# Exploiting TCP Simultaneous Connections to bring out some Security Shenanigans

- Ahamed Nafeez

C0c0n, 2011

# Who am I ?

- Final Year ,Computer Science, B.E. Student
- Security Reseacher
- Almost a CCIE Security
- Facebook Whitehat , Reported several security bugs to Facebook.
- Anything Security

Twitter : @skeptic_fx

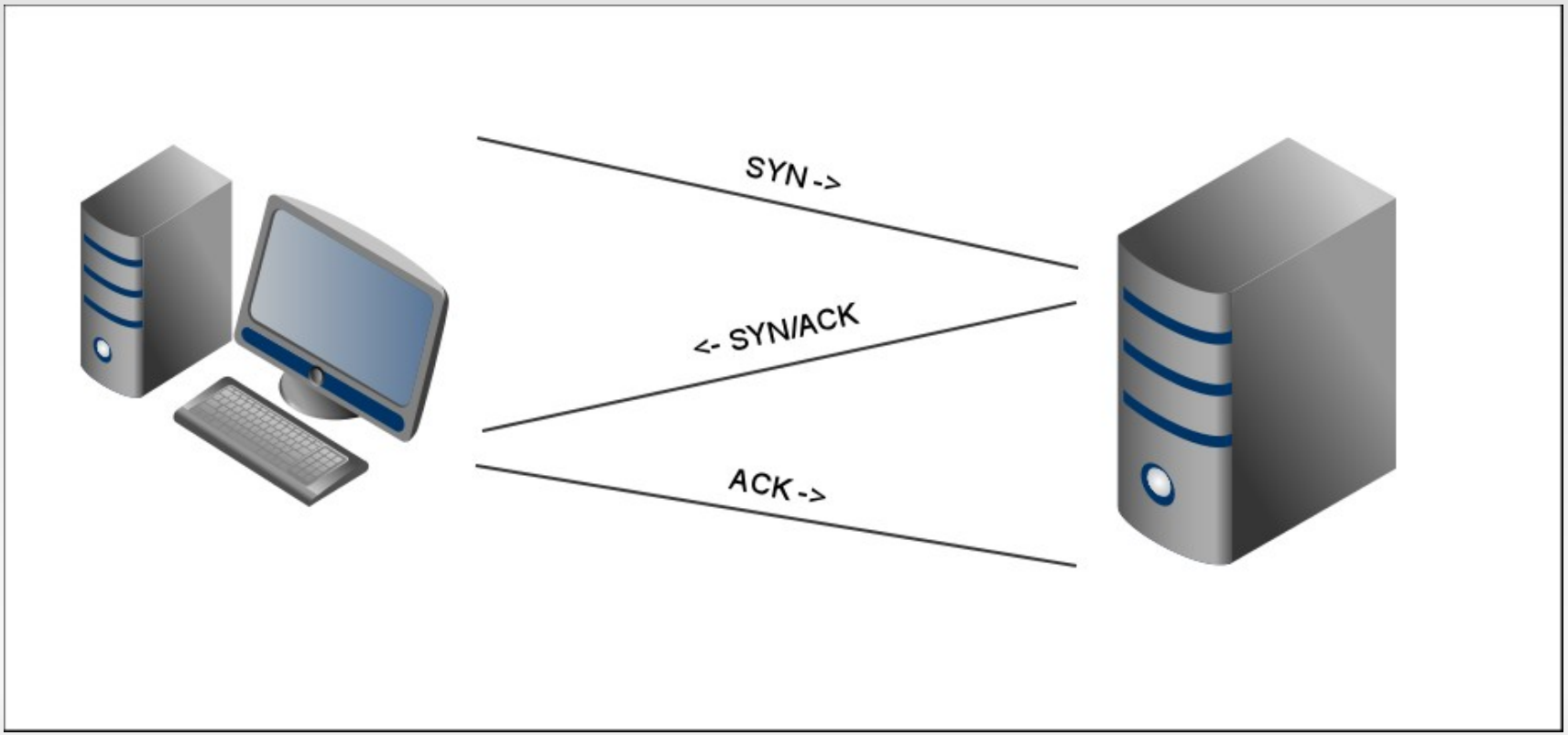
Github : github.com/skepticfx

# What to Expect ?

- Review of TCP Split Handshake
  - Impacts on Security Devices
  - Client Side Attacks
- TCP Simultaneous Open
  - Impacts
  - Some SYN Flood & SYN Cookies
  - DoS Mitigation

# The Classical TCP 3-Way Handhsake

- Reliable Connection Establishment.
- The SYN, SYN-ACK, ACK trio.



