ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

CSSE 490
Network Security

Day 17: TCP State Exhaustion

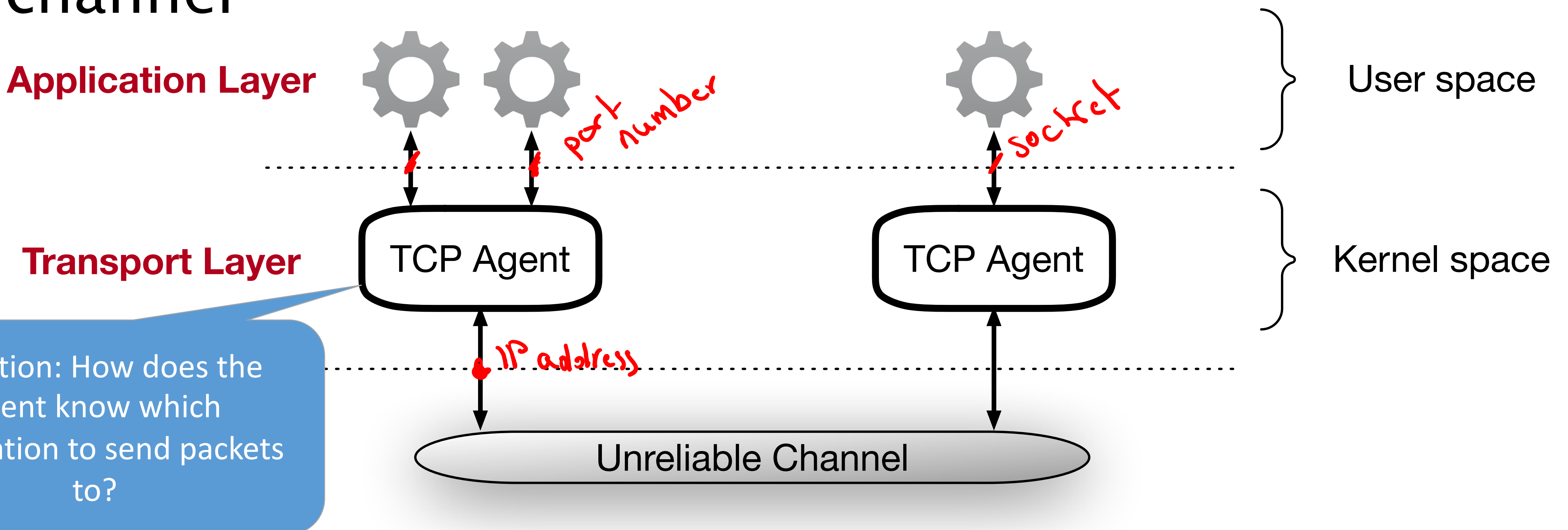
Transport Layer Protocols: TCP vs UDP

	TCP	UDP
Connection	Connection based	Connectionless
Packet Boundary	Stream based	Preserving packet boundaries
Reliability	✓	✗
Ordering	✓	✗
Speed	✗	✓
Broadcast	✗	✓

Transmission Control Protocol

Goal

Provide reliable communication over an unreliable channel



Application Layer

Transport Layer

TCP Agent

TCP Agent

Unreliable Channel

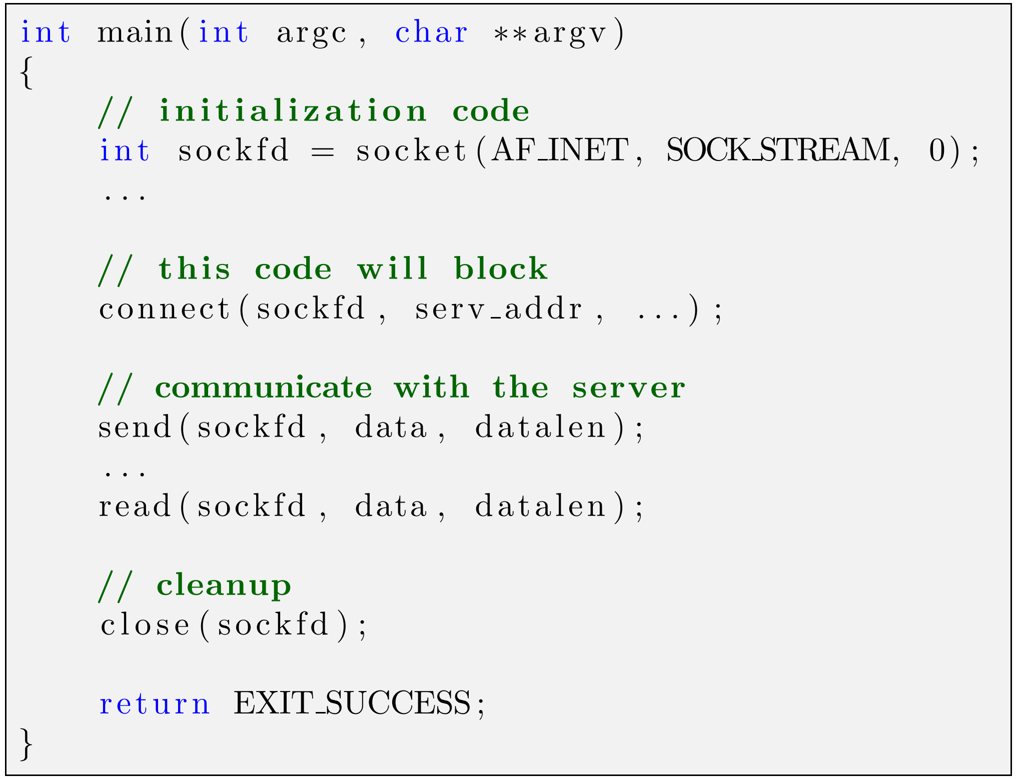
User space

Kernel space

Question: How does the agent know which application to send packets to?

