|  |
| --- |
| **보고서**  보고서 및 논문 윤리 서약   1. 나는 보고서 및 논문의 내용을 조작하지 않겠습니다. 2. 나는 다른 사람의 보고서 및 논문의 내용을 내 것처럼 무단으로 복사하지 않겠습니다. 3. 나는 다른 사람의 보고서 및 논문의 내용을 참고하거나 인용할 시 참고 및 인용 형식을 갖추고 출처를 반드시 밝히겠습니다. 4. EMB000004b8484c나는 보고서 및 논문을 대신하여 작성하도록 청탁하지도 청탁 받지도 않겠습니다.   나는 보고서 및 논문 작성 시 위법 행위를 하지 않고, 명지인으로서 또한 공학인으로서 나의 양심과 명예를 지킬 것을 약속합니다.  보고서명 : 상세 분석 보고서  학 과 : 컴퓨터공학과  과 목 : 공개 SW 실무  담당교수 : 류연승  마 감 일 : 2018-11-28  제 출 일 : 2018-11-27  조원1 학번 : 60112423 이름 : 차건희 (서명)  조원2 학번 : 60142249 이름 : 김민준 (서명)  조원3 학번 : 60142281 이름 : 박현웅 (서명)  조원4 학번 : 60162118 이름 : 민경찬 (서명) |

목차

**1. Domain analyzer 개요** **1~2**

**2. 요구사항2**

**3. 상세 분석 3**

**3.1. Domain analyzer global 변수3**

**3.2. Domain analyzer 함수3**

**4. 추가 기능 제안4**

1. Domain analyzer 개요

Domain analyzer는 주어진 도메인에 대한 정보를 자동으로 찾고, 보고하는 보안 분석 도구이다. 주된 목적은 도메인을 자동으로 분석하고 이를 통해 웹페이지 보안 강화에 도움을 주는 것이다. 이를 위해 Domain name server의 정보(국가 정보, 등록 유형, ip, hostname)와 mail server의 정보 등을 찾는다.

1. 요구사항

Domain analyzer를 사용하기 위해서 python2가 설치 되어 있어야 하며, 또한 Python 라이브러리인 nmap와 패키지인 dnspython가 설치 되어 있어야 한다. Dnspython은 저레벨에서 Domain name server에 접속하도록 도움을 준다. Nmap은 port를 검색하는데 사용되는 라이브러리이다. Domain analyzer에서는 사용자의 컴퓨터에서 Nmap의 GUI인 zenmap이 설치되어 있다면 zenmap의 기능을 사용 할 수 있도록 되어있다.

1. 상세 분석

Domain analyzer는 domain\_analyzer.py와 ansistrm.py, pyText2pdf.py의 총 3개의 python파일로 구성되어 있다. 이중 ansistrm.py는 단지 출력문에 색을 입히는 기능을 가진다. 또 pyText2pdf.py는 출력되는 텍스트를 pdf로 저장하는 단순한 기능만을 지닌다. 따라서 ansistrm.py와 pyText2pdf.py는 제외하고 분석을 진행했다.

* 1. Domain\_analyzer.py

Domain\_analyzer.py는 메인 프로그램으로, 프로그램 실행시 도메인 주소를 입력 받고 도메인에 대한 네임서버, 메일서버, 호스트 등을 검색한다. Domain\_anlyzer.py에서는 다음과 같은 글로벌 변수들을 가진다.

Domain\_date은 딕셔너리 타입으로 도메인을 키로 갖는 정보들을 저장한다.

Domain\_date[‘ipsInfo’]에는 해당 도메인의 ip 관련 정보(국가, 위치 등)가 저장된다.

Domain\_date[‘DomainInfo’]에는 해당 도메인의 관련 정보가 저장된다.

Check\_common\_hosts\_names은 프로그램 실행시 -j 옵션을 줄 경우 true가 되는데, 이 경우 리스트 변수인 common\_hostnames에 저장 되어있는 호스트 네임이 들어있는지 탐색하게 된다.

Common\_hostname 변수는 ‘www’, ‘ftp’, ‘windows’, ‘oracle’, ‘sql’과 같은 일반적인 호스트 네임을 저장하고 있다.

