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

Agenda

- Members introduction
- Research and Finding
  - Part 1: Layer-7 DoS vulnerability analysis and discovery.
  - Part 2: Powerful Zombie
  - Part 3: Defense Model

Biographies

Tony Miu (aka MT)

- Apart from a being a researcher at VXRL, MT currently holds the post of Deputy SOC Manager at Nexusguard Limited, a global provider of premium end to end web security solutions that specializes in Anti-DDoS & web application security services. 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.

Biographies

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.
- 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.

Biographies

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.

Biographies

Kelvin Wong (aka Captain)

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

Research and Findings

Research Methodology

- We have applied Layer 7 techniques for DoS:
  - HTTP GET and POST methods
  - Malformed HTTP
  - HTTP Pipelining
  - 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.

Techniques Overview : Attack Techniques

Attack Combo #1:

- Manipulate the TCP and HTTP characteristics and vulnerabilities
- Find URL which accepts POST -> Change Content Length to 9999 (i.e. abnormal size > 1500 bytes) bytes -> See whether it keeps the connection alive

(Attack Combo #1 Detailed explanation in Part 2)

Techniques Overview : Post-Attack Techniques

Attack Combo #1:

With POST method allowed, 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
- It is an incomplete HTTP packet, which cannot be detected and it is treated as a data trunk.

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
- Submit a packet to the target with wrong IP checksum and check whether there is any replied packet.

Techniques Overview : Post-Attack Techniques

Goals

- Calculation of resources to bring down the target
- Estimation of the detection
- Guess its DDoS mitigation
- 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 incomplete 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, referral value.
- It means there is NO HARDENING ʎ
- User-Agent value: Mozilla/4.0\r\n is a common botnet used “label”, however, it still could accept

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\nGET /?123
    HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\nGET /?123
    HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\nGET /?123
    HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\nGET /?123
    HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\nGET /?123
    HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\nGET /?123
    HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\nGET /?123
    HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\nGET /?123
    HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\nGET /?123
    HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\r\nGET /?123
    HTTP/1.1\r\nHost: www.xxxxxx.com\r\nUser-Agent: Mozilla/4.0\r\nConnection: keep-alive\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 the packet size with 1460 byte size in PSH/ACK packet
- A packet could be multiplied 7 times or more
- For pipelining, for example, HTTP packet is not properly ended without \r\n\r\n, which may bypass the detection and filter, as it is not deemed as a HTTP packet.

Techniques: Attack Techniques

Attack Combo #2:

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

Techniques: Attack Techniques

Attack Combo #2:

- Session – Force to get a new session and connection without cache. It could “guarantee” bypass loadbalancer and Proxy.
- It is hard to remove it.
- If trying to drop the URL with “?”, it causes dropping the normal request:
  - For example, http://www.abc.com/submitform.asp?234732893845DS4fjs9....
  - Cache:no cache and expiry date is 1994

Techniques: Attack Techniques

- Attack Combo #2

```
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
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
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
GET /download/doc.pdf?121234234fgsefasdfl44 HTTP/1.1\r\n
Host: www.xxxxyyyyzzzz.com\r\n
User-Agent: Mozilla/4.0\r\n
Connection: keep-alive\r\n
```

