Network Security (NetSec)

IN2101 – WS 16/17

Prof. Dr.-Ing. Georg Carle

Cornelius Diekmann

Version: October 17, 2016

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Chapter 4: Attacks on TCP

- Recap: TCP
  - TCP Header Format [rfc793]
  - TCP 3-Way Handshake
- TCP SYN Flood Attack
- TCP SYN Cookies
  - SYN Flood Protection with TCP SYN cookies
  - TCP 3-Way Handshake with SYN Cookies
  - SYN Cookies – Advantages
  - SYN Cookies – Disadvantages
- TCP SYN Cookies in the Linux Kernel
- Literature

Recap: TCP

- TCP Header Format [rfc793]
- TCP 3-Way Handshake

TCP SYN Flood Attack

TCP SYN Cookies

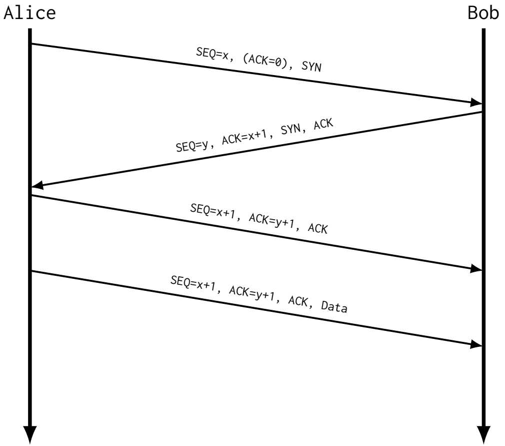
TCP SYN Cookies in the Linux Kernel

Literature

TCP Header Format [rfc793]

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TCP 3-Way Handshake



Basic 3-Way Handshake for Connection Synchronization [rfc793]

TCP 3-Way Handshake

- Can an attacker successfully complete a TCP 3-way handshake?

TCP 3-Way Handshake

- Can an attacker successfully complete a TCP 3-way handshake?
  - Yes!

TCP 3-Way Handshake

- Can an attacker successfully complete a TCP 3-way handshake?
  - Yes!
- Can an attacker with spoofed source address successfully complete a TCP 3-way handshake?

TCP 3-Way Handshake

- Can an attacker successfully complete a TCP 3-way handshake?
    - Yes!
- Can an attacker with spoofed source address successfully complete a TCP 3-way handshake?
    - Yes!

TCP 3-Way Handshake

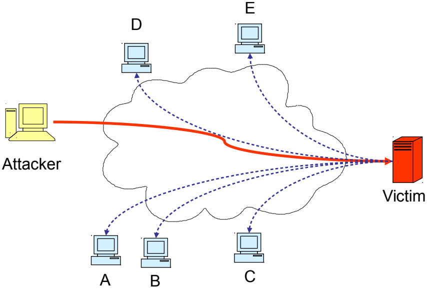
- Can an attacker successfully complete a TCP 3-way handshake?
  - Yes!
- Can an attacker with spoofed source address successfully complete a TCP 3-way handshake?
  - Yes! Recall our default attacker model.

TCP 3-Way Handshake

- Can an attacker successfully complete a TCP 3-way handshake?
  - Yes!
- Can an attacker with spoofed source address successfully complete a TCP 3-way handshake?
  - Yes! Recall our default attacker model.
- Can an attacker with spoofed source address, limited by position such that she does not receive answers to spoofed packets successfully complete a TCP 3-way handshake?

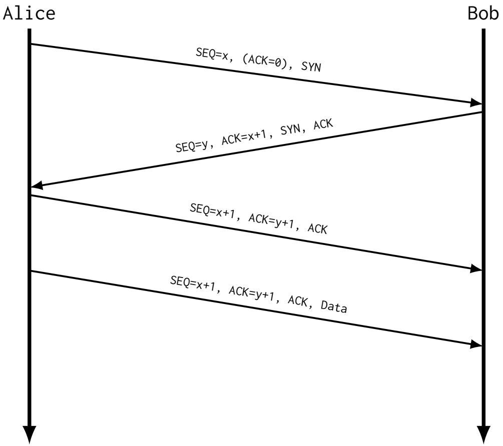
TCP 3-Way Handshake

- Can an attacker successfully complete a TCP 3-way handshake?
  - Yes!
- Can an attacker with spoofed source address successfully complete a TCP 3-way handshake?
  - Yes! Recall our default attacker model.
- Can an attacker with spoofed source address, limited by position such that she does not receive answers to spoofed packets successfully complete a TCP 3-way handshake?
  - No



