Use of the IPv6 Flow Label as a TCP Nonce

Steven Blake
sblake@extremenetworks.com

IETF 73
November 2008

The Problem

- TCP (and other transports) are vulnerable to blind spoofed packet injection attacks from off-path hosts.
  - Attackers can spoof SYN, ACK, DATA, and RST segments, resulting in connection reset, thruput reduction, or data corruption.
  - Attackers can also spoof ICMP error messages
- Attacker has to be able to correctly guess <IPSA, SRCPORT, IPDA, DSTPORT>, plus an in-receive window sequence number.
- Vulnerability grows quadratically with attacker's access link speed.
- Long-running TCP sessions are most vulnerable (e.g., BGP).

Mitigations (1)

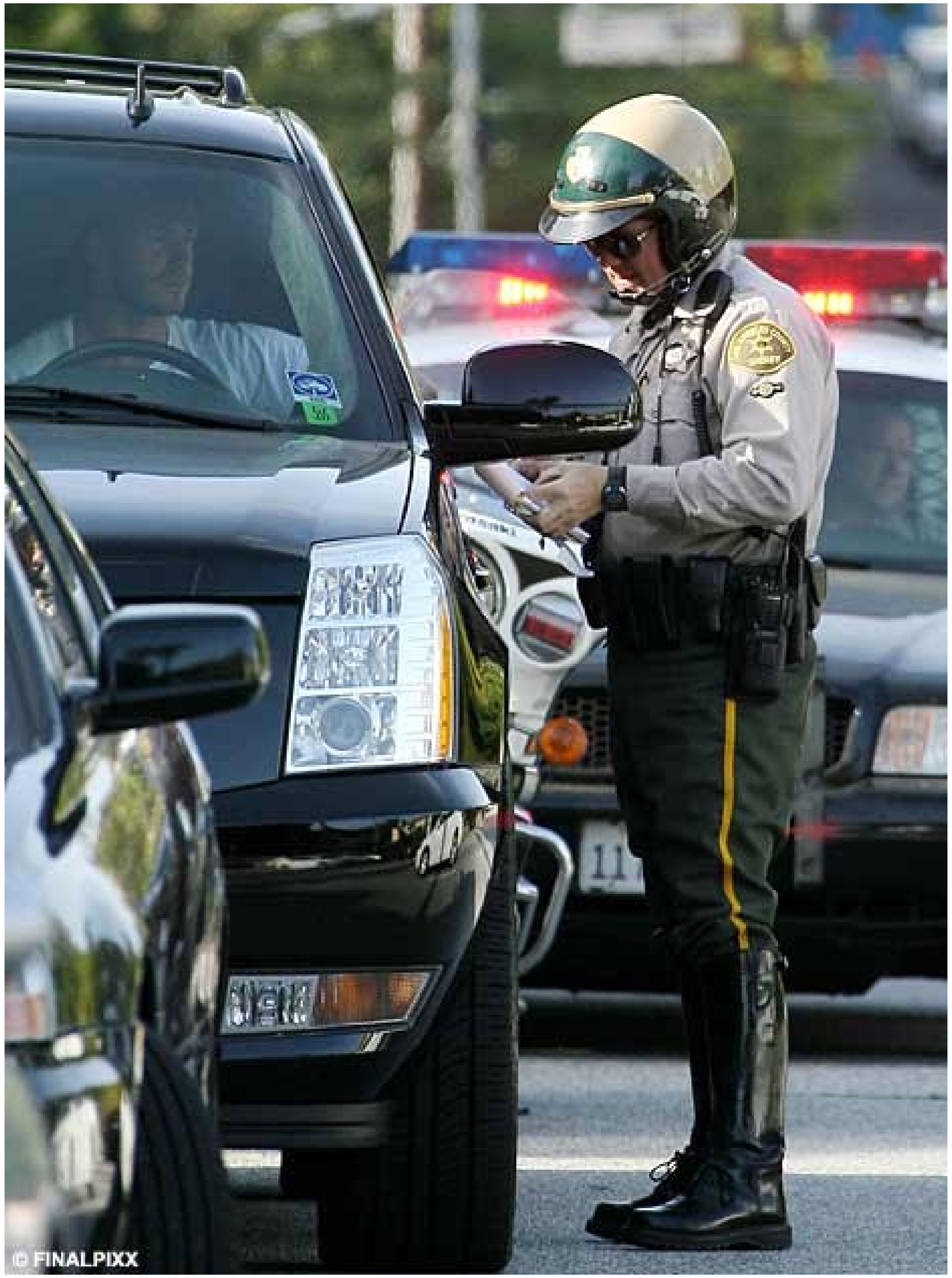
- RFC 4953 surveys the mitigation options.
- Network Ingress Filtering [RFC 2827, RFC 3704]
  - Not (yet) universally deployed.
  - Doesn't protect against ICMP spoofing.
  - With large BOTNETs, more likely that an attack can be launched from a network close to the victim.
- Cryptographic Authentication
  - IPsec AH
  - TCP-MD5 option
  - TCP Authentication Option
  - Also protects against (some) on-path attacks.
  - Computationally expensive.
  - Key management overhead.
  - SHOULD be used in high-threat environments.

