

# 토르 브라우저 핑거프린팅

[https://youtu.be/K\\_wOzprPUXI](https://youtu.be/K_wOzprPUXI)

# Contents

서론

실험 환경

하위주소에 따른 분류

포털별 서비스에 대한 분류

서버의 물리적 위치에 대한 분류



# 목표

- 인터넷을 사용하는 사용자가 어느 사이트에 접속하는지 확인
- 해당 사이트의 카테고리(뉴스, 부동산 등)에 대한 분류를 통해 사용자에게 대한 핑거프린팅 진행

# 배경

- 전송 중인 패킷을 가로채서 하는 분석은 암호화가 많이 되어 있기에 어려움
- 사용자의 PC에서 직접 발생하는 패킷을 이용하여 분석

# 실험 환경

- 가상머신 (VMware Workstation 15 Player (Non-commercial use only))
- Ubuntu 64-bit 16.04 LTS
- Torsocks 2.1.0
- Intel® Core™ i3-8145U CPU @ 2.10GHz

# Torsocks

- CUI 기반
- 토르 네트워크를 이용한  
SOCKS 애플리케이션

```
# The world's leading software development platform - GitHub (p1 of 13)
# GitHub

Skip to content
GitHub no longer supports this web browser. Learn more about the
browsers we support.

Sign up (BUTTON)
(BUTTON)

* Why GitHub?
  Features →
    + Code review
    + Project management
    + Integrations
    + Actions
    + Packages
    + Security
    + Team management
    + Hosting
    + Mobile
    + Customer stories →
    + Security →
* Team
* Enterprise
* Explore
  + Explore GitHub →

Learn & contribute
  + Topics
(NORMAL LINK) Use right-arrow or <return> to activate.
Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Torsocks를 이용해 Github에 접속

# SOCKS

- **SOCK**et **S**ecure Protocol
- SOCKS 서버는 클라이언트의 요청을 받아 클라이언트와 서버 사이에서 릴레이
- 네트워크 방화벽이 도입되면서 인터넷이 내부와 외부로 나뉘게 되고, 이를 통과할 수 있도록 하는 프레임워크 및 강력한 인증이 필요해지면서 등장
- SOCKSv4는 TCP 기반으로 작동
- SOCKSv5는 UDP에서도 작동하도록 확장 및 IPv6 주소화 방식 사용 가능

# 이용 프로그램

- <https://github.com/wisepythagoras/website-fingerprinting.git>
- 캡처 프로그램을 실행시킨 뒤 torsocks를 통해 페이지를 접속하며 패킷 수집
- 수집된 패킷을 학습 (머신러닝) 및 분류 (kNN)



# 수집 자동화

- 파이썬 매크로 프로그램 작성을 통해 수집 자동화

```
hoon@ubuntu: ~/website-fingerprinting
GNU nano 2.5.3 File: test4.py

#!/usr/bin/python

from time import sleep
import os
import json

if __name__ == '__main__':

    with open('config.json') as json_file:
        json_data = json.load(json_file)

    numberOfCollect = 20

    for name in json_data['pcaps']:
        print('[[ ' + name + ' ]]')
        for i in range(numberOfCollect):
            print( str(i+1) + '/' + str(numberOfCollect) + '...' )
            os.system('sudo gnome-terminal -x ./test5.py ' + name)
            sleep(11.5)
        print('collecting done\n')
```

