Question.

What valid response should be on presented stimulus
a) TCP RST
b) ICMP ECHO-REPLY
c) ICMP RST
d) TCP ECHO-REPLY
Answer b

Detect 2.

```
[**] IDS188/probe-back-orifice [**]
05/27-11:08:58.210095 63.21.88.112:31337 -> gateway.my.net:31337
UDP TTL:116 TOS:0x0 ID:16475
Len: 27
```

2.1 Source of trace

gateway.my.net

2.2 Detect was generated by

Snort 1.6 with arachNIDS. Operating system is FreeBSD 3.4

2.3 Probability the source address is spoofed

0 - this guy want to get respond on his BackOrifice ping

2.4 Description of attack

Attempt to find if BackOrifice trojan was installed on my gateway.

2.5 Attack mechanism

BackOrifice is a trojan program usually planted in e-mail attachments or "internet utilities". Very popular place to plant BackOrifice is the "download speedup" programs. Runs on Windows (UNIX client is available). When executed installs "server" side on the compromised computer. "Server" allows "client" (attacker machine) to take control over compromised host. 31337 is the default port "server" binds to. First "public" appearance is August 1998. Major revision one in July 1999 named BackOrifice 2K (BO2K). Features include keystroke
capturing, remote file management (up/download), remote command shells access, direct registry editing, TCP connection redirection, etc.

2.6 Correlation

Script kiddies outdoors. Host address back resolves to the uunet customer address. There is the possibility also that the customers' computer was compromised and now used to do further recon. No correlation that could

© SANS Institute 2000 - 2002, Author retains full rights.

be tied to the above detect in my scope of visibility. Several unrelated detects a month. I would guess it happens more often on home (@home, pacbell) networks than on production sites.

2.7 Evidence of active targeting

No. It was not even Windows machine. Going after specific service but without previous knowledge of OS.

2.8 Severity

Criticality - 5 (gateway)
Lethality - 1 (attack is unlikely to succeed)
System countermeasures - 5 (this beast do not run under *BSD, yet)
Network countermeasures - 5 (restrictive firewall)
SEVERITY = -4

2.9 Defensive recommendations

We are fine for now. Watch for possible return. If this a home network install BackOrifice friendly - this should not hurt your GIAC certification.

2.10 Question

If above traffic is the only traffic between two computers you can tell that BackOrifice active on
a) both computers
b) destination computer
c) source computer
d) not active
Answer d

Detect 3

May 26 02:38:22 firewall ipmon[30168]: 02:38:22.500799
le0 @0:11 b 204.157.27.201,1079 -> firewall.my.net,161 PR udp len 20 72

3.1 Source of trace

firewall.my.net

3.2 Detect was generated by

Ipfilter firewall. OS is OpenBSD 2.6

3.3 Probability the source address is spoofed

0 - respond is wanted.

3.4 Description of attack

Attempt to discover SNMP agent on my firewall. SNMP based on request-reply schema and runs over UDP (port 161,162)

3.5 Attack mechanism

Can use one of defined PDU (Protocol Data Unit) to change object values in the MIB database (GetRequest, SetRequest), discover MIB objects (GetNextRequest). Can lead to routing tables modification, interface status modification (and as a result changes in effective routing).

3.6 Correlation

Strange, but this was a very lonely attempt. There were no attempts to exploit SNMP nor from that, nor from other addresses. At least up to the time I am writing this.

3.7 Evidence of active targeting

I do not think so BUT ... Unfortunately I can not check upper router logs... Maybe guy was going down the router path. In this case my firewall is the default route on the upper level router for my routable network so maybe we can suspect network mapping. Or the targeted system is upstream.

3.8 Severity

Criticality - 5 (gateway)
Lethality - 1 (attack is unlikely to succeed, there no SNMP there))
System countermeasures - 5 (It is simply not there)
Network countermeasures - 5 (restrictive firewall)

© SANS Institute 2000 - 2002, Author retains full rights.

