정보보안 7기

유해사이트 차단 프로젝트

C조 [노우택, 윤찬호, 이선우, 지영광]

Contents

프로젝트 소개

PART 1

조원 소개

윤찬호

팀장 / 개발총괄

- 전체적인 흐름도 제작
- 코드 수정 및 추가 (got_packet내부 함수, DB연결 및 SELECT함수, Send_row 함수)

이선우

Test Client 환경 구축

- DB 테이블 설계
- 도메인 서버 생성
- DB연동 함수

노우택

WEB 서버 구축

- Web 서버 구축 및 Web에서의 차단 도메인 추가 설계
- DB 테이블 설계

지영광

함수 / DB 설계

- 코드 병합 및 ppt제작
- 데이터 베이스 설정
- DB연동 및 INSERT함수 생성

PART 2

개발환경 및 라이브러리

ubuntu

Apache Tomcat

PART 3

구상도


웹 사이트
5
정상 패킷 반환
(Response)
차단 도메인 목록
전송
차단 도메인 목록 DB
유해차단 프로그램
1
패킷 요청
(Request)
2
패킷 캡쳐
차단 도메인일 경우
변조된 패킷 전송
3
차단 도메인 저장
4
사용자 PC
차단 페이지
차단 페이지로 연결
관리자 웹 페이지

구현 방식

main함수 구성

```
pcap_findalldevs(&devs, errbuf);
printf("INFO: dev name = %s .\n" , (*devs).name );
dev = (*devs).name ;
```

캡쳐할 네트워크 장치 이름을 반환해주는 함수

```
if (pcap_lookupnet(dev, &net, &mask, errbuf) == -1) {
    fprintf(stderr, "Couldn't get netmask for device %s: %s\n",
            dev, errbuf);
    net = 0;
    mask = 0;
}
```

IPv4 network numbers 와 그에 상응하는 network mask를 반환 해주는 함수

```
handle = pcap_open_live(dev, BUFSIZ, 1, 1000, errbuf);
if (handle == NULL) {
    fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);
    return(2);
}
```

패킷 캡쳐를 위한 handle을 반환 해주는 함수

main함수 구성

```
if (pcap_compile(handle, &fp, filter_exp, 0, net) == -1) {
    fprintf(stderr, "Couldn't parse filter %s: %s\n", filter_exp,
            pcap_geterr(handle));
    return(2);
}
```

패킷 캡쳐를 위한 필터를 적용시키기 위해 기계어 수준의 필터로 변환하여 fp에 저장해주는 함수

```
if (pcap_setfilter(handle, &fp) == -1) {
    fprintf(stderr, "Couldn't install filter %s: %s\n", filter_exp,
        pcap_geterr(handle));
    return(2);
}
```

기계어 수준의 필터가 적용된 fp 를 handle에 적용 시켜주는 함수

main함수 구성

```
mysql_init(&conn);
db_connection = mysql_real_connect(
        &conn,
        host,
        user,
        passwd,
        db,
        port,
        (char*)NULL,
        0
);
```

→ DB에 연결하기 위한 함수

```
int result = 0;
result = pcap_loop(handle, 0, got_packet, NULL) ;
```

→ 본격적으로 패킷을 캡쳐 하는 함수
콜백 함수인 got_packet 에서
패킷을 캡쳐

got_packet 함수 구성

```
if((payload_len = pcap_len(&ethernet, &ip,
                &tcp, &payload, packet)) == -1)
{
    printf("pcap len error\n");
    return;
}
```

→ 캡쳐 한 Packet을 각 헤더별로 나눠주는 함수
성공 시 payload 길이 값, 실패 시 -1 반환

```
domain_len = get_domain(payload, &domain);
if(domain_len == 0)
    return;
```

→ payload에 있는 도메인 주소를 가져오는 함수
성공 시 domain의 길이 , 실패 시 0 반환

```
chk_domain_cnt = get_check_domain(chk_domain,
                                  chk_domain_cnt);
```

→ DB에 저장된 차단 도메인 주소 리스트를 가져오는 함수
성공 시 차단 도메인 주소의 개수, 실패 시 ERROR 메시지 반환

PART 2-2

got_packet 함수 구성

```
result = domain_check(domain, domain_len,
                      chk_domain_cnt, chk_domain);
```

Payload 부분의 도메인 주소와
DB에서 가져 온 차단 도메인 주소 리스트를 비교
일치 한다면 0, 불일치 한다면 -1을 반환

```
//insert log to db .
insert_db_func(ip,tcp,src_ip_str,
        dst_ip_str,domain,result);
```