```
hoon@ubuntu: ~/website-fingerprinting
GNU nano 2.5.3 File: test5.py

#!/usr/bin/python

from time import sleep
import os
import sys

if __name__ == '__main__':
    os.system('./pcaps/capture.sh ' + sys.argv[1] + ' &')
    sleep(.5)
    os.system('torsocks lynx https://' + sys.argv[1] + ' &')
    sleep(10)
```

# 하위주소가 다를 경우에 대한 구분

- 이용 웹사이트 : MSN (<https://www.msn.com/>)
- 원래는 접속 지역에 따라 패킷 구분이 가능한지를 파악하려고 했으나, 의도와 다른 결과 얻음
- msn 사이트에 접속할 경우 접속 지역에 따라서 각기 다른 하위주소로 연결됨  
(ex. <https://www.msn.com/ko-kr/>, <https://www.msn.com/en-us> 등등)
- 이를 이용하여 각기 다른 exit relay에 대해 패킷 수집 및 분석

# 하위주소가 다를 경우에 대한 구분 - case #1

- case 1
- 하위주소를 직접 입력하여 접속한 사이트들에 대한 구분
- msn.com 으로 접속하는 것이 아닌, msn.com/ko-kr/ msn.com/en-us/ 처럼 하위주소까지 입력하여 접속하여 패킷 확보
- msn.com/ko-kr/ 과 msn.com/en-us/ 등 각기 다른 하위주소에 대한 구분이 가능한지에 대한 실험

# 하위주소가 다를 경우에 대한 구분 - case #1

- ko-kr (한국)
- en-us (미국)
- ja-jp (일본)
- zh-cn (중국)
- de-at (오스트리아)
- fr-fr (프랑스)
- es-es (스페인)
- it-it (이탈리아)
- nl-nl (네덜란드)
- da-dk (덴마크)
- 수집 및 학습에 이용된 하위주소 총 10개

# 하위주소가 다를 경우에 대한 구분 - case #1

- 서로에 대한 구분 가능
- 전체 패킷을 이용해 학습을 시킨 뒤, ko-kr이나 en-us 등의 하위주소에 따라 구분 가능

# 하위주소가 다를 경우에 대한 구분 - case #2

- case 2
- 하위주소를 입력하지 않고 접속한 사이트들에 대한 구분
- msn.com 으로 접속하여 리다이렉트된 하위주소의 패킷을 얻음
- exit relay가 랜덤으로 잡히기 때문에, 위치를 특정해서 얻지는 못함

## 하위주소가 다를 경우에 대한 구분 - case #2

- en-xl
  - de-at
  - fr-fr
  - it-it
  - 수집 및 학습에 이용된 하위주소 총 4개
- 
- 결과 : 서로에 대한 구분 가능

# 하위주소가 다를 경우에 대한 구분 - case #3

- case 3
- msn.com으로 접속해서 학습시킨 분류기가 msn.com/URL/ 으로 접속해서 얻은 패킷을 구분하는지
- 반대의 경우도 성립하는지
- 결과 : 구분하지 못함 (아예 msn으로 인식하지 못함)



# 포털별 서비스에 대한 분류

- 포털 사이트들의 하위 서비스 사이트들이 각 포털 사이트로 분류가 되는지 테스트
- 학습이 되지 않은 패킷을 이용하기에, 일종의 오픈월드 테스트로 볼 수 있음

# 수집 웹사이트

- 네이버(64개), 다음(46개), 네이버(34개)의 하위 사이트를 수집에 이용



# 수집 웹사이트

- 크롤링을 통해 URL 수집

```
public static void naver() throws Exception {  
  
    final String html = " HTML ELEMENTS ";  
    Document doc = Jsoup.parse(html);  
    Elements elem = doc.select("#category-div");  
  
    HashSet<String> set = new HashSet<>();  
    Elements as = elem.select("a");  
    for (Element a : as) {  
        String href = a.attr("href");  
        int cut;  
        if ( href.startsWith("https") ) cut = 8;  
        else cut = 7;  
        href = href.substring(cut);  
        try {  
            href = href.substring(0, href.indexOf("/"));  
            set.add(href);  
        } catch (StringIndexOutOfBoundsException e) {  
            set.add(href);  
        }  
    }  
  
    for (String s : set) {  
        System.out.println( "\"" + s + "\", " );  
    }  
}
```

```
public static void kakao() throws Exception {  
  
    Document doc = Jsoup.connect("https://www.daum.net/sitemap/index.html").get();  
    Elements elem = doc.select("#mArticle > div.box_sitemap");  
  
