Protecting Mobile Devices From TCP Flooding Attacks

Yogesh Swami[%] and Hannes Tschofenig[*]

[%] Nokia Research Center, Palo Alto, CA, USA.

[*] Siemens Corporate Technology, Munich, DE.

Motivation

- Anatomy of a DoS Attack: Identify a resource constraint, then find a means to exhaust it!
- TCP SYN flooding attack is well understood for wire-line networks
  - Memory is the resource constraint
  - SYN Cookies or personal firewalls work reasonably well
- Wireless networks have new resource constraints
  - Battery life -- Depends upon “radio off” time periods during idle periods
  - Wireless Spectrum -- Cellular network spectrum is expensive to end user
- SYN Cookies and other solutions co-located on the mobile device cannot defend against these kinds of attack
  - Waking up the device by sending random SYN packets will exhaust the battery, even if the SYN packet is ultimately dropped by personal firewall on mobile
  - A SYN flood will take radio bandwidth someone needs to pay for it in cellular networks.

NOKIA
Connecting People

Problem Statement

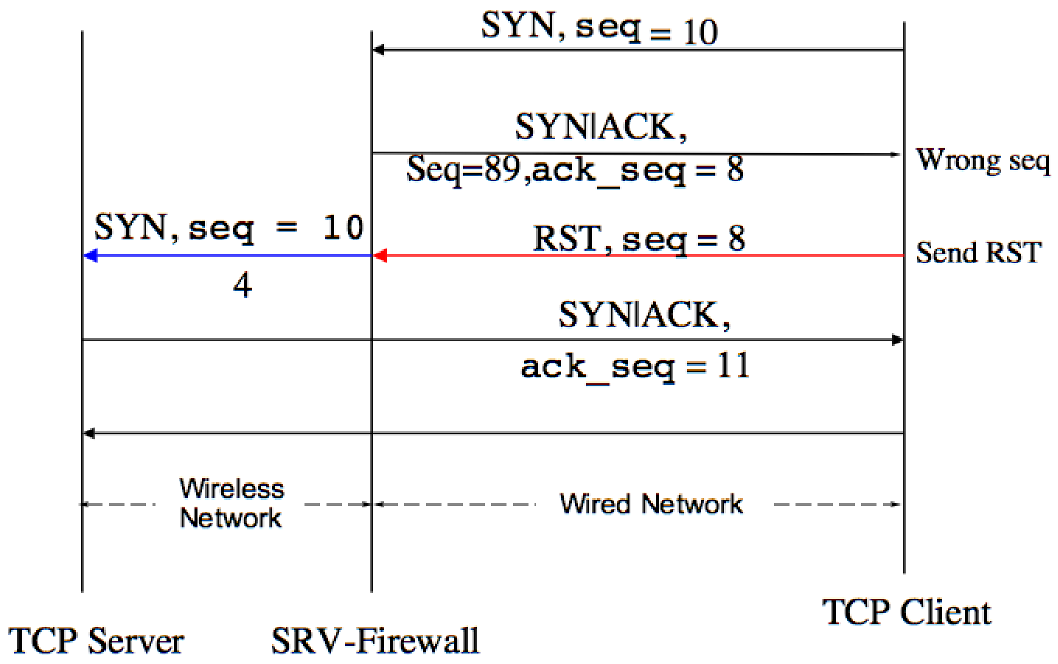
- Firewalls deployed at the wireless-wired network boundary a popular choice for defense
  - Firewall blocks almost all incoming unsolicited packets
  - Assumption is that mobile devices cannot run as servers/peer nodes in P2P networks (no longer true with Apache running on Symbian phones)
  - Dedicated pin-holes for server ports doesn't prevent against battery exhaustion and spectrum waste
  - Some P2P applications (e.g., Skype) have mechanism built into protocol to traverse such firewalls, but not all of them do.
- Our Goal: Can we design a system that
  - Woks with any P2P application or any server application running on mobile phone
  - Does not require changes to existing protocols (e.g., IP or TCP tweaking out of picture)
  - Does not require massive change to the Internet routing infrastructure (e.g., no firmware/IOS upgrades)
  - Treasure end to end nature of TCP

Prior Art

- SYN Cookies -- of course!
- Mechanisms to determine the attack source (e.g., IP Traceback, IP pushback).
  - Requires changes to internet infrastructure -- beyond our solution realm
  - Might help in detection, but not always in prevention
- Intrusion and Misuse Detection Schemes (e.g., Snort, Bro, DOMINO)
  - Work mainly in detection, not in prevention
- Many other proposals on architectural changes to the Internet (e.g., i3, Hi3)
  - Possibly clean, long-term solutions, but cannot be immediately deployed
- Split TCP solutions where firewall splits the connection across wired and wireless networks
  - Requires massive state in the firewall and creates performance bottlenecks
  - Breaks the end-to-end semantics of TCP