# Now wait, Its not a 3-way handshake

- The 3-Way Handshake, is a Cliché
- In Fact, Its a Four way handshake , Piggy Backing made it Three.
- RFC 793, Transmission Control Protocol says something else.
- *“The synchronization requires each side to send it's own initial sequence number and to receive a confirmation of it in acknowledgment from the other side. Each side must also receive the other side's initial sequence number and send a confirming acknowledgment.”*

  **1) A --> B SYN my sequence number is X**

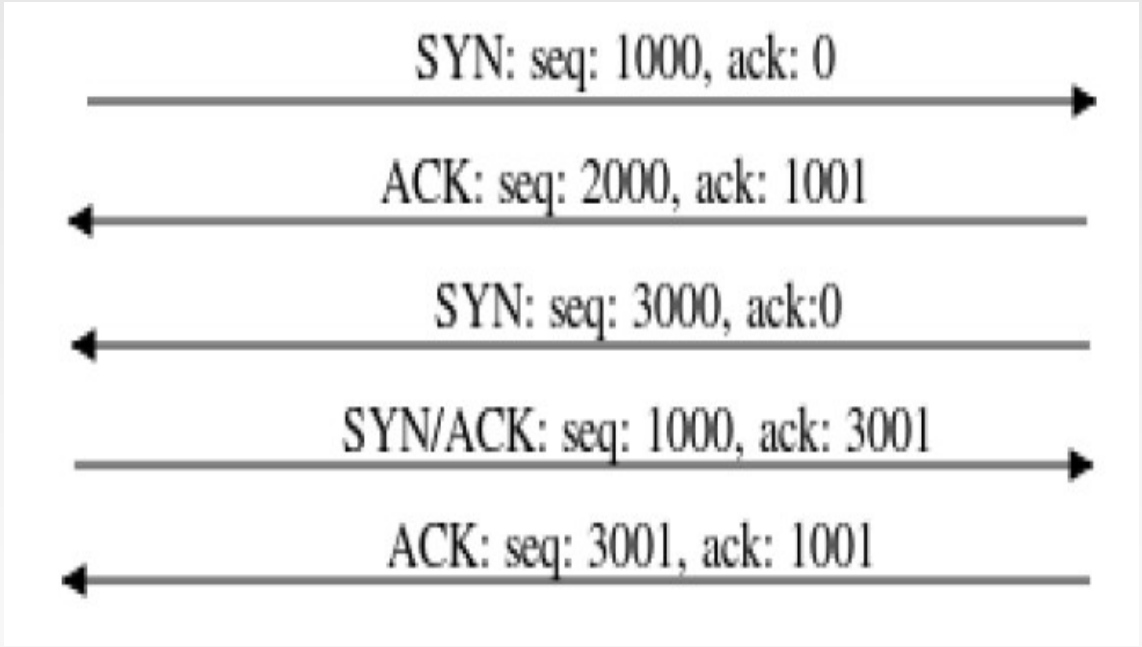
  **2) A <-- B ACK your sequence number is X**

  **3) A <-- B SYN my sequence number is Y**

  **4) A --> B ACK your sequence number is Y**

# The Split Handshake

- Tod Beardsley, a Security Researcher , found a novel handshake method called the SneakAck aka. Split Handshake.
- A few well known Security Devices were vulnerable.



# A Compact Split Handshake

- The First Ack is not mandatory , ignoring it, doesn't make any difference, making this a Four-Way Split Handshake.