Not\_goog\_mail 변수는 bool 타입으로 프로그램 실행시 -g 옵션을 줄 경우 true가 된다. false 경우 프로그램은 구글 그룹스에서 도메인 이름으로 된 구글 그룹스 메일을 찾는다. 그 후 email\_list 딕셔너리 변수에 이메일 정보를 저장한다.

Output\_directory, output\_file, output\_file\_handler는 domain\_analyzer의 결과를 출력하는 출력파일의 정보를 저장하고, 관리하는데 쓰인다.

Net\_block은 bool 변수로 프로그램 실행시 -n 옵션을 줄 경우 true가 된다. 이 경우 프로그램은 nmap라이브러리를 사용해 damain\_data[‘IpsInfo’]에 저장되어 있는 ip 정보들을 reverse domain하여 ip주소로 도메인을 검색한다. 그리고 ip 주소의 netblock이 24인지 확인한다.

Create\_pdf는 bool 변수로 프로그램 실행시 -f 옵션을 줄 경우 true가 된다. 이 경우 출력 파일을 pdf 파일로 저장한다.

Robin\_hood는 bool 변수로 프로그램 실행시 -r 옵션을 줄 경우 true가 된다. 이 경우 robin\_hood\_send() 함수가 실행되는데, 자세한 내용은 밑에 기술한다.

Countrys는 bool 변수로 프로그램 실행시 -c 옵션을 줄 경우 true가 된다. 이 경우 GeoIp 라이브러리가 있을 때, Geoip\_cache 변수에 ip별 국가 정보를 저장하는데 사용된다.

Zenmap은 0으로 초기화되는 변수이다. 프로그램 실행시 -e 옵션을 줄 경우 zemap=1이 된다. 이경우 zenmap이 설치되어 있는 폴더에서 zenmap을 실행한다.

Color는 bool 변수로 프로그램 실행시 -C(대문자 C) 옵션을 줄 경우 true가 된다. 이 경우 출력 파일에 색상을 넣을 수 있다.

Normal\_port\_list는 리스트 타입으로 일반적인 포트 번호들을 저장하고 있다. 호스트 정보를 확인할 때, 호스트의 port가 normal\_port\_list 내의 포트번호와 같게되면 경고 메시지가 출력되지 않는다.

check\_spf는 bool변수로 프로그램 실행시 -q 옵션을 줄 경우 true가 된다. false가 되면 Sender Policy Framework record를 확인하게 된다.

Zone\_transfer는 bool 변수로 프로그램 실행시 -t 옵션을 줄 경우 true가 된다. 해당 도메인의 zone transfer가 취약한지 확인한다.

Not\_subdomains는 bool 변수로 프로그램 실행시 -s 옵션을 줄 경우 true가 된다. False 일 경우 해당 도메인이 subdomain인지 확인한다. 또한 해당 도메인에 subdomain이 있다면 subdomains\_found[]에 저장하게 된다.

Webcrawl 변수는 bool 변수로 프로그램실행시 -w 옵션을 주면 true가 된다. False일 경우 웹사이트를 모두 크롤링 한다.

max\_amount\_to\_crawl는 크롤링시 최대 어느정도 크롤링 할건지 입력받는 변수이다. 프로그램 시작시 -m 옵션으로 입력 받을 수 있다.

amount\_of\_random\_domains은 0으로 초기화된다. 프로그램 시작시 -k 옵션과 함께 정수를 줄 수 있다. 주어진 정수만큼 무작위의 도메인을 분석하게 된다.

domains\_still\_to\_analyze은 리스트 변수로, 현재 분석중인 도메인을 저장한다.

ignore\_host\_pattern은 스트링으로, 프로그램 시작시 -v 옵션과 함께 호스트 패턴을 주어주면, 같은 패턴의 호스트는 무시하게 된다.

Robtex\_domains는 bool 변수로 프로그램 시작시 -b 옵션을 주면 true가 된다. false일 때, zone transfer가 발견된 도메인 네임 서비스 서버에 대해 https://www.robtex.com/ 에서 해당 도메인을 검색해 분석하게 된다.

all\_robtex는 Robtex\_domains와 유사하지만 zone transfer가 발견되지 않아도 해당 도메인을 [www.robtex.com](http://www.robtex.com)에서 검색,분석한다.

