DNS based DDoS mitigation solution

Overlapping Issue & Implementation Method

Sukhun Yang

Seoul National University

August 28, 2023

Contents

1. Overlapping Issue
   - Diagram Case 1: Accept
   - Diagram Case 2: Abort
   - Overlapping Issue
2. Implementation Method

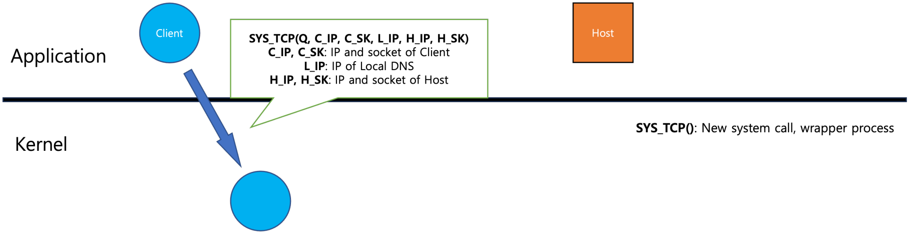
Table of Contents

1. Overlapping Issue
   - Diagram Case 1: Accept
   - Diagram Case 2: Abort
   - Overlapping Issue
2. Implementation Method

Puzzle policy

Assumed that the puzzle policy already exists.

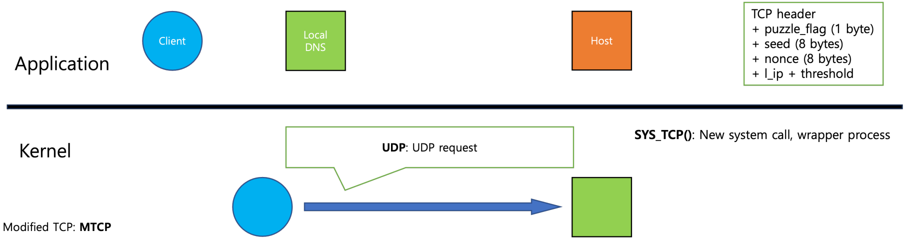
Case 1: Accept



SYS_TCP(Q, C_IP, C_SK, L_IP, H_IP, H_SK)
C_IP, C_SK: IP and socket of Client
L_IP: IP of Local DNS
H_IP, H_SK: IP and socket of Host

SYS_TCP(): New system call, wrapper process

Case 1: Accept



Application

Client

Local DNS

Host

TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

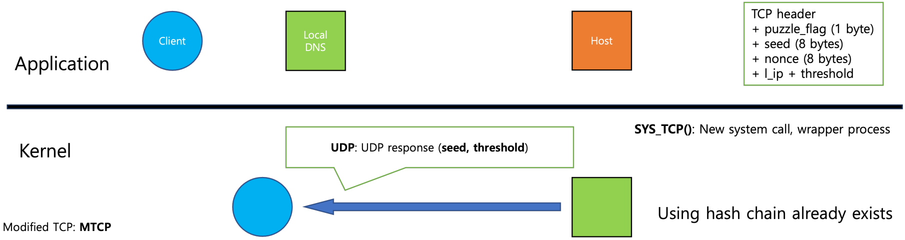
Kernel

UDP: UDP request

SYS_TCP(): New system call, wrapper process

Modified TCP: **MTCP**

Case 1: Accept



Application

Client

Local DNS

Host

TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

Kernel

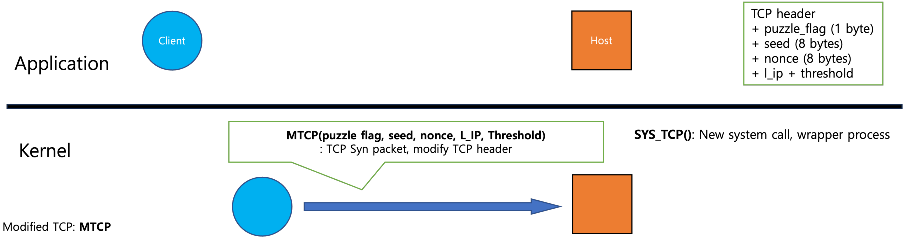
SYS_TCP(): New system call, wrapper process

UDP: UDP response (**seed, threshold**)

Modified TCP: **MTCP**

Using hash chain already exists

Case 1: Accept



Application

TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

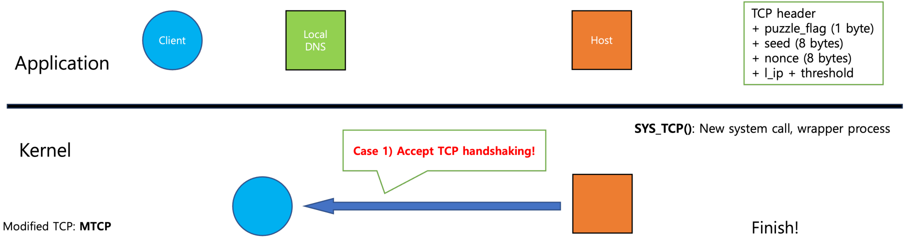
Kernel

MTCP(puzzle flag, seed, nonce, L_IP, Threshold)
: TCP Syn packet, modify TCP header

