# Knock: Stealthy TCP Servers

Julian Kirsch, Maurice Leclaire & Christian Grothoff

Fakultät für Informatik
Technische Universität München

December 29, 2013

# Outline

# Motivation

Why?

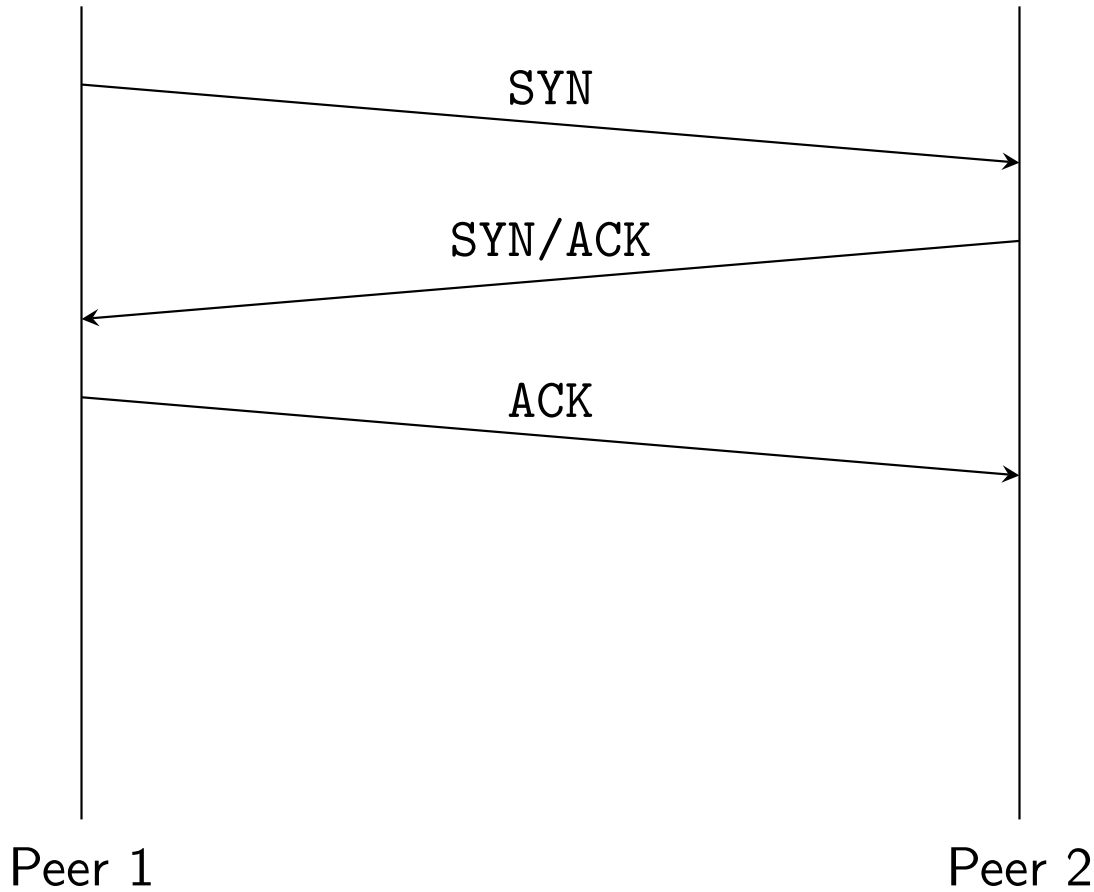
- minimize visible footprint
- anti-censorship
- national restrictions on how to use the internet

# Goals

- invisible for portscanners
- no easy way to distinguish communication with a hidden service from a normal connection by traffic observation
- (optionally) integrity protection of TCP payload to prevent man-in-the-middle attacks
- easy to deploy & use