SEVERITY = -4

3.9 Defensive recommendations

Check all routers for writable attributes to default ("public" and "private") communities and rename the communities. Filter SNMP traffic from bad guys at router ACL.

3.10 Question

In the above detect destination port is associated with following service

a) SMPN
b) SNMP
c) SMTP
d) SPNM

Answer b

Detect 4.

```
[**] IDS277/named-probe-iquery [**]
05/26-03:27:27.287542 141.106.40.42:3737 -> nameserver.my.net:53
UDP TTL:50 TOS:0x0 ID:17452
Len: 35

[**] IDS277/named-probe-iquery [**]
05/31-21:34:45.733011 141.106.40.42:3737 -> gateway.my.net:53
UDP TTL:50 TOS:0x0 ID:55251
Len: 35
F7 A0 09 80 00 00 00 01 00 00 00 00 00 00 01 00  ................
01 00 00 7A 69 00 04 04 03 02 01                 ...zi......


May 31 21:34:46 gateway ipmon[30168]: 21:34:44.386556
le0 @0:11 b 141.106.40.42,3737 -> gateway.my.net,53 PR udp len 20 55
```

4.1 Source of trace

gateway.my.net

4.2 Detect was generated by

Ipfilter firewall. OS is OpenBSD 2.7 snapshot.

4.3 Probability the source address is spoofed.

0 - respond wanted. Discovery of service that could be exploited. TTLs are same so same host -same route.

4.4 Description of attack

Attempt to discover and exploit (not very) recent buffer overflow in named. Crafted tool uses regular syscalls.

4.5 Attack mechanism

As lot of the boxed *NIX operation systems installing BIND by default and sometimes system administrators are not aware of orphan name servers running. The buffer overflow vulnerability in named daemon can lead to execution of the code with the same privileges named daemon was running.
The packet that was stopped by my firewall and noted by Snort contains signature that advises that it could be possibly generated by modified source of iquery.c by ROTShB. The original source of exploit (iquery.c) was able to recognize following versions of bind and exploit buffer overflow vulnerability:

```
struct target_type target[] =
{
  {"x86 Linux 2.0.x named 4.9.5-REL (se)",0,0xbffff21c,0xbffff23c,4},
  {"x86 Linux 2.0.x named 4.9.5-REL (le)",0,0xbfffeedc,0xbfffeefc,4},
```

© SANS Institute 2000 - 2002, Author retains full rights.

```
  {"x86 Linux 2.0.x named 4.9.5-P1 (se)",0,0xbffff294,0xbffff2cc,4},
  {"x86 Linux 2.0.x named 4.9.5-P1 (le)",0,0xbfffef8c,0xbfffefb4,4},
  {"x86 Linux 2.0.x named 4.9.6-REL (se)",0,0xbffff3e3,0xbffff403,4},
  {"x86 Linux 2.0.x named 4.9.6-REL (le)",0,0xbffff188,0xbffff194,4},
  {"x86 Linux 2.0.x named 8.1-REL (se)",0,0xbffff6a4,0xbffff6f8,5},
  {"x86 Linux 2.0.x named 8.1-REL (le)",0,0xbffff364,0xbffff3b8,5},
  {"x86 Linux 2.0.x named 8.1.1 (se)",0,0xbffff6b8,0xbffff708,5},
  {"x86 Linux 2.0.x named 8.1.1 (le)",0,0xbffff378,0xbffff3c8,5},
  {"x86 FreeBSD 3.x named 4.9.5-REL (se)",1,0xefbfd260,0xefbfd2c8,4},
  {"x86 FreeBSD 3.x named 4.9.5-REL (le)",1,0xefbfd140,0xefbfd1a8,4},
  {"x86 FreeBSD 3.x named 4.9.5-P1 (se)",1,0xefbfd260,0xefbfd2c8,4},
  {"x86 FreeBSD 3.x named 4.9.5-P1 (le)",1,0xefbfd140,0xefbfd1a8,4},
  {"x86 FreeBSD 3.x named 4.9.6-REL (se)",1,0xefbfd480,0xefbfd4e8,4},
  {"x86 FreeBSD 3.x named 4.9.6-REL (le)",1,0xefbfd218,0xefbfd274,4},
  {{0},0,0,0,0}
};
```

The data boundaries are not checked in all of above versions of bind when inverse query request is received.