    HashSet<String> set = new HashSet<>();  
    Elements lis = elem.select("li");  
    for (Element li : lis) {  
        String href = li.child(0).attr("href");  
        int cut;  
        if ( href.startsWith("https") ) cut = 8;  
        else cut = 7;  
        href = href.substring(cut);  
        try {  
            href = href.substring(0, href.indexOf("/"));  
            set.add(href);  
        } catch (StringIndexOutOfBoundsException e) {  
            set.add(href);  
        }  
    }  
  
    for (String s : set) {  
        System.out.println( "\"" + s + "\", " );  
    }  
}
```

```
public static void nate() throws Exception {  
  
    Document doc = Jsoup.connect("https://www.nate.com/sitemap/index.html").get();  
    Elements elem = doc.select("#list0 > ul");  
  
    HashSet<String> set = new HashSet<>();  
    Elements as = elem.select("a");  
    for (Element a : as) {  
        String href = a.attr("href");  
        int cut;  
        if ( href.startsWith("https") ) cut = 8;  
        else if ( href.startsWith("http") ) cut = 7;  
        else cut = 2;  
        href = href.substring(cut);  
        try {  
            href = href.substring(0, href.indexOf("/"));  
            set.add(href);  
        } catch (StringIndexOutOfBoundsException e) {  
            set.add(href);  
        }  
    }  
  
    for (String s : set) {  
        System.out.println( "\"" + s + "\", " );  
    }  
}
```

# 실험 세팅

- 패킷 수집 시간 15초
- 패킷 개수 100개
- 사이트별로 서비스를 2,3개로 나눈 뒤 1개를 제외하고 전체를 학습시킨 뒤 그 1개가 가리키는 곳을 바탕으로 성공 여부를 판단
- ex. 네이버 64개 서비스를 22개, 22개, 20개로 나눈 뒤 첫 22개를 제외한 42개를 나머지 서비스들과 함께 학습에 이용한 뒤, 제외시켰던 22개 서비스를 분류
- 분류 기준 : 네이버 64개를 전부 “네이버” 카테고리로 묶은 뒤 “네이버”를 가리키면 분류

# 추가 수집 데이터

- 네이버, 다음, 네이트 외에 Alexa's top sites 10개 데이터 추가
- <https://www.alexac.com/topsites>
- tmail.com과 login.tmail.com은 중복될 수 있으므로 제외
- 11위의 360.cn 추가

	Site
1	Google.com
2	Youtube.com
3	Tmall.com
4	Baidu.com
5	Qq.com
6	Facebook.com
7	Sohu.com
8	Login.tmall.com
9	Taobao.com
10	Yahoo.com
11	360.cn

# 결과

- 네이버 정답률 : 53/64 (82.81%)
- 카카오 정답률 : 18/46 (39.13%)
- 네이트 정답률 : 8/34 (23.53%)
- 네이버는 높은 정답률을 보였으나, 나머지는 높지 않음

# 가정

- 서버의 물리적 위치가 분류에 영향을 줄 수 있지 않을까?

# 서버의 물리적 위치

- IP 주소를 이용해 WHOIS를 통해 서버의 물리적 위치 조회
- WHOIS 오픈API를 이용해 파이썬으로 물리적 위치 정보 확보

## WHOIS 조회

WHOIS 조회

210.89.164.90

query : 210.89.164.90

# KOREAN(UTF8)

조회하신 IPv4주소는 한국인터넷진흥원으로부터 아래의 관리대행자에게 할당되었으며, 할당 정보는 다음과 같습니다.

[ 네트워크 할당 정보 ]

IPv4주소 : 210.89.160.0 - 210.89.191.255 (/19)

기관명 : 케이넷비즈니스플랫폼 주식회사

서비스명 :

주소 : 경기도 성남시 분당구 분당내곡로 117

우편번호 : 13529

할당일자 : 20170116

[ 할당기관 연락처 보기 ]

조회하신 IPv4주소는 위의 관리대행자로부터 아래의 사용자에게 할당되었으며, 할당 정보는 다음과 같습니다.

[ 네트워크 할당 정보 ]

IPv4주소 : 210.89.160.0 - 210.89.175.255 (/20)

기관명 : 케이넷비즈니스플랫폼 주식회사

네트워크 구분 : INFA

주소 : 경기도 성남시 분당구 분당내곡로 117

우편번호 : 13529

할당내역 등록일 : 20170101

[ 할당기관 연락처 보기 ]

### WHOIS OpenAPI 사용안내

인터넷주소 검색요청 URL

도메인 검색 결과 예시

<http://whois.kisa.or.kr/openapi/whois.jsp?query=도메인이름&key=발급받은KEY값&answer={xml,json}> XML JSON

IP주소 검색 결과 예시

<http://whois.kisa.or.kr/openapi/whois.jsp?query=IP주소&key=발급받은KEY값&answer={xml,json}> XML JSON

※ 도메인 조회인 경우 국가도메인(.kr, .한국)만 조회가 가능합니다.

※ 해외 IP주소/AS번호 조회인 경우 국가코드만 조회가 가능합니다.

IP주소/AS번호 국가코드 검색요청 URL

<http://whois.kisa.or.kr/openapi/ipascc.jsp?query=AS번호&key=발급받은KEY값&answer={xml,json}>

<http://whois.kisa.or.kr/openapi/ipascc.jsp?query=IP주소&key=발급받은KEY값&answer={xml,json}>

요청 변수

변수명	구분	설명
query	필수	WHOIS OpenAPI Key
key	필수	조회 할 [도메인 이름, IP주소, AS번호]
answer	선택	응답형식 [JSON, XML] (기본값은 XML)