SYS_TCP(): New system call, wrapper process

Modified TCP: **MTCP**

Case 1: Accept

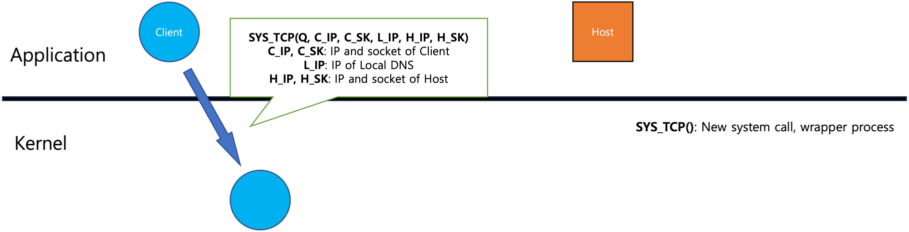


TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

SYS_TCP(): New system call, wrapper process

Case 1) Accept TCP handshaking!

Case 2: Abort

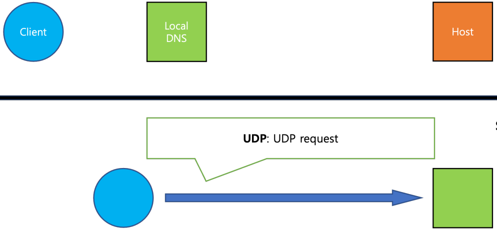


SYS_TCP(Q, C_IP, C_SK, L_IP, H_IP, H_SK)
C_IP, C_SK: IP and socket of Client
L_IP: IP of Local DNS
H_IP, H_SK: IP and socket of Host

SYS_TCP(): New system call, wrapper process

Case 2: Abort

Application



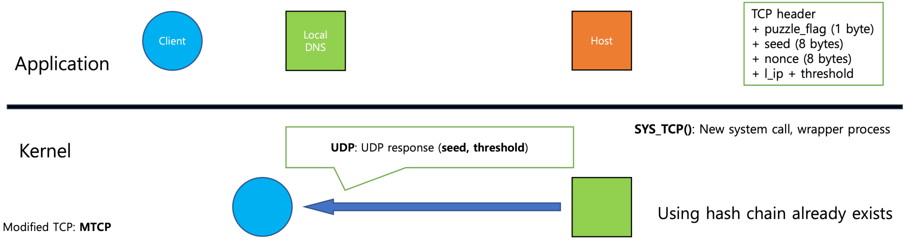
TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

Kernel

SYS_TCP(): New system call, wrapper process

Modified TCP: **MTCP**

Case 2: Abort



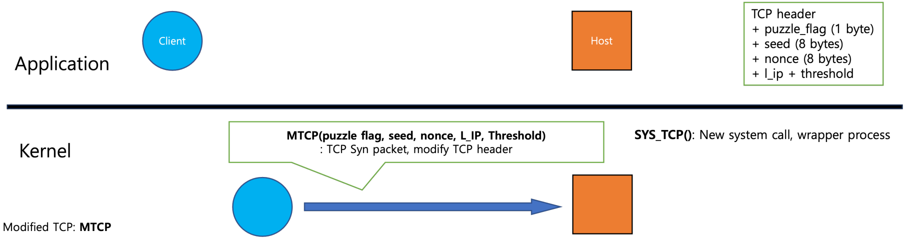
TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

SYS_TCP(): New system call, wrapper process

UDP: UDP response (**seed, threshold**)

Using hash chain already exists

Case 2: Abort



TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

Application

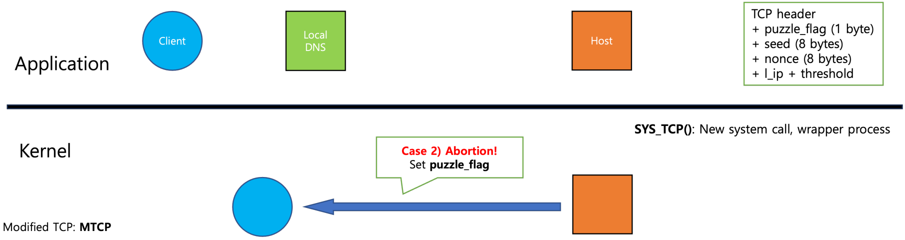
Kernel

MTCP(puzzle flag, seed, nonce, L_IP, Threshold)
: TCP Syn packet, modify TCP header