DB의 로그 table에 패킷의 정보를
INSERT 하는 함수

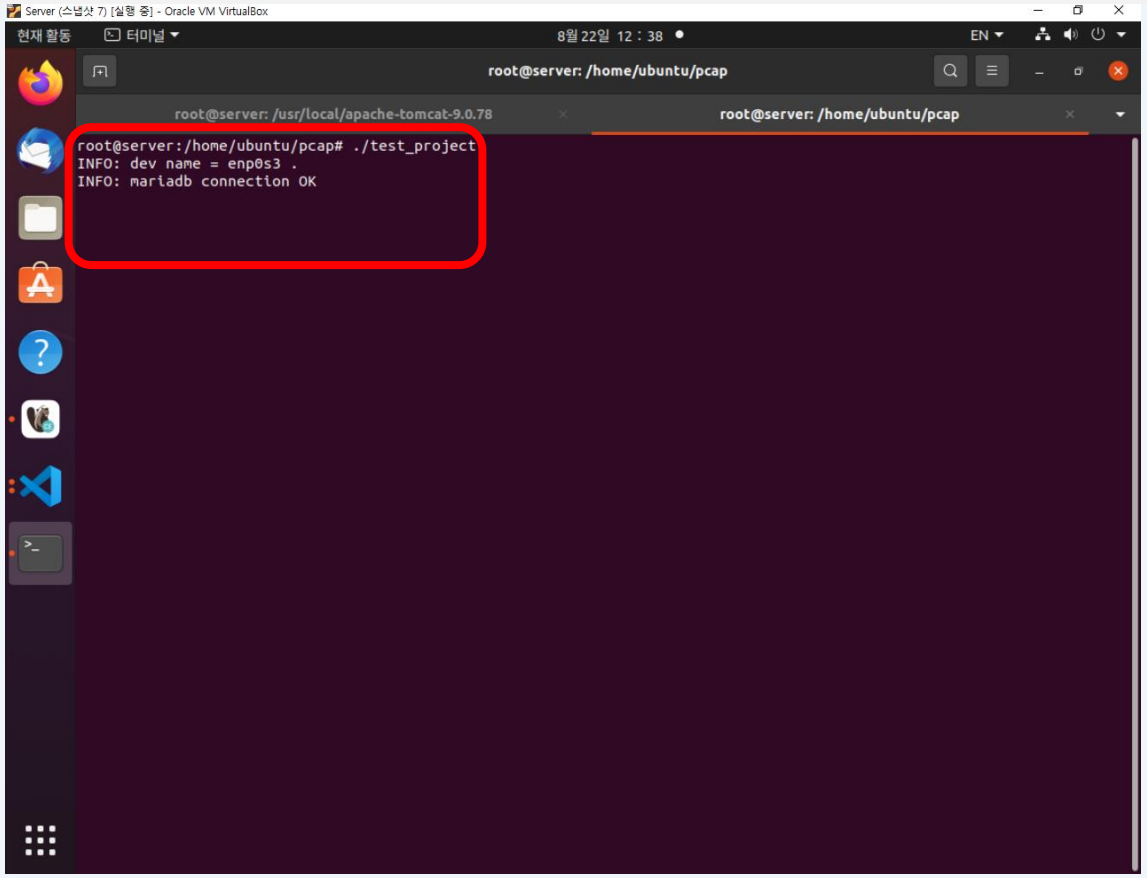
```
if( result == 0)
{
    print_info(ethernet, ip, tcp, payload, domain);

    int sendraw_ret = sendraw(packet , sendraw_mode);
    return;
}
```

