GPU-based

NSEC3 Hash Breaking

Matthäus Wander, Lorenz Schwittmann

<dnssec@vs.uni-due.de>

DNS-OARC Workshop

Dublin, May 13, 2013

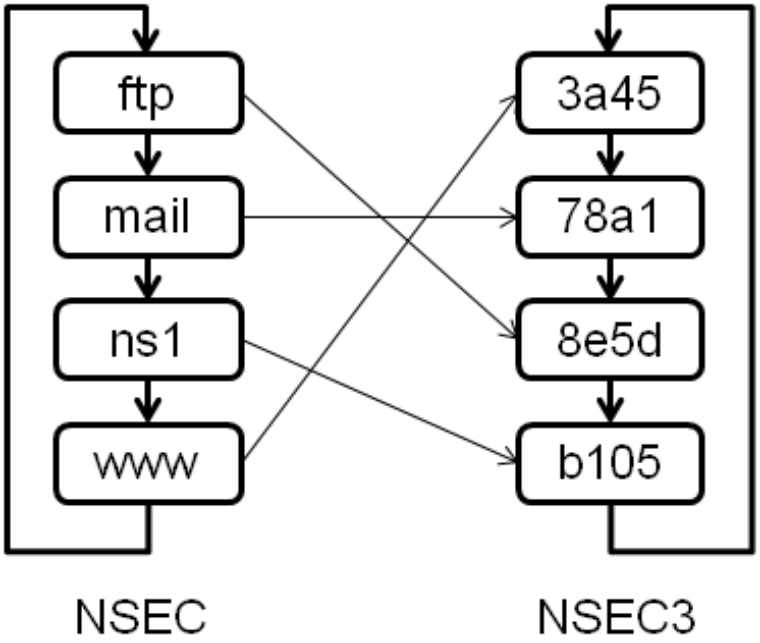
Secure Denial of Existence

- Goal: offline signing of NXDOMAIN responses
- Sort owner names in canonical order
- NSEC: sign proof of non-existence

```
ftp    IN  NSEC    mail
```

- New requirement: hide names
- NSEC3: hash names before sorting

```
3a45   IN  NSEC3   78a1
```



- "There is no name with $3a45 < h(\text{name}) < 78a1$"
- 31 top-level domains use NSEC, 81 use NSEC3

Reasons for and against NSEC3

- Opt-out
- Privacy
- Complexity
- CPU performance

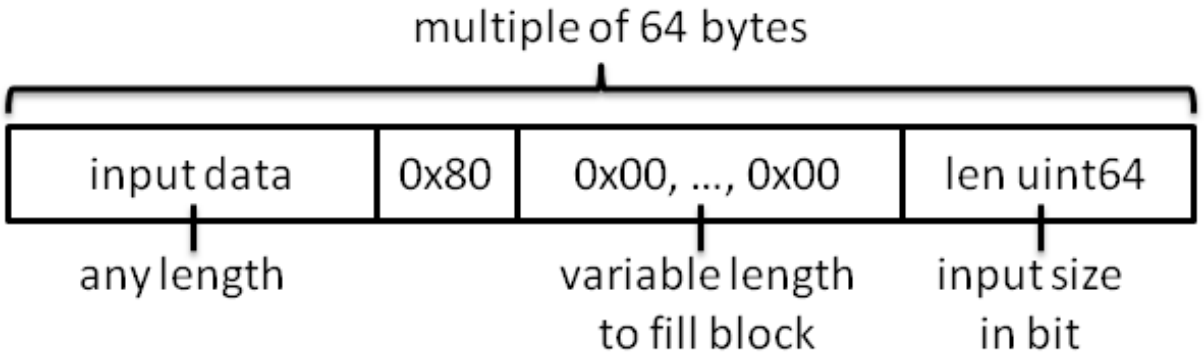
⇨ How well does NSEC3 prevent zone enumeration?