4.6 Correlation

One of the most popular attacks. It is in SANS top 10. Every day in SANS detect. Couple of times a week probes on my networks. The list of visitors is long and that is why not presented here.

4.7 Evidence of active targeting.

Yes. Every domain has a name server. So every domain is a potential target. The first target was name server in SOA, and than probable orphan name server.

4.8 Severity

Criticality - 5 (DNS)
Lethality – 5 (attacker can gain root across net)
System Countermeasures – 5 (OpenBSD named clinically proven)
Network countermeasures – 5 restrictive firewall
SEVERITY – 0

4.9 Defensive recommendations.

Read bugtraq – update nameds. Better in critical points run secure operating system (OpenBSD in this case). Update to the latest version of bind.
If this is a home, research network or honeypot and you are running bind 8 set

```
Options {
      version “4.9.6”;
      }
```

and wait – you will get some useful information in your IDS soon.

4.10 Question

The above detect shows
a) Host discovery.
b) DNS query.
c) DNS attack.
d) Zone transfer attempt.
Answer b.

Detect 5.

```
May 30 12:20:15 perimeter ipmon[32240]: 12:20:15.106323
le0 @0:19 b 24.7.194.60,1426 -> perimeter.my.net,5632 PR udp len 20 30  IN
May 30 12:20:15 perimeter ipmon[32240]: 12:20:15.111889
le0 @0:19 b 24.7.194.60,1426 -> perimeter.my.net,22 PR udp len 20 30  IN
```

5.1 Source of detect

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

One of the computers on my.net perimeter. Has only external interface.

5.2 Detect was generated by

Ipfilter firewall. OS is OpenBSD 2.6 patched.

5.3 Probability the source address was spoofed.

0 – this is home computer on @Home cable modem network. Backresolves to c970174-a.frmt1.sfba.home.com. Could be compromised though.

5.4 Description of attack.

More computers at home means more need to access from outside. Easy solution for Windows is PC anywhere. Attempt to find PC anywhere installations.

5.5 Attack mechanism

There are several tricks that could be performed with PC anywhere.

1) Easy DoS attack affects computer.
2) Easy DoS attack affects application itself.
3) PEOPLE, PLEASE CHANGE THE DEFAULT PASSWORDS.
4) Default encryption algorithm in certain versions weak and password can be sniffed and decrypted.

First attack ends with computer 100% processor utilization and hanged computer. All you need is connect to PCAnywhere port (5632) and on kind prompt "Please Press <Enter>" enter a decent amount of garbage. The exploit was discovered by pasting a binary file, but any significant (100s K) amount will work. Use PCAnywhere 8.0 for fun.

Second attack works with 8.01, 8.02, 9.0, 9.2 and hangs only application itself. TCP connect scan with nmap or similar tool will do the job.

Third attack is old like the world itself and will live longer that we all.

Third. Default passwords should be changed immediately after installation. Also your name and the name of your dog is probably not the best choice.

Last. The encryption in 9.0 is not the strongest in the world and the simple thing like

```
void main() {

  char password[n];
  char cleartext[n];
  int  i;

  password[0]=0xfist_hex_value_of_sniffed_password;
  password[1]=0xsecond_hex_value_of_sniffed_password;
  password[x]=0x…....;
  password[x]=0x…....;
  password[n]=0xlast_hex_value_of_sniffed_password;
  password[n+1]='\0';

      cleartext[0]=0xca-password[0]+0x61;
      for (i=1;i<strlen(password);i++)
        cleartext[i] = password[i-1] ^ password[i] ^ i-1;

      cleartext[strlen(password)]='\0';

      printf("password is %s \n",cleartext);

}
```

will present you password in clear text. On cable modem networks where sniffing is not extremely difficult because of cable network nature this is a particular threat.