패킷의 정보를 print 하는 함수

패킷을 변조 후 전송하는 함수

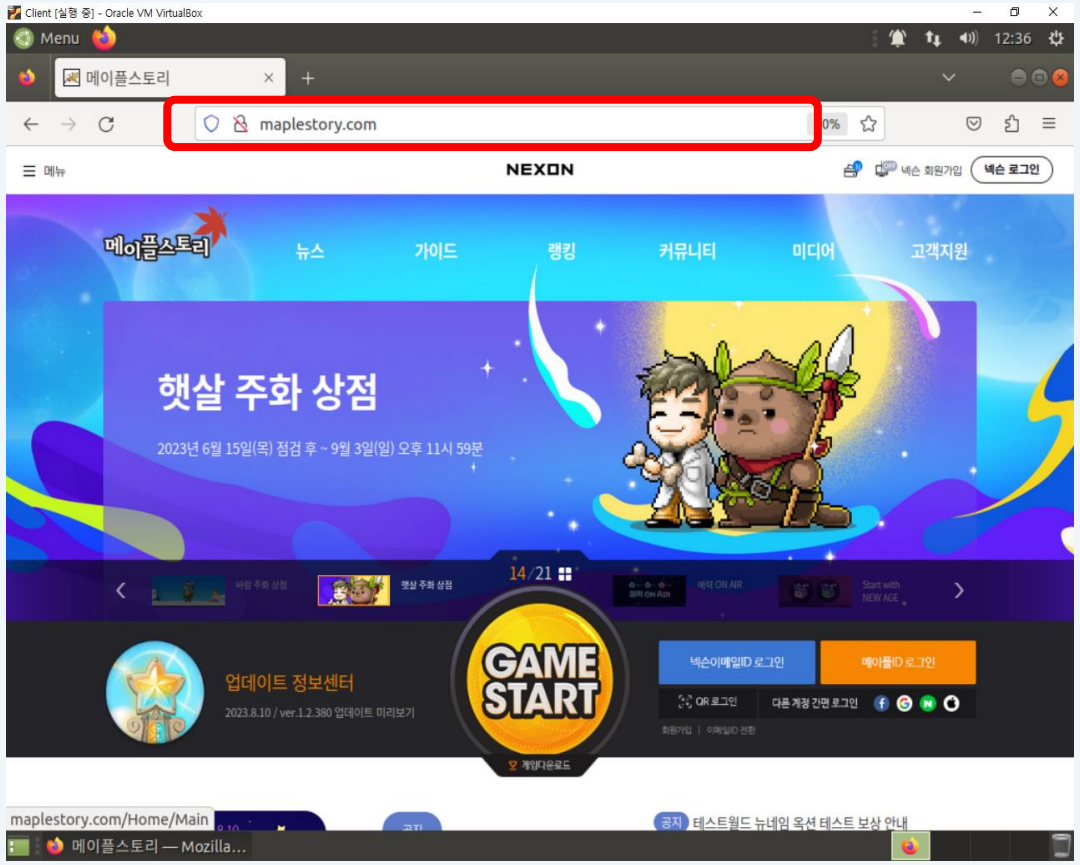
PART 3-1

작동 화면



❶

유해서비스 차단 프로그램 **실행**

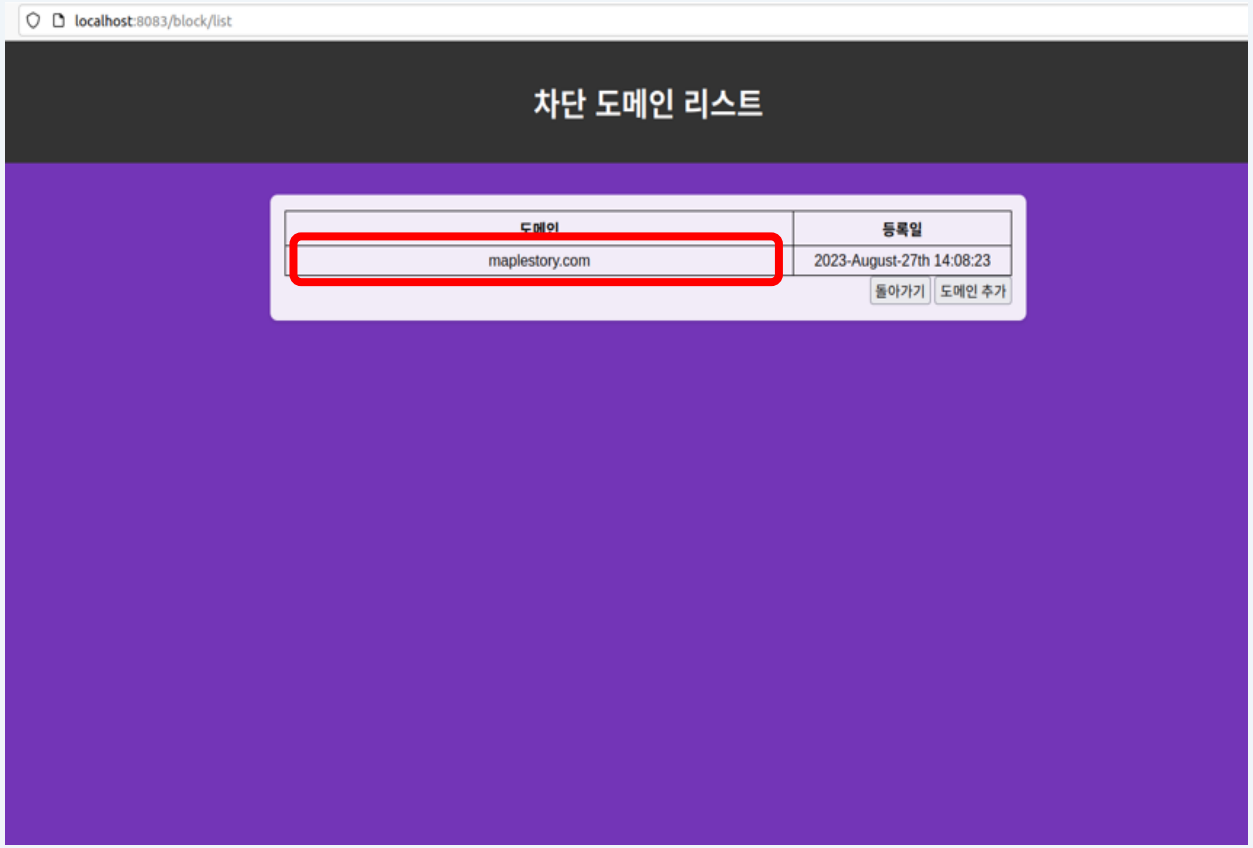


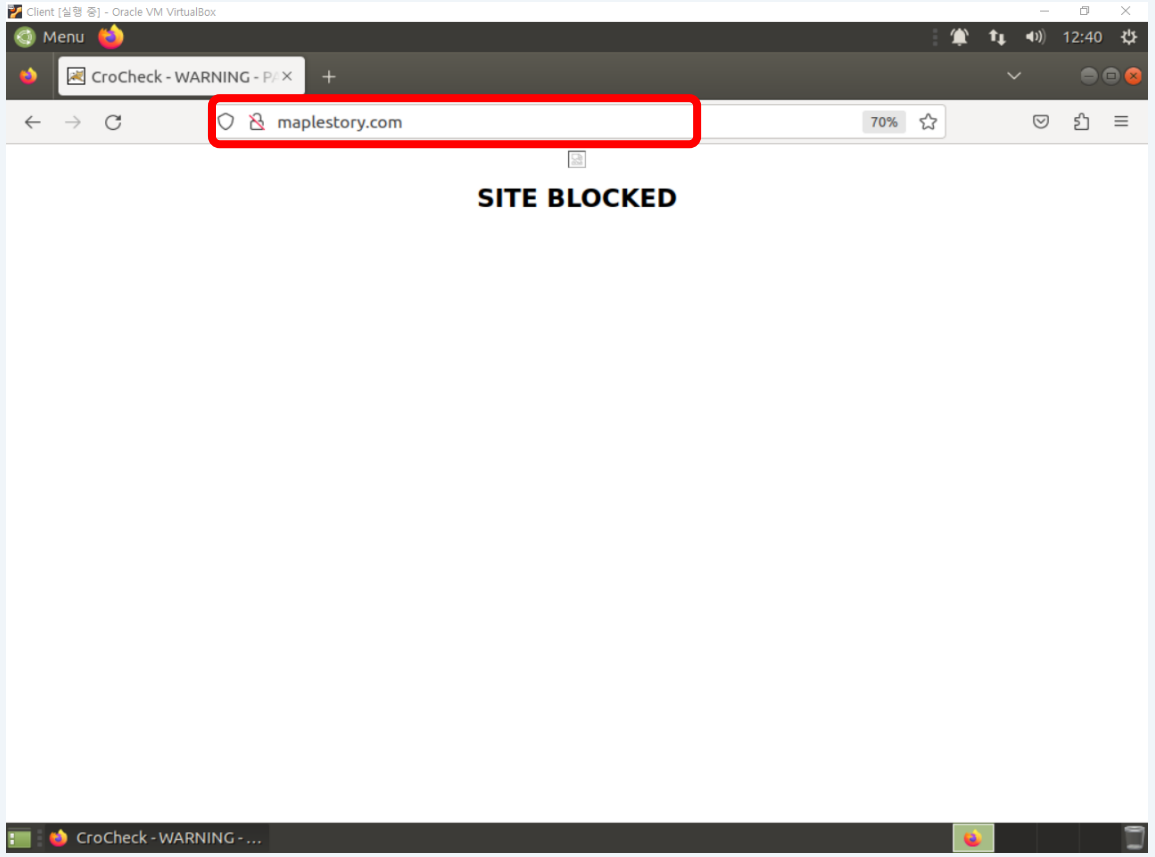
❷

(**차단 전**) 도메인 접속

PART 3-2

작동 화면





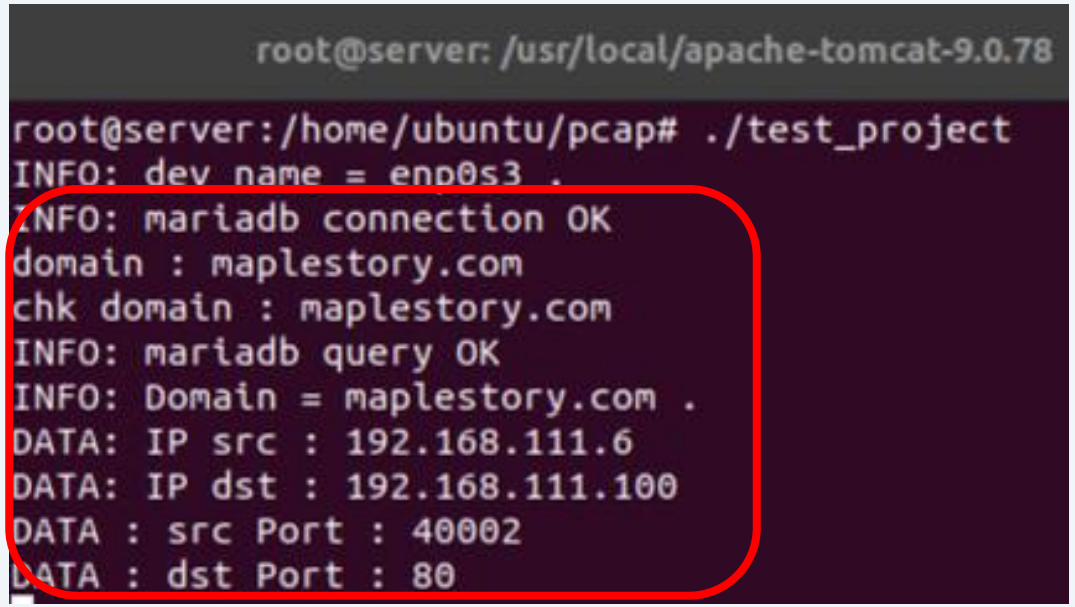
3

관리자 웹 페이지 -> 차단 도메인 추가

4

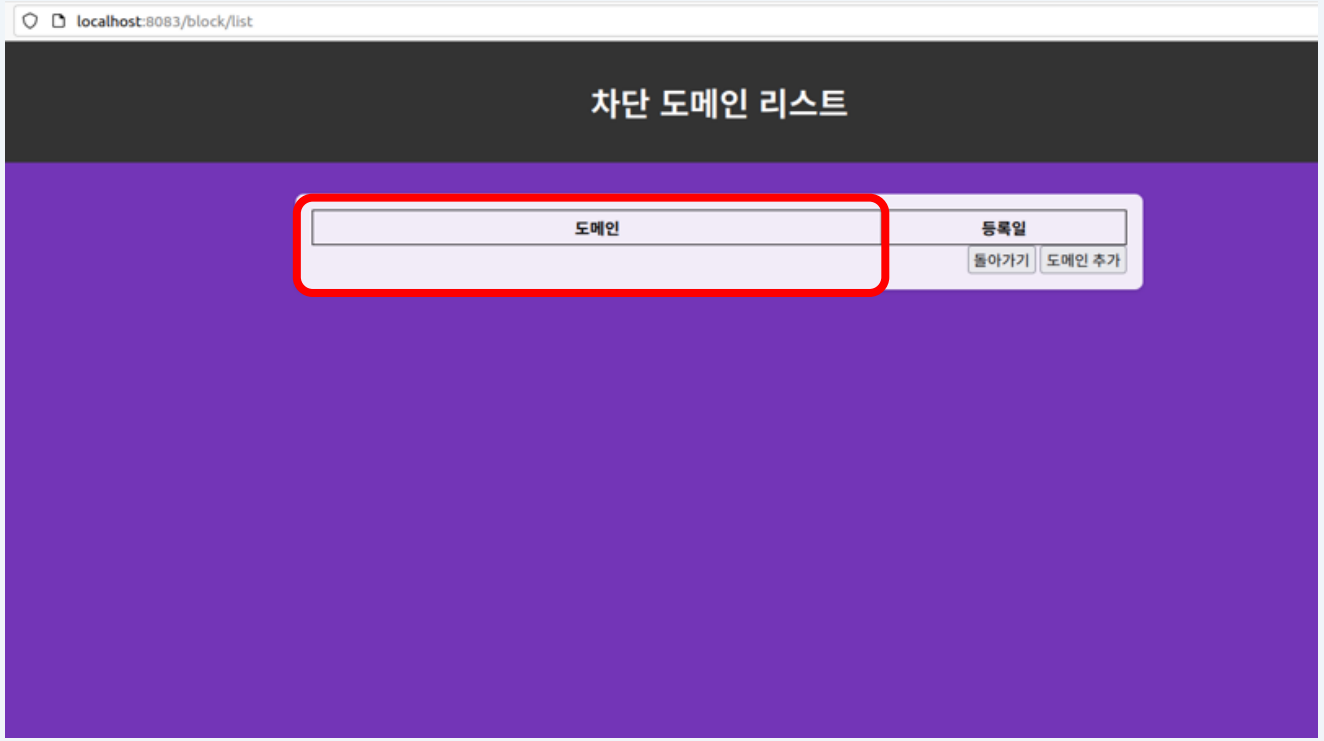
(차단 후) 도메인 접속

작동 화면



5

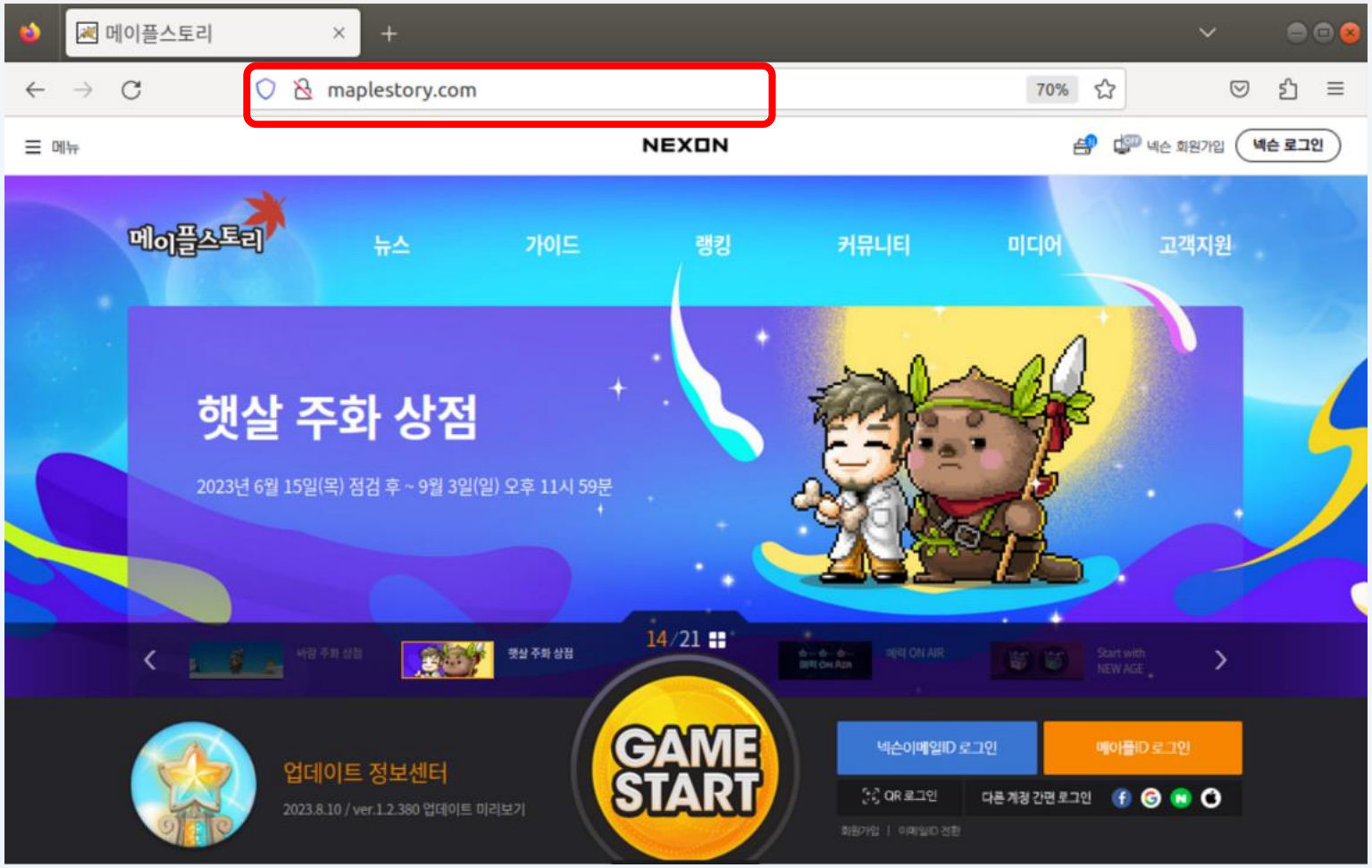
차단 도메인에 접속 시 작동하는 모습



6

차단 도메인 목록 삭제

작동 화면



7

(차단 도메인 삭제 후) 접속 가능

코드수정 및 추가 함수

코드 수정

전

```