```
1 {
2   "whois": {
3     "query": "210.89.164.90",
4     "queryType": "IPv4",
5     "registry": "KRNIC",
6     "countryCode": "KR",
7     "korean": {
8       "ISP": {
9         "netinfo": {
10          "range": "210.89.160.0 - 210.89.191.255",
```

```
getinfo.py (~/Desktop/whois) - gedit
Open  Save
getinfo.py  config.json  test.py
#!/usr/bin/python
import json
import socket
from urllib2 import urlopen

key = '2020080203155481725353'

with open('config.json') as json_file:
    json_data = json.load(json_file)

for name in json_data['pcaps']:
    ip = socket.gethostbyname(name)
    query = 'http://whois.kisa.or.kr/openapi/whois.jsp?query=' + ip + '&key=' + key + '&answer=json'
    req = urlopen(query).read().decode('utf-8')
    dic = json.loads(req)
    try:
        print( name + ' : ' + dic['whois']['korean']['ISP']['netinfo']['addr'] )
    except:
        print( name + ' : none ' )
```



# 서버의 물리적 위치

- 네이버, 다음, 네이트 하위 서비스들을 물리적 위치에 따라 구분한 결과 얻어낸 주소는 7개
- 서울특별시 용산구
- 서울특별시 중구
- 서울특별시 구로구
- 용인시 수지구
- 성남시 분당구 분당내곡로
- 성남시 분당구 불정로
- 제주시 첨단로

# 서버의 물리적 위치

- 각 위치별로 2,3묶음으로 나누어서 제대로 가리키는지에 대해 확인해본 결과

• 서울특별시 용산구	1/13 (7.69%)
• 서울특별시 중구	0/20 (0%)
• 서울특별시 구로구	2/12 (16.67%)
• 용인시 수지구	2/15 (13.33%)
• 성남시 분당구 분당내곡로	30/52 (57.69%)
• 성남시 분당구 불정로	5/24 (20.83%)
• 제주시 첨단로	0/8 (0%)

# 향후 연구 계획

- 뉴스, 부동산 등과 같이 특정 카테고리로 묶을 수 있는지
- “Improved Website Fingerprinting on Tor” 논문에서 언급된 방식을 통해 분류 진행

Q & A