5.6 Correlation.

Often. On the home network of my friends. You can also get some ideas from SANS detect on December 31, 1999.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

5.7 Evidence of active targeting
Do not think so. More looks like scanning all the neighbors for PCAnywhere. And the system is running wrong"(for PCAnywhere) operation system.
5.8 Severity
Criticality - 3 (important host)
Lethality - 1 (attack is very unlikely to succeed)
System Countermeasures - 5 (PCAnywhere does not run on UNIX boxes thanks Symantec)
Network countermeasures - 5 restrictive firewall
SEVERITY (-6)
5.9 Defensive recommendations
Check for latest update from Symantec. Use personal firewalls on windows and restrict access. Change the default passwords. Use public key encryption.
5.10 Question
The detect shows attempt to discover.
a) ssh
b) PCAnywhere
c) ssh-22
d) Sub7 on port 5632
Answer b

Detect 6

```
May 30 02:11:47 perimeter ipmon[32240]: 02:11:47.388976
le0 @0:18 b 155.100.120.53,2755 -> perimiter.my.net,27665 PR tcp len 20 48 -S IN
May 30 02:11:50 perimeter ipmon[32240]: 02:11:50.382206
le0 @0:18 b 155.100.120.53,2755 -> perimiter.my.net,27665 PR tcp len 20 48 -S IN
```

6.1 Source of detect
Same computer that is in detect 5. One of the servers on my.net perimeter.
6.2 Detect was generated by
Ipfilter firewall. OS is OpenBSD 2.6 patched.
6.3 Probability the source address was spoofed.
No. Medical students in Utah Health Sciences Center need some Masters.
6.4 Description of attack.
Trolling for Trin00. This specific attempt is to discover trin00 Master. Seems like the guy looking for unchanged default passwords on deployed Masters (see 6.5)
6.5 Attack mechanism.
Trin00 is one of the DDoS tools that are so popular in media this days. The architecture is no different from similar tools (TFN, TFN2K). There are three parts involved in action. Client, Master (terminology may vary) and Broadcast. Client is usually just a telnet programm. With client - attacker connects to Master to start the attack. Master communicates with number of Broadcasts(aka Daemons) and "distributes" the attack. The default settings are:
Communication:
```
Attacker  -> Master     27665/tcp
Master    -> Broadcasts 27444/udp
Broadcasts-> Master     31335/udp
```
Passwords:
```
Attacker  -> Master      betaalmostdone (or gOrave after master ?? local prompt)
Master    -> Broadcasts  l44dsl in form of <command> <pass> <command>
```
6.6 Correlation.
None on my network. I would think that this detect should be quite popular and grow with the popularity of DDoS tools.
Because when tools like that are in script kiddies hands quite significant amount of work could be saved for attackers. Kids do not change default settings

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

or passwords making an easy trophy. Even one master with its IP address hardcoded into Broadcasts adds quite a power to the army of Flooding Soldiers.

6.7 Evidence of active targeting

No. Attempt to find a treasure in the dirt.

6.8 Severity

Criticality - 3 (important host)
Lethality - 1 (attack is very unlikely to succeed)
System Countermeasures - 5 (modern OS)
Network countermeasures - 5 (restrictive firewall)
SEVERITY (-6)

6.9 Defensive recommendations

Check for open ports on questionable host. Check all neighbors. It is a possibility that if the attempt to compromise given host faild other hosts on the same network will be probed.