tcphdr->source = ((struct tcphdr *)(pre_packet + size_vlan + 14 + 20))->dest ;
tcphdr->dest = ((struct tcphdr *)(pre_packet + size_vlan + 14 + 20))->source ;
tcphdr->seq = ((struct tcphdr *)(pre_packet + size_vlan + 14 + 20))->ack_seq ;
tcphdr->ack_seq = ((struct tcphdr *)(pre_packet + size_vlan + 14 + 20))->seq + 
tcphdr->window = ((struct tcphdr *)(pre_packet + size_vlan + 14 + 20))->window ;
```

중략 ...

반복된 구조체 포인터의

주소연산 간소화

후

```
struct iphdr* ip_p = (struct iphdr *)(pre_packet + size_vlan + SIZE_ETHERNET);
struct tcphdr* tcp_p = (struct tcphdr *)(pre_packet + size_vlan +
                                         SIZE_ETHERNET + SIZE_IP);
```

코드 수정

변조될 패킷의 payload를 배열로 선언함으로써
strlen() 을 통해 payload_size를 구함

전

```
post_payload_size = 230 + 65  ;
memcpy ( (char*)packet + 40, "HTTP/1.1 200 OK\x0d\x0a"
        "Content-Length: 230\x0d\x0a"
        "Content-Type: text/html"
        "\x0d\x0a\x0d\x0a"
        "<html>\r\n"
        "<head>\r\n"
        "<meta charset=\"UTF-8\">\r\n"
        "<title>\r\n"
        "CroCheck - WARNING - PAGE\r\n"
            "SITE BLOCKED - WARNING - \r\n"
        "</title>\r\n"
        "</head>\r\n"
        "<body>\r\n"
```

중략 …

후

```
char change_payload[] =
    "HTTP/1.1 200 OK\x0d\x0a"
    "Content-Length: 230\x0d\x0a"
    "Content-Type: text/html"
    "\x0d\x0a\x0d\x0a"
    "<html>\r\n"
        "<head>\r\n"
            "<meta charset=\"UTF-8\">\r\n"
            "<title>\r\n"
                "CroCheck - WARNING - PAGE\r\n"
                    "SITE BLOCKED - WARNING - \r\n"
            "</title>\r\n"
        "</head>\r\n"
        "<body>\r\n"
```