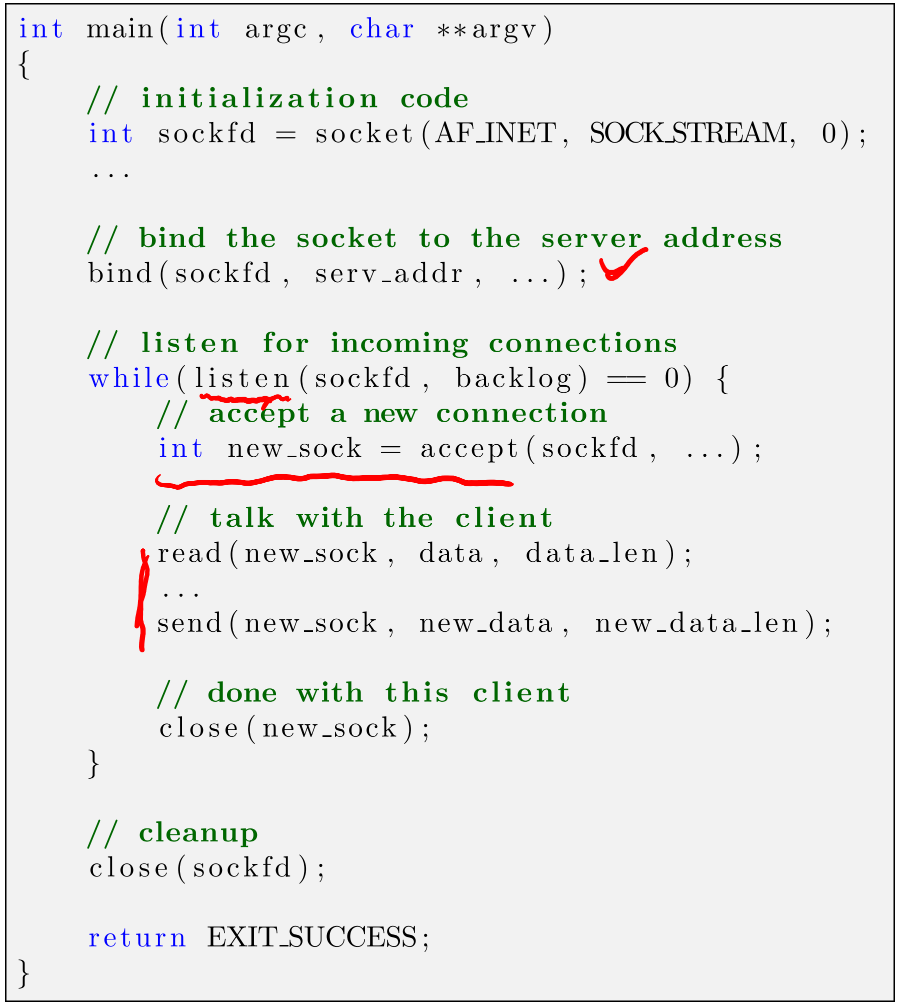
TCP Initilization

Client



```
int main(int argc, char **argv)
{
    // initialization code
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    ...

    // this code will block
    connect(sockfd, serv_addr, ...);

    // communicate with the server
    send(sockfd, data, datalen);
    ...
    read(sockfd, data, datalen);

    // cleanup
    close(sockfd);

    return EXIT_SUCCESS;
}
```

Server



```
int main(int argc, char **argv)
{
    // initialization code
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    ...

    // bind the socket to the server address
    bind(sockfd, serv_addr, ...);

    // listen for incoming connections
    while(listen(sockfd, backlog) == 0) {
        // accept a new connection
        int new_sock = accept(sockfd, ...);

        // talk with the client
        read(new_sock, data, data_len);
        ...
        send(new_sock, new_data, new_data_len);

        // done with this client
        close(new_sock);
    }

    // cleanup
    close(sockfd);

    return EXIT_SUCCESS;
}
```

TCP Initialization

Client

```
int main(int argc, char **argv)
{
    // initialization code
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    ...
...

connect(sockfd, serv_addr, ...);

...
    ...
    read(sockfd, data, datalen);

    // cleanup
    close(sockfd);

    return EXIT_SUCCESS;
}
```

Server

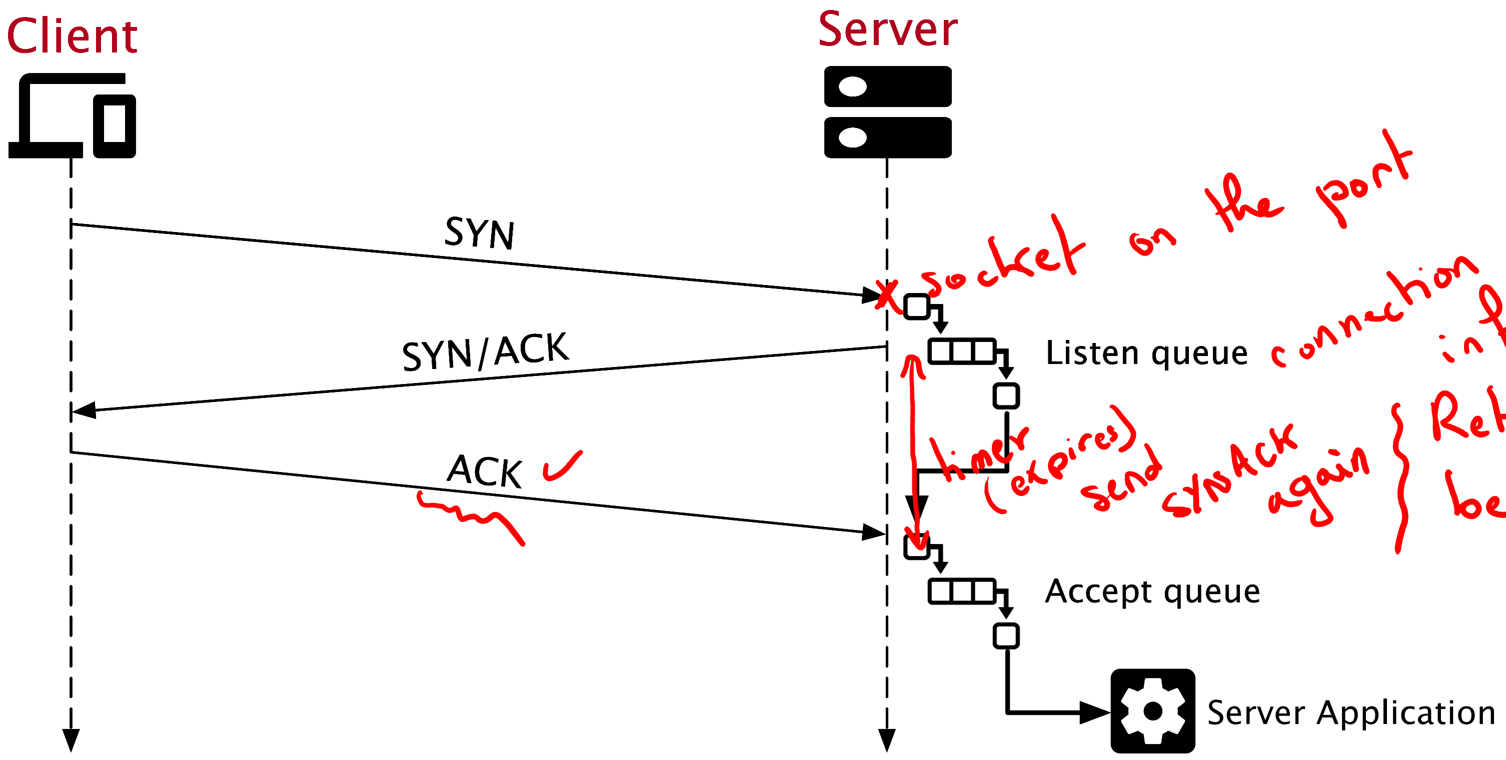
```
int main(int argc, char **argv)
{
    // initialization code
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    ...

    // bind the socket to the server address
    bind(sockfd, serv_addr, ...);
...

while(listen(sockfd, backlog) == 0) {
    int new_sock = accept(sockfd, ...);
    ...
}
        read(new_sock, data, datalen);
        ...
        send(new_sock, new_data, new_data_len);

        // done with this client
        close(new_sock);
    }

    // cleanup
    close(sockfd);

    return EXIT_SUCCESS;
}
```

size of listen queue

The 3-way Handshake

listen : trying to talk to

accept : actively talking to



socket on the port

connection information

timer (expires) send SYNACK again

Retry ~~6~~ 4 5 times before time out

k=2,3

State-Exhaustion Attacks

Definition:

State-exhaustion attacks attempt to deplete computational or memory resources at a victim server.

They are often combined with volumetric attacks for more effective *Distributed Denial of Service (DDoS)* attacks.