6.10 Question

IDS could detect the above traffic because of
a) SYN, PUSH, RESET flags
b) destination port
c) source port
d) size of the packet
Answer b

Detect 7

```
May 31 10:32:08 perimeter ipmon[32240]: 10:32:07.513563
le0 @0:18 b 209.209.135.216,1679 -> perimeter.my.net,12345 PR tcp len 20 48 -S
IN
May 31 10:32:12 perimeter ipmon[32240]: 10:32:11.695314
le0 @0:18 b 209.209.135.216,1933 -> perimeter.my.net,12346 PR tcp len 20 48 -S
IN
May 31 10:32:16 perimeter ipmon[32240]: 10:32:15.976095
le0 @0:18 b 209.209.135.216,2187 -> perimeter.my.net,31337 PR tcp len 20 48 -S
IN
May 31 10:32:20 perimeter ipmon[32240]: 10:32:20.176236
le0 @0:18 b 209.209.135.216,2442 -> perimeter.my.net,1243 PR tcp len 20 48 -S IN
May 31 10:32:24 perimeter ipmon[32240]: 10:32:24.401802
le0 @0:18 b 209.209.135.216,2696 -> perimeter.my.net,27374 PR tcp len 20 48 -S
IN
```

7.1 Source of detect

Host on the perimeter of my.net.

7.2 Detect was generated by

Ipfilter firewall. OS OpenBSD 2.6 patched.

7.3 Probability the source address was spoofed.

Do not think so. Unknown but inquisitive user from Kentucky asking us do we run all this at once?

7.4 Description of attack.

3 big and well known trojans discovery attempt. They were kind enough to even sort it for us. The first two packets for old and newer NetBus, the second is for BackOrifice (see detect 2), the third is old and new version of Sub7 trojan. The step between probes is almost exactly 4 seconds so I assume the discovery running by script or program. I will dare to say that the connectivity is good because deviations from 4 seconds are in 0.01 sec. Also if you will subtract the source port numbers you will get an interesting clue 1933-1679=2187-1933=254. This guy is sweeping all my class C for one trojan after another. Nice.

7.5 Attack mechanism

All of those programs are Windows trojans. Description of BackOrifice is in

© SANS Institute 2000 - 2002, Author retains full rights.

Detect 2. NetBus is kind of an old fashioned so let me concentrate on the hottest one - Sub7. Sub7's home is subseven.slak.org. Feature set is quite impressive my favorite is text2speach conversion - is a must for all future trojans. Now bad guys can _say_ you that U R B33N H4CK3D :). From noticable and quite dangerous are: udate from URL - could bypass any firewall because you usually allowing outgoing http traffic, all messenger spies (Yahoo, AOL, MS) dangerous because nowdays administrators of distributed sites like to communicate via messenger, video camera capture - you do not want remote monitoring of your room, don't you?
Distribution mechanism is as for any trojans: e-mail, electronic greeting cards, freeware and shareware software.
The big danger of modern windows trojans on my opinion is that every feature is easily configurable and there is no *NIX code patching and editing. Anybody can use it.
Ports are configurable so below listed ports are default ports for different releases, and preconfigured downloads. Starting I think from verion 2.1 Sub7 jumped to the higher port range. You might consider all 4 digits ports pre 2.X and all 5 digits after 2.X
1243/tcp, 6711/tcp, 6712/tcp, 6713/tcp, 6776/tcp, 27374/udp, 27573/tcp/udp
There is no signature, at least I have not came with one so this makes detect even more difficult.

7.6 Correlations.

I've seen very similar detect among SANS posted detects, can not find the date. More often windows trojan probes seen on home cable and DSL networks. My DMZ/collocation is checked once a week. There was no firm correlations so far. In general more often seen source address is one of end-user networks (@home, pacbell, KIH (London, Kentucky) :)

7.7 Evidence of active targeting

No. Class C sweep.

7.8 Severity

Criticality - 3 (important host)
Lethality - 1 (attack is very unlikely to succeed)
System Countermeasures - 5 (modern OS)
Network countermeasures - 5 (restrictive firewall)
SEVERITY (-6)

7.9 Defensive recommendations

Hard to detect - hard to recommend anything except taking security seriously. There is no matter how your perimeter is protected. One download in marketing ;) and you have a breach.