Techniques: Attack Techniques

- Attack Combo #2

```
GET /download/doc.pdf?121234234fgsefasdfl55 HTTP/1.1\r\n
Host: www.xxxxyyyyzzzz.com\r\n
User-Agent: Mozilla/4.0\r\n
Connection: keep-alive\r\n
GET /download/doc.pdf?121234234fgsefasdfl66 HTTP/1.1\r\n
Host: www.xxxxyyyyzzzz.com\r\n
User-Agent: Mozilla/4.0\r\n
Connection: keep-alive\r\n
GET /download/doc.pdf?121234234fgsefasdfl77 HTTP/1.1\r\n
Host: www.xxxxyyyyzzzz.com\r\n
User-Agent: Mozilla/4.0\r\n
Connection: keep-alive\r\n
GET /download/doc.pdf?121234234fgsefasdfl88 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 (40)**
- Europe (20)
- Asia Pacific (20)

Case Studies:

It will be discussed on stage

Agenda

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

Before taking Appetizer, let us do the demo

Let us give three demos:

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.

1.Attack target server and stuck TCP state TIME_WAIT

2.Attack target server and stuck TCP state FIN_WAIT1

3.Attack target server and stuck TCP state Established

Attack Goal

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

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.

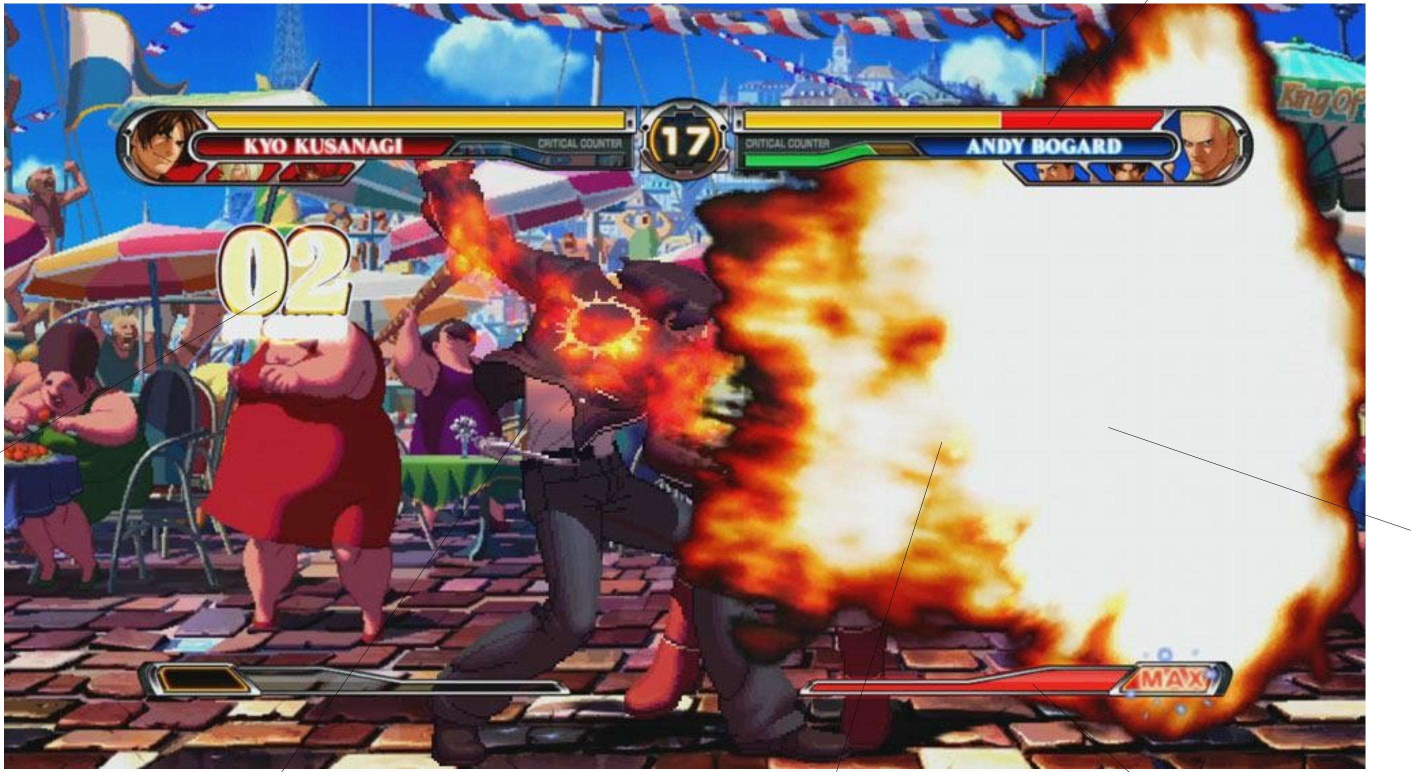
- Use the HTTP as an example to control the TCP state.

- 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 30…etc.

Logical Diagram



Super combo Period =TCP State

Health Point = Server resource

Hits = TCP Connection

Andy in fire = Web server

Kyo = Attack server

Super combo = HTTP Request

High CPU

Keep Super Combo to Andy

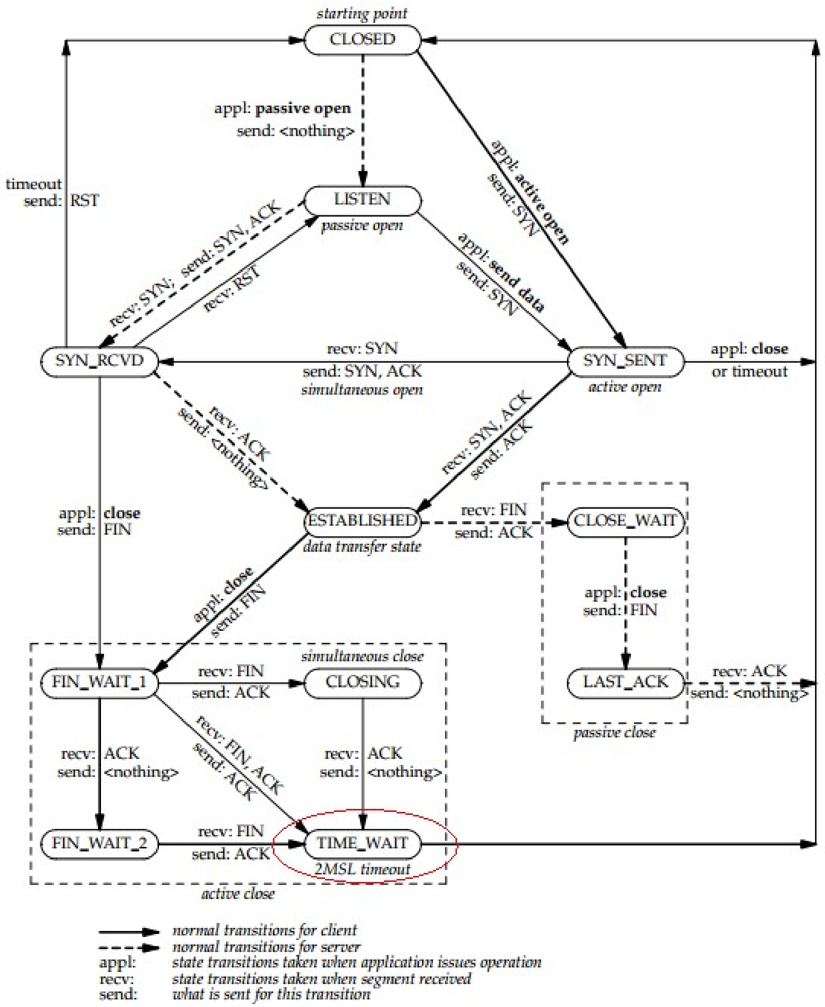
- We wish to extent the super combo period!!!

- We will discuss the 3 different TCP states.

- Targeted TCP state:
  - Demo 1. TCP TIME_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 TIME_WAIT



TCP state transition diagram.

Demo 1. TCP STAT TIME_WAIT

From RFC:

“When a connection is closed actively, it MUST linger in TIME-WAIT state for a time 2xMSL (**Maximum Segment Lifetime**). However, it MAY accept a new SYN from the remote TCP to reopen the connection directly from TIME-WAIT state, if it:
(1)assigns its initial sequence number for the new connection to be larger than the largest sequence number it used on the previous connection incarnation, and

(2) returns to TIME-WAIT state if the SYN turns out to be an old duplicate”

Demo 1. TCP STAT TIME_WAIT

- Demo 1 is simulating the most common DDoS attack.

- RFC: “Server is waiting for a connection termination request from the local user.” Depends on OS, time out around 60s.

- Web server are only with high CPU usage and in unstable status

Demo 1. TCP STAT TIME_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)

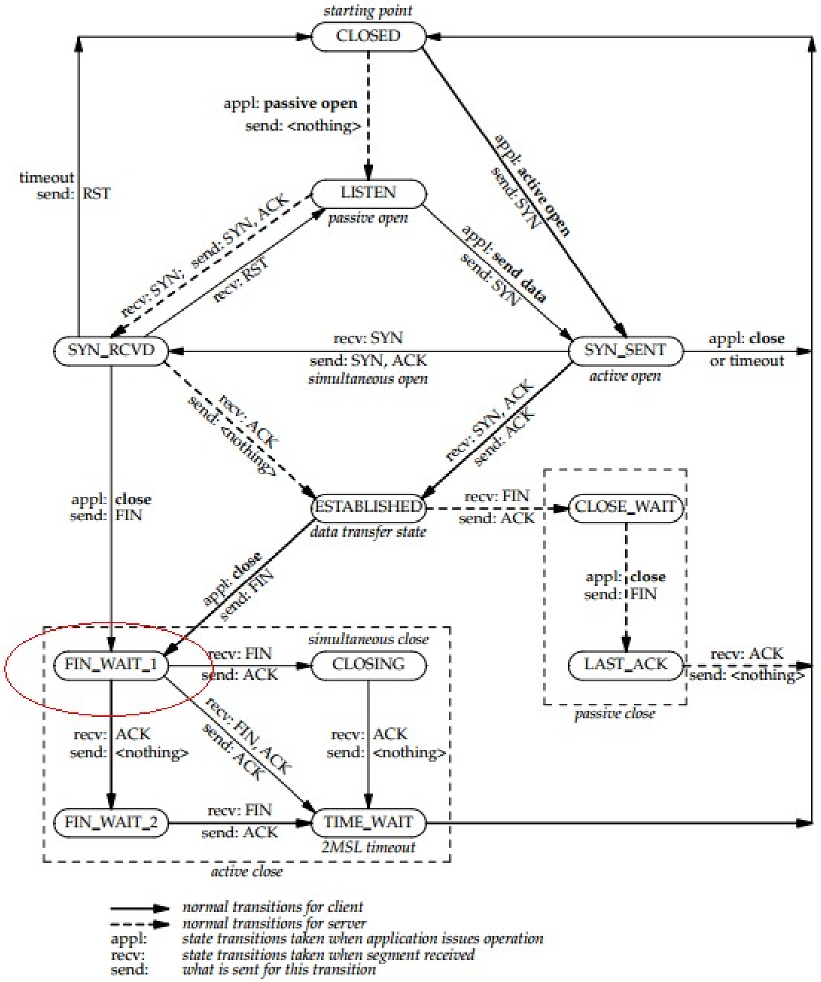
Demo 1 – The Key for goal

- Check the TCP last ACK timeout value
- Wait for RST – Initiated by device like IDS, IPS, etc.

Demo1 – The Key for Goal

TBC

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

- Most of network security devices can set the timeout value, but easy to crush the web application….

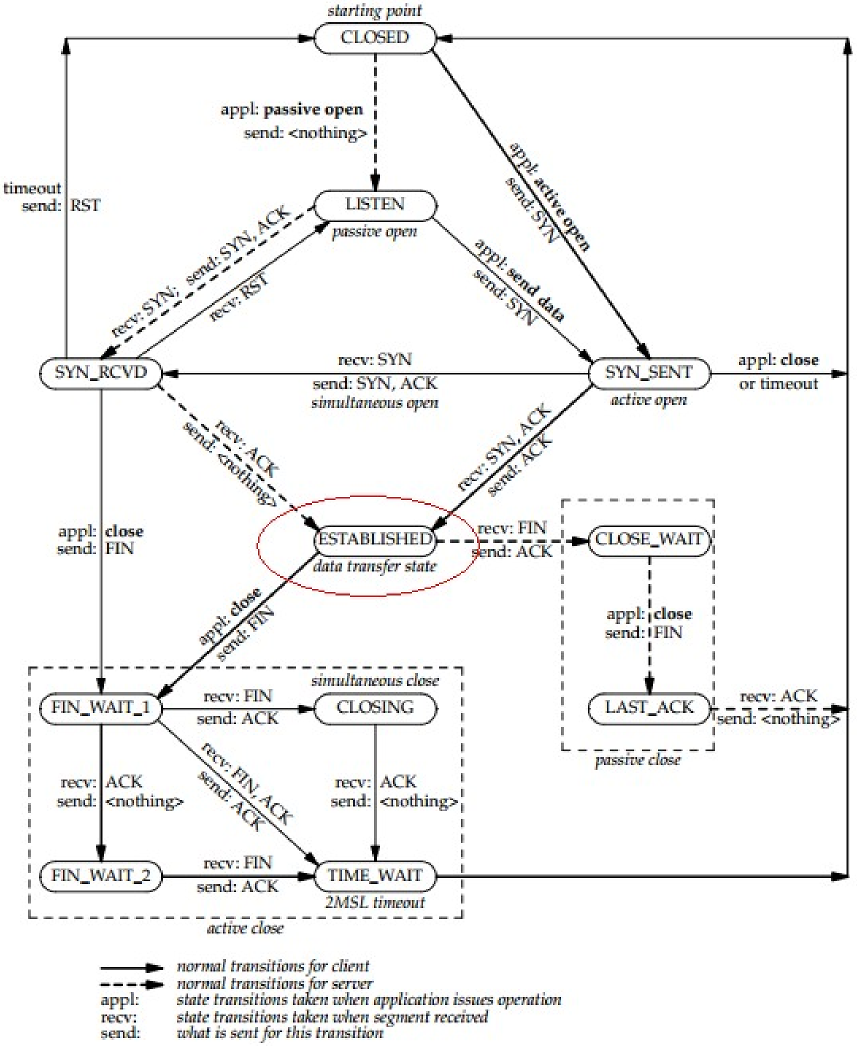
Demo 2 - The Key for goal

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

Demo 2 - The Key for goal

TBC

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.

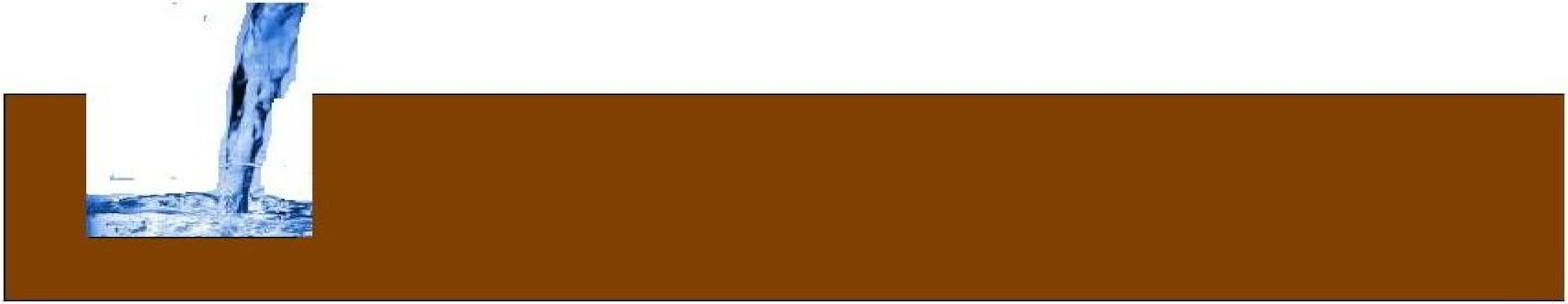
    - It is an incomplete HTTP request

- Timeout depends on application and server, may be 30s, 5mins, 10mins or more.

- Incomplete HTTP request can bypass the Network Security devices.

