[TOOL] Fast SYN Scanner (libnet, libpcap)

Source: http://www.derkeiler.com/Mailing−Lists/Securiteam/2004−07/0027.html

---

From: SecuriTeam (*support_at_securiteam.com*)
Date: 07/11/04

```
To: list@securiteam.com
Date: 11 Jul 2004 11:27:48 +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/mailinglist.html

− − − − − − − − −

Fast SYN Scanner (libnet, libpcap)
−−

SUMMARY

DETAILS

The following tool is a fast SYN scanner written in C.

Tool source:

```
/*
 This is a fast and portable (i think). 48 bytes syn, w2k emulation, we
are still working on it,
 drop an email to drbios2000@yahoo.com if something goes wrong.
 libnet and libpcap is required, the options are pretty self explanatory,
 stripped static binary included for lamers.
 Greets to kauggie (kaugex), nebunu, amidax, jhony si la ce tovarasi mai
avem noi pe internetu asta.
 BAG PULA IN TOTI ADMINII CARE SE CRED DUMNEZEI CA SUNT CU CONSOLA IN FATA
 MUIE CUI SE SIMTE LUAT IN VIZOR DE HAITATEAM
*/

#include <libnet.h>
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
```

```
#include <arpa/inet.h>
#include <sys/types.h>
#include <unistd.h>
#include <pcap.h>
#include <time.h>

int main(int argc, char **argv)
{
 libnet_t *l;
 libnet_ptag_t t;
 unsigned short burst=50;
 unsigned short ct=0;
 char errbuff[LIBNET_ERRBUF_SIZE];
 unsigned long myip;
 struct in_addr sc;
 unsigned char tcpopt[]="\x02\x04\x05\xb4\x01\x01\x04\x02";

 unsigned short port;
 unsigned long usec;
 //unsigned char outstr[1024];
 char cc;
 int i;
 pid_t pid;
 pcap_t *handle;
 char *temp_char;
 bpf_u_int32 mask;
 bpf_u_int32 net;
 char errbuf[PCAP_ERRBUF_SIZE];
 char filter[1024];
 struct bpf_program cfilter;
 struct pcap_pkthdr header;
 const unsigned char *packet;
 struct in_addr ekkt;
 unsigned char ip[50];

 unsigned long dstip=0;
 unsigned short sport;
 char *interface=NULL;
 unsigned char bclass=0;
 unsigned char aclass=0;
 unsigned char rclass=1;
 unsigned int a=0,b=0,c=0,d=0;

 srand(time(NULL));
 sport=rand();
 usec=1000000;
 if(argc<2)
 {
 printf("usage: %s <port> [−a <a class> | −b <b class>] [−i <interface]
[−s <speed>]\n",argv[0]);
 printf("speed 10 −> as fast as possible, 1 −> it will take bloody ages
(about 50 syns/s)\n");
 printf("by DrBIOS <drbios2000@yahoo.com> & Bagabontu
<bagabonturo@yahoo.com>\n");
 exit(0x01);
 }
 for(i=1;i<argc;i++)
 {
 if(strstr(argv[i],"−s"))
 {
  if(i+1<argc)
  {
 switch (atoi(argv[i+1]))
 {
  case 1:usec=1000000;break;
  case 2:usec=500000;break;
  case 3:usec=250000;break;
  case 4:usec=125000;break;
  case 5:usec=60000;break;
  case 6:usec=30000;break;
  case 7:usec=10000;break;
  case 8:usec=1000;break;
  case 9:usec=100;break;
  case 10:usec=0;burst=65535;
 }

  }
  else
  {
 printf("−s requires an argument\n");
 exit(0x01);
  }
 }

 if(strstr(argv[i],"−i"))
 {
  if(i+1<argc) interface=argv[i+1];else
  {
 printf("−i requires an argument\n");
 exit(0x01);
  }
 }
 if(strstr(argv[i],"−a"))
 {
  if(i+1<argc)
  {
 aclass=1;
 bclass=0;
 rclass=0;
 a=atoi(argv[i+1]);
 b=0;
 c=0;
 d=0;
 //printf("%d\n",a);
 if((a<1) || (a>254))
 {
  printf("A must be between 1 and 254\n");
  exit(0x02);
 }
 printf("scanning network %d.*.*.*\n",a);
  }
  else
  {
 printf("−a requires an A network as argument\n");
 exit(0x01);
  }
 }
 if(strstr(argv[i],"−b"))
 {
  if(i+1<argc)
  {
 aclass=0;
 bclass=1;
 rclass=0;
 a=atoi(strtok(argv[i+1],"."));
 temp_char=strtok(NULL,".");
 if(temp_char==NULL)
 b=0;else b=atoi(temp_char);
 c=0;
 d=0;
 //printf("%d\n",a);
 if((a<1) || (a>254))
 {
  printf("A must be between 1 and 254\n");
  exit(0x02);
 }
 printf("scanning network %d.%d.*.*\n",a,b);
  }
  else
  {
 printf("−b requires an B network as argument(e.g. 192.168)\n");
 exit(0x01);
  }
 }
 }
 printf("usec: %ld, burst packets %d\n",usec,burst);
 port=(unsigned short)atoi(argv[1]);
 if((port<1) || (port>65535)) exit(printf("damn dude, port numbers are in
 1 .. 65535\n"));
 if(interface!=NULL) printf("using inteface %s\n",interface);