# Impacts on Security Devices

- Avoid Detection
- Bypass Firewall Restrictions

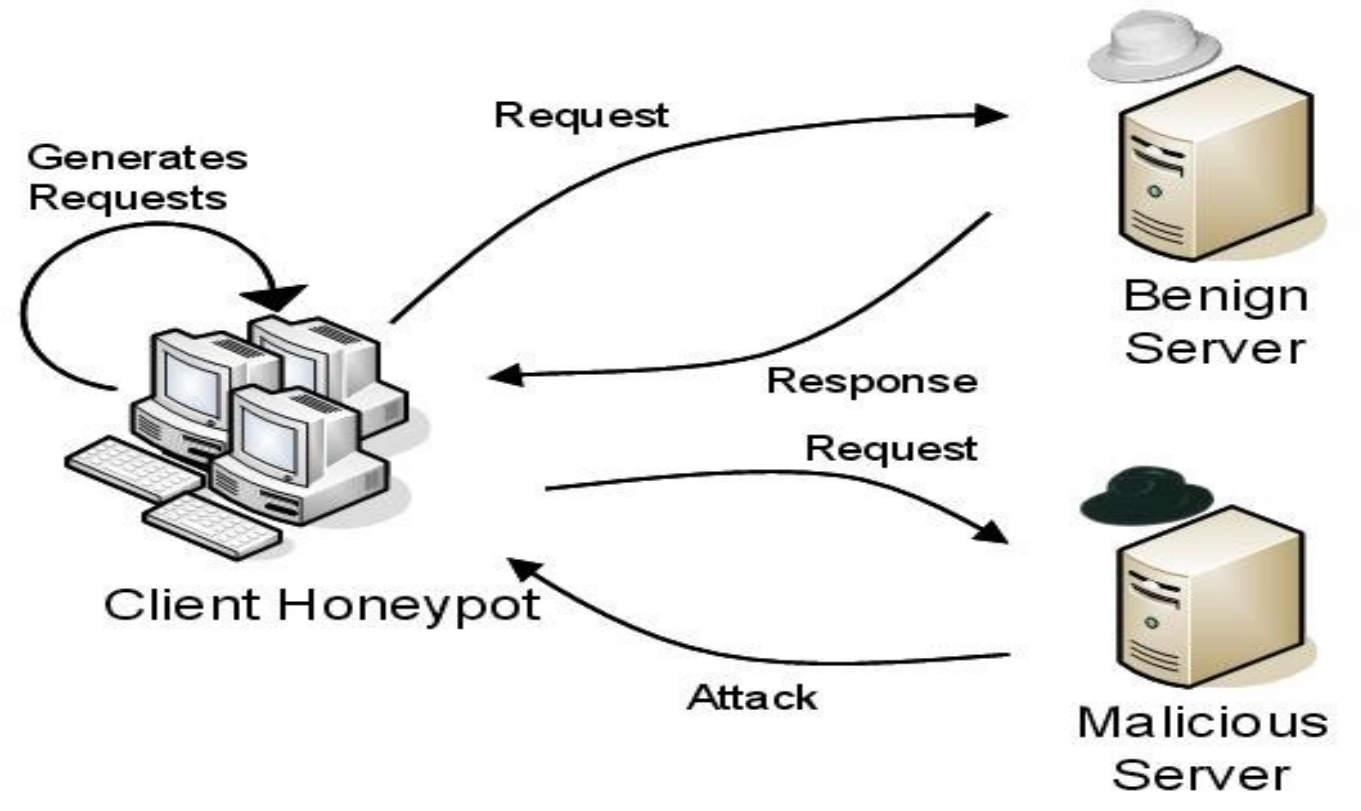
  *If the firewall tracks sessions based on ISN and only one socket pair, the Pure SYN packet can be crafted to bypass the firewall restrictions.*

- Fool the IPS Devices

  *The directionality of the connection flow is changed. IPS devices, are fooled to think the server is the client and vice versa.*

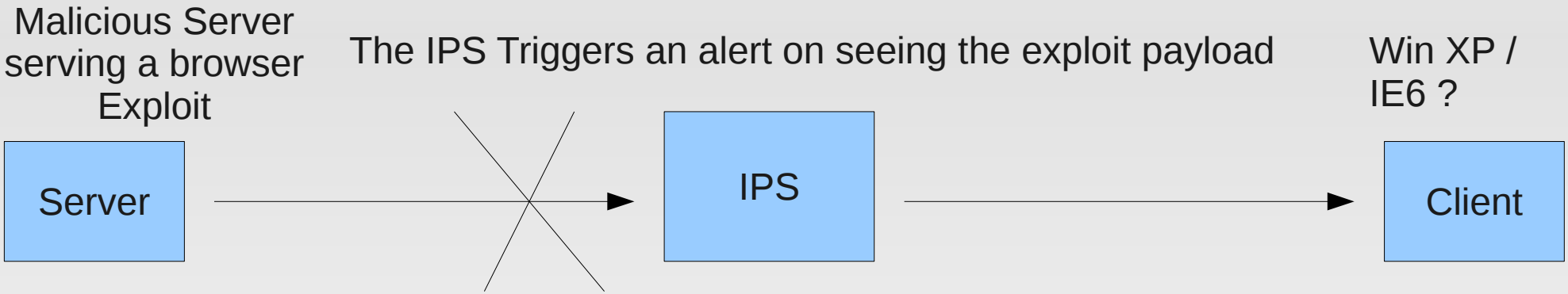
# Client Side Attacks

- Attacks that target vulnerabilities in client applications that interact with a malicious server or process malicious data..