7.10 Question

What is the number of trojans attacker addressing
a) 1
b) 3
c) 5
d) none, this is a portscan
Answer c.

Detect 8

```
[**] IDS198/SYN FIN Scan [**]
05/30-18:26:13.112201 146.169.14.38:53 -> dmz.my.net.50:53
TCP TTL:19 TOS:0x0 ID:39426
**SF**** Seq: 0x11809B9F   Ack: 0x5ABA7B4B   Win: 0x404
63 7F EA 21 5A 9D                                c..!Z.

[**] IDS198/SYN FIN Scan [**]
05/30-18:26:13.212195 146.169.14.38:53 -> dmz.my.net.60:53
```

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
TCP TTL:19 TOS:0x0 ID:39426
**SF**** Seq: 0x11809B9F   Ack: 0x5ABA7B4B   Win: 0x404
43 43 77 65 4B 4D                                CCweKM

[**] IDS198/SYN FIN Scan [**]
05/30-18:26:13.329894 146.169.14.38:53 -> dmz.my.net.70:53
TCP TTL:19 TOS:0x0 ID:39426
**SF**** Seq: 0x11809B9F   Ack: 0x5ABA7B4B   Win: 0x404
00 00 00 00 00 00
```

8.1 Source of detect
IDS on DMZ at my.net
8.2 Detect was generated by
Snort 1.6 with ArachNIDS. OS is FreeBSD 3.4-STABLE
8.3 Probability the source address was spoofed.
No. Imperial College of Science, Department of Computing, London, UK.
8.4 Description of attack.
The port is not enough. This is network mapping. Illegal combination of TCP flags SYN and FIN. IP IDs, Sequence numbers are the same - so if this is NMAP than quite and old version of it. Tempo is vivo - swept all my class C in less than two seconds.
8.5 Attack mechanism.
This is not an attack this is a prelude to attack. This is mapping of my DMZ network. Too bad this guys were not aware that probably there are no IDS (even freeware) that do not register SYN-FIN combination.
8.6 Correlation.
I see a LOT of those on my net.

```
[**] IDS198/SYN FIN Scan [**]
05/30-14:30:07.343789 203.149.232.20:53 -> dmz.my.net.50:53
TCP TTL:27 TOS:0x0 ID:39426
**SF**** Seq: 0x17808333   Ack: 0x385521C9   Win: 0x404

[**] IDS198/SYN FIN Scan [**]
05/30-14:30:07.542535 203.149.232.20:53 -> dmz.my.net.60:53
TCP TTL:27 TOS:0x0 ID:39426
**SF**** Seq: 0x44FD3E3A   Ack: 0x2D0E01CE   Win: 0x404

[**] IDS198/SYN FIN Scan [**]
05/30-14:30:07.764234 203.149.232.20:53 -> dmz.my.net.70:53
TCP TTL:27 TOS:0x0 ID:39426
**SF**** Seq: 0x44FD3E3A   Ack: 0x2D0E01CE   Win: 0x404
```

Now this is interesting - same IP IDs from Taiwanese ISP as from London?

```
[**] IDS198/SYN FIN Scan [**]
06/02-07:19:48.246674 210.196.222.18:53 -> dmz.my.net.50:53
TCP TTL:30 TOS:0x0 ID:39426
**SF**** Seq: 0x176E9A2D   Ack: 0x89FC621   Win: 0x404
01 01 08 0A 04 4A                                .....J

[**] IDS198/SYN FIN Scan [**]
06/02-07:19:48.447120 210.196.222.18:53 -> dmz.my.net.60:53
TCP TTL:30 TOS:0x0 ID:39426
**SF**** Seq: 0x457FC724   Ack: 0x7DB57EAE   Win: 0x404
4A 4B 54 AC AB B1                                JKT...
```

© SANS Institute 2000 - 2002, Author retains full rights.