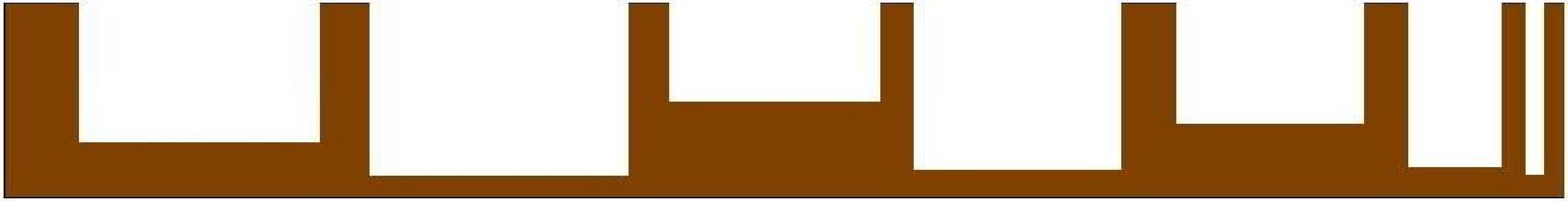
Demo 3. Vs Slowloris

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 3 - The Key for goal

- Check the TCP establishment timeout value
- Check the TCP first PSH/ACK timeout value
- Check the TCP continuous ACK timeout value
- Wait for FIN/ACK – Initiated by target’s server
- Submit a packet to the target with wrong IP checksum and check whether there is any replied packet.
- It is an incomplete HTTP packet, which cannot be detected and it is treated as a data trunk.

Demo 3 - The Key for goal

TBC

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 and test the boundary values of rate limit of any security and detection devices.

- Confuse the security detection device with “look-like” real HTTP request.

PoC of Case study

- Slowloris is a good example for Demo 3
- Demo 1-3 are PoC for the analysis result and impact in Part 1.

We have a great weapon and need a best solider

7/11/12

Before taking empower Zombie ...

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 Zombie “without” established any connection and intensive use of memory and CPU

Demo

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

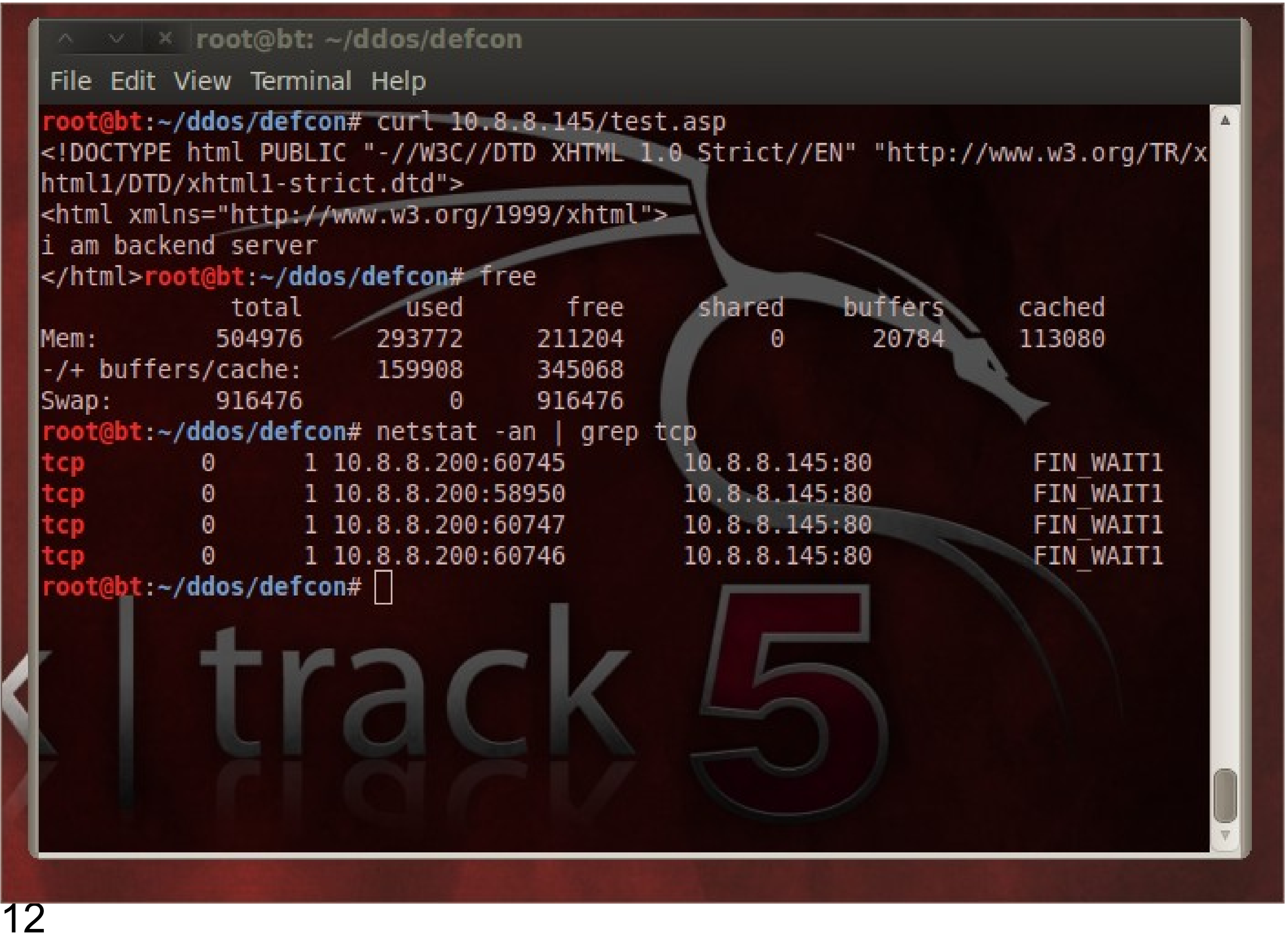
Demo time

Our Zombie’s Design

Design Overview

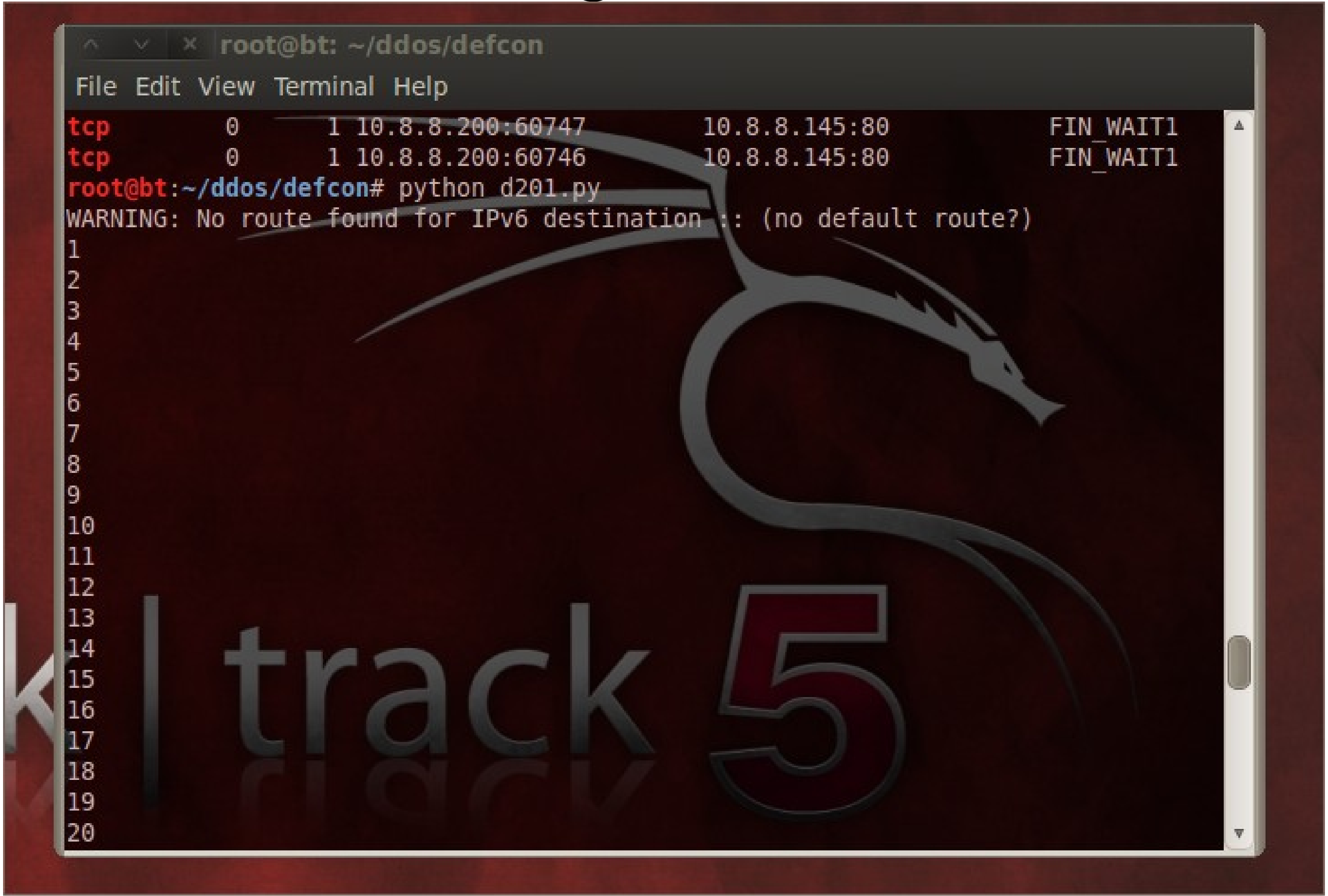
- Current DDoS mitigation method violates RFC standard.
- Our Zombie also adopt DDoS mitigation methods into the design
- Our Zombie’s protocol design “looks like” fulfilling a RFC standard. We simply adopt the DDoS mitigation method and design into our Zombie.
- This solider design is for our Demo 3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

1. Show Attack Server's Resources Status