Mitigations (2)

- Obfuscation techniques:
  - Source port randomization: draft-ietf-tsvwg-port-randomization
  - Initial sequence number randomization: draft-ietf-tcpm-tcpsecure
  - Randomization increases the work factor for an attacker to successfully spoof a valid TCP packet.
  - Both schemes in combination introduce ~ 32 bits of entropy.
  - A host on a high-speed link may be able to spoof a connection in less than an hour.

IPv6 Flow Label

- IPv6 introduced the concept of an interworking-layer flow.
  - FlowID: 20 bit field in IPv6 header
  - RFC 1883 defined a flow as a sequence of packets from a source to a particular (set of) destination(s), which require special handling by routers.
  - Flows are identified by <IPSA, FlowID>, where FlowID is <u>non-zero</u>.
- RFC 3697 redefined flow identity as <IPSA, IPDA, FlowID>.
- We want to utilize the FlowID as a per-connection nonce, to increase the work factor of spoofing attacks.
  - Randomization of FlowID, SRCPORT, and ISN increases entropy to > 51 bits.



Warning!

Layering Violation

Existing Flow Label Rules

- Source MUST keep FlowID constant for the duration of a flow.
- FlowID MUST remain unchanged end-to-end.
- Source SHOULD assign each transport connection or application datastream to a unique flow.
- Source SHOULD select an unused FlowID if not explicitly selected by an application.
- FlowIDs MUST be unique at a source host at any instant in time.
- Source MUST NOT reuse the same FlowID to the same destination for a quarantine period after flow termination (>= 120 seconds).

Flow Label Nonce Use

- Each host assigns each transport connection to a flow.
- Host selects an outgoing FlowID per-connection.
- Host records the incoming FlowID from the peer and checks it against every received packet in the connection.
- Host silently discards packets with invalid FlowIDs.
- Excessive FlowID errors SHOULD be logged.
- Scheme is incrementally deployable:
  - If a destination does not check FlowID, nothing broken (*but attack resistance not improved*).
  - If source does not support this scheme, FlowID = 0. Destination check will not fail.
- MUST NOT rely on this mechanism in high-threat environments.

Additional Flow Label Rules

- Host MUST assign each transport connection to a new flow.
- Host MUST be able to select unused FlowIDs when the application does not request a specific value.
- FlowID MUST be practically unguessable (e.g., selected by a RFC 4086-compliant RNG).
- Host MUST clean-up flow state when cleaning up transport state.
- Quarantine period must be no less than the duration where transport state may linger (e.g., TIME_WAIT state).

TCP Operation (1)

- Client TCP stack selects OUTGOING_FLOW_ID at connection creation.
  - Compute at same time as SRCPORT and ISN.
  - Save OUTGOING_FLOW_ID in connection TCB.
- Client sends SYN with its OUTGOING_FLOW_ID.
- Server records SYN packet's FlowID as INCOMING_FLOW_ID in connection TCB *(ignoring SYN cache/cookie case here)*.
- Server selects OUTGOING_FLOW_ID (same procedure as client).
  - Value can (but does not have to) equal INCOMING_FLOW_ID.
- Server sends SYN-ACK with its OUTGOING_FLOW_ID.
- Client records SYN_ACK packet's FlowID as INCOMING_FLOW_ID in connection TCB.

TCP Operation (2)

- Both ends always send packets with their OUTGOING_FLOW_ID.
- Both ends always check received packet's INCOMING_FLOW_ID.
- If the INCOMING_FLOW_ID check fails, silently discard the packet.
- When the connection closes, FlowID cannot be reused to the same destination for MAX(2 x MSL, 120 sec).

Applicability to UDP

- Also useful for UDP, since it only has source port randomization as an obfuscation technique.
- Ex/ use FlowID as nonce in DNS queries to protect against DNS cache poisoning attacks.
  - DNS server sends the reply with the same FlowID as used in the query.
  - Client verifies the received FlowID.
- Text in draft for UDP-Lite is probably wrong: should use FlowID as with UDP.
- Issues:
  - UDP/IP stack does not have the equivalent of a TCP connection TCB (except for connected sockets).
  - Ergo, setting/checking of FlowID needs to happen in the application (above the socket API).
  - No standard sockets API for setting/retrieving FlowID.

Further Work

- Examine applicability to SCTP, DCCP, and RTP (over UDP or DCCP).
- Prototype in Linux.