Impact of the Attacks

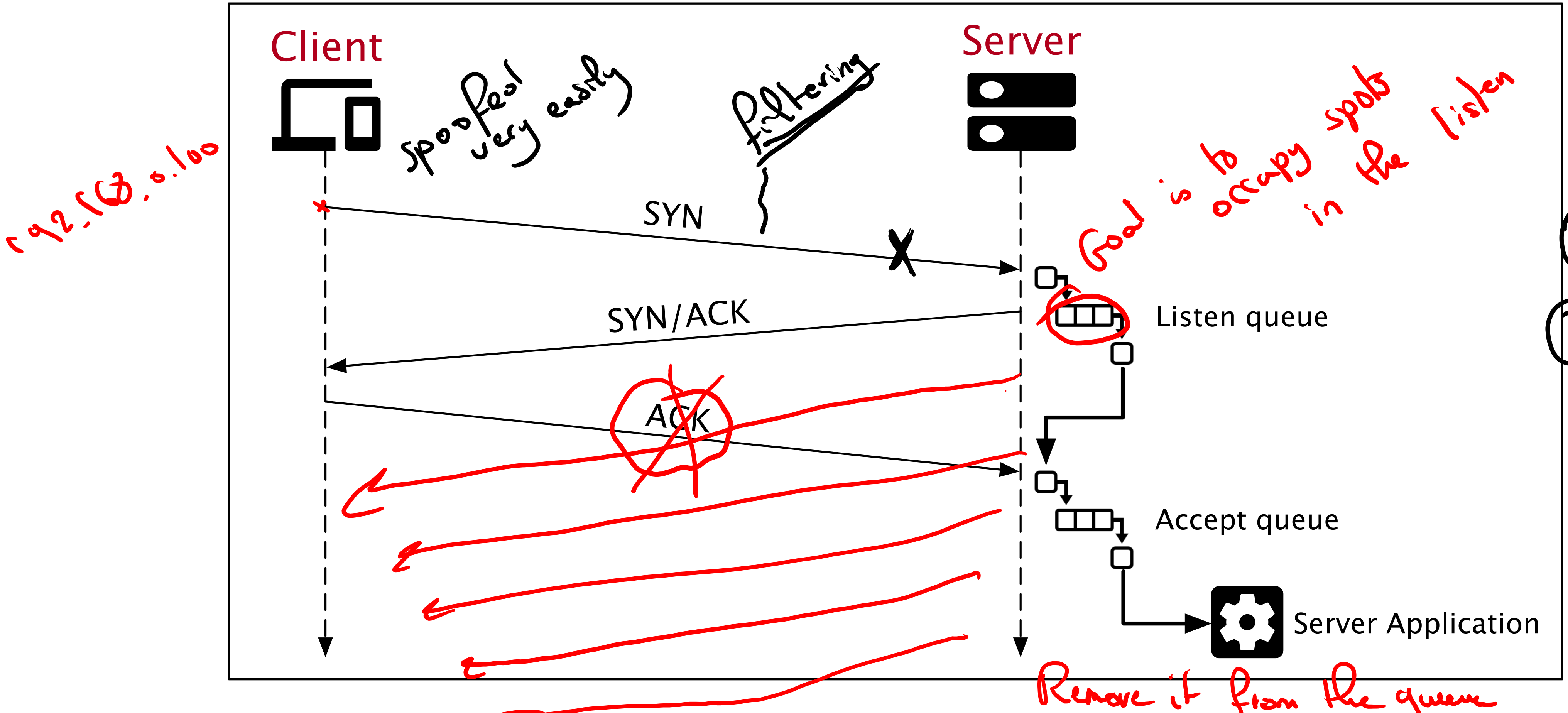
- Can you recall any recent DDoS attacks?

  2016 2018 2014

  - Dyn, Github, Krebsonsecurity, Blizzard, etc.
- Let's see an attack while it is taking place!

Breaking TCP for Fun & Profit

Recall: State-exhaustion attacks attempt to deplete computational or memory resources at a victim server.