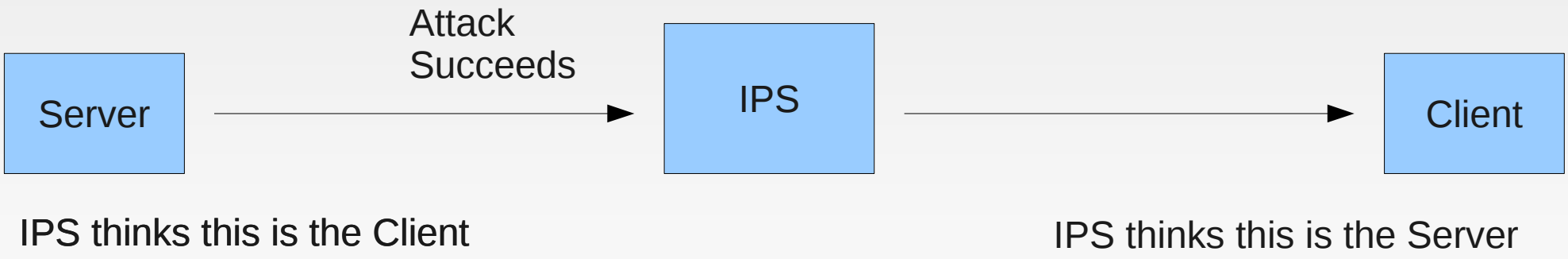


# Imagine, Browser exploits can evade IPS

- IPS detects malicious activity depending up on who the internal network is. (Client/ Server)