```
root@bt:~/ddos/defcon# curl 10.8.8.145/test.asp
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/x
html1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
i am backend server
</html>root@bt:~/ddos/defcon# free
             total       used       free     shared    buffers     cached
Mem:        504976     293772     211204          0      20784     113080
-/+ buffers/cache:     159908     345068
Swap:       916476          0     916476
root@bt:~/ddos/defcon# netstat -an | grep tcp
tcp        0      1 10.8.8.200:60745        10.8.8.145:80           FIN_WAIT1
tcp        0      1 10.8.8.200:58950        10.8.8.145:80           FIN_WAIT1
tcp        0      1 10.8.8.200:60747        10.8.8.145:80           FIN_WAIT1
tcp        0      1 10.8.8.200:60746        10.8.8.145:80           FIN_WAIT1
root@bt:~/ddos/defcon# 
```

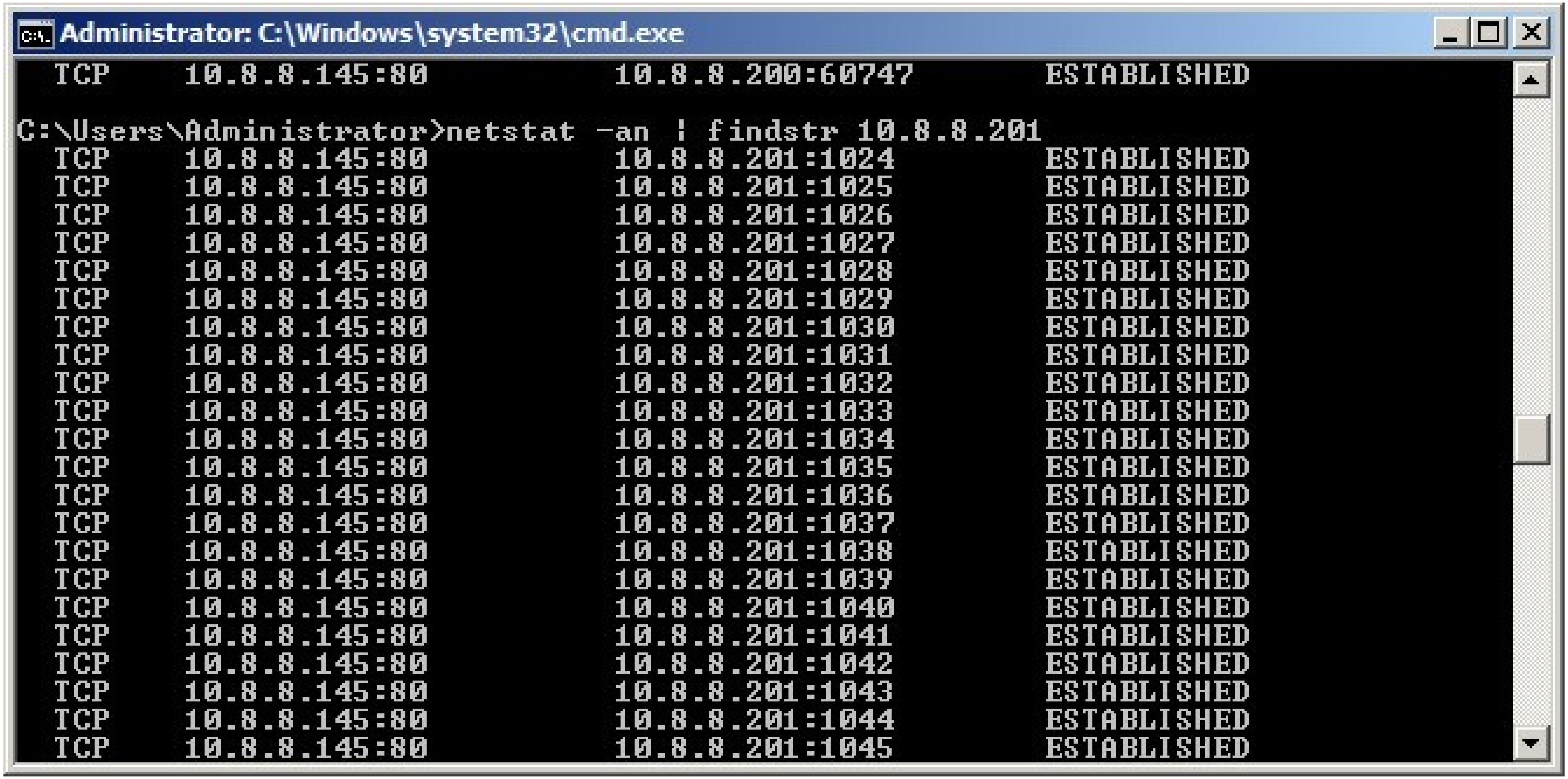
2. Generating an attack