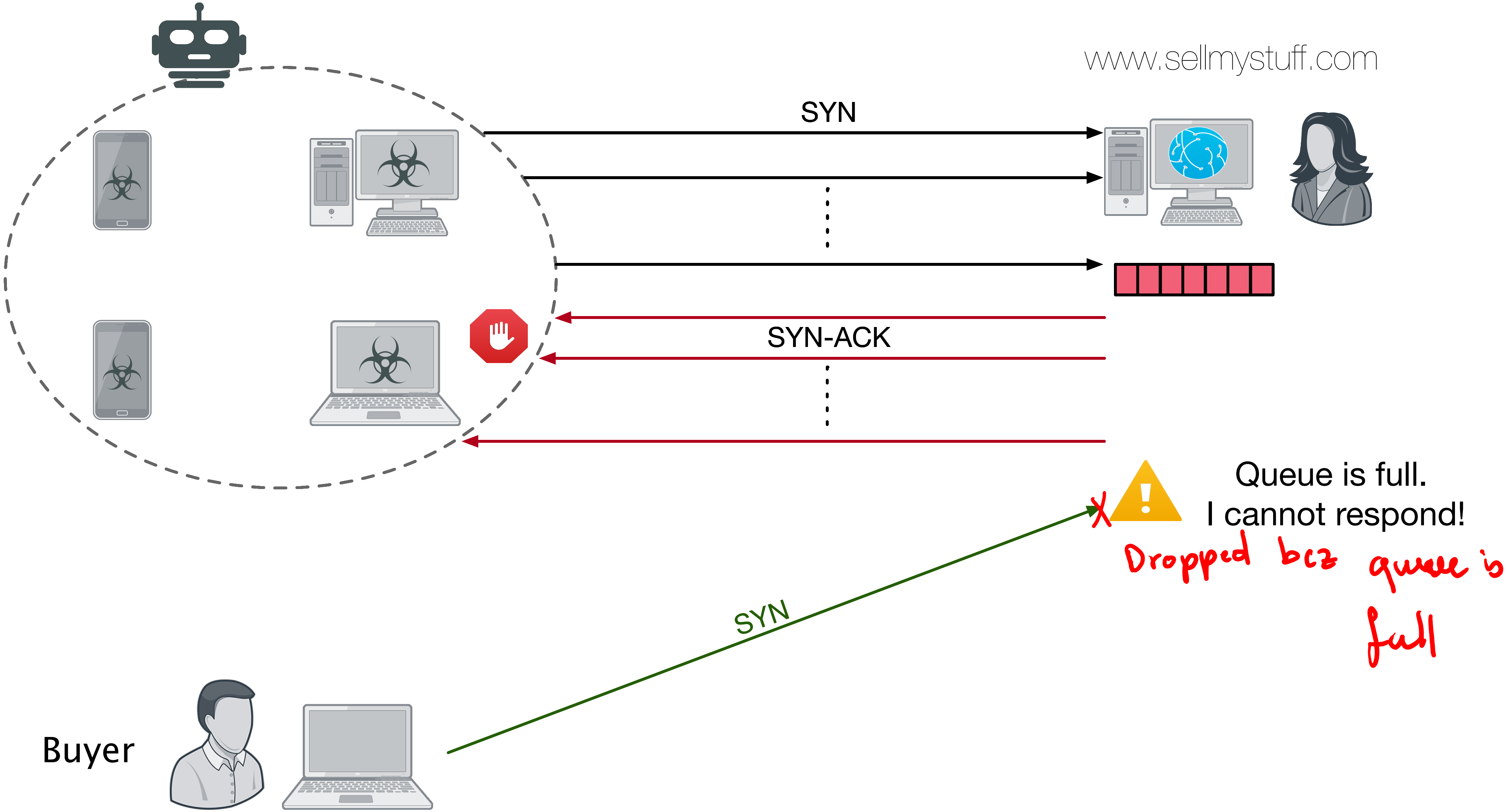


① Adjust to

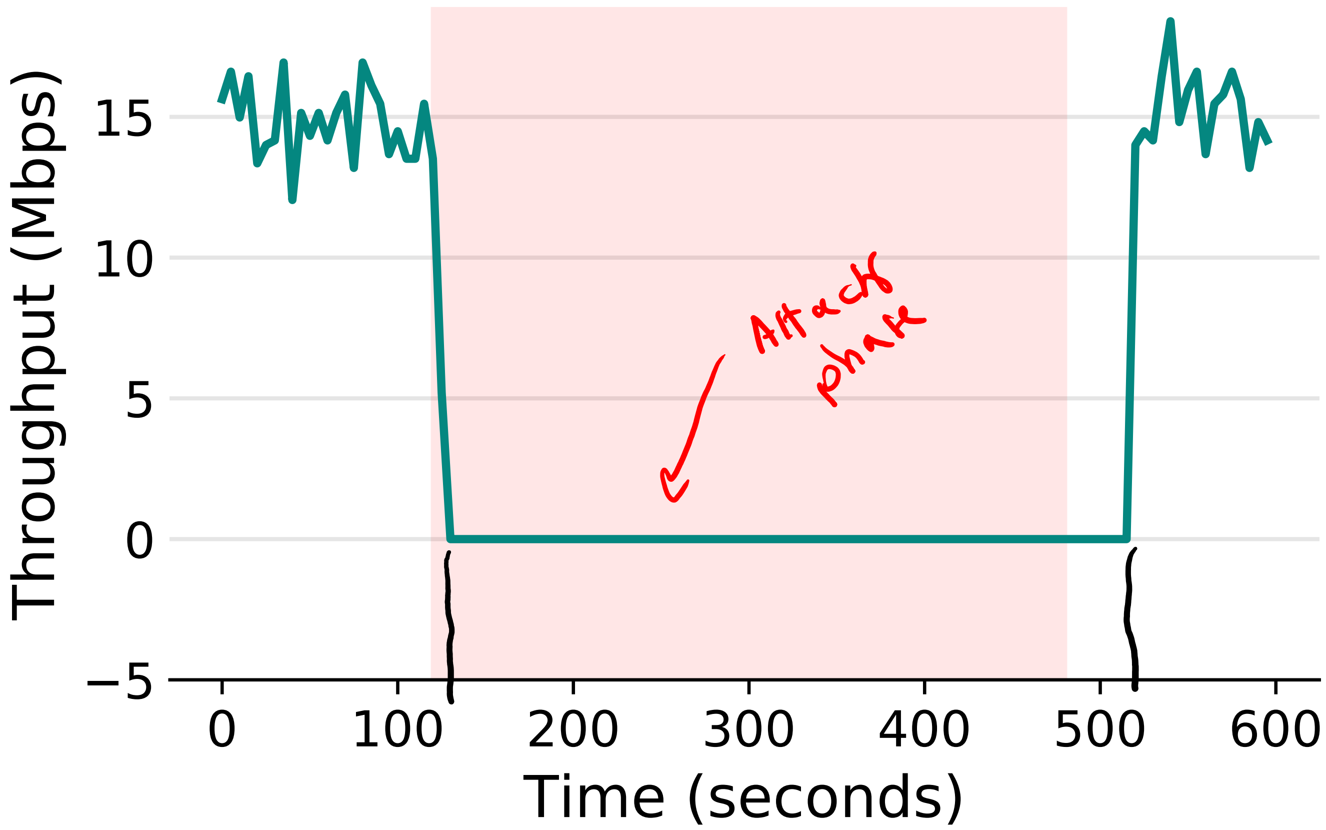
② Filtering

③ Authorization

SYN Flood Attack

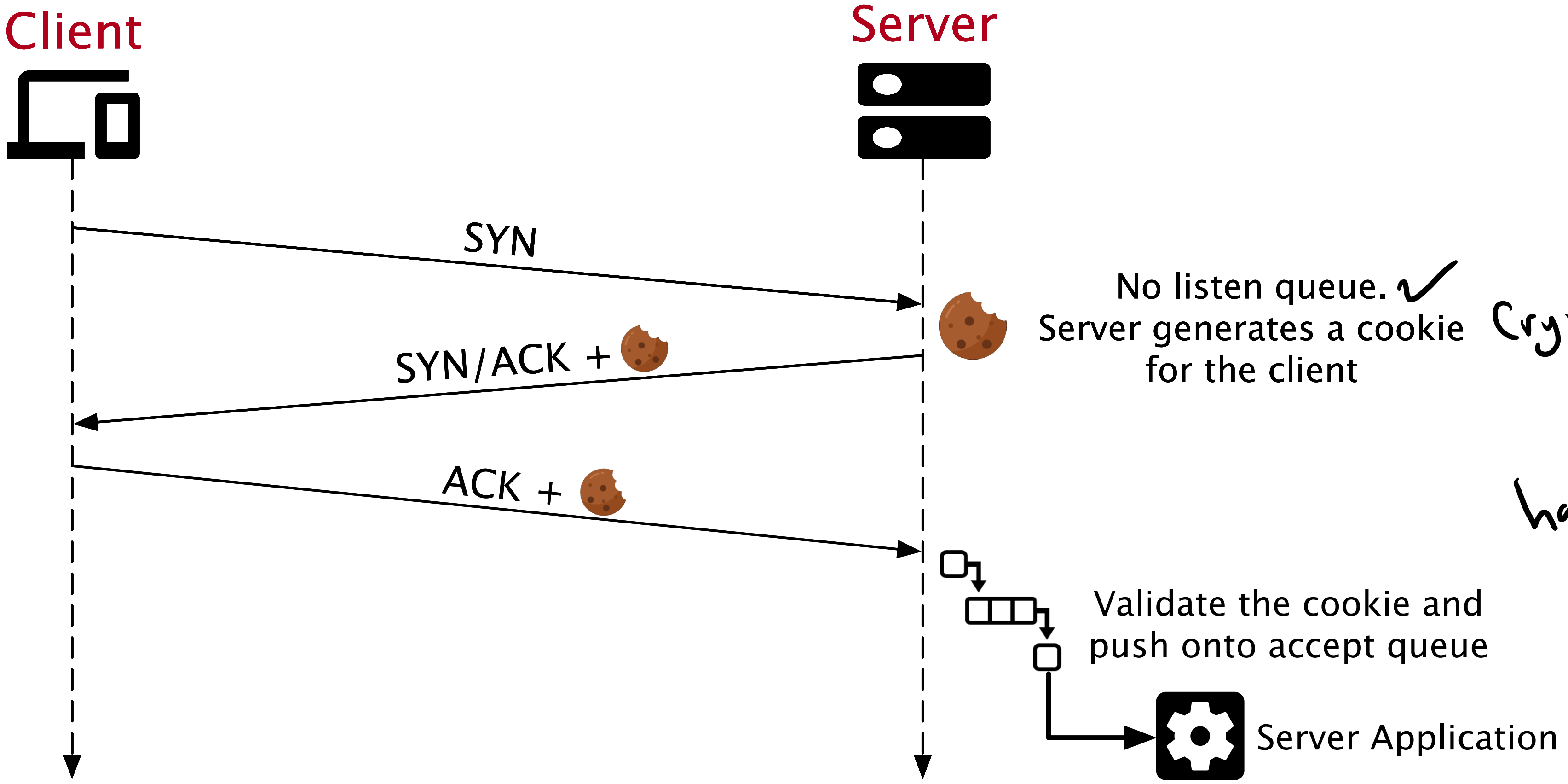


A SYN Flood in Action



TCP Defenses: SYN Cookies (Authorization)

- ❑ Server replaces the listen queue with a cookie



Client

Server

SYN

No listen queue. ✓
Server generates a cookie
for the client

Crypto generated cookie

hash(IP, port, ...)