NSEC3 Hashing

```
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

NSEC3 Top-Level Domain Survey

TLD	It	Salt	TLD	It	Salt	TLD	It	Salt
1. ac.	5	a9f1a97445	28. hn.	10	6a9f5952	55. pw.	150	46a94223
2. ag.	1	d399eaab	29. in.	1	d399eaab	56. re.	1	badfe11a
3. am.	10	76931f	30. info.	1	d399eaab	57. ru.	3	00ff
4. asia.	1	d399eaab	31. io.	5	28fb2159b6	58. sc.	1	d399eaab
5. at.	5	c8ea5a6104	32. jp.	8	d8cb49d670	59. sh.	5	2dc36b4ec0
6. be.	5	1a4e9b6c	33. kr.	10	96e920	60. si.	5	380b3abbb6
7. bz.	1	d399eaab	34. la.	150	61ca116859	61. su.	3	00ff
8. ca.	5		35. lc.	1	d399eaab	62. sx.	10	4321
9. cat.	12	ae7a5f02ad	36. li.	2	025e	63. tf.	1	badfe11a
10. cc.	0		37. lt.	5	797ecdb87f	64. th.	10	
11. ch.	2	d01f	38. lu.	3	83e2faf0	65. tm.	5	d03dd9e26c
12. cl.	2	a45f80464b	39. lv.	8	02c42256bc	66. tt.	10	72183e36
13. com.	0		40. me.	1	d399eaab	67. tv.	0	
14. cr.	10	00000000	41. mil.	10	febc	68. tw.	10	23411313
15. cx.	10	34f35594	42. mn.	1	d399eaab	69. ua.	13	c0ffee
16. cz.	10	cf089385be	43. museum.	12	3b616ccef9	70. ug.	10	6f1f6f40
17. de.	15	ba5eba11	44. my.	10	5439b2	71. uk.	0	
18. dk.	17	0c8f6e891c	45. nc.	10	140238c4	72. vc.	1	d399eaab
19. edu.	0		46. net.	0		73. wf.	1	badfe11a
20. eu.	1	5ca1ab1e	47. nf.	10	45b970ec	74. xn--3e0b707e.	10	96e920
21. fi.	5	d3f06270dd	48. nl.	5	272ea647c5	75. xn--h2brj9c.	1	d399eaab
22. fo.	10	69bf459c	49. nu.	12	5544da4b	76. xn--kprw13d.	10	23411313
23. fr.	1	badfe11a	50. org.	1	d399eaab	77. xn--kpry57d.	10	23411313
24. gi.	1	d399eaab	51. pl.	12	28fb33b947	78. xn--mgbx4cd0ab.	10	eb9717
25. gl.	10	6cd13fb2	52. pm.	1	badfe11a	79. xn--o3cw4h.	10	
26. gov.	8	4c44934802	53. post.	1	d399eaab	80. xn--p1ai.	3	00ff
27. gr.	10	beef	54. pt.	10	fcce	81. yt.	1	badfe11a

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 $O(log(n))$
   - Challenge on GPU, global memory access is slow
   - Bloom filter before binary search for probabilistic test: 300% speedup

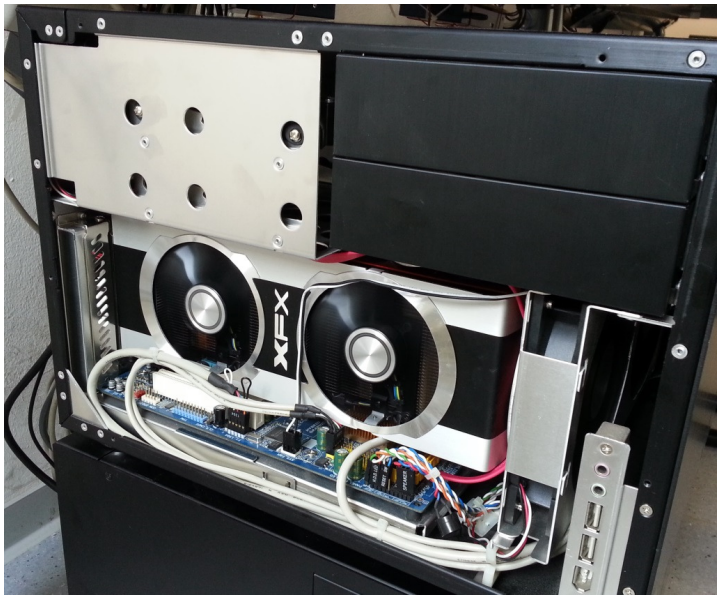
- Limited feasiblity
  - 9 characters: ~1 week
  - 10 characters: ~37 weeks

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

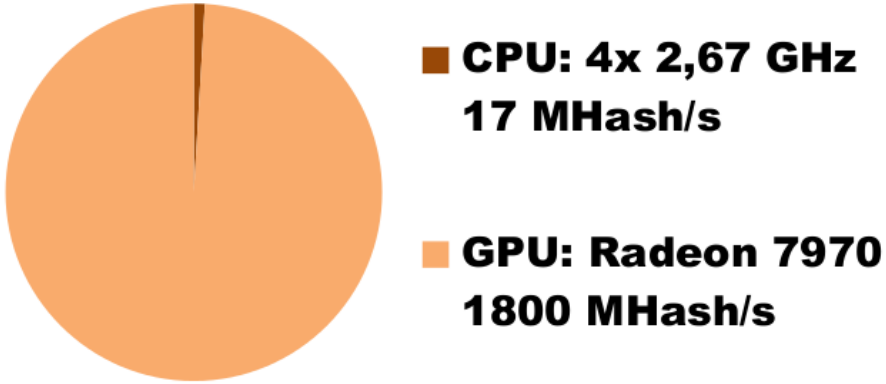
- Written in Python and OpenCL
- OpenCL runs on CPU or GPU
  - AMD/ATI graphics card: runs best
  - NVIDIA graphics card: runs decently
  - CPU with AMD APP SDK: runs
- http://dnssec.vs.uni-due.de/nsec3



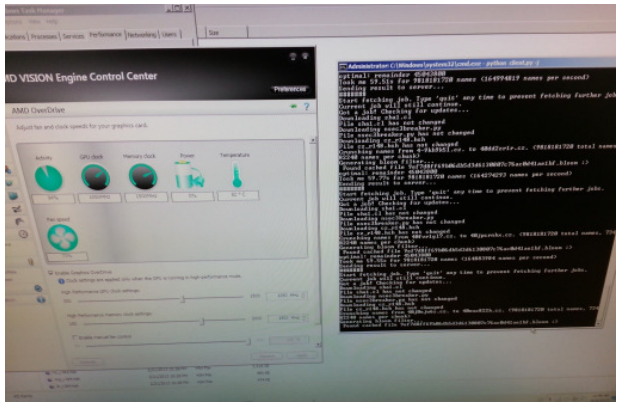

```
svn co https://www.vs.uni-due.de/svn/dnssec/nsec3breaker/trunk

(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?