SDN-Assisted Network-Based Mitigation of Slow HTTP Attacks

Thomas Lukaseder, Lisa Maile, Frank Kargl | Institute of Distributed Systems
12. Oct 2017
Slow HTTP attacks

- Attack Goal: Reach maximum amount of server connections
- No malformed requests
- Low data rate and few packets
- Highly efficient, one attacker is sufficient
Slowloris

GET / HTTP/1.1 CRLF
Host: www.xy.de CRLF
Connection: keep-alive CRLF
User-Agent: Mozilla/5.0 CRLF
Referer: http://www.xy.com/x/ CRLF

...
Overview

- Mitigation: reduce and limit timeouts
  - also blocks slow normal clients

- Bots: special behavior like constant packet rate
Our Solution

- DDoS mitigation framework
- No action from the admin required
- Mitigate attacks without support of the server operators
- Based on SDN
The Framework

Phase 1 – Attack Detection:

Controller / Observer → OpenFlow Switch

Webserver

available?

CAPTCHA Server
The Framework

Phase 2 – Attacker Identification:

Controller / Observer
Host1: 10
Host2: 50

OpenFlow Switch

Webserver

CAPTCHA Server

mirrored
The Framework

Phase 3 – Attack Mitigation:

Which are the attacking clients?

Institute of Distributed Systems, Ulm University
Method 1: Max. Duration

- SYN
- ACK
- Data Part

Server overloaded

t_{duration} > t_{timeout}?

really high suspiciousness score

 Eventually: client blocked
Method 1: Max. Duration

- **Pro**
  - Clear identification

- **Contra**
  - False Positives: normal but slow clients blocked
  - Long identification phase
Method 2a: Even Packet Rate

SYN  ACK Data Part

Server overloaded

\[ t_{\text{delta1}} = t_{\text{delta2}} \]

High suspiciousness score
Method 2a: Even Packet Rate

**Pro**
- Few connection details to remember

**Contra**
- False Positives: happens sporadically to normal clients
- Forgets older messages
Method 2b: Even Packet Rate

Server overloaded

\[
\text{packerate} = \frac{\#\text{packets}}{t_{\text{duration}}}
\]

\[
\text{packerate}^{(1)} = \text{packerate}^{(2)} = \text{packerate}^{(3)}
\]

high suspiciousness score
Method 2b: Even Packet Rate

- **Pro**
  - Few connection details to remember

- **Contra**
  - TCP handshake packets are sent fast
    - Packet rate higher
    - Packet rate only decays slowly, therefore long identification phase
Method 2c: Even Packet Rate

Server overloaded

SYN ACK DataPart

if SYN or ACK nothing
else
  if #packet = 0
    #packet ++
    t_{duration} = 0
  else
    #packet ++
    t_{duration} += t_{delta}(i)

packetrate^{(1)} = \frac{#packets}{t_{duration}}

high suspiciousness score

packetrate^{(2)} = packetrate^{(3)}?

packetrate^{(3)}
Method 2c: Even Packet Rate

- **Pro**
  - Clear identification because of very low packet rate

- **Contra**
  - Large management effort
Method 3: Incomplete Packets

Server overloaded

SYN  ACK  Data Part

if HTTP-Method = GET
    if not packet contains “CRLF CRLF”
        #packetsIncomplete ++
Method 3: Incomplete Packets

- **Pro**
  - Clear identification

- **Contra**
  - Large effort for identification of incomplete packets
Method 4: Connections

Client A

SYN ACK DataPart

DataPart

Server overloaded

high suspiciousness

ss score

#clientAConnection >> #connection threshold?

#clientAConnection ++

#clientAConnection ++

#clientAConnection ++
Method 4: Connections

- Pro
  - Little management effort

- Contra
  - Only for non-distributed DoS Attacks
Summary

Completely automate detection, identification and mitigation of slow HTTP attacks.

Possibility to identify best identification technique.

Framework offers decent support against most DDoS attacks.
Thank you for your attention!

For details about the framework, please refer to: 

Thanks to pixabay.com