```
[**] IDS198/SYN FIN Scan [**]
06/02-07:19:48.668979 210.196.222.18:53 -> dmz.my.net.70:53
TCP TTL:30 TOS:0x0 ID:39426
**SF**** Seq: 0x457FC724   Ack: 0x7DB57EAE   Win: 0x404
00 00 00 00 00 00                                  ......
```

Same IP IDs and source address resolves to dns1.udc-c.dion.co.jp. Compromised DNS? Quite possible.

So this is not nmap this is some other automated tool out there in wild.

One of the SANS posted detects on 3/22/00 shows similar scan (same speed, unfortunately no IP IDs in posted scan to compare.

Recalling GIAC training we saw the same scan for port 137 and for port 109 (It is good be trained on real stuff). So page 202 from last day class by Stephen Northcutt in San Jose. SAME ID 39426, same window 0x404, same speed 0.02 sec per host, other port => same tool.

If I will get the tool I'll let you know. If anybody knows the tool I would appreciate the source.

8.7 Evidence of active targeting

Yes. Both network mapping and DNS discovery. RST packet will be sent if service is present.

8.8 Severity

Criticality - 5 (important hosts, one of them gateway)
Lethality - 1 (there are no attack yet)
System Countermeasures - 3 (some computers are not patched and running samba)
Network countermeasures - 5 (restrictive firewall)
SEVERITY (-2)

8.9 Defensive recommendations

Everybody is hunting your DNS. Make sure you do not have stray ones. And those you have are in a good shape. Create a list of probers - it will be long but it probably worth watching for this IPs for a while.

8.10 Question

What's wrong with the presented packets
a) flags
b) IP IDs
c) seq #s
d) all of above
Answer d

Detect 9

```
[**] IDS177/netbios-name-query [**]
05/31-12:10:04.949375 208.180.61.124:137 -> dmz.my.net.50:137
UDP TTL:108 TOS:0x0 ID:34747
Len: 58
3D 90 00 10 00 01 00 00 00 00 00 20 43 4B 41  =.......... CKA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 00 00 21  AAAAAAAAAAAA..!
00 01                                         ..

[**] IDS177/netbios-name-query [**]
05/31-12:10:06.401728 208.180.61.124:137 -> dmz.my.net.50:137
UDP TTL:108 TOS:0x0 ID:35003
```

© SANS Institute 2000 - 2002, Author retains full rights.

```
Len: 58
3D 92 00 10 00 01 00 00 00 00 00 20 43 4B 41  =.......... CKA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 00 00 21  AAAAAAAAAAAAA..!
00 01                                          ..

[**] IDS177/netbios-name-query [**]
05/31-12:12:00.788879 208.180.61.124:137 -> dmz.my.net.60:137
UDP TTL:108 TOS:0x0 ID:54459
Len: 58
3D F4 00 10 00 01 00 00 00 00 00 20 43 4B 41  =.......... CKA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 00 00 21  AAAAAAAAAAAAA..!
00 01                                          ..

[**] IDS177/netbios-name-query [**]
05/31-12:12:02.205801 208.180.61.124:137 -> dmz.my.net.60:137
UDP TTL:108 TOS:0x0 ID:54715
Len: 58
3D F6 00 10 00 01 00 00 00 00 00 20 43 4B 41  =.......... CKA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 00 00 21  AAAAAAAAAAAAA..!
00 01                                          ..