**Split handshake can change the direction of the flow , from the IPS' context.**

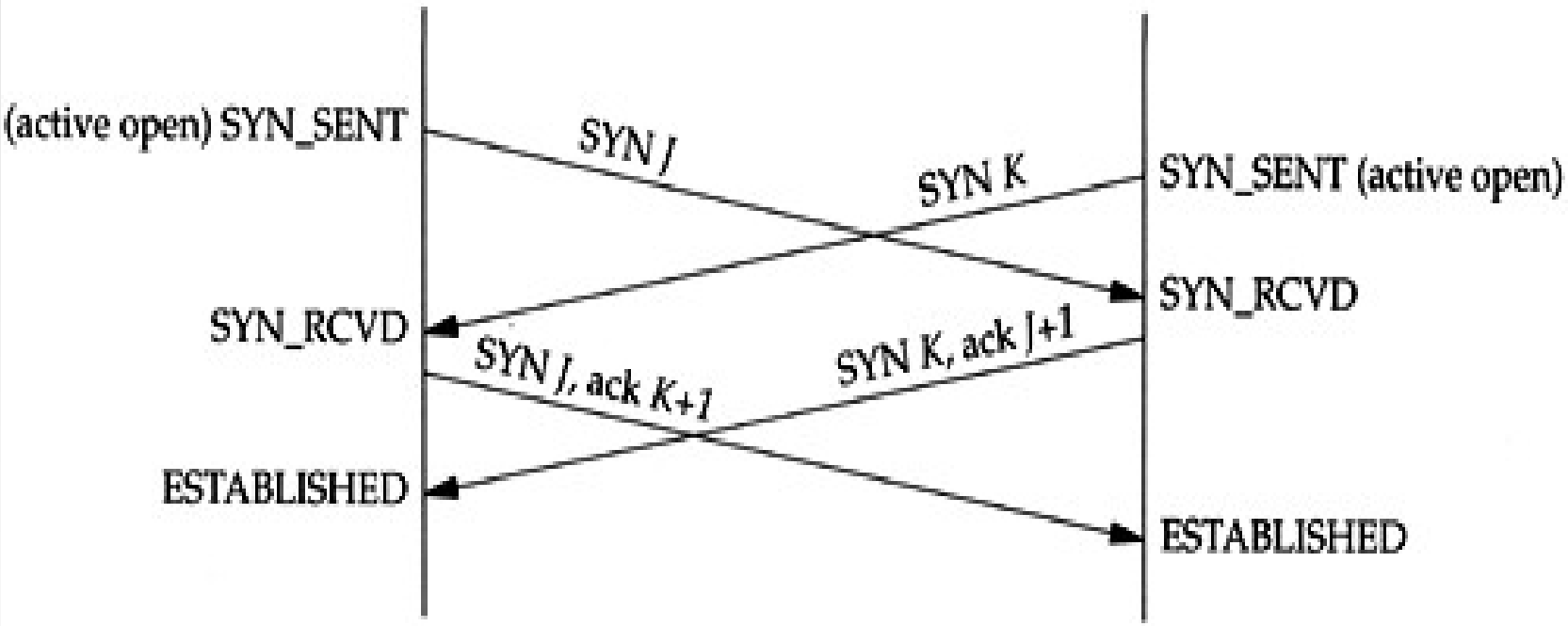


# TCP Simultaneous Open

- According to the RFC on TCP, The TCP Three way handshake is not the only means by which TCP Sessions can be established.
- A pair of TCP hosts can simultaneously attempt to open a connection to each other via a SYN packet.
- Something which is quite rare in the Internet today.
- The applications require a fairly contrived set of circumstances to initiate a Simultaneous-Open Connection.
- Both hosts need to send the SYN packets at virtually the same time, with mirrored port pairs.

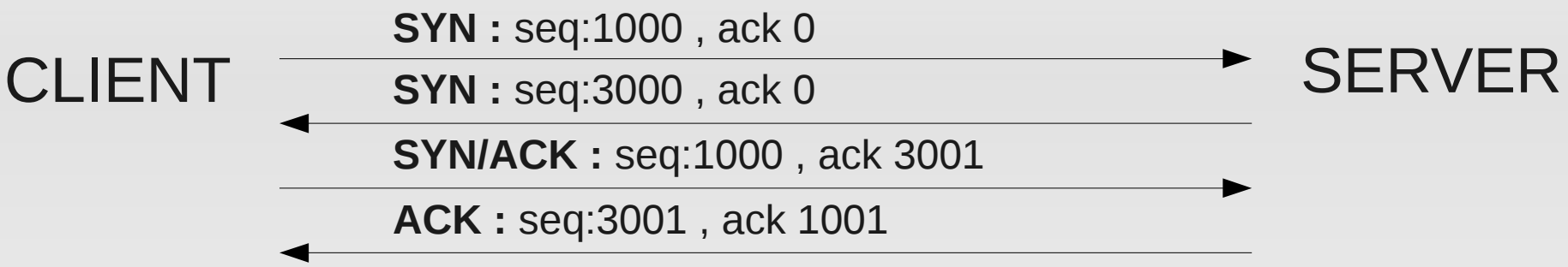
# This actually inspired the Split Handshake

- Its again  a 4 way handshake , a bit different from the Split Handshake.