SYS_TCP(): New system call, wrapper process

Modified TCP: **MTCP**

Case 2: Abort

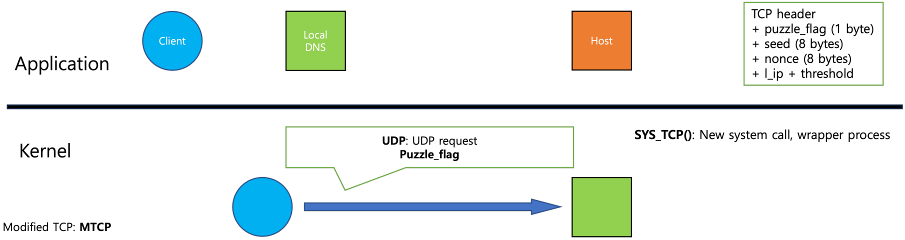


TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

SYS_TCP(): New system call, wrapper process

Case 2) Abortion!
Set **puzzle_flag**

Case 2: Abort

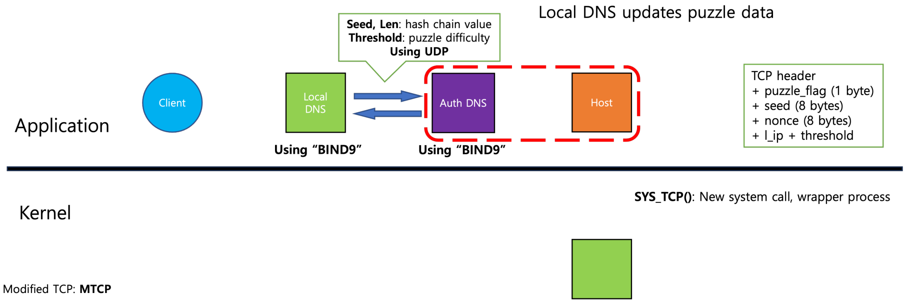


TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

UDP: UDP request
Puzzle_flag

SYS_TCP(): New system call, wrapper process

Case 2: Abort



Local DNS updates puzzle data

Seed, Len: hash chain value
Threshold: puzzle difficulty
Using UDP

Application

Client

Local DNS

Auth DNS

Host

Using "BIND9"

Using "BIND9"

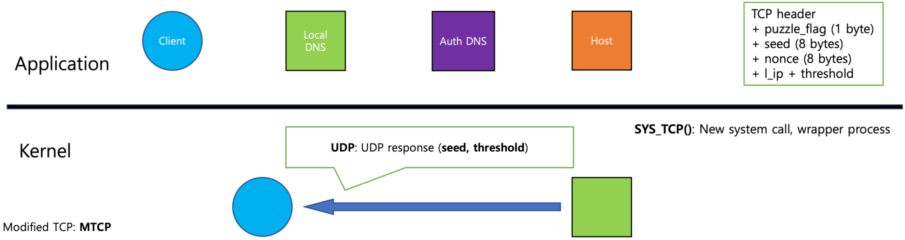
TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

Kernel

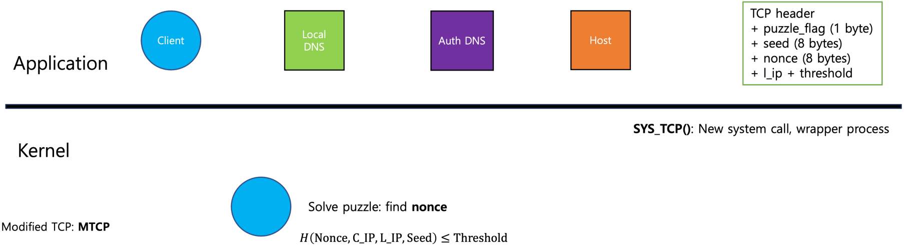
SYS_TCP(): New system call, wrapper process

Modified TCP: **MTCP**

Case 2: Abort


Application
Client
Local DNS
Auth DNS
Host
TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold
Kernel
SYS_TCP(): New system call, wrapper process
UDP: UDP response (seed, threshold)
Modified TCP: MTCP

