컴퓨터공학부 | 20123434 | 황정담

[ 네트워크 최신기술 ]

[ MSNORT ]

제출일 : 2015.05.6

과제 요구사항

1. TCP 트래픽만 출력되도록 한다 🡪 mSnort.c의 int filter(mPacket\_t \*p) 함수 수정

void dissect (unsigned char \*user, const struct pcap\_pkthdr \*h, const unsigned char \*p)

{

…

// filter

if (filter (&pkt)) return;

// view

view (&pkt);

…

}

^dissect함수에서 filter()가 1을 return하면 dissect는 반환(종료)되고 0 을 return하면 해당 트래픽을 보여준다

* Filter()에서는 TCP트래픽일 때 0, 그렇지 않을 때 1을 return하게함.

수정 전)

int filter (mPacket\_t \*p)

{

#if 0

if (p->proto == IPPROTO\_TCP &&

(ntohs (p->th.tcph->th\_sport) == 22 ||

ntohs (p->th.tcph->th\_dport) == 22)) return 1;

if (p->proto == IPPROTO\_TCP &&

(ntohs (p->th.tcph->th\_sport) == 3389 ||

ntohs (p->th.tcph->th\_dport) == 3389)) return 1;

#endif

return 0;

}

수정 후)

int filter (mPacket\_t \*p)

{

if (p->proto == IPPROTO\_TCP ||

(ntohs (p->th.tcph->th\_sport) == 22 ||

ntohs (p->th.tcph->th\_dport) == 22)) return 0;

if (p->proto == IPPROTO\_TCP ||

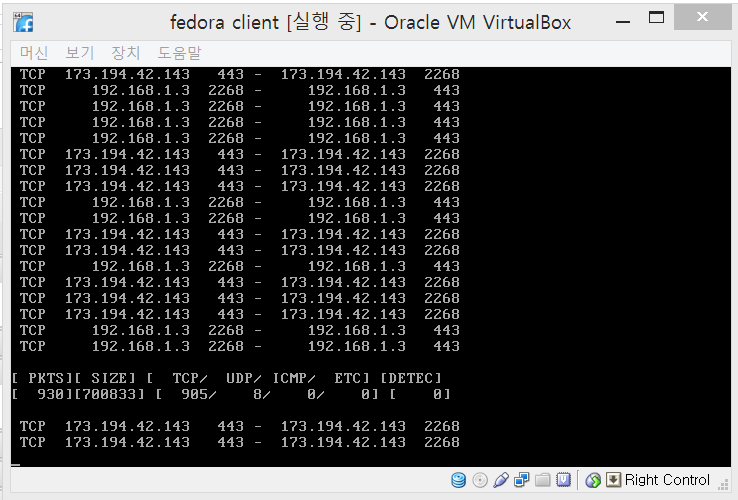
(ntohs (p->th.tcph->th\_sport) == 3389 ||

ntohs (p->th.tcph->th\_dport) == 3389)) return 0;

return 1;

}

실행결과)



2. 두개의 패턴을 기준으로 탐지되어야 한다 (“GET” & “html”) 🡪 mSnort.c의 int search (mPacket\_t \*p) 함수 수정

void dissect (unsigned char \*user, const struct pcap\_pkthdr \*h, const unsigned char \*p)

{

…

// detection

if (search (&pkt)){

fprintf (stderr, "\t\t---- Detection !! -----\n");

}

return;

}

^dissect함수에서 serach()가 1을 return하면 패킷을 Detection하여 보여주고, 0을 return하면 함수가 반환(종료)된다.