추가 함수 생성 전 함수

```
void got_packet(u_char *args, const struct pcap_pkthdr *header,
    const u_char *packet) {

    /* ethernet headers are always exactly 14 bytes */
    #define SIZE_ETHERNET 14

    u_int size_ip;
    u_int size_tcp;

    const struct sniff_ethernet *ethernet; /* The ethernet header */
    const struct sniff_ip *ip; /* The IP header */
    const struct sniff_tcp *tcp; /* The TCP header */
    const char *payload; /* Packet payload */

    ethernet = (struct sniff_ethernet*)(packet);
    ip = (struct sniff_ip*)(packet + SIZE_ETHERNET);
    size_ip = IP_HL(ip)*4;
    if (size_ip < 20) {
        printf("   * Invalid IP header length: %u bytes\n", size_ip);
        return;
    }
    tcp = (struct sniff_tcp*)(packet + SIZE_ETHERNET + size_ip);
    size_tcp = TH_OFF(tcp)*4;
    if (size_tcp < 20) {
        printf("   * Invalid TCP header length: %u bytes\n", size_tcp);
        return;
    }
    payload = (u_char *)(packet + SIZE_ETHERNET + size_ip + size_tcp);

    unsigned short int  payload_len = 0;
    payload_len = ntohs(ip->ip_len) - size_ip - size_tcp ;
    //printf("INFO: payload_len = %u .\n", payload_len);

    //printf("Jacked a packet with "
    //       "length of [%d]\n", header->len);

    // print Ethernet address .
    if ( 0 ) {
    printf("DATA: dest MAC : %02x:%02x:%02x:%02x:%02x:%02x\n" ,
                ethernet->ether_dhost[0],
                ethernet->ether_dhost[1],
                ethernet->ether_dhost[2],
                ethernet->ether_dhost[3],
                ethernet->ether_dhost[4],
                ethernet->ether_dhost[5]
            );
```

중략 ...

got_packet 함수 내부에서 한번에 처리하던 기능들을 각각의 목적에 맞게 함수로 만듦으로써 코드의 효율성과 가독성을 높임

추가 함수 생성 후 함수

```
void got_packet(u_char *args, const struct pcap_pkthdr *header,
    const u_char *packet) {

    struct sniff_ethernet *ethernet;
    struct sniff_ip *ip;
    struct sniff_tcp *tcp;
    char *payload;
    unsigned short int payload_len = 0;
    int result;

    char src_ip_str[SIZE_IP_STR];
    char dst_ip_str[SIZE_IP_STR];


    if((payload_len = pcap_len(&ethernet, &ip, &tcp, &payload, packet)) == -1)
    {
        printf("pcap len error\n");
        return;
    }

    if(payload_len == 0)
        return;

    u_char* domain;
    int domain_len = 0;


    domain_len = get_domain(payload, &domain);
    if(domain_len == 0)
        return;

    chk_domain_cnt = get_check_domain(chk_domain, chk_domain_cnt);

    result = domain_check(domain, domain_len, chk_domain_cnt, chk_domain);

    //insert log to db .
    insert_db_func(ip,tcp,src_ip_str,dst_ip_str,domain,result);

    if( result == 0)
    {
        int sendraw_ret = sendraw(packet , sendraw_mode);
        print_info(ethernet, ip, tcp, payload, domain);
        return;
    }
```

추가 함수

```
int get_domain(char* payload, u_char ** domain)
```

캡쳐된 패킷의 payload 부분에서
도메인의 주소를 get 하는 함수

```
if (domain_str != NULL) {
    domain_end = strstr(domain_str, "\x0d\x0a");
    if (domain_end != NULL){
        domain_len = domain_end - domain_str - HOST_LEN;
        *domain = (char *)malloc(domain_len + 1);
        strncpy(*domain, domain_str + HOST_LEN, domain_len);
        strcpy((*domain)+domain_len,"\0");
    }
}
```

도메인 주소를 담을 변수에 도메인 주소의 길이만큼 메모리를 동적 할당

도메인 주소의 끝에 NULL문자를 추가

추가 함수

```
int get_check_domain(char* chk_domain[256], int row_cnt)
```

DB에 저장된 차단 도메인을 get 하는 함수

```
if(row_cnt == 0)
{
    row_cnt = mysql_num_rows(result);

    for ( int i = 0; i < row_cnt; i++){

        chk_domain[i] = malloc(256);
        if ( chk_domain[i] == NULL ) {
            fprintf(stderr, "ERROR: malloc fail !!\n");
        }

        row=mysql_fetch_row(result);
        strcpy(chk_domain[i],row[0]);
    }
}
```

DB에서 가져온 차단 도메인의 수 만큼 배열에 메모리 동적 할당 후 strcpy() 를 통해서 문자열 복사

PART 2-3

추가 함수

```
else if (row_cnt != (new_row_cnt = mysql_num_rows(result)))
```

DB에 추가로 도메인주소가 들어왔을 때 동작

```
if(row_cnt < new_row_cnt)
{
    new_select_command = malloc(100);
    sprintf( new_select_command , "SELECT domain FROM blocked_domain_list"
                "ORDER BY created_at DESC LIMIT %d", (new_row_cnt - row_cnt) );
    mysql_query( db_connection , new_select_command );
    result = mysql_store_result(db_connection);

    while(row_cnt < new_row_cnt)
    {
        printf("chk_domain : %s\n",chk_domain[row_cnt-1]);
        chk_domain[row_cnt] = malloc(256);
        if ( chk_domain[row_cnt] == NULL ) {
            fprintf(stderr, "ERROR: malloc fail !!\n");
        }
        row=mysql_fetch_row(result);
        strcpy(chk_domain[row_cnt] ,row[0]);
        row_cnt++;
    }
}
```