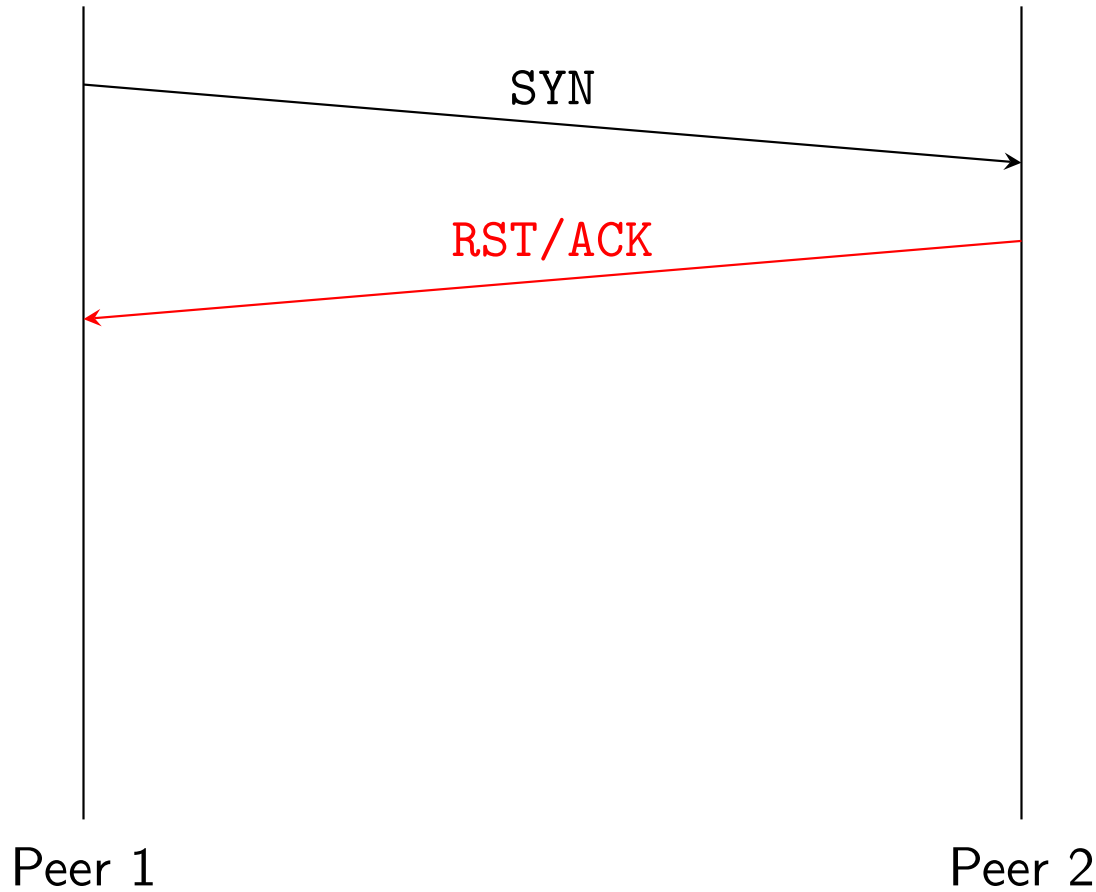
# Goal

## Normal TCP 3-way-handshake



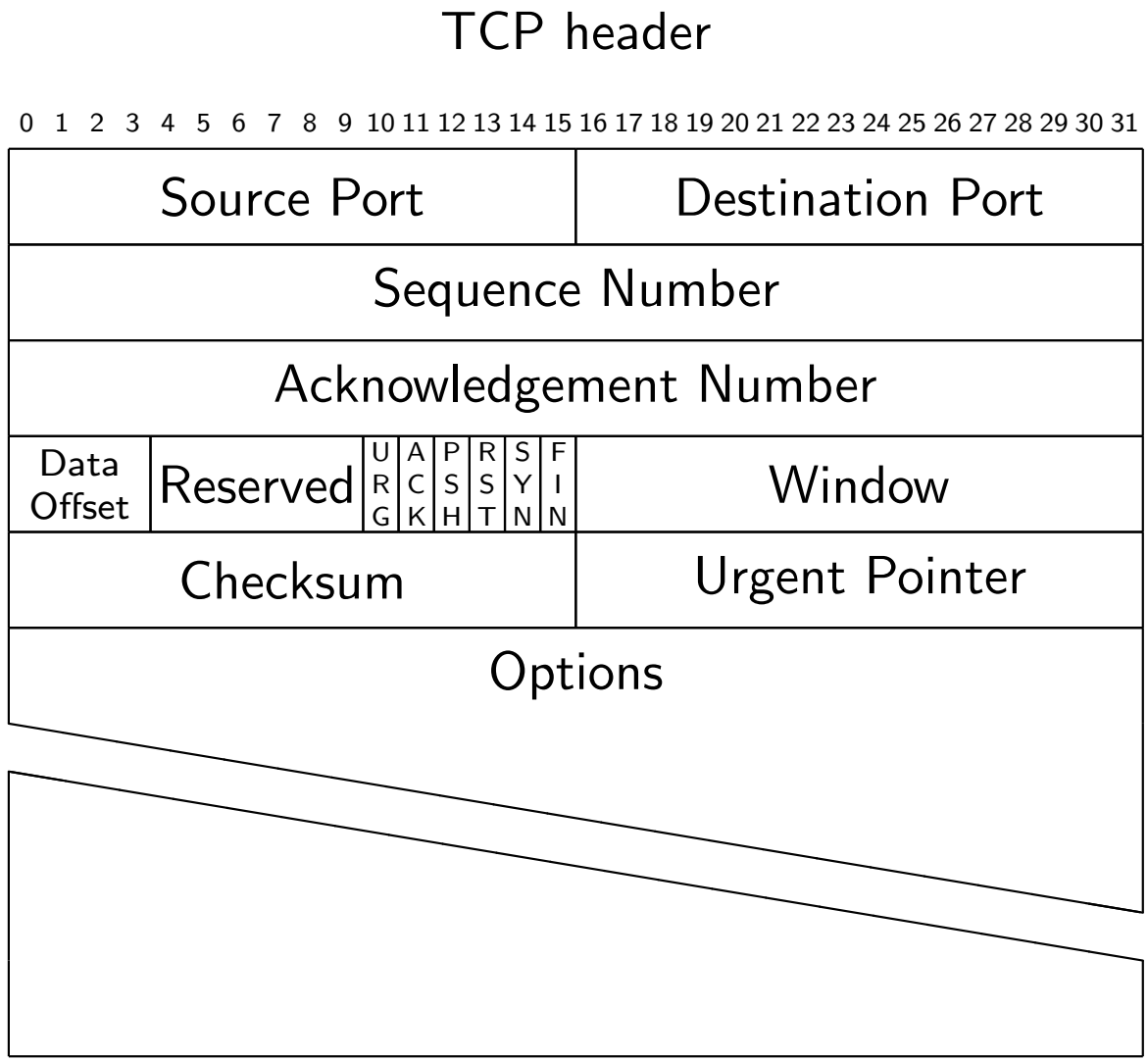
# Goal

## Closed port



# Goal

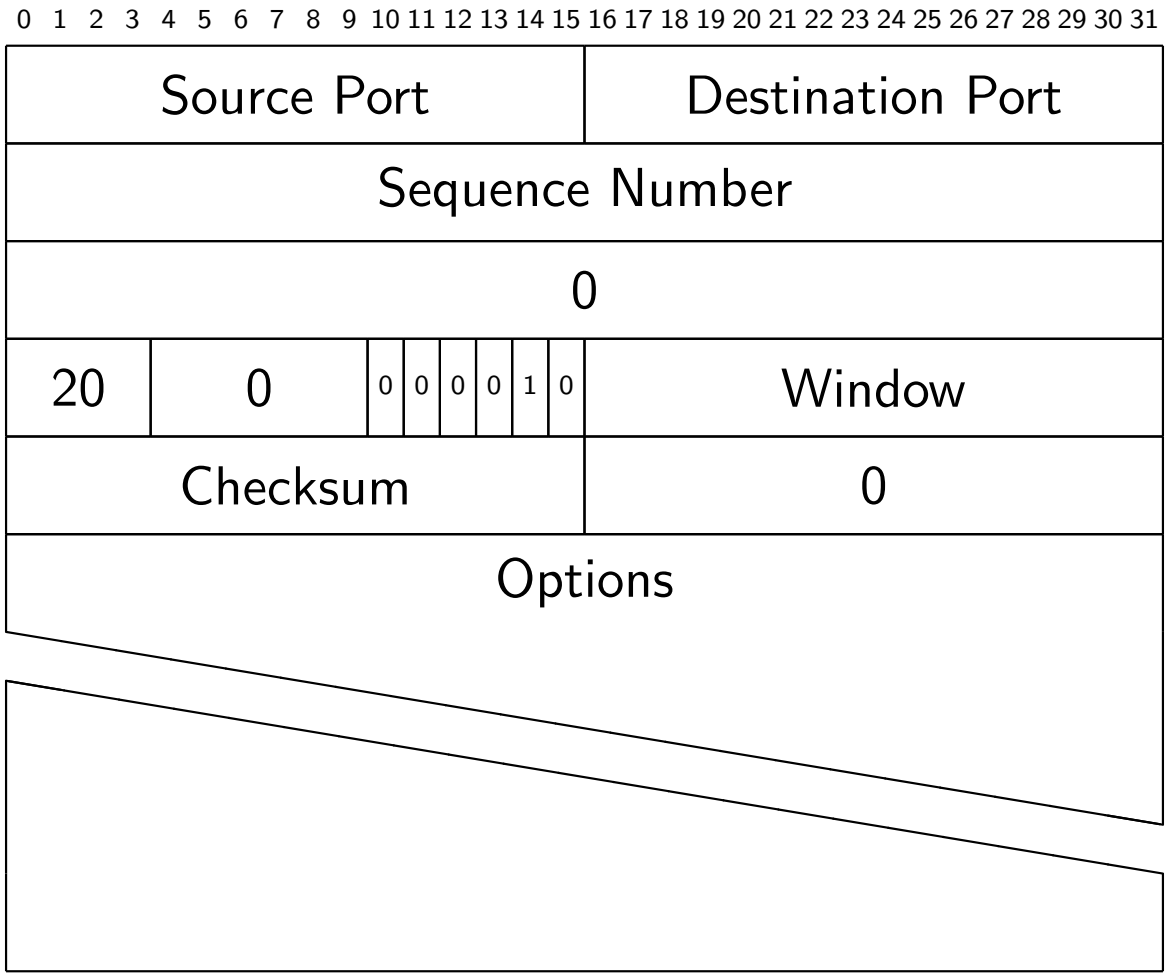
## Hiding Information within TCP



# Goal

## Hiding Information within TCP

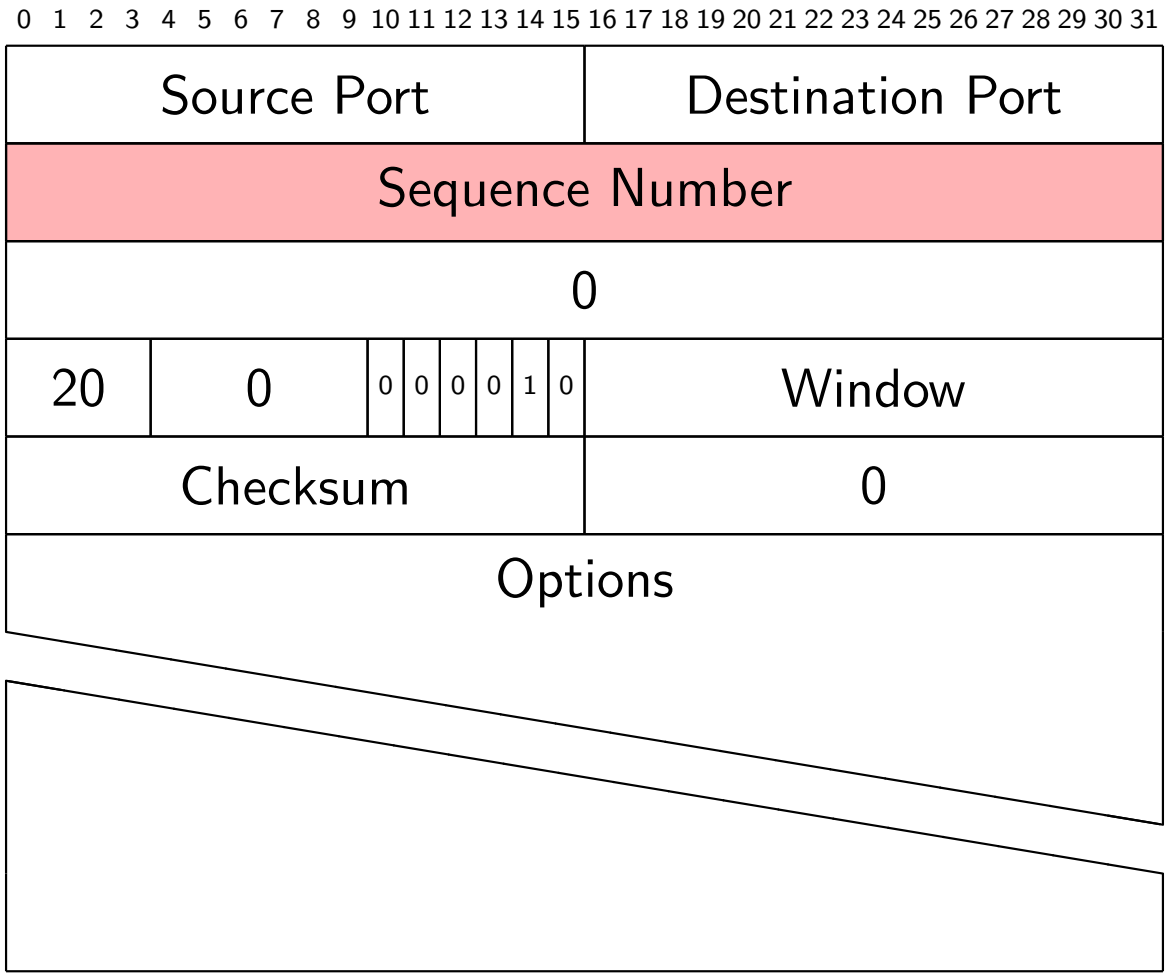
TCP header – reserved values for SYN



# Goal

## Hiding Information within TCP

TCP header – reserved values for SYN



# Related work

## SilentKnock / 2007

- use cryptographic MAC of size 32 bit for authentication
- hide the MAC inside TCP SYN packet (ISN)
- claim that the linux TCP ISN has only 24 bits entropy
  - no longer true (changed between 2.6.31 and 2.6.32)
- use these 24 bits and 8 bits of the timestamp
- need to delay packets for a realistic timestamp
- the TCP timestamp is an optional field
- implemented as server daemon and client side proxy

# Related work

## BridgeSPA / 2011

- similar to SilentKnock
- same TCP header fields
- SHA256-HMAC
- embed a timestamp to avoid replay attacks
- only works if client and server are time synchronized
