A Method to Compress and Anonymize Packet Traces

Markus Peuhkuri

2001-11-02
Problems in packet capture?

- Data volume
  - pre-filtering and processing on capture card
  - persistent storage problem

- Data privacy
  - packets include sensitive data in
    * header
    * payload
  - TLS, SSH and IPSec helps for payload
Packet capture, why to do it?

• Measure sum effect of multiple
  – users
  – applications
  – operating systems
  – protocols
  – hardware

• with one (or a few) device(s)

• to provide data for
  – analysis
  – simulation
  – models
Compression by flows

- Better compression rate if you utilize structure of data than if data is “just bits”
  - RFC1144, RFC2507

- Data in flows (5-tuple)
  - TCP sequence, ack numbers proceed, possibly same size
  - UDP possibly same size

1. keep track of every active flow (large id space)
2. compare to previous packet
3. short codes for common cases
What data not to include

- IP identification + fragment word
  - changes randomly
  - for most studies no-use
  - adds 32 or 0/24/40 bits for each packet

- checksums
  - no use afterwards, just check if ok (if possible)

- length and header length fields implicit

- TTL field and TOS/DS byte should be constant in a flow
  $\Rightarrow$ record changes
What is sensitive

**Person identification** identifies communication parties to a single person (or one’s family)
⇒ IP address

**Application identifier** TCP/UDP port numbers – what applications are used

**Payload**

If you open a protected message, or learn the content of a message or learn that a message is sent or received.
⇒ *violation of communication secret*
⇒ a fine or maximum 1 years, 3 if special device is used
How to keep address secret

secret key → MD5 → Blowfish key, 128 bit

IP address → 32 bit of key → ECB → encrypted value

replacement address

hash table index

data structure
Steps

Initiate encryption (blowfish), then for each IP address

1. Encrypt

2. Check if in hash, if not then
   (a) insert into hash
   (b) write out record to stream

3. Replace real IP with 8 bits of clear and 24 bits from encrypted
   ⇒ codeIP

On decoding (off-line) each time encryption record is found

1. Check if known mapping secret ⇒ anonIP, if not then
   (a) pick random unused IP from that network
   (b) store (secret,anonIP) to (persistent) database
   (c) maintain hash table of codeIP ⇒ (secret,anonIP) mapping

Replace codeIP with anonIP in headers
Possible disclosure

Discloser

capture equipment
# Performance

<table>
<thead>
<tr>
<th>Compression</th>
<th>Size [MiB]</th>
<th>Time [s]</th>
<th>Pkts/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>4,886</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>gzip</td>
<td>2,218</td>
<td>5,108</td>
<td>9,120</td>
</tr>
<tr>
<td>anon+flow</td>
<td>770</td>
<td>1,374</td>
<td>33,906</td>
</tr>
<tr>
<td>anon+flow+gzip</td>
<td>318</td>
<td>2,023</td>
<td>23,028</td>
</tr>
<tr>
<td>plain</td>
<td>1759</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Conclusions

- Compression and desensitization needed

- Utilize flow nature of traffic in compression

- Desensitization works, but vulnerable to *chosen plain-text*
  \[\Rightarrow\] Some control to trace archives needed

- Same real IP maps always to the same anon IP if secret is the same
  \[\Rightarrow\] Possible to relate measurements in multiple locations

- Performance feasible, also memory requirements