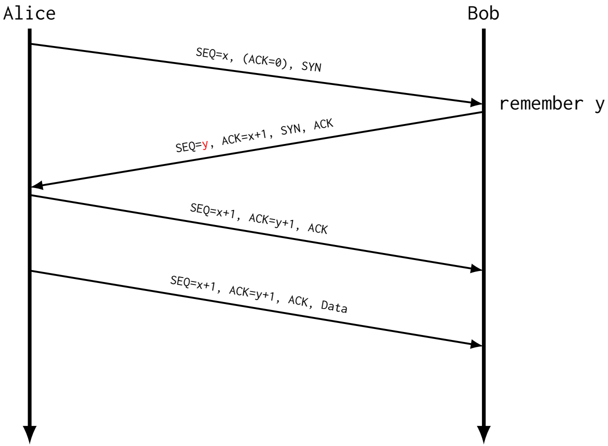
TCP 3-Way Handshake

- Bob needs to track sequence numbers



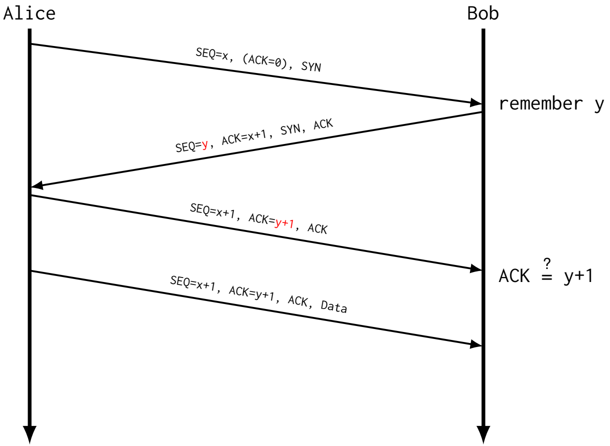
TCP 3-Way Handshake

- Bob needs to track sequence numbers



TCP 3-Way Handshake

- Bob needs to track sequence numbers



Chapter 4: Attacks on TCP

Recap: TCP

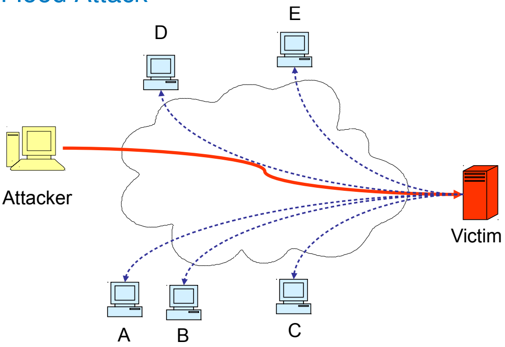
TCP SYN Flood Attack

TCP SYN Cookies

TCP SYN Cookies in the Linux Kernel

Literature

TCP SYN Flood Attack

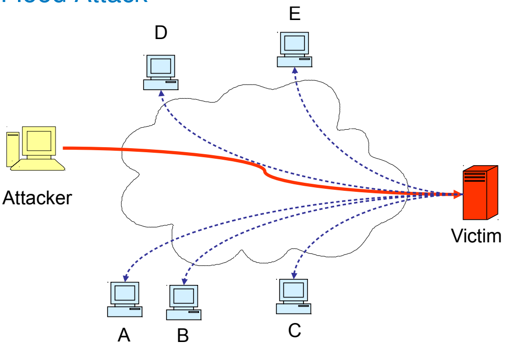


Connection Table
A
B
C
D
E
...