 l=libnet_init(LIBNET_RAW4,interface,errbuff);
 if(!l)
```

```
 {
 printf("ERROR: %s\n",errbuff);
 exit(0x02);
 }
 myip=libnet_get_ipaddr4(l);
 sc.s_addr=myip;
 sprintf(filter,"(tcp[tcpflags]=0x12) and (src port %d) and (dst port
%d)",port,sport);
 printf("using \"%s\" as pcap filter\n",filter);
 printf("my detected ip on %s is %s\n",l−>device,inet_ntoa(sc));
 pcap_lookupnet(l−>device, &net, &mask, errbuf);
 pid=fork();
 handle=NULL;
 handle = pcap_open_live(l−>device, BUFSIZ, 1, 0, errbuf);
 if(handle==NULL)
 {
 printf("ERROR: pcap_open_live() : %s\n",errbuff);
 exit(0x05);
 }
 cc=pcap_compile(handle, &cfilter, filter, 0, net);
 if(cc!=0)
 {
  printf("ERROR: pcap_compile() failed!!!\n");
  exit(0);
 }
 cc=pcap_setfilter(handle, &cfilter);
 if(cc!=0)
 {
  printf("ERROR: pcap_setfilter() failed!!!\n");
  exit(0);
 }
 if(pid==0)
 {
 /* sniff */
  while(1)
 {
   packet = pcap_next(handle, &header);
 memcpy(&ekkt.s_addr,packet+26,4);
 printf("%s\n",inet_ntoa(ekkt));
 FILE * fp;
 fp=fopen("bios.txt","a+");
 fprintf(fp,"%s\n",inet_ntoa(ekkt));
 fclose(fp);
 }
 }
 if(pid > 0)
 {
 printf("capturing process started pid %d\n",pid);
  usleep(500000);
  while(1)
  {
   t=LIBNET_PTAG_INITIALIZER;
  t=libnet_build_tcp_options(tcpopt, 8, l,0);
  //t=LIBNET_PTAG_INITIALIZER;

t=libnet_build_tcp(sport,port,rand(),rand(),TH_SYN,65535,0,0,LIBNET_TCP_H+8,NULL,0,l,0);
  if(rclass) dstip=rand();
  if(aclass)
  {
 if(d==0) printf("scanning %d.%d.%d.*\n",a,b,c);
 d++;
 if(d>255) {c++;d=0;}
 if(c>255) {b++;c=0;}
 sprintf(ip,"%d.%d.%d.%d\n",a,b,c,d);

 //printf("%s\n",ip);
 if((b==255)&& (c==255) && (d==255))
 {
  printf("aici trebuie stop\n");
  sleep(10);
  kill(pid,2);
  return 0;
 }
 sc.s_addr=inet_addr(ip);
 dstip=sc.s_addr;
  }
  if(bclass)
  {
 if(d==0) printf("scanning %d.%d.%d.*\n",a,b,c);
 d++;
 if(d>255)
 {
  c++;d=0;
 }
 sprintf(ip,"%d.%d.%d.%d",a,b,c,d);
 if((c==255) && (d==255))
 {
  printf("%s\n",ip);
  printf("aici trebuie stop\n");
  sleep(10);
  kill(pid,2);
  return 0;
 }
 sc.s_addr=inet_addr(ip);
 dstip=sc.s_addr;
  }


libnet_build_ipv4(LIBNET_TCP_H+LIBNET_IPV4_H+8,0,rand(),0,128,IPPROTO_TCP,0,myip,dstip,NULL,0,l,0);
   cc=libnet_write(l);
  if(cc<=0) printf("libnet_write() wtf %d\n",cc);
  libnet_clear_packet(l);
  if(ct==burst)
  {
 usleep(usec);
 ct=0;
  };
  ct++;
  }

 }
 if(pid<0)
 {
  printf("cannot fork()\n");
  exit(0x05);
 }
 return 0;
}
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:drbios2000@yahoo.com> Doctor
BIOS.