```
tcp        0      1 10.8.8.200:60747        10.8.8.145:80           FIN_WAIT1
tcp        0      1 10.8.8.200:60746        10.8.8.145:80           FIN_WAIT1
root@bt:~/ddos/defcon# python d201.py
WARNING: No route found for IPv6 destination :: (no default route?)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

7/11/12

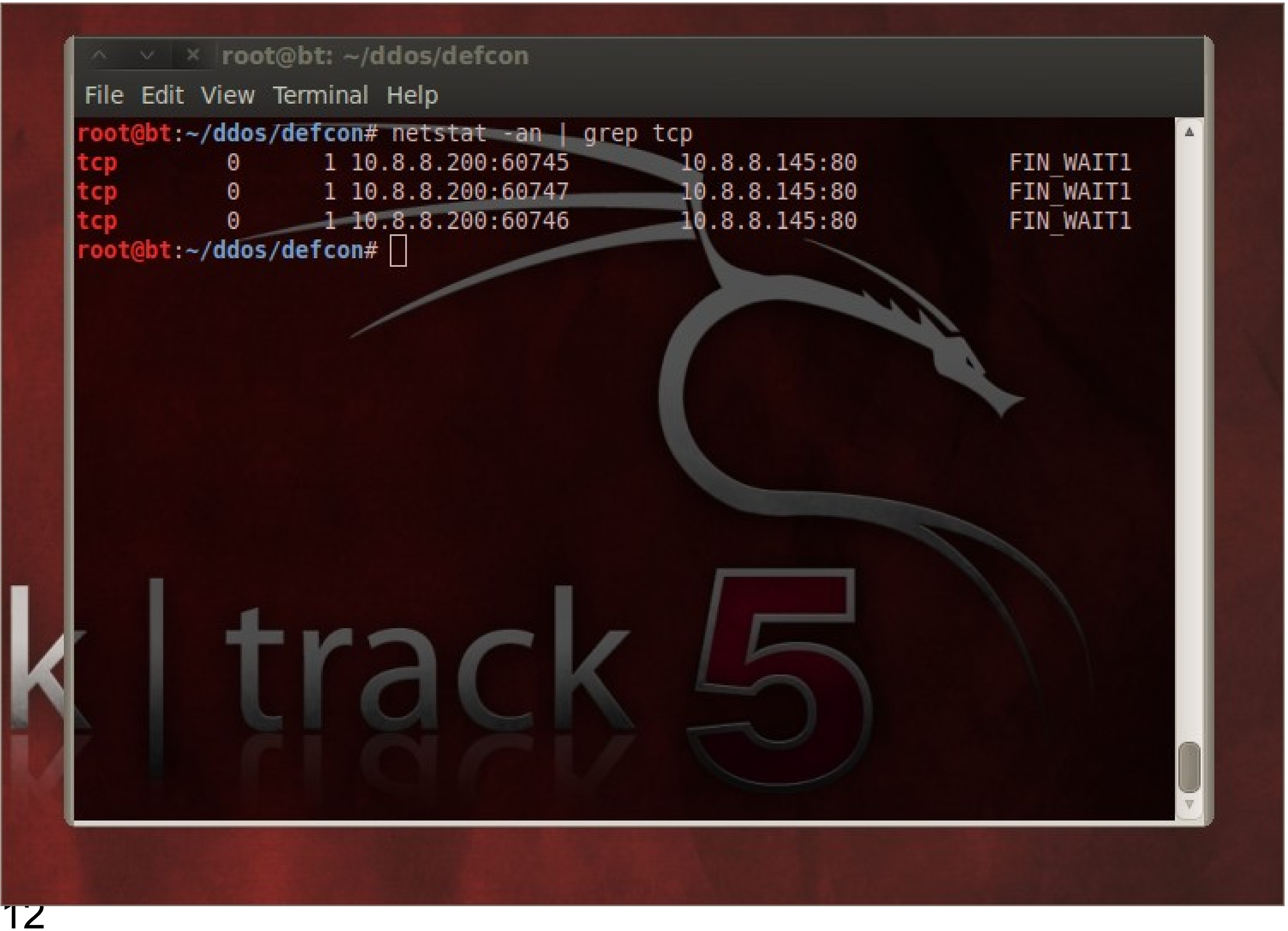
3. Show the target's server status



```
  TCP    10.8.8.145:80          10.8.8.200:60747       ESTABLISHED

C:\Users\Administrator>netstat -an | findstr 10.8.8.201
  TCP    10.8.8.145:80          10.8.8.201:1024        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1025        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1026        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1027        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1028        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1029        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1030        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1031        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1032        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1033        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1034        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1035        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1036        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1037        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1038        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1039        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1040        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1041        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1042        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1043        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1044        ESTABLISHED
  TCP    10.8.8.145:80          10.8.8.201:1045        ESTABLISHED
```

4. Show attack server status AFTER attack



```
root@bt:~/ddos/defcon# netstat -an | grep tcp
tcp        0      1 10.8.8.200:60745        10.8.8.145:80           FIN_WAIT1
tcp        0      1 10.8.8.200:60747        10.8.8.145:80           FIN_WAIT1
tcp        0      1 10.8.8.200:60746        10.8.8.145:80           FIN_WAIT1
root@bt:~/ddos/defcon# 
```

Zombie Features

- Our designed zombie 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 zombie

It could have many different types of solider.