* serach ()에서는 패턴을 감지했을 때 1을, 그 외에 0을 반환해야함

수정 전)

int search (mPacket\_t \*p)

{

// pattern, "GET "

char \*pat = "GET ";

int len = strlen (pat);

if (p->payload\_len < len) return 0;

if (strstr (p->payload, pat) == NULL) return 0;

return 1;

}

수정 후) pattern “html”추가.

int search (mPacket\_t \*p)

{

// pattern, "GET " & "html"

char \*pat1 = "GET ";

int len1 = strlen (pat1);

char \*pat2 = "html";

int len2 = strlen (pat2)

if ( (p->payload\_len < len1) || (p->payload\_len <len2)) return 0;

if ( (strstr (p->payload, pat1) == NULL) || (strstr (p->payload, pat2) == NULL) ) return 0;

return 1;

}

3. 탐지되었을 때, 패킷의 페이로드를 화면에 출력한다

수정 전)

void dissect (unsigned char \*user, const struct pcap\_pkthdr \*h, const unsigned char \*p)

{

…

// detection

if (search (&pkt)){

fprintf (stderr, "\t\t---- Detection !! -----\n");

}

return;

}

수정 후) detection출력후, dump\_data실행

void dissect (unsigned char \*user, const struct pcap\_pkthdr \*h, const unsigned char \*p)

…

// detection

if (search (&pkt)){

fprintf (stderr, "\t\t---- Detection !! -----\n");

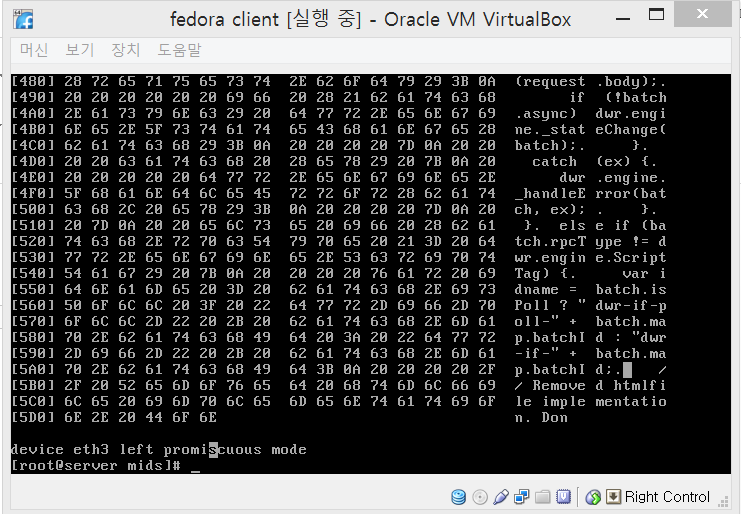
Dump\_data(p, (unsigned int)h->caplen);

}

return;

}

실행결과)



전체 소스코드)

다운로드 🡪 (클릭)

#include <unistd.h>

#include <string.h>

#include <ctype.h>

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

#include <stdarg.h>

#include <getopt.h>

#include <errno.h>

#include <time.h>

#include <unistd.h>

#include <signal.h>

//#include <sys/stat.h>

#include <fcntl.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netdb.h>

#include <sys/time.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <dirent.h>

#include <linux/types.h>

#include <errno.h>

//#include <netinet/tcp.h>

#include <sys/file.h>

#ifdef LINUX

#include <linux/unistd.h>

#endif // LINUX

#ifdef \_\_NR\_gettid

#else // \_\_NR\_gettid

#define \_\_NR\_gettid 224

#endif // \_\_NR\_gettid

#ifndef MIN

#define MIN(a, b) ((a) < (b) ? (a) : (b))

#endif /\* MIN \*/

#include "mSnort.h"

mStat\_t stat;

void help ()

{

printf ("mSnort Ver 0.1.0\n"

"Usage: ./mSnort -i [Interface] -c [COUNT] -f [FILE]\n"

" -h Help\n"

" -i [INTERFACE] Interface ex) eth0\n"

" -f [FILE] File output ex) a.txt\n");

exit (0);

}

void print\_asc (unsigned char \*buf,int len)

{

int i;

for (i = 0; i < len; i++)

fprintf(stderr, "%c",isprint(buf[i])?buf[i]:'.');

}

void Dump\_Data (unsigned char \*buf, int len)