→ TCP SYN packets with forged source addresses (“SYN Flood”)

⇢ TCP SYN ACK packet to assumed initiator (“Backscatter”)

TCP SYN Flood Attack



Connection Table

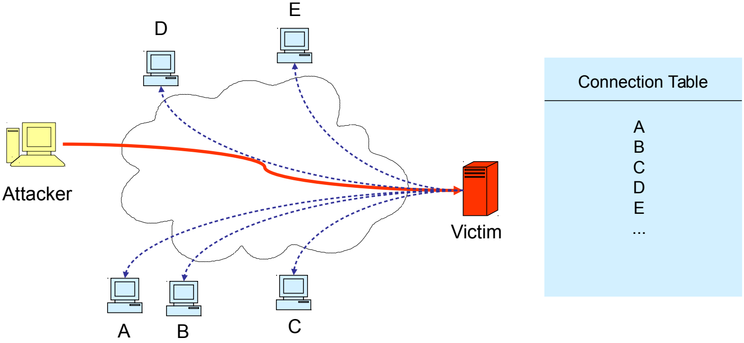
A
B
C
D
E
...

→ TCP SYN packets with forged source addresses (“SYN Flood”)

⇢ TCP SYN ACK packet to assumed initiator (“Backscatter”)

- Bob’s connection table fills up with many half-opened connections.

TCP SYN Flood Attack



Connection Table

A
B
C
D
E
...

⟶ TCP SYN packets with forged source addresses (“SYN Flood”)

⇢ TCP SYN ACK packet to assumed initiator (“Backscatter”)

- Bob’s connection table fills up with many half-opened connections.
- Legitimate users can not establish new TCP connection.

Chapter 4: Attacks on TCP

Recap: TCP

TCP SYN Flood Attack

TCP SYN Cookies

- SYN Flood Protection with TCP SYN cookies
- TCP 3-Way Handshake with SYN Cookies
- SYN Cookies – Advantages
- SYN Cookies – Disadvantages

TCP SYN Cookies in the Linux Kernel

Literature

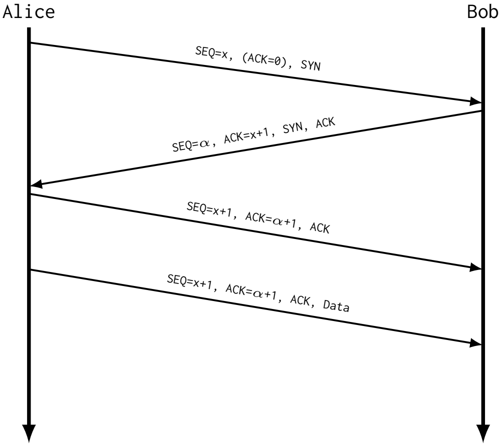
SYN Flood Protection with TCP SYN cookies

- SYN cookie: particular choice of the initial seq number by Bob.
- Bob generates the initial sequence number α such as:
  - $\alpha = h(K, S_{\mathrm{SYN}})$
  - K: a secret key
  - S_{SYN}: source addr of the SYN packet
  - h is a one-way function.
- At arrival of the ACK message, Bob calculates α again.
- Then, he verifies if the ACK number is correct.
- If yes, he assumes that the client has sent a SYN message recently and it is considered as normal behavior.

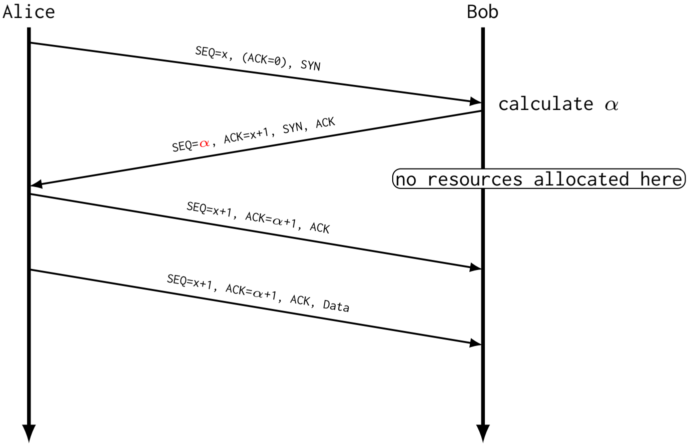
SYN Flood Protection with TCP SYN cookies

- SYN cookie: particular choice of the initial seq number by Bob.
- Bob generates the initial sequence number α such as:
  - $\alpha = h(K, S_{\text{SYN}})$
  - K: a secret key
  - S_{SYN}: source addr of the SYN packet
  - h is a one-way function.
    - Usually, h is a cryptographic hash function (implies one-way function)
- At arrival of the ACK message, Bob calculates α again.
- Then, he verifies if the ACK number is correct.
- If yes, he assumes that the client has sent a SYN message recently and it is considered as normal behavior.

