Payload Attribution via Hierarchical Bloom Filters

Kulesh Shanmugasundaram, Hervé Brönnimann
Nasir Memon
http://isis.poly.edu/projects/fornet/
Kulesh,

This was a close call, almost fell for it!

- Can we find out who else got this email at Poly?
- And who actually visited the site. (Probably there are lot of other sites as well!)

Bye,
Nasir

------------------------------

Recently there have been a large number of identity theft attempts targeting Citibank customers. In order to safeguard your account, we require that you update your Citibank ATM/Debit card PIN.

This update is requested of you as a precautionary measure against fraud. Please note that we have no particular indications that your details have been compromised in any way.

This process is mandatory, and if not completed within the nearest time your account may be subject to temporary suspension.

To securely update your Citibank ATM/Debit card PIN please go to:


Please note that this update applies to your Citibank ATM/Debit card - which is linked directly to your checking account, not Citibank credit cards.

Thank you for your prompt attention to this matter and thank you for using Citibank!

Regards,

Madeline Walter

Head of Citi® Identity Theft Solutions

Copyright © 2004 Citicorp.
Majority of Home Computers Infected with Spyware

Another Phishing Hole Found in Google
By Michael McCue
October 21, 2004
Google Inc. said on Thursday it fixed a flaw in its site that could allow outside attackers to exploit a similar vulnerability. The flaw, which was discovered and posted to Symantec Corp.'s Bugtraq mailing list on Tuesday, The bulletin demonstrated the ability of hackers using JavaScript to gain access to the user's computer.
Payload Attribution

- The problem:
  - Identify the sources and/or the destinations of a bit-string in a network
  - We may only have an arbitrary portion of payload

- Our contribution:
  - Payload attribution system
    - Currently in alpha-test
  - Not a foolproof system
  - Very useful in most cases
Recently there have been a large number of identity theft attempts targeting Citibank customers. In order to safeguard your account, we require that you update your Citibank ATM/Debit card PIN.

This update is requested of you as a precautionary measure against fraud. Please note that we have no particular indications that your details have been compromised in anyway.

This process is mandatory, and if not completed within the nearest time your account may be subject to temporary suspension.

To securely update your Citibank ATM/Debit card PIN please go to:

https://www.citibank.com/signin/citifin/scripts/login2/update_pin.jsp

Please note that this update applies to your Citibank ATM/Debit card – which is linked directly to your checking account, not Citibank credit cards.

Thank you for your prompt attention to this matter and thank you for using Citibank!

Regards,

Madeline Walter

Head of Citibank Identity Theft Solutions

Copyright © 2004 Citibank.
Packet-loggers:
• Need ~1TB/day
• IS is not going to like this

Hash-packets:
• Storage is better
• Need packets for attribution!
  • But we only have emails/excerpts!

Have Problem. Will Solve.
The Problem: Identify the sources/destinations of a bit-string in a network

Our Solution: Create digests of payload s.t we can:
- Attribute excerpts of payload
- Reduce storage requirements
Bloom Filters

- **Bloom Filter:**
  - Randomized data structure for representing a set in order to support membership queries.
  - **Insert**(*x*):
    - Flip bits $H_1(x)\ldots H_k(x)$ to ‘1’
  - **IsMember**(*y*):
    - If $H_1(y)\ldots H_k(y)$ all ‘1’ “yes” otherwise “no”

Can tradeoff memory (m), compute power (k), and accuracy (FP)

- m – length of bit vector (range of $H(.)$)
- k – number of hashes per element
- n – number of elements in the set

$$FP = \left(1 - (1 - 1/m)^{kn}\right)^k$$
Packet Digests & Bloom Filters

- Snoeren et. al. used it successfully in SPIE for single packet traceback ("Hash-Based IP Traceback")

- Space Efficient:
  - 16-bits per packet (m/n=16) and 8 hashes (k=8)
    - false positive (FP) = $5.74 \times 10^{-4}$
    - No false negatives!

- However, we don’t have packets.
  - We only have some excerpt of payload
  - Don’t know where the excerpt was aligned in the packet

- Extend Bloom Filters to support excerpt/substring matching
Block-based Bloom Filter

\[ P = (p^0 \mid 0) (p^1 \mid s) (p^2 \mid 2s) \ldots \ldots \ldots (p^{(n/s)} \mid n/s) \]

*Create s-byte blocks of payload*

*Append blocks’ offset (in payload)*

*Insert each block into a Bloom Filter*
create s-byte blocks of query string

Try all possible offsets

“q^0q^1q^2” was seen in a payload at offset ‘s’
**P1 =**

| A | B | R | A | C | A |

**P2 =**

| C | D | A | B | R | A |

**BBF =**

| (A|0) | (B|s) | (R|2s) | (A|3s) | (C|4s) | (A|5s) |
|-----|------|-------|-------|-------|-------|
| (C|0) | (D|s) | (A|2s) | (B|3s) | (R|4s) | (A|5s) |

**“Offset Collisions”**

| (A|0) | (B|s) | (R|2s) | (A|3s) | (C|4s) | (A|5s) |
|-----|------|-------|-------|-------|-------|
| (C|0) | (D|s) | (A|2s) | (B|3s) | (R|4s) | (A|5s) |

For query strings: “AD”, “CB”, “DR”, “AA” etc. BBF falsely identifies them as seen in the payload!