{

int i = 0;

if (len <= 0) return;

fprintf(stderr, "[%03X] ", i);

for (i = 0; i < len;) {

fprintf(stderr, "%02X ", (int)buf[i]);

i++;

if (i%8 == 0) fprintf(stderr, " ");

if (i%16 == 0) {

print\_asc(&buf[i-16], 8); fprintf(stderr, " ");

print\_asc(&buf[i-8], 8); fprintf(stderr, "\n");

if (i < len) fprintf(stderr, "[%03X] ",i);

}

}

if (i%16) {

int n;

n = 16 - (i%16);

fprintf(stderr, " ");

if (n > 8) fprintf(stderr, " ");

while (n--) fprintf(stderr, " ");

n = MIN(8,i%16);

print\_asc(&buf[i-(i%16)],n); fprintf(stderr, " ");

n = (i%16) - n;

if (n > 0) print\_asc(&buf[i-n],n);

fprintf(stderr, "\n");

}

fprintf(stderr, "\n");

}

int enable\_core ()

{

int n;

struct rlimit rl;

if (getrlimit (RLIMIT\_NOFILE, &rl) == -1){

return -1;

}

n = rl.rlim\_cur = rl.rlim\_max;

setrlimit (RLIMIT\_NOFILE, &rl);

if (getrlimit (RLIMIT\_CORE, &rl) == -1){

return -1;

}

rl.rlim\_cur = rl.rlim\_max;

setrlimit (RLIMIT\_CORE, &rl);

return 0;

}

int decode (mPacket\_t \*p, char \*pkt, unsigned short plen)

{

struct ip \*iph = (struct ip \*)(pkt + sizeof(struct ether\_header));

struct tcphdr \*th = NULL;

unsigned int ihlen = iph->ip\_hl \* 4;

unsigned int thlen = 0;

unsigned int offset;

unsigned int length;

if (iph->ip\_v != 4) return -1;

if (iph->ip\_off & htons(IP\_MF|IP\_OFFMASK)) return -1;

//Dump\_Data (pkt, plen);

switch (iph->ip\_p) {

case IPPROTO\_TCP:

th = (struct tcphdr \*)((const char \*)iph + ihlen);

thlen = th->th\_off \* 4;

stat.tcp++;

break;

case IPPROTO\_UDP:

thlen = sizeof(struct udphdr);

stat.udp++;

break;

case IPPROTO\_ICMP:

stat.icmp++;

break;

default:

return -1;

}

p->mac.raw = pkt;

p->nh.iph = iph;

p->th.raw = (char\*)iph + ihlen;

p->proto = iph->ip\_p;

offset = sizeof(struct ether\_header) + ihlen + thlen;

length = sizeof(struct ether\_header) + ntohs(iph->ip\_len) - offset;

p->payload = pkt + offset;

p->payload\_len = length;

return 0;

}

void view (mPacket\_t \*p)

{

static unsigned int ti = 0, now;

now = time (NULL);

if (now > ti){

ti = now + 5;

fprintf (stderr, "\n[%5s][%5s] [%5s/%5s/%5s/%5s] [%5s]\n",

"PKTS", "SIZE", "TCP", "UDP", "ICMP", "ETC", "DETEC");

fprintf (stderr, "[%5llu][%5llu] [%5llu/%5llu/%5llu/%5llu] [%5llu]\n\n",

stat.cnt, stat.size,

stat.tcp, stat.udp, stat.icmp, stat.etc,

stat.detec);

}

// proto sip sport - dip sport

switch (p->proto){

case IPPROTO\_TCP:

fprintf (stderr, "%4s %15s %5d - %15s %5d\n",

"TCP",

inet\_ntoa (p->nh.iph->ip\_src),

ntohs (p->th.tcph->th\_sport),

inet\_ntoa (p->nh.iph->ip\_dst),

ntohs (p->th.tcph->th\_dport));

break;

case IPPROTO\_UDP:

fprintf (stderr, "%4s %15s %5d - %15s %5d\n",

"UDP",

inet\_ntoa (p->nh.iph->ip\_src),

ntohs (p->th.udph->uh\_sport),

inet\_ntoa (p->nh.iph->ip\_dst),

ntohs (p->th.udph->uh\_dport));

break;

case IPPROTO\_ICMP:

fprintf (stderr, "%4s %15s %5s - %15s %5s\n",

"ICMP",

inet\_ntoa (p->nh.iph->ip\_src),

" ",

inet\_ntoa (p->nh.iph->ip\_dst),

" ");

break;