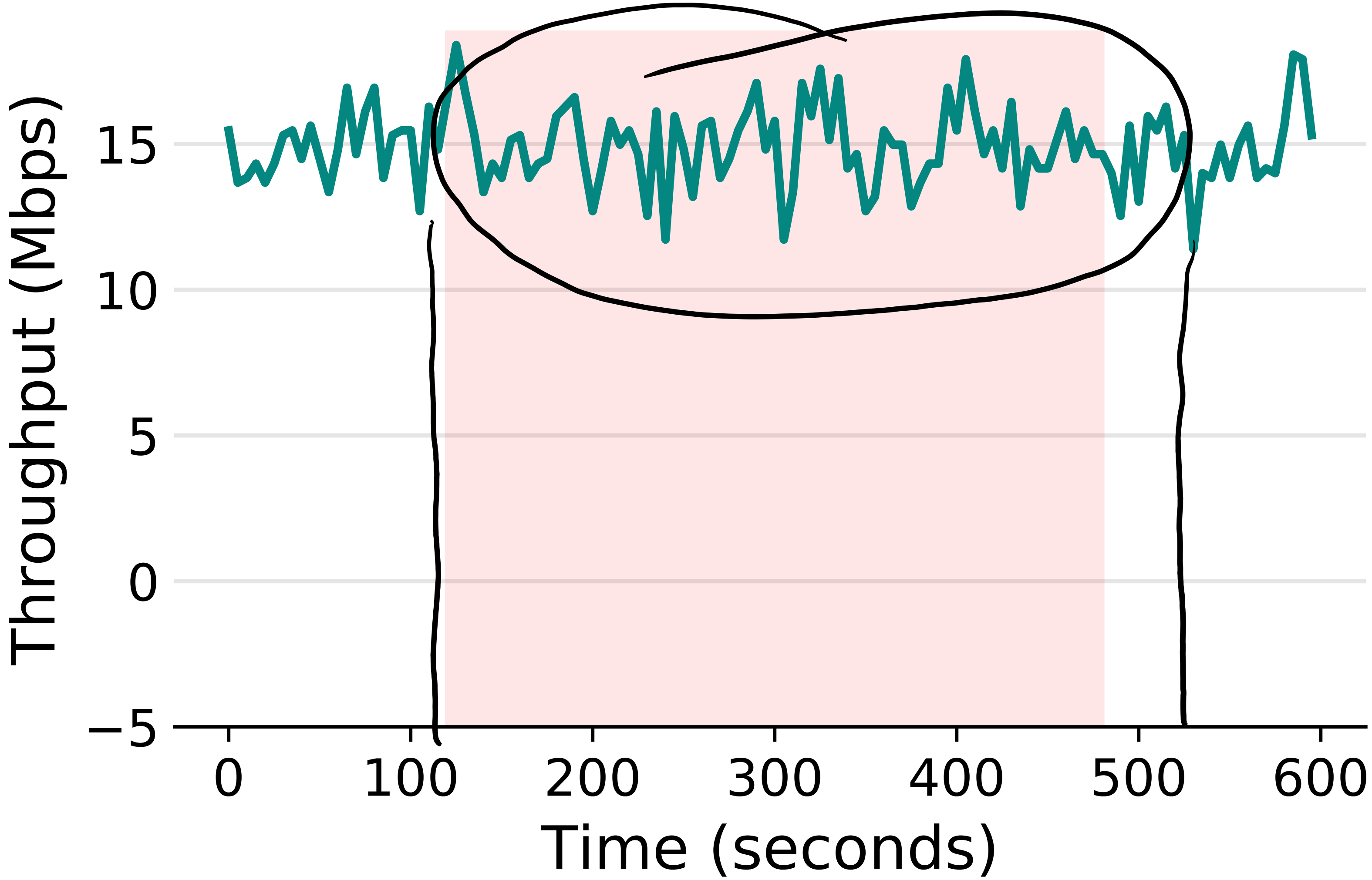
SYN/ACK + 🍪

ACK + 🍪

Validate the cookie and
push onto accept queue

Server Application

SYN Cookies in Action



Pushing the Envelop

- ❑ What is the main objective in a SYN flood attack?

  Fill up the listen {Overwhelm the state at the server}

- ❑ Why do SYN cookies work in this case?

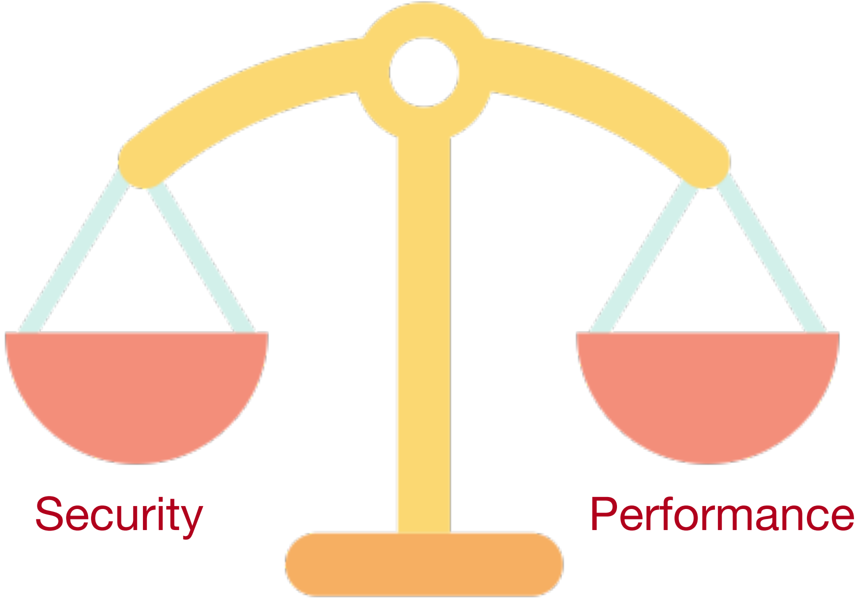
  Don't need a queue anymore.

- ❑ What could be the next logical target?