- replay within a minute is still possible

# Related work

## KnockKnock / 2009

- similar to SilentKnock
- same TCP header fields
- sends two SYN packets
  - one for authentication, one to connect

# Related work

- KnockKnock, BridgeSPA and SilentKnock are all vulnerable to man-in-the-middle attacks

# Design

- initial TCP sequence number (ISN) as authenticator
  - Similar concept to port knocking
- two variants, with and without payload authentication

# Design

## Knock without payload authentication

$$ISN = H_K(IP_{Dest}, Port_{Dest}, TS, 0)$$

- $IP_{Dest}, Port_{Dest}$ represent the destination address
- $TS$ is a timestamp obtained from the (optional) TCP header
- $K$ is a shared secret

# Design

## Knock with payload authentication

$$
\begin{aligned}
ISN &= (A, I) \\
A &= H_K(IP_{Dest}, Port_{Dest}, TS, I) \\
I &= MD5(K \circ Data)
\end{aligned}
$$

- $IP_{Dest}, Port_{Dest}$ represent the destination address
- $TS$ is a timestamp obtained from the (optional) TCP header field
- $K$ is a shared secret
- $Data$ represents the first $n$ bytes of the payload

# Design

## Hashing

- MD5 hash with fixed message length and no pre-processing
- close to what linux normally does
- fast (server needs to compute the hash for each SYN packet)

# Implementation

kernel patch (kernel 3.12)

- generating special sequence numbers
- checking for those sequence numbers
- control via setsockopt()
- IPv4 and IPv6 support

# Usability

- easy to use for every application
- only 1 additional setsockopt() call (2 for data authentication)
- one time password possible (changed via setsockopt())
- possible integration into mainline kernel

# Usage

```
1  //...
2  int sock;
3  struct sockaddr_in addr;
4  u8 secret[64] = "This is my magic ID.";
5  char payload[42];
6
7  //...
8
9  sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
10 setsockopt(sock, SOL_TCP, TCP_STEALTH,
11              secret, sizeof(secret));
12 setsockopt(sock, SOL_TCP, TCP_STEALTH_INTEGRITY,
13         payload, sizeof(payload));
14 connect(sock, &addr, sizeof(addr));
15 write(sock, payload, sizeof(payload));
