Collaborative TCP Sequence Number Inference Attack - How to Crack Sequence Number Under A Second

Zhiyun Qian - University of Michigan
Z. Morley Mao - University of Michigan
Yinglian Xie - Microsoft Research

Presented By:
Narayanan Seshan
Discussion Topics

- Goals
- TCP Sequence Number Inference Attack
- Inference using Sequence-Number-Dependent Counters in Linux
- Inference using Sequence-Number-Dependent Counters in BSD/MacOS
- Design & Implementation of these TCP Attacks
- Attack Impact From Case Studies
- Conclusion
Goals

• Use of “sequence-number-dependent” packet counters in OpenBSD/MacOS/Linux/Android

• Collaboration with Malware to infer TCP Sequence Numbers

• Inference carried out without Firewall middlebox

• Inference completed in 4-5 RTT’s
TCP Sequence Number Inference Attack

Figure 1: Threat model
- Attacker needs 2 things - 4 tuples + sequence number
- On device malware - “netstat”
- Attacker -> probe packets using 4 tuples + prediction of sequence number
- TCP Stack increments counters - incorrect sequence number
TCP Incoming Packet Validation

Figure 2: Incoming packet validation logic
- Error Check: MD5, Timestamp, Packet Length, Checksum

- Sequence Number: seq_end >= X, seq <= X + rcv_win

- ACK: Valid ACK -> [Y, Y+remaining_bytes]

- Payload: If payload = 0, drop

- Retransmission: If seq_end <= X, drop

- In window vs Out of window 0 payload packets
Inference using Sequence-Number-Dependent Counter in Linux

- Binary Search ~ 32*RTT, ie, 32 packets.
- N-ary Search ~ eliminates (N-1)/N of search space.
- Similar approach for Server-Side & Client-Side Inference

```c
if (TCP_SKB_CB(skb)->end_seq != TCP_SKB_CB(skb)->seq
&& before(TCP_SKB_CB(skb)->seq, tp->rcv_nxt)) {
    NET_INC_STATS_BH(sock_net(sk),
                     LINUX_MIB_DELAYEDACKLOST);
}
```

Figure 3: tcp_send_dupack() source code snippet in Linux
Inference using Sequence-Number-Dependent Counter in BSD/Mac OS

- `rcvdupack & rcvdupbyte` : N-ary search
- `rcvpackafterwin & rcvbyteafterwin` : $\text{seq} > X + \text{rcv\_win}$
- `rcvooopack & rcvoobyte` : $\text{seq} > X$ && $\text{seq} < X + \text{rcv\_win}$
- `rcvdupack & rcvacktoomuch (Client Side)` : $\text{seq} \leq \text{seq\_unack}$, $\text{ack} > \text{seq}$
Design & Implementation of TCP Attacks

- Client-Side & Server-Side TCP Injection
- Passive TCP Hijacking
- Active TCP Hijacking
Client-Side & Server-Side TCP Injection

- Server \( \sim > 100\text{ms} \) to process a request
- Single TCP connection - reused for multiple HTTP requests
- Injection of malicious Javascripts
Figure 9: Passive TCP hijacking sequence
Figure 10: Active TCP hijacking sequence
Attack Impact From Case Studies

- Facebook Javascript Injection - Client Side Injection
- Phishing Facebook Login Page - Passive TCP Hijack
- Command Injection on Windows Live Messenger - Server Side Injection
- Restricted Facebook Login Page Hijack - Active TCP Hijack
Conclusions

- Systems today have too much shared state
- OS Statistics leak critical information
- Always use SSL/TLS
- Remove unnecessary global state or restrict access
- Improve isolation among resources
Q & A ??
Thank You!!!