IPv4 fragmentation is worse than we thought

draft-mathis-frag-harmful-00.txt
3-Aug-2004

Matt Mathis <mathis@psc.edu>
John Heffner <jheffner@psc.edu>
Ben Chandler <bchandle@psc.edu>
Slides: http://www.psc.edu/~mathis/papers/frag200408

Observation

- IPv4 fragmentation is not appropriate for modern data rates
  - The IP ID field is only 16 bits - 65536 packets
  - Trivial to wrap IP ID within fragment lifetime
    - Less than 1 second at 1 Gb/s
- This creates the opportunity for missassociated fragments
  - The TCP/UDP checksums are not sufficient prevent delivery

The failure

- Assume a single flow
  - Fast enough to wrap IP ID within fragment lifetime
  - 100 Mb/s @ 1500+ bytes is 8 kpps
  - 8 seconds to wrap
- Loose one low fragment
  - High fragment remains in reassembly queue
  - Transport protocol has to retransmit segment
- When the IP ID wraps
  - New low fragment missassociated with old high fragment
    - Delivered to TCP/UDP/SCTP, etc
  - New high fragment remains in reassembly queue
    - Self sustaining loop hammers on the checksum!
- Behavior on lost high fragment depends on reassembly code
  - Details not specified, but most implementations do it right

Checksum strength

Rely on TCP/UDP checksum to toss corrupted data

- Checksums are only 16 bits
  - Random data - 1 in 65536 false pass
  - Real data is not random
- Pathological data - always false pass
- Note that the data resembles
  - TCP using 1500 byte MTU
  - IP in IP tunnels that ignore DF

The experiment

- Precomputed file for UDP transfer
  - 1524 byte datagrams (1468 + 56)
    - Packets are 1468+32 and 56+20 Bytes (SABLE)
    - Designed to resemble DF ignorant encapsulation
  - Wrap IP ID every 65536 datagrams
    - About 100 MBytes
  - Precompute fragment boundaries to label data
    - fragment number + random data
    - md5sum of the rest of fragment
    - optionally construct pathological data - zero checksum
- Transfer "4 wraps" at 90 Mb/s - 36 seconds elapsed time
  - Second stream burst 1024 packets in 1 second, to cause losses near start

The result

- 250 runs of random data
  - 100 GBytes total data
  - 41k UDP checksums errors
  - 1 corrupted file
- Pathological data
  - No checksum errors
  - Observed error offsets are periodic

Failure at low rate

- Assume busy server
  - Fast enough to wrap IP ID within fragment lifetimes
  - Many slow clients receiving fragmented data
    - e.g. Due to tunnels near the client
- For each lost low fragment
  - Every new low fragment
    - Match IP ID 1 in 1^16
    - Match checksum 1 in 1^16
  - Or 1 in 2^32 chance of delivering corrupted data

Beware that this is summed across all losses on all fragmented flows from all busy servers