Case 2: Abort



Application

Client

Local DNS

Auth DNS

Host

TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

SYS_TCP(): New system call, wrapper process

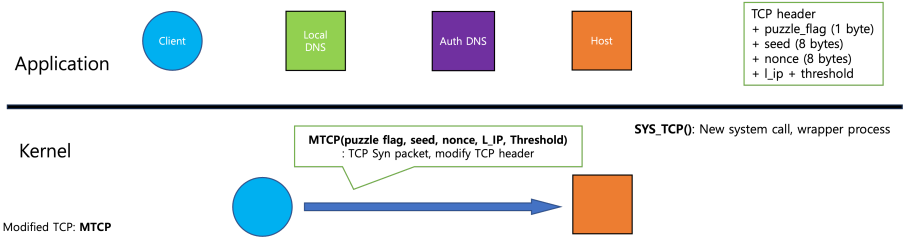
Kernel

Modified TCP: **MTCP**

Solve puzzle: find **nonce**

$H(\text{Nonce}, \text{C\_IP}, \text{L\_IP}, \text{Seed}) \leq \text{Threshold}$

Case 2: Abort



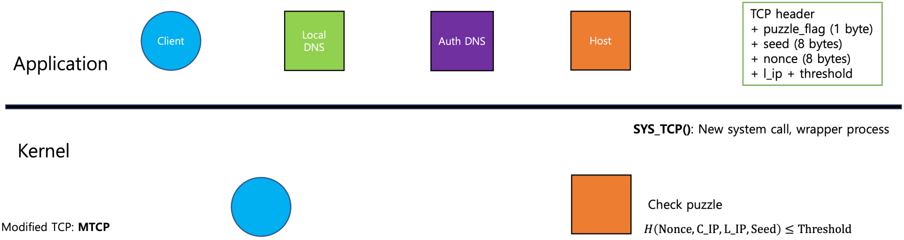
TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

SYS_TCP(): New system call, wrapper process

MTCP(puzzle flag, seed, nonce, L_IP, Threshold)
: TCP Syn packet, modify TCP header

Modified TCP: **MTCP**

Case 2: Abort



TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

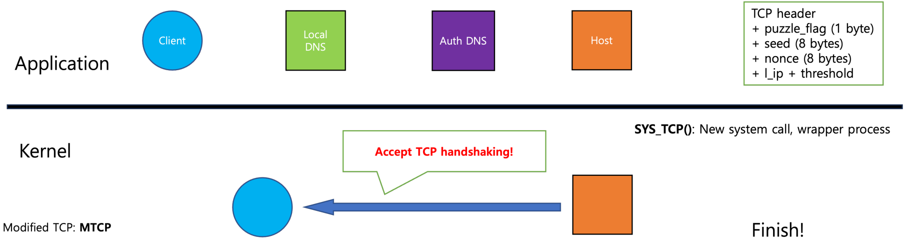
SYS_TCP(): New system call, wrapper process

Modified TCP: **MTCP**

Check puzzle

$H(\text{Nonce}, \text{C\_IP}, \text{L\_IP}, \text{Seed}) \leq \text{Threshold}$

Case 2: Abort



TCP header
+ puzzle_flag (1 byte)
+ seed (8 bytes)
+ nonce (8 bytes)
+ l_ip + threshold

Questions

Timing of update puzzle information

- Client → Local DNS
- Local DNS → Authoritative Name Server

Issue

- Case 1: Client get exist puzzle from Local DNS → Accept TCP 3-way handshaking
- Case 2: Client get exist puzzle from Local DNS → Abort TCP 3-way handshaking → Client request puzzle to Local DNS → Local DNS get puzzle from Auth. NS → Local DNS send puzzle to Client → Accept TCP 3-way handshaking

1. If there is an overlapping period, then corresponds to Case 1 because handshaking accepts
2. Local DNS updates puzzle information only when Aborted (Case 2)
3. If there is an overlapping period, the puzzle is not updated in the current diagram

Table of Contents

1. Overlapping Issue
   - Diagram Case 1: Accept
   - Diagram Case 2: Abort
   - Overlapping Issue
2. Implementation Method

Implementation Method

Implementation Method

- Modify linux kernel → Now working!
- Application TCP 3-way simulator

Progress

Puzzle information keeps missing during TCP 3-way handshaking.
TCP header exists as an internal data structure of IP header, but modification of TCP header is not reflected in IP header.
New modification to IP header is required...

Source Code

https://github.com/Sagit25/DNS-based-DDoS-mitigation/tree/ysh-kernel

Implementation Method

Implementation Method

- Modify linux kernel
- Application TCP 3-way simulator → ...!

Issues

- Existence of overlapping period
- Modification of IP header
- DNS mapping at host kernel

In fact, I don't even know what other people are doing.

Thinking

Implementing simulator (about 3 days) ⇒
Basic test and set parameters ⇒
Modifying linux kernel & Start writing ⇒
Measure accurate data ⇒
Finish!

Simulator

Now set basic puzzle method!
(ref. https://github.com/zakgilbert/TCP_handshake_simulator)

Source Code

https://github.com/Sagit25/DNS-based-DDoS-mitigation/tree/ysh-simulator