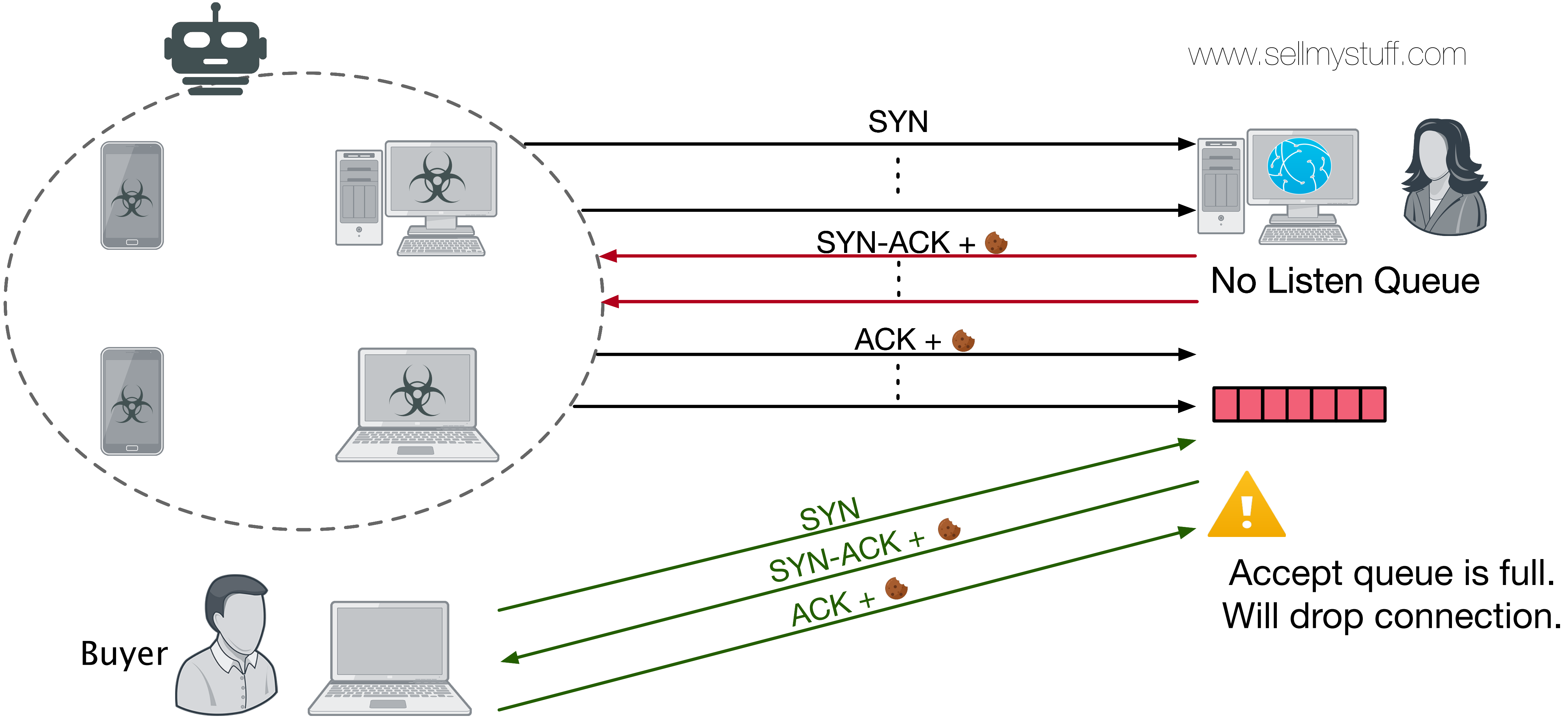
  Accept queue

Security Tradeoff

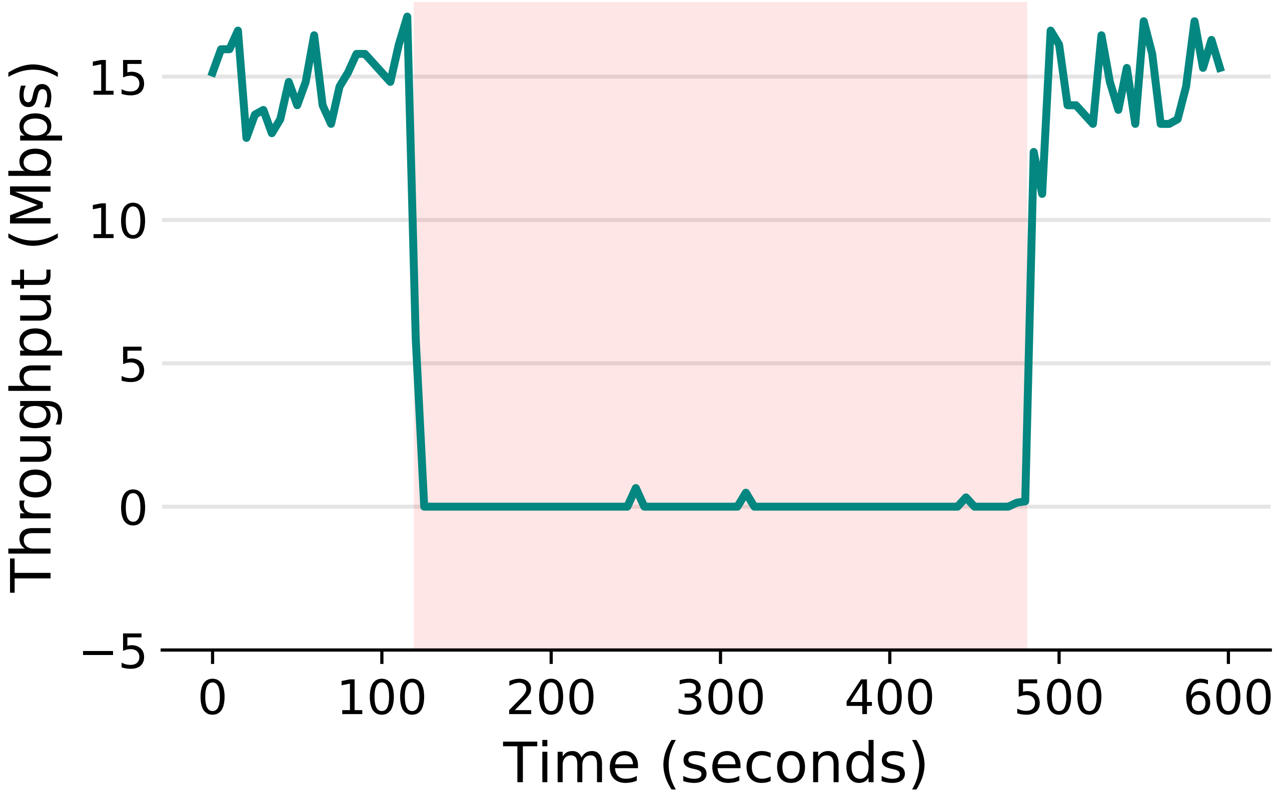
- ❑ The server must keep accepting new connections.
- ❑ There is thus a tradeoff between **security** and **performance**
- ❑ Figuring out the right balance is the job of a good engineer