Because BBF cannot distinguish between P1 and P2
Hierarchical Bloom Filter

- An HBF is basically a set of BBF for geometrically increasing sizes of blocks.
Hierarchical Bloom Filter

- Querying is similar to BBF.
- Matches at each level can be confirmed a level above.
Hierarchical Bloom Filter

- Now we have a data structure that:
  - Allows us to do substring matching
  - Avoids “offset collisions”
  - Improves accuracy over standard Bloom Filter and BBF

\[ FP_e^{\text{HBF}} \ll FP_e^{\text{BBF}} \ll FP \]

HBF Performance Summary:
- Using a Bloom Filter with
  - \( m/n=5, k=2, FP=0.1090 \)
  - Block size of 128-bytes
- Achieves about 100 fold savings in storage
- For a 512-byte query \( FP_{\text{HBF}} = 2.00 \times 10^{-5} \)
Adapting an HBF for PAS

- So far an HBF can attest for the presence of a bit-string in payloads

- We need to tie this bit-string to a source and/or destination hosts

- Our Approach:
  - Similar to tying an offset to a block/bit-string
  - In addition to inserting (block||offset) also insert (block||offset||hostid)
  - Hostid could be (srcIP||dstIP)
A Payload Attribution System (PAS)

- System design of a payload attribution system
  - Packet id or host identifier is (SourceIP||DestIP)
  - Although host identifier can be obtained from firewalls and routers (Netflow), a list of host ids is maintained by the system
How to run a query?
A Bayesian approach

- Trying to measure:
  - $P = \text{conditional probability of offset collision}$ (i.e., probability that when two equal blocks at the same position in two packets, the next blocks are also equal)
  - *Turns out to be too pessimistic for practical use:*

- **FP rate analysis must depend on the query string to be useful!**

<table>
<thead>
<tr>
<th>Query Blocks</th>
<th>SMTP</th>
<th>HTTP</th>
<th>FTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0.542259</td>
<td>0.562018</td>
<td>0.168082</td>
</tr>
<tr>
<td>32</td>
<td>0.762793</td>
<td>0.547919</td>
<td>0.050411</td>
</tr>
<tr>
<td>64</td>
<td>0.410678</td>
<td>0.362524</td>
<td>0.006253</td>
</tr>
<tr>
<td>128</td>
<td>0.313077</td>
<td>0.322957</td>
<td>0.003306</td>
</tr>
<tr>
<td>256</td>
<td>0.213254</td>
<td>0.282814</td>
<td>0.003717</td>
</tr>
</tbody>
</table>

Table 1: Probabilities of offset collisions by protocol, measured on a network trace of 1M packets.
Note: these are *not* the false positive rates of our filters, see Table 2 for those.
FP Rate Analysis

\[ FP_e = \text{effective FP rate for payload query} \]
\[ \text{(after all the filtering)} \]
\[ FP_o = \text{individual FP rate of (content)} \text{ and} \]
\[ \text{(content||offset) BF for BBF or HBF}, \]
\[ FP_p = \text{individual FP rate of} \]
\[ \text{(content||offset||packetID) BF for BBF} \]
\[ \text{or HBF} \]
\[ n_b = \text{number of blocks inserted in the BBF}, \]
\[ n_h = \text{number of blocks inserted in the HBF} \]
\[ (n_b \leq n_h \leq 2n_b), \]
\[ N = \text{the number of packetIDs to check for} \]
\[ \text{the payload digest}. \]
FP Rate (BBF)

- $L$ = max length of a payload
- $u$ = number of blocks not in the query or at wrong offsets (offset collisions)
- $v$ = number of blocks at correct offsets but wrong packet (packet collisions)

To prevent offset collisions, since we must test $N$ packet IDs for each passing combination of blocks with offsets, we expect to see a false positive after both (content||offset) and (content||offset||packetID) BF's with probability at most

$$FP_e^{(BBF)} = L \cdot N \cdot FP_b^u \cdot FP_p^{u+v}.$$
FP Rate (HBF)

- \( L \) = max length of a payload
- \( u \) = number of blocks not in the query or at wrong offsets (offset collisions)
- \( v \) = number of blocks at correct offsets but wrong packet (packet collisions)
- \( u' \) and \( v' \) induced by the hierarchy, and
- \( u'' \) = offset collisions in the hierarchy

To prevent offset collisions, since we must test \( N \) packet IDs for each passing combination of blocks with offsets, we expect to see a false positive after both (content||offset) and (content||offset||packetID) BF's with probability at most

\[
FP^{(HBF)}_e = L \cdot N \cdot (FP'_o)^u + u' + u'' (FP'_p)^u + u' + u'' + v + v'.
\]

As a conclusion, with equal memory requirements, HBF is never worse than BBF, and sometimes much better.

