GPU-based

NSEC3 Hash Breaking

Matthäus Wander, Lorenz Schwittmann
<dnssec@vs.uni-due.de>

DNS-OARC Workshop
Dublin, May 13, 2013
Secure Denial of Existence

- Goal: offline signing of NXDOMAIN responses
- Sort owner names in canonical order
- NSEC: sign proof of non-existence
  
  \[
  \text{ftp IN NSEC mail}
  \]
- New requirement: hide names
- NSEC3: hash names before sorting
  
  \[
  \text{3a45 IN NSEC3 78a1}
  \]
- “There is no name with 3a45 < h(name) < 78a1”
- 31 top-level domains use NSEC, 81 use NSEC3
Reasons for and against NSEC3

- Opt-out
- Privacy
- Complexity
- CPU performance

⇒ How well does NSEC3 prevent zone enumeration?
def nsec3(name, iterations, salt):
    digest = hashlib.sha1(name + salt).digest()
    for i in xrange(0, iterations): # for(i=0; i < iterations; i++)
        digest = hashlib.sha1(digest + salt).digest()
    return digest

- **name**: label-encoded FQDN (RFC 1035 Section 3.1)
  - foo.example.org. = \x03foo\x07example\x03org\x00

- **iterations**: SHA-1 operations – 1
  - 0 iterations = 1 SHA-1 operation
  - 10 iterations = 11 SHA-1 operations

- **salt**: any binary blob 0–255 bytes
  - hexadecimal presentation format
SHA-1 Hash Function