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

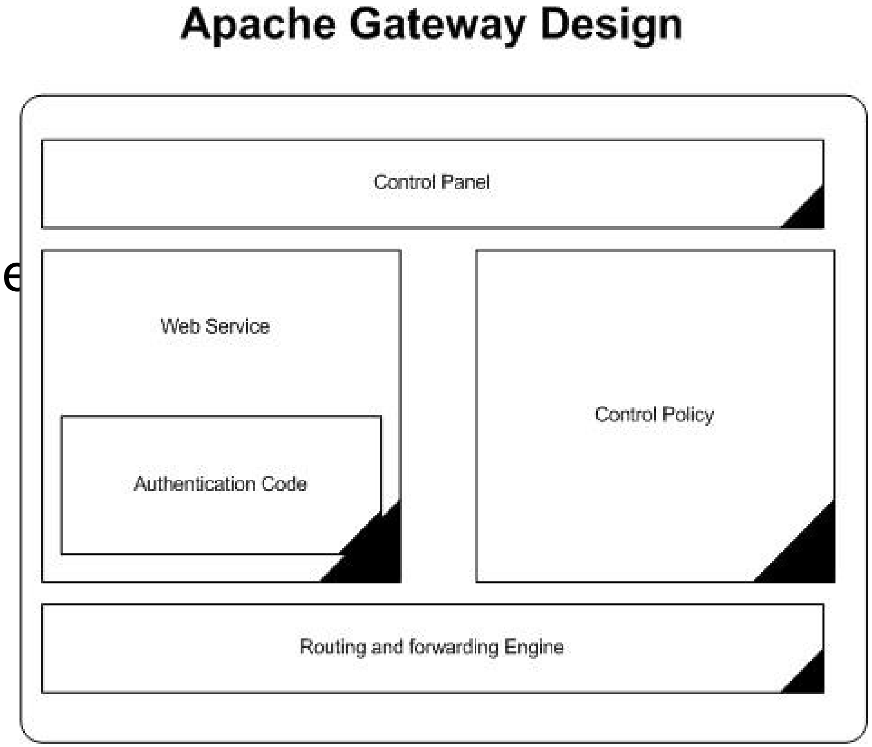
Part 3: Defense Model

Existing DDoS mitigation countermeasure

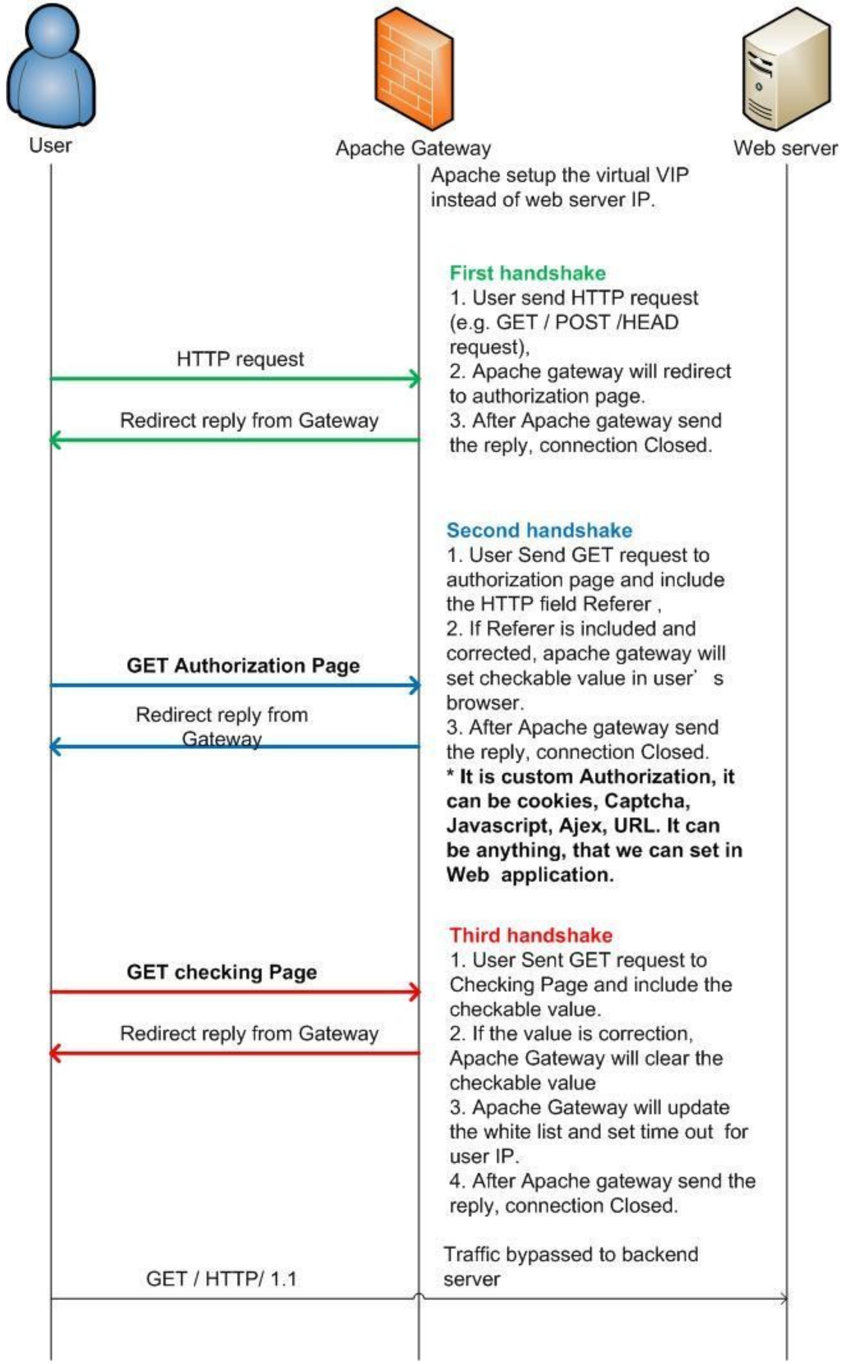
- TCP Layer Authentication
- HTTP Layer Authentication (redirect, URL)
- HTTP Layer interrupt (Captcha)
- Signature base application detection
- Rate limit

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



Layer 7 3-Way handshakes Concept



Apache setup the virtual VIP instead of web server IP.

First handshake

1. User send HTTP request (e.g. GET / POST /HEAD request),
2. Apache gateway will redirect to authorization page.
3. After Apache gateway send the reply, connection Closed.

Second handshake

1. User Send GET request to authorization page and include the HTTP field Referer ,
2. If Referer is included and corrected, apache gateway will set checkable value in user’ s browser.
3. After Apache gateway send the reply, connection Closed.

*** It is custom Authorization, it can be cookies, Captcha, Javascript, Ajex, URL. It can be anything, that we can set in Web application.**