[**] IDS177/netbios-name-query [**]
05/31-12:14:15.912861 208.180.61.124:137 -> dmz.my.net.70:137
UDP TTL:110 TOS:0x0 ID:11964
Len: 58
3E 62 00 10 00 01 00 00 00 00 00 20 43 4B 41  >b.......... CKA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 00 00 21  AAAAAAAAAAAAA..!
00 01
```

9.1 Source of detect
IDS on DMZ at my.net
9.2 Detect was generated by
Snort 1.6 with ArachNIDS. OS is FreeBSD 3.X-STABLE.
9.3 Probability the source address was spoofed.
No. Cox Internet user at above ip forgot windows names of computers at my DMZ. He wants to get the names.
9.4 Description of attack.
Discovering SMB or Windows computers using wildcard pattern. Seems like payload in packets is valid - ends with regular 00 01. Source port is lower than 1024 - program probably running with root privileges. IP IDs are probably generated via normal syscalls and give us 256 increment between two UDP packets at the same computer. Where all other packets destined to other addresses in the same class C? Possible.
9.5 Attack mechanism.
Recon. Discover netbios and domain names, logged users. The other possibility is Virus (LOVEBUG-like) using one of its distribution mechanisms.
9.6 Correlation.
A lot of thouse on my network. Couple of false positives with high management laptops taken home as submitter of SANS detects posts on 02/11/00. List of detected IP addresses could be used to study geography. Total percentage of same detects is close to 4% on my.net.
9.7 Evidence of active targeting
Possible. One of the recon attempts before the attack.

© SANS Institute 2000 - 2002, Author retains full rights.

9.8 Severity
Criticality - 5 (important host, one of them is a gateway)
Lethality - 1 (there is no attack yet)
System Countermeasures - 3 (some computers are not patched and running samba)
Network countermeasures - 5 (restrictive firewall)
SEVERITY (-2)
9.9 Defensive recommendations
Check SMB daemon version on computer running samba. Check packet filtering setup on samba host to allow connections from internal network only.
9.10 Question
The above detect is
a) DoS against netbios
b) Smurf attack
c) Attempt to connect to windows shares
d) recon
Answer d

Detect 10

```
May 26 16:01:22 202.103.237.30:5078 -> dmz.my.net.50:80 SYN **S*****
May 26 16:01:22 202.103.237.30:5080 -> dmz.my.net.50:3128 SYN **S*****
May 26 16:01:22 202.103.237.30:5519 -> dmz.my.net.60:80 SYN **S*****
May 26 16:01:22 202.103.237.30:5522 -> dmz.my.net.60:3128 SYN **S*****
May 26 16:01:23 202.103.237.30:5547 -> dmz.my.net.70:80 SYN **S*****
May 26 16:01:23 202.103.237.30:5549 -> dmz.my.net.70:3128 SYN **S*****
```

10.1 Source of detect
DMZ at my.net
10.2 Detect was generated by
Snort 1.6. OS is FreeBSD 3.X-STABLE.
10.3 Probability the source address was spoofed.
No. Now Cox Internet users are looking for proxy.
10.4 Description of attack.
An attempt to discover open proxy running. Quite popular since a lot of organization started to implement "surf controls" on their networks. Port 80 could also lead to the web server with following attempts to exploit numerous CGI
vulnerabilities. I am quite surprised not seeing 8080 or 1080. Source port increments normally but it took so many port increments to get from my host 50 to 60 and if port increments by one where is the missed increment between attempt to connect to port 80 and attempt to connect to port 3128?
10.5 Attack mechanism
A open proxy can work as a middle tier in the path to the information that forbidden to access directly. If company X has an open proxy - people from company Y can access playboy.com even if access from their company in forbidden. Also the list of open proxies is of a great demand especially from China where internet policies is extremely restrictive.
There was a squid vulnerability that allowed to bypass authentication (with external authentication server)
10.6 Correlation.
None at my scope. Not on my network.
10.7 Evidence of active targeting
No. Broad search.
10.8 Severity.
Criticality - 5 (important host, one of them is a gateway)
Lethality - 1 (there is no proxies there)
System Countermeasures - 3 (some computers are not patched and running samba)

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Network countermeasures - 5 (restrictive firewall)
SEVERITY (-2)

10.9 Defensive recommendations

We are OK for now. If this is a research network I can suggest plugging honeypot to distinguish searchers for freedom from searchers for CGI exploits. This is not quite defensive though.

10.10 Question

What service this scan tries to discover
a) web
b) web proxy
c) web on non standard port
d) a)and c)
Answer b

© SANS Institute 2000 - 2002, Author retains full rights.