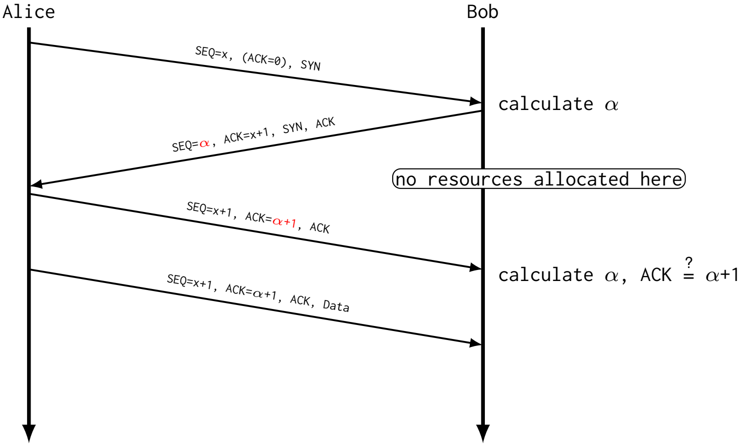
TCP 3-Way Handshake with SYN Cookies



TCP 3-Way Handshake with SYN Cookies


Alice
Bob
SEQ=x, (ACK=0), SYN
calculate α
SEQ=α, ACK=x+1, SYN, ACK
no resources allocated here
SEQ=x+1, ACK=α+1, ACK
SEQ=x+1, ACK=α+1, ACK, Data

TCP 3-Way Handshake with SYN Cookies



Alice
Bob
SEQ=x, (ACK=0), SYN
calculate α
SEQ=α, ACK=x+1, SYN, ACK
no resources allocated here
SEQ=x+1, ACK=α+1, ACK
calculate α, ACK $\overset{?}{=}$ α+1
SEQ=x+1, ACK=α+1, ACK, Data

SYN Cookies – Advantages

- Server does not need to allocate resources after the first SYN packet.
- Client does not need to be aware that the server is using SYN cookies.
- SYN cookies don’t requires changes in the specification of the TCP protocol.

SYN Cookies – Disadvantages

- Calculating α may be CPU consuming.
    - Moved the vulnerability from memory overload to CPU overload.
- TCP options cannot be negotiated (e.g. large window option)
    - Use only when an attack is assumed.
- ACK/SEQ number are only 32 Bit long.
- Efficient implementation (fast but insecure crypto) may be vulnerable to cryptoanalysis after receiving a sufficient number of cookies.
    - The secret needs to be changed regularly, e.g. by including a timestamp.

Chapter 4: Attacks on TCP

Recap: TCP

TCP SYN Flood Attack

TCP SYN Cookies

TCP SYN Cookies in the Linux Kernel

Literature

TCP SYN Cookies in the Linux Kernel

Linux/net/ipv4/syncookies.c

http://lxr.free-electrons.com/source/net/ipv4/syncookies.c?v=4.2

- Calculating cookie helper

```
static u32 syncookie_secret[2][16-4+SHA_DIGEST_WORDS] __read_mostly;

static DEFINE_PER_CPU(__u32 [16 + 5 + SHA_WORKSPACE_WORDS],
                      ipv4_cookie_scratch);

static u32 cookie_hash(__be32 saddr, __be32 daddr, __be16 sport, __be16 dport,
                       u32 count, int c)
{
        __u32 *tmp;

        net_get_random_once(syncookie_secret, sizeof(syncookie_secret));

        tmp  = this_cpu_ptr(ipv4_cookie_scratch);
        memcpy(tmp + 4, syncookie_secret[c], sizeof(syncookie_secret[c]));
        tmp[0] = (__force u32)saddr;
        tmp[1] = (__force u32)daddr;
        tmp[2] = ((__force u32)sport << 16) + (__force u32)dport;
        tmp[3] = count;
        sha_transform(tmp + 16, (__u8 *)tmp, tmp + 16 + 5);

        return tmp[17];
}
```