Connection Flood Attacks

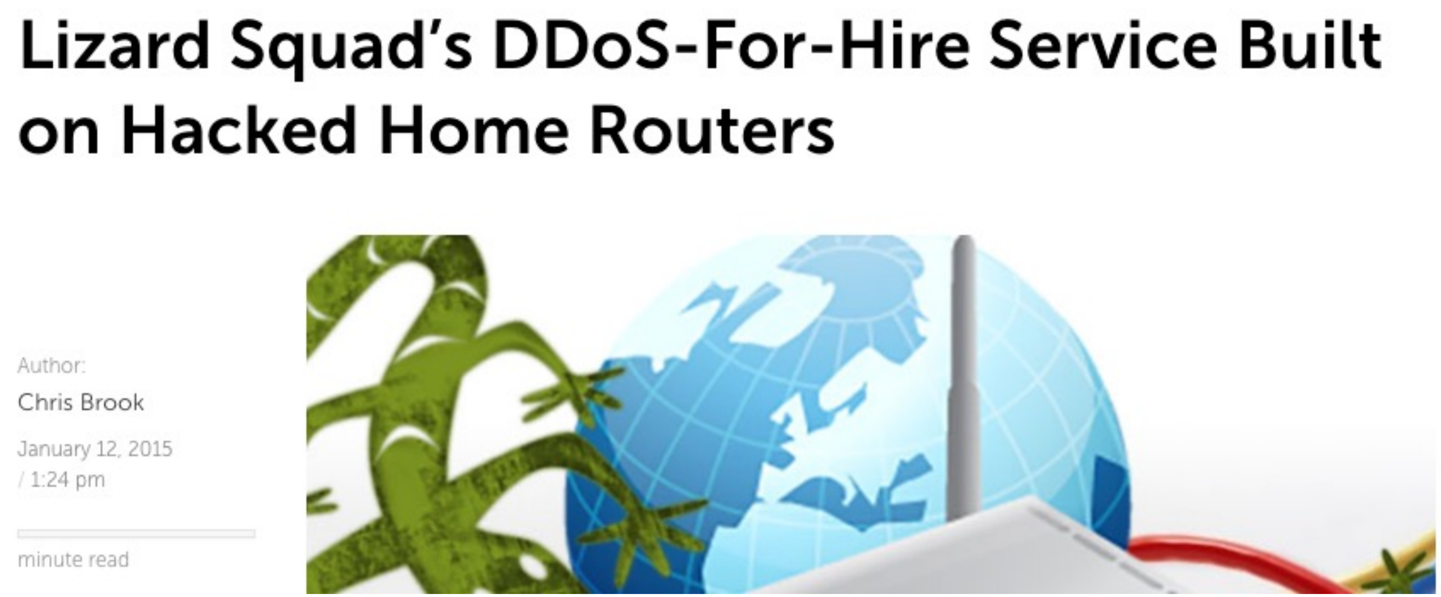


Connection Flood Attacks



Why do Connection Floods Work?