* 1. Domain\_analyzer.py의 함수

**Version(), usage()** 함수는 각각 프로그램의 버전과 사용법에 대해 출력한다.

def version():

print( "+----------------------------------------------------------------------+")

print( "| "+ sys.argv[0] + " Version "+ vernum +" |")

print( "| This program is free software; you can redistribute it and/or modify |")

……

def usage():

print( "options:")

print( " -h, --help Show this help message and exit.")

print( " -V, --version Output version information and exit.")

print( " -D, --debug Debug.")

print( " -d, --domain Domain to analyze.") |"

…..

**Get\_NS\_records(domain)**함수는 domain을 받아와 해당 도메인의 네임 서버 레코드 정보를 가져온다. 그 정보는 네임 서버의 ip, hostname, 등록 유형, ip의 국가 정보이다.

def get\_NS\_records(domain):

…..

for rdata in ns\_servers:

if debug:

logging.debug('\t\t> Looking for {0} IP address'.format(rdata.to\_text()))

# We search for the IP of each NSs

ip\_list = dns.resolver.query(rdata.to\_text()[:-1], 'A')

# For each IP we store its information

for ip in ip\_list:

ip\_registry=[]

if debug:

logging.debug('\t\t> NS IP: {0}'.format(ip.to\_text()))

try:

# If already exists this IP in the registry

# We search for this IP in the main dict

ip\_registry=domain\_data['IpsInfo'][ip.to\_text()]

# Here we store the hostname in a dictionary. The index is 'HostName'

hostname['HostName']=rdata.to\_text()[:-1]

ip\_registry.append(hostname) ……

**def dns\_request(domain)**함수는 get\_NS\_records, get\_MX\_records 함수를 호출하여 도메인 관련 정보를 가져온다.

def dns\_request(domain):

…..

get\_NS\_records(domain)

get\_MX\_records(domain)

……..

**get\_MX\_records(domain)**함수는 domain을 받아와 해당 도메인의 메일 서버 레코드 정보를 가져온다. 그 정보는 메일 서버의 ip, hostname, 등록 유형, ip의 국가 정보이다.

def get\_MX\_records(domain):

mail\_servers = dns.resolver.query(domain, 'MX')

for rdata in mail\_servers:

# We search for the IP of each NSs

ip\_list = dns.resolver.query(rdata.exchange.to\_text()[:-1], 'A')

# For each IP we store its information

for ip in ip\_list:

ip\_registry=[]

….

**def check\_PTR\_record(ip)**함수는 ip로부터 도메인 주소를 역으로 변환한다.

def check\_PTR\_record(ip):

temp\_ip=ip.split('.')

temp\_ip.reverse()

reverse\_ip=""

for i in temp\_ip:

reverse\_ip=reverse\_ip+i+'.'

reverse\_ip=reverse\_ip+'in-addr.arpa'

reverse\_name\_result = dns.resolver.query(reverse\_ip, 'PTR')

for i in reverse\_name\_result:

reverse\_name=i.to\_text()[:-1]

return reverse\_name

**check\_SPF\_record(domain)**함수는 도메인에서 SPF 레코드 값을 찾아온다.

def check\_SPF\_record(domain):

temp\_spf = dns.resolver.query(domain, 'TXT')

# For each spf record...

for spf\_record in temp\_spf:

if 'v=spf' in spf\_record.to\_text():

# We found a SPF record

if debug:

print( '\t\t> SPf record found: {0}'.format(spf\_record.to\_text()))

hosttype={}

ip\_registry=[]

# Split it in parts

spf\_record\_splitted=spf\_record.to\_text().split()

# For each part of the spf record

for part in spf\_record\_splitted:

# Look for hostnames

if 'a:' in part:

…..

**check\_A\_record(domain,text=””)** 함수는 도메인에서 A 레코드 값을 찾아온다. 해당 A 레코드의 호스트가 있는지, 하위 도메인이 있는지 분석한다.

def check\_A\_records(domain,text=""):

…….

if common\_host != domain.split('.')[0]:

# We add the new subdomain for later analysis

ip\_registry=[]

subdomain={}

for ip in host\_name\_ips:

subdomain['SubDomain']=common\_host+'.'+domain

ip\_registry.append(subdomain)

# We store it in the main dictionary

a=[]

a=copy.deepcopy(ip\_registry)

domain\_data['IpsInfo'][ip.to\_text()]=a

…..