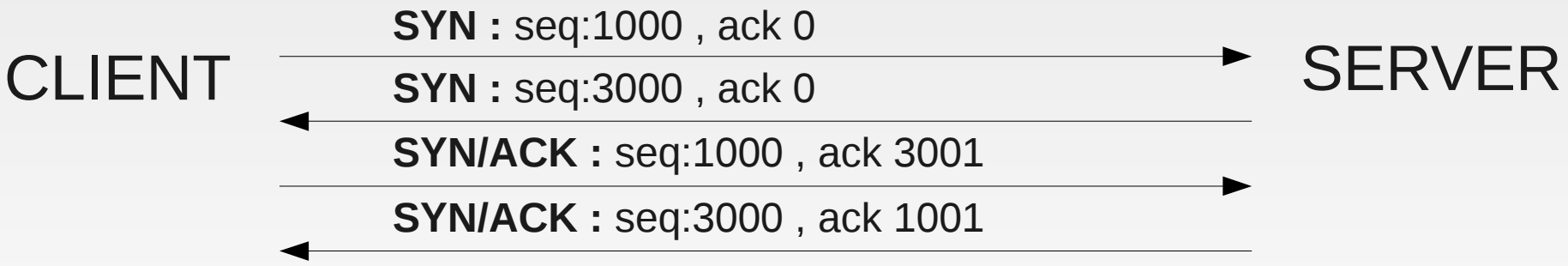


# Heres the difference

## Split Handshake




## Simultaneous Open Connection



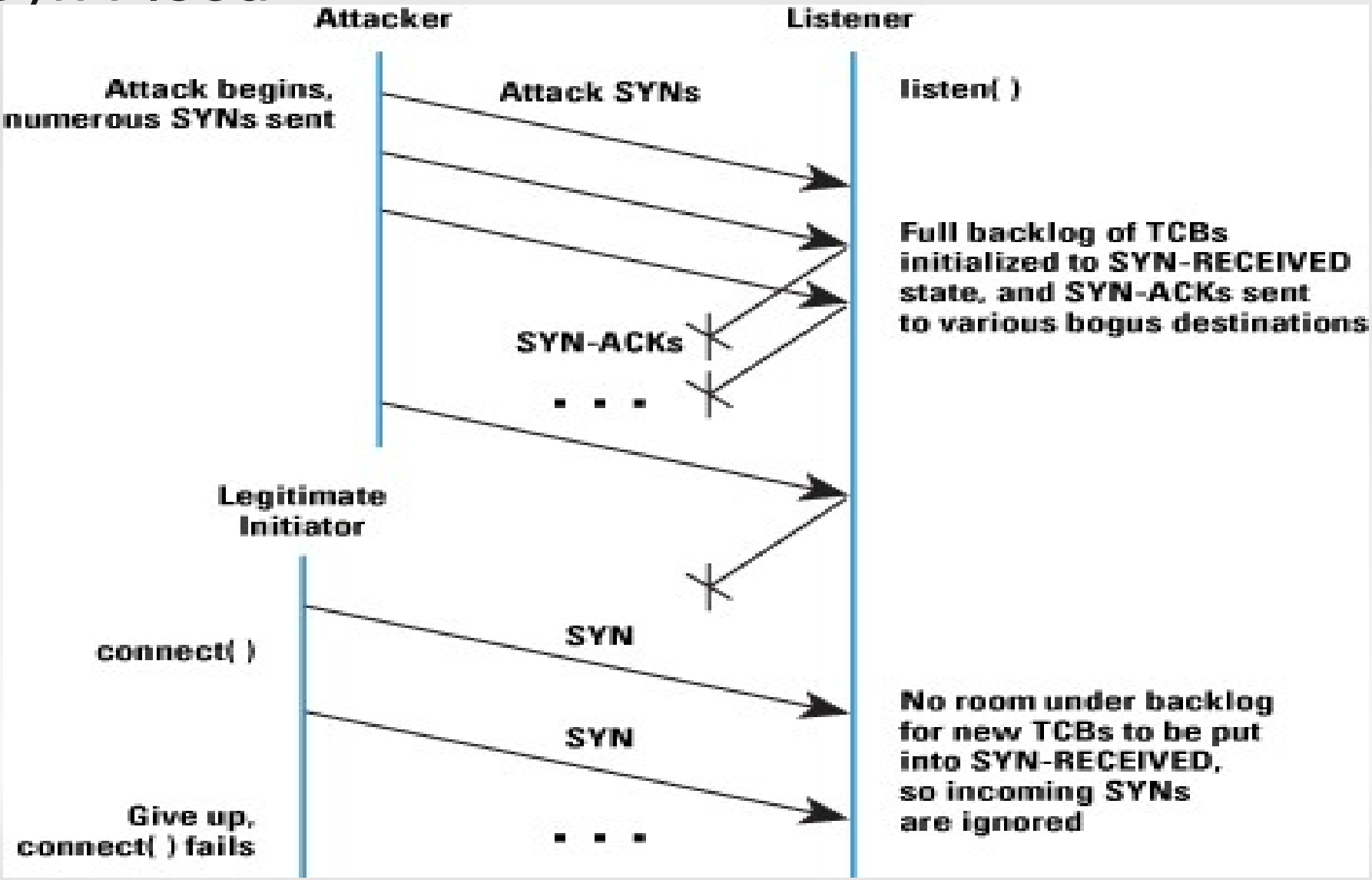

**Not the Last packet Sent by the Server in each of the cases.**
**Also note the Sequence Number.**

# Impacts

- The impacts are mostly similar to the Split handshake.
- Port scanners and host firewalls behave in an abnormal way.
- With the stack setup to behave in  a Simultaneous Open way , Port Scanners can be fooled.
- Tested on Nmap Port Scanner and Host Firewalls of Windows XP, Ubuntu 10.04