Note: \( u + u' \geq 2u - 1 \)
\( u'' \approx v \)
\( FP' \approx \sqrt{FP} \)
### Actual FP Rates

<table>
<thead>
<tr>
<th>Blocks</th>
<th>0.3930</th>
<th>0.2370</th>
<th>0.1550</th>
<th>0.1090</th>
<th>0.0804</th>
<th>0.0618</th>
<th>0.0489</th>
<th>0.0397</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.000000</td>
<td>0.999885</td>
<td>0.996099</td>
<td>0.976179</td>
<td>0.933179</td>
<td>0.870477</td>
<td>0.798657</td>
<td>0.728207</td>
</tr>
<tr>
<td>2</td>
<td>0.063758</td>
<td>0.064569</td>
<td>0.048981</td>
<td>0.036060</td>
<td>0.026212</td>
<td>0.021024</td>
<td>0.015881</td>
<td>0.012538</td>
</tr>
<tr>
<td>3</td>
<td>0.012081</td>
<td>0.002620</td>
<td>0.000744</td>
<td>0.000275</td>
<td>0.000172</td>
<td>0.000046</td>
<td>0.000023</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0.000820</td>
<td>0.000230</td>
<td>0.000060</td>
<td>0.000020</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>&gt; 4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2: Measured effective false positive rate ($FP_e$) of HBF as a function of both the basic false positive rate ($FP_o$) and the length of the query (in blocks; 1block=32 bytes). Note that for blocks > 4, we encountered no false positives, hence the measured $FP_e$ is equal to 0 (indicated by –).

<table>
<thead>
<tr>
<th>Query Blocks</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBF</td>
<td>0.049621</td>
<td>0.035129</td>
<td>0.000560</td>
<td>0.000088</td>
</tr>
<tr>
<td>HBF</td>
<td>0.016457</td>
<td>0.000720</td>
<td>0.000110</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 3: Performance comparison of a BBF and an HBF with the same memory footprint. (Query strings of size > 5 resulted in 0 measured false positives for both BBF and HBF, hence are not listed.)
Tracking MyDoom

- Recorded all email traffic for a week
  - Using HBF and raw traffic
  - Was not aware of MyDoom during this collection
- When signatures became available we used them to query the system
  - To find hosts that are infected in our network
  - How the hosts were infected
- Some statistics:
  - 679 hosts originated at least one copy of the virus
    - 52 of which were in our network
  - These hosts sent out copies of the virus to 2011 hosts outside our network boundary
MyDoom’s Weekly Progress…
MyDoom’s Daily Progress

![Graph showing daily progress of MyDoom with different signature sizes and actual propagation rate.](image)
HBF Cannot Account for Duplicates
False Positives

- Actual number of virus instances seen is 25328
- Number of incorrect attributions for various lengths of the signature used to query the system

<table>
<thead>
<tr>
<th>Length</th>
<th>96</th>
<th>128</th>
<th>160</th>
<th>192</th>
<th>224</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect</td>
<td>1375</td>
<td>932</td>
<td>695</td>
<td>500</td>
<td>293</td>
<td>33</td>
</tr>
</tbody>
</table>
Attacks on PAS

- **Streaming transformations:**
  - Encryption, compression

- **Malicious Transformation:**
  - Create packets of length \((\text{blocksize} - 1)\)

- **Stuffing:**
  - Stuff every other block with application dependent escape characters
  - For smaller blocks we can try to guess for larger blocks it is not possible!

- **Resource Exhaustion:**
  - Flood the network with random bits of data

- **Exploiting Collisions:**
  - Hash collisions
    - Very unlikely for strong hash functions
    - We use a random seed for every HBF so it makes it more difficult
  - Packet collisions
    - A possibility in Block-based Bloom Filters but not in HBFs
Conclusion & Future Work

- **Summary:**
  - A data structure for digesting payloads
    - Supports queries on excerpts
    - Reduces storage requirements
    - Provides some privacy guarantees
  - Payload Attribution System:
    - Capable of attributing excerpts of payload to source/destination
    - In alpha-tests in our campus network

- **Future work:**
  - Value-based blocking using Rabin fingerprints
  - Enhancing storage of host ids
  - Hardware implementations
Questions