새롭게 추가된 도메인주소를 가져오는 query문

추가된 도메인주소의 개수만큼 배열에 메모리 동적할당 및 문자열 복사

PART 2-4

추가 함수

```
int domain_check(const u_char* domain, const int domain_len,
        const int chk_domain_cnt, char* chk_domain[chk_domain_cnt])
```

캡쳐된 패킷의 도메인주소와
DB에 저장된 차단 도메인주소를
비교 해주는 함수

```
if(domain_len) {
    int cmp_ret = 1;

    for ( int i = 0; i < chk_domain_cnt; i++) {
        int str1_len = strlen(chk_domain[i]);

        if ( str1_len != domain_len ){
            continue;
        } else {
            printf("domain : %s \n", domain);
            printf("chk domain : %s \n", chk_domain[i]);
        }
        cmp_ret = strcmp(domain, chk_domain[i]);

        if(cmp_ret == 0){
            return cmp_ret;
        }
    } //end for loop 1.
```

차단 도메인주소의 길이와 캡쳐된 패킷
의 도메인주소의 길이가 같을때만 동작
하도록 만듦

두 도메인주소의 길이가 같다면
차단 되어야할 도메인 이므로
strcmp()를 통해서 sendrow()
함수 동작의 여부를 결정

Trouble Shooting

문제점

```
int get_domain(char* payload, u_char * domain)
```

중략 ...

```
if (domain_str != NULL) {
    domain_end = strstr(domain_str, "\x0d\x0a");
    if (domain_end != NULL){
        domain_len = domain_end - domain_str - HOST_LEN;
        domain = (char *)malloc(domain_len + 1);
        strncpy(*domain, domain_str + HOST_LEN, domain_len);
        strcpy((*domain)+domain_len,"\0");
    }
}
```

→ 1차원 포인터의 메모리 동적 할당 과정에서 오류발생

```
domain(get_domain function) : naver.com

domain : `N
```

→ 함수 내부에서는 정상적으로 나오지만,
함수 외부에서는 비정상으로 나옴

해결안

```
int get_domain(char* payload, u_char ** domain)
```

중략 ...

```
if (domain_str != NULL) {
    domain_end = strstr(domain_str, "\x0d\x0a");
    if (domain_end != NULL){
        domain_len = domain_end - domain_str - HOST_LEN;
        *domain = (char *)malloc(domain_len + 1);
        strncpy(*domain, domain_str + HOST_LEN, domain_len);
        strcpy((*domain)+domain_len,"\0");
    }
}
```

→ 함수의 매개변수를 2차원 포인터로 바꿈으로써 문제 해결

```
domain : naver.com
```

→ 함수 외부에서도 정상적으로 나옴

문제점

```
root@server:/home/ubuntu/sockTest/project# ./3_project
INFO: mariadb connection OK
DEBUG: befor domain == fifaonline4.nexon.com
DEBUG: after domain == fifaonline4.nexon.com
INFO: mariadb query OK
```

→ 도메인 차단전 접속시도

Name	Value
Updated Rows	1
Query	INSERT INTO chk (domain) value("fifaonline4.nexon.com")
Start time	Thu Aug 10 18:50:13 KST 2023
Finish time	Thu Aug 10 18:50:13 KST 2023

→ 차단 도메인 추가

```
DEBUG: befor domain == fifaonline4.nexon.com
세그멘테이션 오류 (코어 덤프됨)
```

→ fifaonlin4.nexon.com을 추가했으나 **차단을 못하는 모습**

문제점

```
root@server:/home/ubuntu/pcap# ./a.out
domain(get_domain function) : fifaonline4.nexon.com
domain : fifaonline4.nexon.com
chk domain : fifaonline4.nexon.com
INFO: Domain = fifaonline4.nexon.com .
DATA: IP src : 10.0.2.15
DATA: IP src : 222.122.222.106
DATA : src Port : 41590
DATA : dst Port : 80
```

→ 프로그램을 다시 실행했을때에는 정상적으로 작동

```
else if (row_cnt != (new_row_cnt = mysql_num_rows(result))){

    if(row_cnt < new_row_cnt)
    {
        sprintf( new_select_command , "SELECT domain FROM blocked_domain_list ORDER BY"
                                      "created_at DESC LIMIT %d", (new_row_cnt - row_cnt));
        mysql_query( db_connection , new_select_command );
```

→ 새로운 query문을 생성하는 과정에서 문제가 발생하는 것을 확인

해결안

```
else if (row_cnt != (new_row_cnt = mysql_num_rows(result))){

    if(row_cnt < new_row_cnt)
    {
        new_select_command = malloc(100);
        sprintf( new_select_command , "SELECT domain FROM blocked_domain_list ORDER BY"
                                      "created_at DESC LIMIT %d", (new_row_cnt - row_cnt));
        mysql_query( db_connection , new_select_command );
```

new_select_command 에
동적 메모리 할당을 함으로써 문제 **해결**

Thank You