- Name+Salt ≤ 55 bytes will fit into one SHA-1 block
- Each SHA-1 block increases the hashing work
- Maximum: 255 bytes name + 255 bytes salt + padding ⇒ 9 blocks
- Hash value is 20 bytes (32 bytes encoded as Base32hex)
- No known pre-image attacks on SHA-1
  - collision attacks not relevant for common NSEC3 usage
<table>
<thead>
<tr>
<th>TLD</th>
<th>It</th>
<th>Salt</th>
<th>TLD</th>
<th>It</th>
<th>Salt</th>
<th>TLD</th>
<th>It</th>
<th>Salt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ac</td>
<td>5</td>
<td>a9f1a97445</td>
<td>28. hn</td>
<td>10</td>
<td>6a9f5952</td>
<td>55. pw</td>
<td>150</td>
<td>46a94223</td>
</tr>
<tr>
<td>2. ag</td>
<td>1</td>
<td>d399eaab</td>
<td>29. in</td>
<td>1</td>
<td>d399eaab</td>
<td>56. re</td>
<td>1</td>
<td>badfe11a</td>
</tr>
<tr>
<td>3. am</td>
<td>10</td>
<td>76931f</td>
<td>30. info</td>
<td>1</td>
<td>d399eaab</td>
<td>57. ru</td>
<td>3</td>
<td>00ff</td>
</tr>
<tr>
<td>4. asia</td>
<td>1</td>
<td>d399eaab</td>
<td>31. io</td>
<td>5</td>
<td>28fb2159b6</td>
<td>58. sc</td>
<td>1</td>
<td>d399eaab</td>
</tr>
<tr>
<td>5. at</td>
<td>5</td>
<td>c8ea5a6104</td>
<td>32. jp</td>
<td>8</td>
<td>d8cb49d670</td>
<td>59. sh</td>
<td>5</td>
<td>2dc36b4ec0</td>
</tr>
<tr>
<td>6. be</td>
<td>5</td>
<td>1a4e9b6c</td>
<td>33. kr</td>
<td>10</td>
<td>96e920</td>
<td>60. si</td>
<td>5</td>
<td>380b3abb6</td>
</tr>
<tr>
<td>7. bz</td>
<td>1</td>
<td>d399eaab</td>
<td>34. la</td>
<td>150</td>
<td>61ca116859</td>
<td>61. su</td>
<td>3</td>
<td>00ff</td>
</tr>
<tr>
<td>8. ca</td>
<td>5</td>
<td>4a9f1a97445</td>
<td>35. lc</td>
<td>1</td>
<td>d399eaab</td>
<td>62. sx</td>
<td>10</td>
<td>4321</td>
</tr>
<tr>
<td>9. cat</td>
<td>12</td>
<td>ae7a5f02ad</td>
<td>36. li</td>
<td>2</td>
<td>025e</td>
<td>63. tf</td>
<td>1</td>
<td>badfe11a</td>
</tr>
<tr>
<td>10. cc</td>
<td>0</td>
<td></td>
<td>37. lt</td>
<td>5</td>
<td>797ecdb87f</td>
<td>64. th</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11. ch</td>
<td>2</td>
<td>d01f</td>
<td>38. lu</td>
<td>3</td>
<td>83e2fa0</td>
<td>65. tm</td>
<td>5</td>
<td>d03dd9e26c</td>
</tr>
<tr>
<td>12. cl</td>
<td>2</td>
<td>a45f80464b</td>
<td>39. lv</td>
<td>8</td>
<td>02c42256bc</td>
<td>66. tt</td>
<td>10</td>
<td>72183e36</td>
</tr>
<tr>
<td>13. com</td>
<td>0</td>
<td></td>
<td>40. me</td>
<td>1</td>
<td>d399eaab</td>
<td>67. tv</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>14. cr</td>
<td>10</td>
<td>00000000</td>
<td>41. mil</td>
<td>10</td>
<td>febc</td>
<td>68. tw</td>
<td>10</td>
<td>23411313</td>
</tr>
<tr>
<td>15. cx</td>
<td>10</td>
<td>34f35594</td>
<td>42. mn</td>
<td>1</td>
<td>d399eaab</td>
<td>69. ua</td>
<td>13</td>
<td>c0ffee</td>
</tr>
<tr>
<td>16. cz</td>
<td>10</td>
<td>cf089385be</td>
<td>43. museum</td>
<td>12</td>
<td>3b616ccef9</td>
<td>70. ug</td>
<td>10</td>
<td>6f1f6f40</td>
</tr>
<tr>
<td>17. de</td>
<td>15</td>
<td>ba5eba11</td>
<td>44. my</td>
<td>10</td>
<td>5439b2</td>
<td>71. uk</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>18. dk</td>
<td>17</td>
<td>0c8f6e891c</td>
<td>45. nc</td>
<td>10</td>
<td>140238c4</td>
<td>72. vc</td>
<td>1</td>
<td>d399eaab</td>
</tr>
<tr>
<td>19. edu</td>
<td>0</td>
<td></td>
<td>46. net</td>
<td>0</td>
<td></td>
<td>73. wf</td>
<td>1</td>
<td>badfe11a</td>
</tr>
<tr>
<td>20. eu</td>
<td>1</td>
<td>5ca1ab1e</td>
<td>47. nf</td>
<td>10</td>
<td>45b970ec</td>
<td>74. xn--3e0b707e</td>
<td>10</td>
<td>96e920</td>
</tr>
<tr>
<td>21. fi</td>
<td>5</td>
<td>d3f06270dd</td>
<td>48. nl</td>
<td>5</td>
<td>272ea647c5</td>
<td>75. xn--h2br9c</td>
<td>1</td>
<td>d399eaab</td>
</tr>
<tr>
<td>22. fo</td>
<td>10</td>
<td>69bf459c</td>
<td>49. nu</td>
<td>12</td>
<td>5544da4b</td>
<td>76. xn--kprw13d</td>
<td>10</td>
<td>23411313</td>
</tr>
<tr>
<td>23. fr</td>
<td>1</td>
<td>badfe11a</td>
<td>50. org</td>
<td>1</td>
<td>d399eaab</td>
<td>77. xn--kpry57d</td>
<td>10</td>
<td>23411313</td>
</tr>
<tr>
<td>24. gi</td>
<td>1</td>
<td>d399eaab</td>
<td>51. pl</td>
<td>12</td>
<td>28fb33b947</td>
<td>78. xn--mgbx4cd0ab</td>
<td>10</td>
<td>eb9717</td>
</tr>
<tr>
<td>25. gl</td>
<td>10</td>
<td>6cd13fbb2</td>
<td>52. pm</td>
<td>1</td>
<td>badfe11a</td>
<td>79. xn--o3cw4h</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>26. gov</td>
<td>8</td>
<td>4c4934802</td>
<td>53. post</td>
<td>1</td>
<td>d399eaab</td>
<td>80. xn--p1ai</td>
<td>3</td>
<td>00ff</td>
</tr>
<tr>
<td>27. gr</td>
<td>10</td>
<td>beef</td>
<td>54. pt</td>
<td>10</td>
<td>fcccc</td>
<td>81. yt</td>
<td>1</td>
<td>badfe11a</td>
</tr>
</tbody>
</table>
Attacking NSEC3