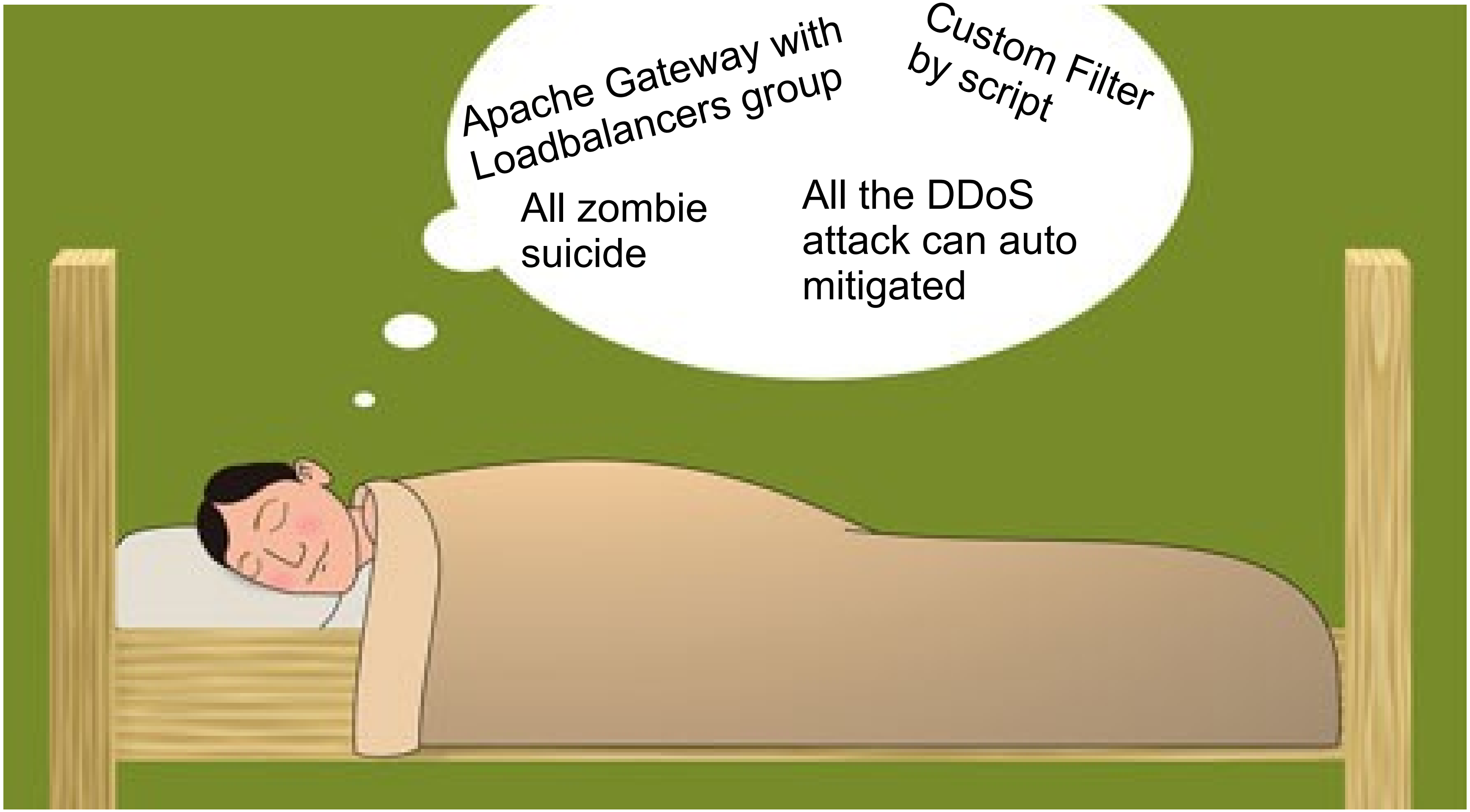
Third handshake

1. User Sent GET request to Checking Page and include the checkable value.
2. If the value is correction, Apache Gateway will clear the checkable value
3. Apache Gateway will update the white list and set time out for user IP.
4. After Apache gateway send the reply, connection Closed.

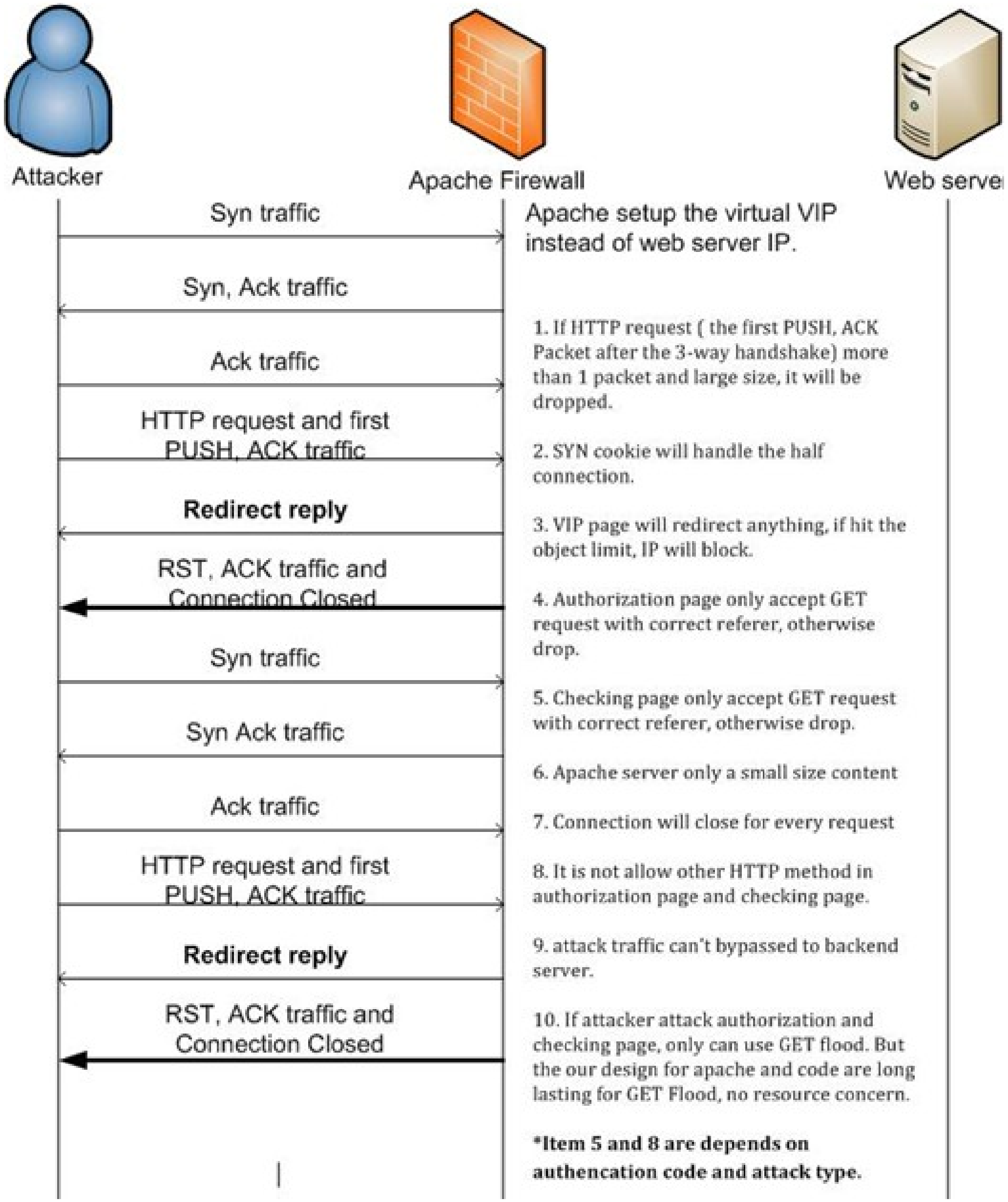
Traffic bypassed to backend server

7/11/12

I have a Dream~


Apache Gateway with Loadbalancers group
Custom Filter by script
All zombie suicide
All the DDoS attack can auto mitigated

Apache Firewall against to GET / POST / Connection Flood



Attacker

Apache Firewall

Web serve

Syn traffic

Syn, Ack traffic

Ack traffic

HTTP request and first PUSH, ACK traffic

Redirect reply

RST, ACK traffic and Connection Closed

Syn traffic

Syn Ack traffic

Ack traffic

HTTP request and first PUSH, ACK traffic

Redirect reply

RST, ACK traffic and Connection Closed

Apache setup the virtual VIP instead of web server IP.

1. If HTTP request (the first PUSH, ACK Packet after the 3-way handshake) more than 1 packet and large size, it will be dropped.

2. SYN cookie will handle the half connection.

3. VIP page will redirect anything, if hit the object limit, IP will block.

4. Authorization page only accept GET request with correct referer, otherwise drop.

5. Checking page only accept GET request with correct referer, otherwise drop.

6. Apache server only a small size content

7. Connection will close for every request

8. It is not allow other HTTP method in authorization page and checking page.

9. attack traffic can't bypassed to backend server.

10. If attacker attack authorization and checking page, only can use GET flood. But the our design for apache and code are long lasting for GET Flood, no resource concern.

***Item 5 and 8 are depends on authencation code and attack type.**

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

Tony: mt[at]vxrl[dot]org

Alan: avenir[at]vxrl[dot]org

Kelvin: captain[at]vxrl[dot]org

Anthony: darkfloyd[at]vxrl[dot]org