Server Friendly Firewalls

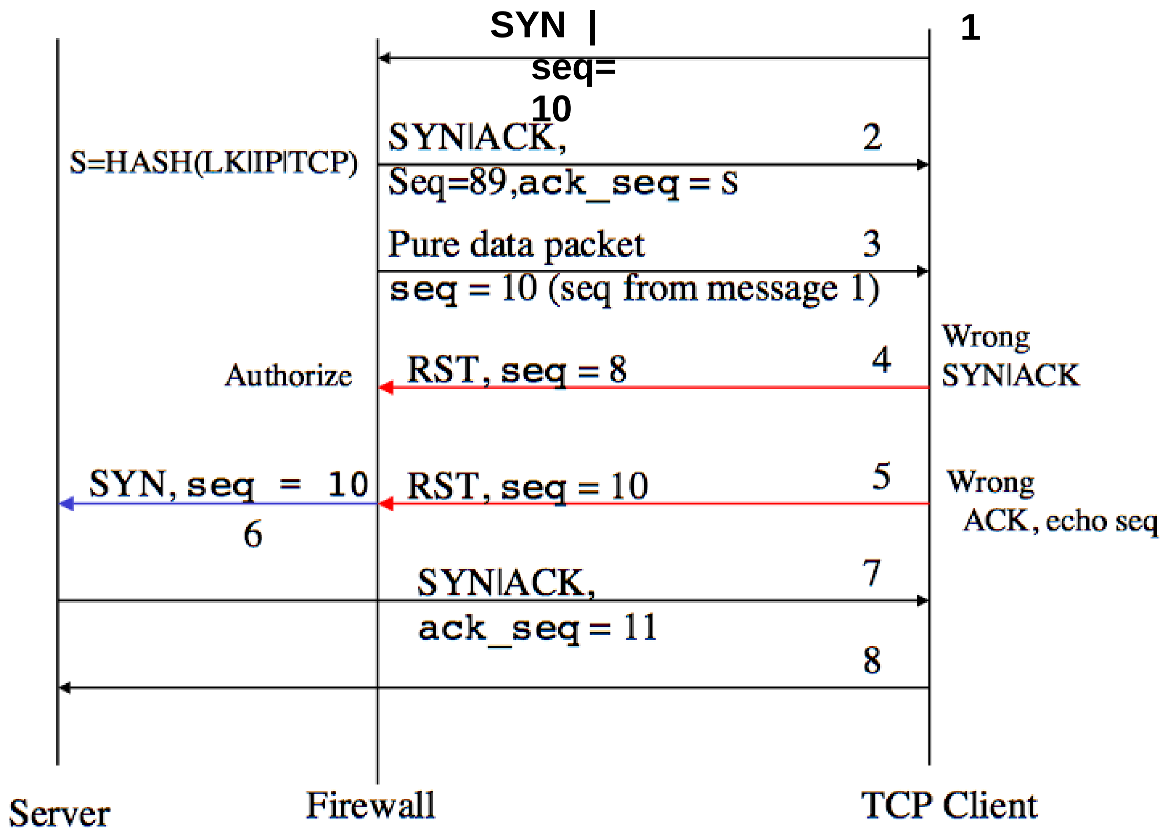
- Goal is to exploit the TCP connection setup procedure
  - TCP sends RST messages to packets it did not expect to receive (I.e., packets that are outside the window)
- Basic Idea
  - Let the firewall send a wrong sequence number to the incoming SYN packet
  - If the SYN packet was not spoofed, the client will respond with an RST message. Use this RST response as a return routability test. If no RST received ==> the SYN was spoofed
  - Firewall reconstructs the SYN packet and forwards it to the mobile server application
  - Beyond this point, connection setup proceeds normally.



NOKIA
Connecting People

Stateless Server Friendly Firewall

- Previous approach works but requires firewall to keep state about the TCP connection so that it can reconstruct the SYN packet
  - Opens firewall itself for DoS attacks
- Stateless Server Friendly firewalls send two “wrong packet”
  - Send SYN|ACK with ack sequence as a cookie
  - Send a Data packet with right sequence number as the packet sequence
  - First RST returns the cookie information (verify based on HMAC computation)
  - Second RST returns the sequence number that was sent in the data packet



Implementation Status

- Implemented as a standalone user-space application using ‘netmon’ on Linux.
- Uses libpcap for packet capture
- Used Komodia PacketCrafter tool for attack traffic generation running in parallel with legitimate application to very usage.
- More details, for example about timers, in paper

Conclusion, Future Work, and Limitations

- Limitations
  - Essentially a hack, driven by short term needs
  - Connection Setup time increased by 1 RTT
  - Sending wrong packet in response to SYN requests could be exploited for reflection attacks
- Future Work
  - Integrate netmon in Linux iptables/netfilter code
- Conclusion
  - Works well for TCP; could be extended for other connection oriented protocols which have an RST equivalent of TCP (e.g., SCTP, etc.)
  - Doesn’t break the end to end semantics of TCP
  - Require minor changes to existing firewalls