- Crawl NSEC3 hashes  implemented
- Break NSEC3 hashes
  - Brute-force attack  implemented
  - Markov chains  implemented
  - Dictionary attack  work in progress
  - Rainbow tables  no plans
    (won't work once everybody starts changing the salt regularly)
Crawl NSEC3 Hashes

1. Calculate hash value of random name
2. Check whether hash value falls into known NSEC3 range
   ◦ If yes, go back to step 1 and try again
3. Send query to server, receive NXDOMAIN and new NSEC3 range
4. Repeat until NSEC3 chain is complete

- Download \( n \) NSEC3 ranges with \( n \) online queries
- For large zones, finding the last NSEC3 ranges can take a while
- We send one query to a zone at a time
  ◦ Host: crawler.vs.uni-duisburg-essen.de.
  ◦ IPv4: 134.91.78.159    IPv6: 2001:638:501:8efc::159
Brute-force Attack

1. Enumerate all names up to a certain length
   ○ aaa, aab, aac, aad, ..., aa8, aa9, aba, abc, ...

2. Hash name

3. Check if hash is in known-hash array
   ○ Efficient approach: binary search with $O\left(\log(n)\right)$
   ○ Challenge on GPU, global memory access is slow
   ○ Bloom filter before binary search for probabilistic test: 300% speedup

- Limited feasiblity
  ○ 9 characters: ~1 week
  ○ 10 characters: ~37 weeks
Markov Chains

- General idea: characters in natural languages follow hidden Markov models
  - Probability of a certain character in a word depends on its predecessors
  - Example for the English language: 'th', 'he', 'in' and 'er' are very common

- Use a data set to calculate character transition probabilities
  - Names from brute-force attack for example
  - Advantage: TLD-specific probabilities are considered

- Use efficient algorithm which enumerates highly probable names
  - Ported Simon Marechal's John the Ripper Markov patch to OpenCL
  - Not exhaustive: omits improbable names
Tool: nsec3breaker

- Written in Python and OpenCL
- OpenCL runs on CPU or GPU
  - AMD/ATI graphics card: runs best
  - NVIDIA graphics card: runs decently
  - CPU with AMD APP SDK: runs
- http://dnssec.vs.uni-due.de/nsec3

```
svn co https://www.vs.uni-due.de/svn/dnssec/nsec3breaker/trunk
(install dnspython, numpy and pyopencl, see HOWTO.txt)

python nsec3breaker.py -o  # Desktop GPU: set also -c 0.01
python client.py -r someusername  # register at server
python client.py -j  # fetch and compute jobs
```
Statistics

- Brute-force attack performance
  - Radeon HD 7970: 1800 MHash/s (360 €)
  - Radeon HD 6970: 550 MHash/s (200 €)
  - 4 cores (out of 12) Intel Xeon X5650 2,67 GHz: 17 MHash/s

- Number of crawled hashes so far: 5,6 M
  - Top 3: ch. (1,46 M), cz. (1,03 M), nl. (1,45 M)

- Number of broken names so far: 1,6 M
  - Top 3: ch. (481 k), cz. (428 k), nl. (390 k)
Outlook

- Finish dictionary attack
  - create dictionary from PTR reverse lookups
- Automatic scheduling
  - crawl unknown zones
  - distributed breaking of unknown hashes
- Goal: full copy of all DNSSEC zones
  - for monitoring and further research
- Join the distributed computing project
  - http://dnssec.vs.uni-due.de/nsec3
- Question to you: publish zone data?