# Using Simultaneous Connections to mitigate DoS attacks.

- Syn Flood



# TCP backlog Queue

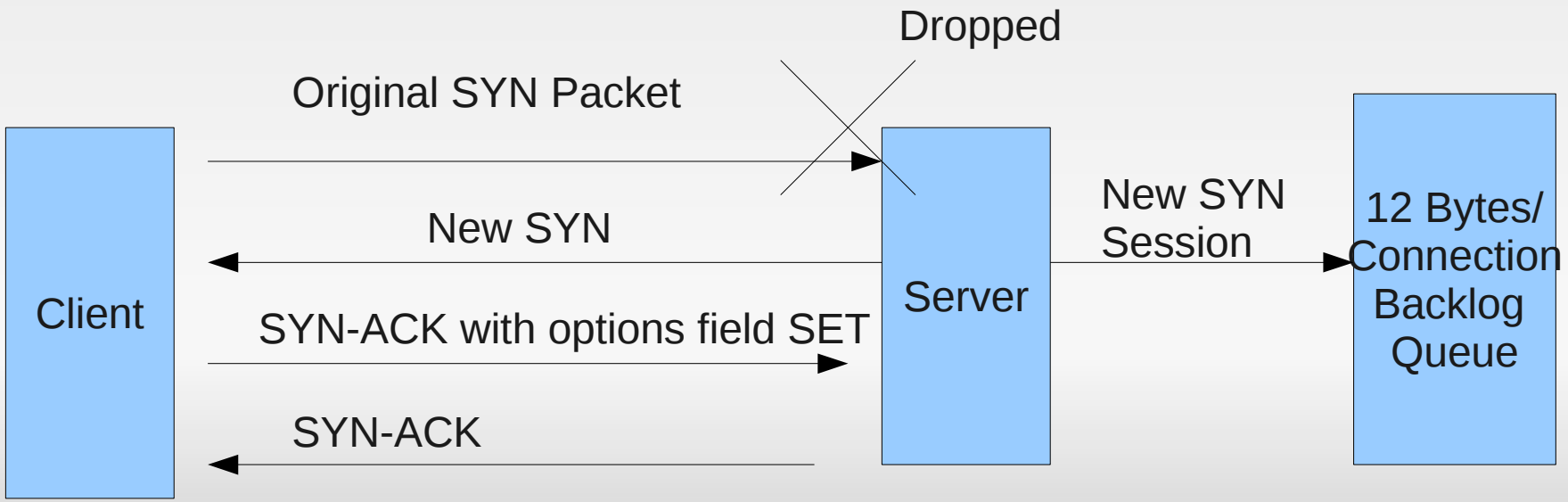
- Large memory structure used to store the incoming connection requests.
- In Linux , it is implemented as 'sk_buff_head' structure, which takes up to **80 bytes** for each connection.
- A tcp_max_syn_backlog variable is used to define the maximum number of connections to hold in the queue.
- If at any instant, the backlog queue is full, then the TCP Stack starts dropping all the packets.
- Having a high value for the tcp_max_syn_backlog will require more main memory.

# Good old SYN Cookies !

- Does not allocate any resources for the incoming SYN packet in the backlog queue.
- Instead, it computes a cookie value and encodes it in the TCP sequence number field of the SYN/ACK packet.
- When the server receives the ACK from the client, it checks whether its secret cookie function yields the right value for the current time. If so, it reconstructs the details of the original SYN packet that it had received earlier from the encoded cookie.
- SYN cookies fail to take advantage of the **TCP options** like large window and selective acknowledgement . Thus the attacker can perform a QoS Degradation and render the network useless, in case of high performance networks.