TCP SYN Cookies in the Linux Kernel

- Calculating cookie (hacking in additional information)

```
static __u32 secure_tcp_syn_cookie(__be32 saddr, __be32 daddr, __be16 sport,
                                   __be16 dport, __u32 sseq, __u32 data)
{
        /*
         * Compute the secure sequence number.
         * The output should be:
         *   HASH(sec1,saddr,sport,daddr,dport,sec1) + sseq + (count * 2^24)
         *      + (HASH(sec2,saddr,sport,daddr,dport,count,sec2) % 2^24).
         * Where sseq is their sequence number and count increases every
         * minute by 1.
         * As an extra hack, we add a small "data" value that encodes the
         * MSS into the second hash value.
         */
        u32 count = tcp_cookie_time();
        return (cookie_hash(saddr, daddr, sport, dport, 0, 0) +
                sseq + (count << COOKIEBITS) +
                ((cookie_hash(saddr, daddr, sport, dport, count, 1) + data)
                 & COOKIEMASK));
}
```

TCP SYN Cookies in the Linux Kernel

- Verifying received cookie

```
/*
 * This retrieves the small ”data” value from the syncookie.
 * If the syncookie is bad, the data returned will be out of
 * range.  This must be checked by the caller.
 *
 * The count value used to generate the cookie must be less than
 * MAX_SYNCOOKIE_AGE minutes in the past.
 * The return value (__u32)-1 if this test fails.
 */
static __u32 check_tcp_syn_cookie(__u32 cookie, __be32 saddr, __be32 daddr,
                                  __be16 sport, __be16 dport, __u32 sseq)
{
        u32 diff, count = tcp_cookie_time();

        /* Strip away the layers from the cookie */
        cookie -= cookie_hash(saddr, daddr, sport, dport, 0, 0) + sseq;

        /* Cookie is now reduced to (count * 2^24) ^ (hash % 2^24) */
        diff = (count - (cookie >> COOKIEBITS)) & ((__u32) -1 >> COOKIEBITS);
        if (diff >= MAX_SYNCOOKIE_AGE)
                return (__u32)-1;

        return (cookie -
                cookie_hash(saddr, daddr, sport, dport, count - diff, 1))
                & COOKIEMASK;   /* Leaving the data behind */
}
```

SYN Cookies Disadvantages Revisited

- Calculating α may be CPU consuming?

- TCP options cannot be negotiated?

- Efficient implementation vulnerable to cryptoanalysis?

SYN Cookies Disadvantages Revisited

- Calculating α may be CPU consuming?
  - Highly efficient. CPU-local, barely any cache misses.
- TCP options cannot be negotiated?
- Efficient implementation vulnerable to cryptoanalysis?

SYN Cookies Disadvantages Revisited

- Calculating α may be CPU consuming?
  - Highly efficient. CPU-local, barely any cache misses.
- TCP options cannot be negotiated?
  - Window size (here MSS) up a certain value hacked into cookie.
- Efficient implementation vulnerable to cryptoanalysis?

SYN Cookies Disadvantages Revisited

- Calculating α may be CPU consuming?
  - Highly efficient. CPU-local, barely any cache misses.
- TCP options cannot be negotiated?
  - Window size (here MSS) up a certain value hacked into cookie.
  - SYN Cookies are only dynamically enabled if `net.ipv4.tcp_max_syn_backlog` is exceeded.
- Efficient implementation vulnerable to cryptoanalysis?

SYN Cookies Disadvantages Revisited

- Calculating α may be CPU consuming?
  - Highly efficient. CPU-local, barely any cache misses.
- TCP options cannot be negotiated?
  - Window size (here MSS) up a certain value hacked into cookie.
  - SYN Cookies are only dynamically enabled if `net.ipv4.tcp_max_syn_backlog` is exceeded.
- Efficient implementation vulnerable to cryptoanalysis?
  - SHA is a proper one-way function
    (but considered broken as cryptographic hash function)
  - A counter is updated every minute.

Chapter 4: Attacks on TCP

Recap: TCP

TCP SYN Flood Attack

TCP SYN Cookies

TCP SYN Cookies in the Linux Kernel

Literature

Literature

- Patrick McManus, *Improving syncookies*, LWN, April 9, 2008, `http://lwn.net/Articles/277146/`
- Linux Kernel Sources, `Linux/net/ipv4/syncookies.c`

Literature

- Patrick McManus, *Improving syncookies*, LWN, April 9, 2008, `http://lwn.net/Articles/277146/` ⇐ recommended
- Linux Kernel Sources, `Linux/net/ipv4/syncookies.c`