- ❑ Compared to a SYN flood, the success of a connection flood is dependent on



Lizard Squad's DDoS-For-Hire Service Built on Hacked Home Routers

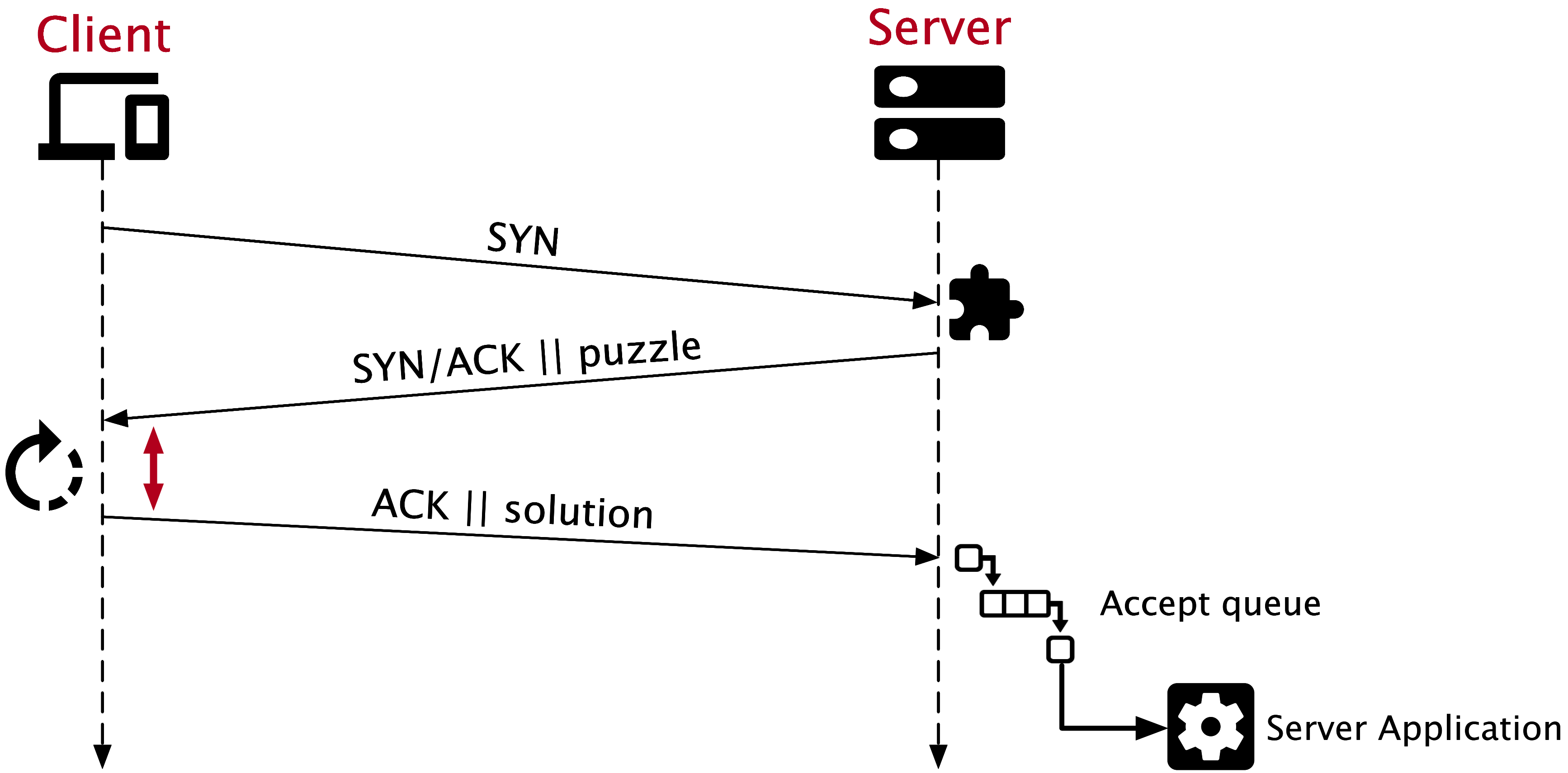
Author:
Chris Brook

January 12, 2015
/ 1:24 pm

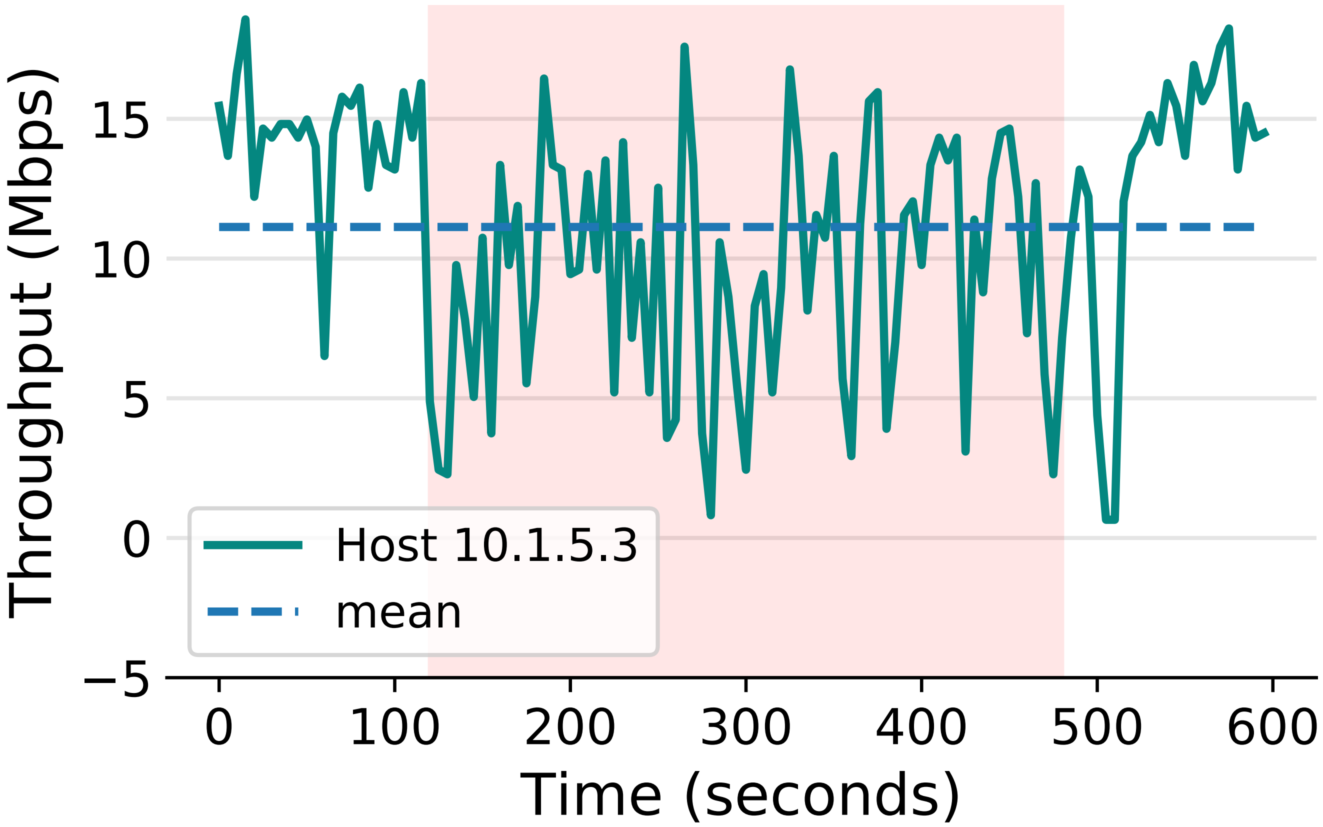
minute read

- ❑ How can
  - ▪ They use large botnets, e.g. the Mirai botnet peak at 650k bot devices!

Client Puzzles

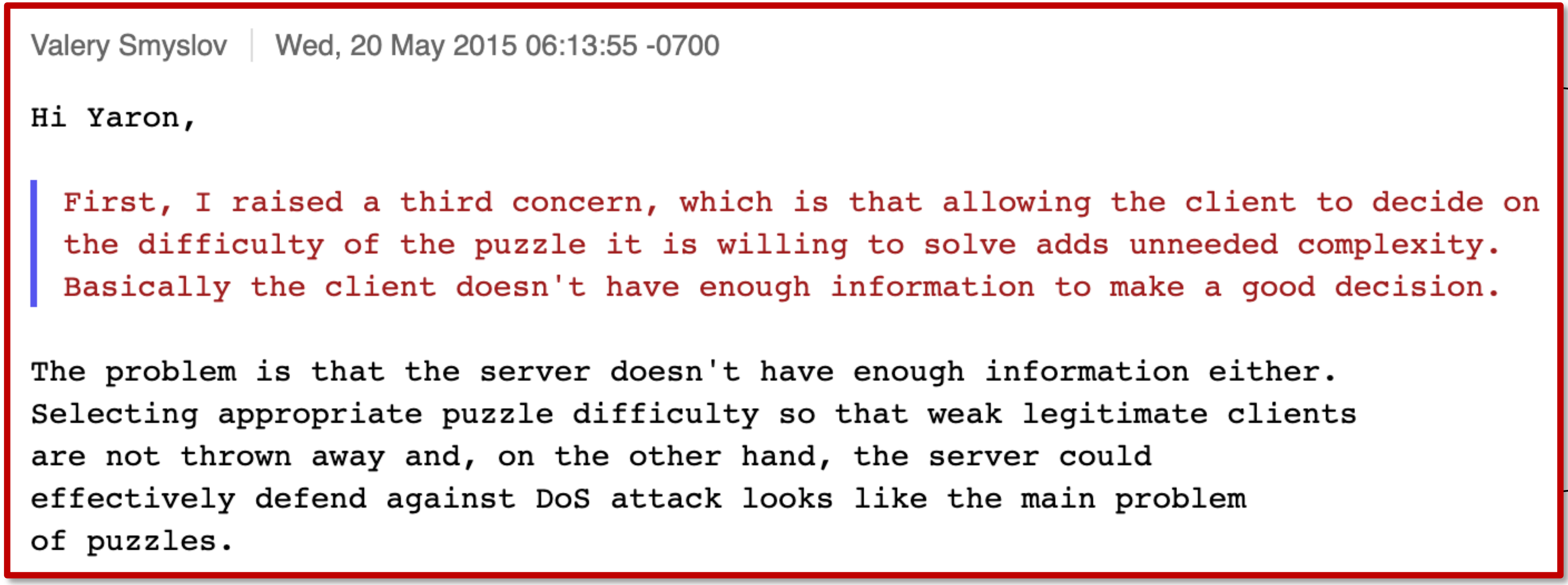


Client Puzzles in Action



Next Steps

❑ Heterogeneous set of devices, setting the puzzle difficulty can be very challenging!

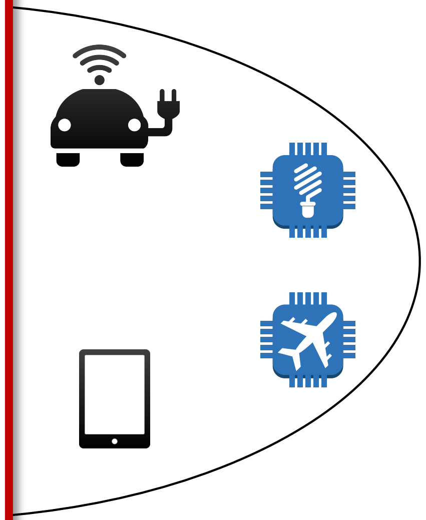


Valery Smyslov | Wed, 20 May 2015 06:13:55 -0700

Hi Yaron,

> First, I raised a third concern, which is that allowing the client to decide on the difficulty of the puzzle it is willing to solve adds unneeded complexity. Basically the client doesn't have enough information to make a good decision.

The problem is that the server doesn't have enough information either. Selecting appropriate puzzle difficulty so that weak legitimate clients are not thrown away and, on the other hand, the server could effectively defend against DoS attack looks like the main problem of puzzles.



Recap

- ❑ Why is TCP vulnerable to state exhaustion attacks?

Exploit	Targets	By	Mitigated by	Limitation of mitigation technique
Syn Flood	The listen queue	Sending a barrage of SYN packets and not ACKing the SYN-ACK	SYN Cookies	Fails when there is a connection flood.
Connection Flood	The accept queue	Completing a lot of connections faster than the application can process them	Client puzzles	Need to determine a balanced puzzle difficulty, especially with heterogeneous devices.