16
17 //...
```

# Usage

```
1  //...
2  int sock;
3  struct sockaddr_in addr;
4  u8 secret[64] = "This is my magic ID.";
5  int len = 42;
6  //...
7
8  sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
9
10 setsockopt(sock, SOL_TCP, TCP_STEALTH,
11              &secret, sizeof(secret));
12 setsockopt(sock, SOL_TCP,
13         TCP_STEALTH_INTEGRITY_LEN,
14         &len, sizeof(len));
15
16 bind(sock, &addr, sizeof(addr));
17 listen(sock, 10);
18
19 //...
```

# Limitations

- an active man-in-the-middle attacker can detect Knock
- NAT
  - Destination IP and port are part of the hash (DNAT / Load balancers)
  - NAT can change the sequence number
- 32 bits can be brute-forced

# Questions?

- authentication:

$$ISN = H_K(IP_{Dest}, Port_{Dest}, TS, 0)$$

- authentication & integrity checking:

$$\begin{aligned} ISN &= (A, I) \\ A &= H_K(IP_{Dest}, Port_{Dest}, TS, I) \\ I &= MD5(K \circ Data) \end{aligned}$$

- web: `https://gnunet.org/knock`
- contact: {`kirschju,leclaire,grothoff`}`@in.tum.de`

Thanks for your attention!