# Proposed Scheme

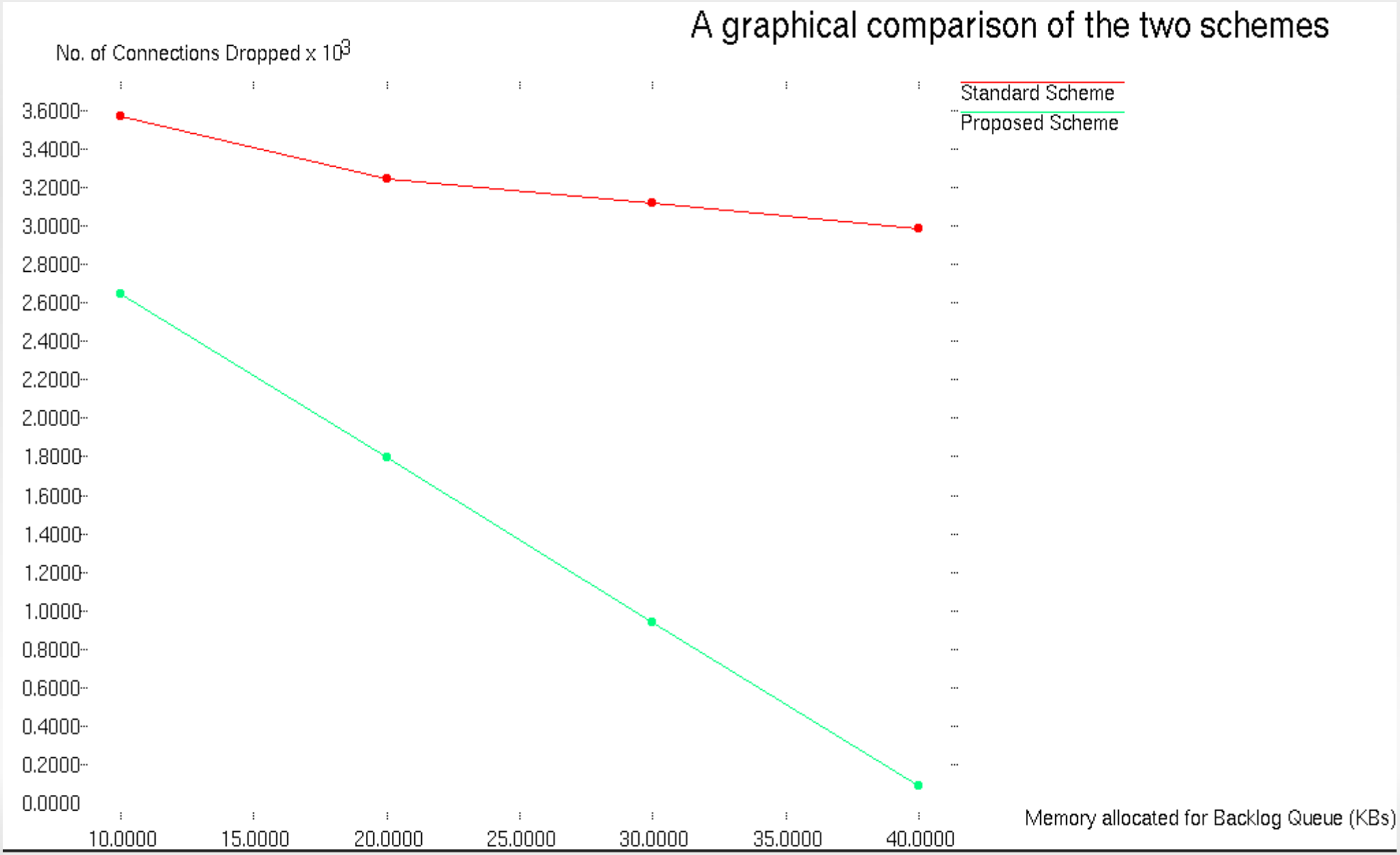
- Once a SYN packet is received by the server from a client, the server generates its own SYN with the same socket pair and with a fresh sequence number.
- The first SYN received is simply dropped and no details about it are saved in thebacklog queue. No cookies are computed.
- However the New SYN packet sent is saved for future connections.



# The minification !

- Destination IP Address [32 bits] + Destination Port Address [16 bits] + Source Port Address [16 bits] = **64 bits.**
- 64 bits + New Initial Sequence Number [4 bytes] = **12 bytes.**
- From 80 Bytes to 12 Bytes , thats about 6 times less.
- Means, less memory cost.
- Increase the maximum number of connections to a massive amount and we have a Low Cost DoS Mitigation Scheme now.
- Reliable and much better than SYN Cookies.
- Doesn't affect the TCP protocols like other schemes often do.

# Some Stats


A graphical comparison of the two schemes
No. of Connections Dropped x 10^3
Standard Scheme
Proposed Scheme
3.6000
3.4000
3.2000
3.0000
2.8000
2.6000
2.4000
2.2000
2.0000
1.8000
1.6000
1.4000
1.2000
1.0000
0.8000
0.6000
0.4000
0.2000
0.0000
10.0000
15.0000
20.0000
25.0000
30.0000
35.0000
40.0000
Memory allocated for Backlog Queue (KBs)

# FIN

You can get the Simultaneous Open Connection and Split handshake connections module at,

http://www.github.com/skepticfx

0x01

@skeptic_fx