}

// payload ??

return;

}

int search (mPacket\_t \*p)

{

// pattern, "GET " & "html"

char \*pat1 = "GET ";

int len1 = strlen (pat1);

char \*pat2 = "html";

int len2 = strlen (pat2)

if ( (p->payload\_len < len1) || (p->payload\_len <len2)) return 0;

if ( (strstr (p->payload, pat1) == NULL) || (strstr (p->payload, pat2) == NULL) ) return 0;

return 1;

}

int filter (mPacket\_t \*p)

{

if (p->proto == IPPROTO\_TCP ||

(ntohs (p->th.tcph->th\_sport) == 22 ||

ntohs (p->th.tcph->th\_dport) == 22)) return 0;

if (p->proto == IPPROTO\_TCP ||

(ntohs (p->th.tcph->th\_sport) == 3389 ||

ntohs (p->th.tcph->th\_dport) == 3389)) return 0;

return 1;

}

void dissect (unsigned char \*user, const struct pcap\_pkthdr \*h, const unsigned char \*p)

{

mPacket\_t pkt;

memset (&pkt, 0, sizeof (mPacket\_t));

stat.cnt++;

stat.size+=h->caplen;

// decode

if (decode (&pkt, p, (unsigned int)h->caplen)) return;

// filter

if (filter (&pkt)) return;

// view

view (&pkt);

// detection

if (search (&pkt)){

fprintf (stderr, "\t\t---- Detection !! -----\n");

Dump\_data(p, (unsigned int)h->caplen);

}

return;

}

void io\_read (char \*dev)

{

char \*net;

char \*mask;

int ret;

char errbuf[PCAP\_ERRBUF\_SIZE];

bpf\_u\_int32 netp; // ip

bpf\_u\_int32 maskp; // submet mask

struct in\_addr addr;

struct pcap\_pkthdr header; // Header that pcap gives us

char \*file = stdout;

if (dev == NULL) return;

pcap\_t \*pcd;

pcd = pcap\_open\_live (dev, BUFSIZ, 1, 1000, errbuf);

if (pcd == NULL) {

fprintf (stderr, "Couldn't open device %s: %s\n", dev, errbuf);

return;

}

fprintf (file, "===================================\n");

fprintf (file, "Interface Info : %s\n", dev);

ret = pcap\_lookupnet (dev, &netp, &maskp, errbuf);

if (ret == -1){

fprintf (file,"%s\n", errbuf);

return;

}

addr.s\_addr = netp;

net = inet\_ntoa (addr);

if (net == NULL){

perror ("inet\_ntoa");

return;

}

fprintf (file,"NET : %s\n", net);

addr.s\_addr = maskp;

mask = inet\_ntoa (addr);

if (mask == NULL){

perror ("inet\_ntoa");

exit(1);

}

fprintf (file,"MASK : %s\n", mask);

fprintf (file,"===================================\n");

pcap\_loop (pcd, 0, dissect, NULL);

pcap\_close (pcd);

return;

}

int init ()

{

memset (&stat, 0, sizeof (mStat\_t));

return 0;

}

int main (int argc, char \*\*argv)

{

char \*dev = NULL;

int opt;

enable\_core ();

while ((opt = getopt(argc, argv, "hi:d:")) != -1){

switch(opt){

case 'h':

help();

break;

case 'i':

dev = optarg;

break;

default:

break;

}

}

if (dev == NULL) {

fprintf(stderr, "Try `%s -h' for more information.\n", argv[0]);

exit(0);

}

init ();

io\_read (dev);

return 0;

}