**find\_and\_analyze\_random\_domains(domain,amount)**함수는 amount만큼의 도메인을 인터넷에서 random하게 찾는다.

def find\_and\_analyze\_random\_domains(domain, amount):

…..

while amount:

# We search in the first 100 pages

page\_counter\_web = random.randrange(1,100,10)

try:

results\_web = 'http://www.google.com/search?q=inurl%3a'+str(domain)+'&hl=en&btnG=Search&aq=f&start='+ repr(page\_counter\_web) + '&sa=N'

request\_web = urllib.request.Request(results\_web)

request\_web.add\_header('User-Agent','Mozilla/4.0 (compatible;MSIE 5.5; Windows NT 5.0)')

opener\_web = urllib.request.build\_opener()

text = opener\_web.open(request\_web).read()

# This re extracts the domains

domains\_web = (re.findall('(http:\/\/\w[\w\.\-]+\.'+domain\_re+')',text))

…..

**check\_domain\_emails(domain)**함수는 주어진 도메인의 구글 그룹스 메일을 찾는다. 메일 앞부분(‘@’ 앞)이 20자 이상인 경우 딕셔너리에 저장 하지 않는다.

def check\_domain\_emails(domain):

…..

while page\_counter < 50 :

results = 'http://groups.google.com/groups?q='+str(domain)+'&hl=en&lr=&ie=UTF-8&start=' + repr(page\_counter) + '&sa=N'

request = urllib.request.Request(results)

request.add\_header('User-Agent','Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)')

opener = urllib.request.build\_opener()

text = opener.open(request).read().decode()

emails = (re.findall('([\w\.\-]+@'+domain+')',StripTags(text)))

for email in emails:

d[email]=1

uniq\_emails=d.keys()

page\_counter = page\_counter + 10

**check\_active\_host()**함수는 nmap을 이용하여 주어진 호스트가 살아있는지 확인한다.

def check\_active\_host():

….

for ip in domain\_data['IpsInfo']:

ignore=False

ip\_registry=domain\_data['IpsInfo'][ip]

reason=""

# If any of the host names has the 'ignored' pattern, do not check it!

if ignore\_host\_pattern:

for dict in ip\_registry:

for i in dict.keys():

if i == 'HostName':

for pattern in ignore\_host\_pattern.split(','):

if pattern in dict['HostName']:

ignore=True

print('\t\tPattern: {0}, Hostname: {1}. Ignoring!'.format(pattern,dict['HostName']))

break

……

**analyze\_domain(domain)**함수는 **check\_active\_host(),check\_domain\_emails(domain) 등** 여러 함수를 호출하여 도메인을 검색하고 정보를 출력한다.

def analyze\_domain(domain):

….

if dns\_request(domain) != -1:

# Check domain related emails using goog-mail.py

if not\_goog\_mail==True:

check\_domain\_emails(domain)

# If nmap is activated

if nmap:

# Check for active hosts with nmap

check\_active\_host()

# Scan ports with nmap

host\_info(domain)

# If we can webcrawl

if webcrawl == True and nmap:

web\_crawl\_domain()

# Print out the final summary

….

**robin\_hood\_send()함수는 domain analyzer가 분석한 결과를 해당 도메인에서 찾은 이메일들에게 전송하는 함수이다.**

def robin\_hood\_send():

…..

for ip in domain\_data['IpsInfo']:

# We extract IP information

ip\_registry=domain\_data['IpsInfo'][ip]

if debug:

print( '\t> For IP : {0}'.format(ip))

# We extract its vector

ip\_vect=domain\_data['IpsInfo'][ip]

# First we search for every MX type!

temp\_host\_name\_to\_crawl=[]

for dicts in ip\_vect:

if 'Type' in dicts:

# Store last domain for this IP

if 'MX' in dicts['Type']:

mail\_server=dicts.get('Type')

…..

1. 추가 기능 제안

현재 Domain\_anlyzer는 python2 버전만 사용할 수 있다. 이에 python3 사용자는 불편함을 느낄 수 있기 때문에, Python3 에서 사용 할 수 있도록 개선한다.

또한 영어로 프로그램이 구성되어 있어 한국어 사용자에게 불편함이 따른다. 이에 한국어 번역이 필요해 보인다. 프로그램에서 언어 설정이 가능하도록 기능을 추가한다.

Domain\_anlyzer가 해당 도메인에서 404\_not found, 408 요청 시간 초과 등, 에러를 확인 하지 못하는 점 또한 개선이 필요하다.