GPU–based NSEC3 Hash Breaking

Matthäus Wander, Lorenz Schwittmann, Christopher Boelmann, Torben Weis

<matthaeus.wander@uni-due.de>

IEEE NCA 2014

Cambridge, August 22, 2014
Secure Denial of Existence with DNSSEC

Query: www?

ftp
mail
ns1
www
Secure Denial of Existence with DNSSEC

- Signed resource record
- Authenticity verified with PKI built into DNSSEC
Secure Denial of Existence with DNSSEC

Query: test?

Not found

- Query name: $n$
Secure Denial of Existence with DNSSEC

- Query name: $n$

![Diagram showing a query process with DNSSEC](image)
Secure Denial of Existence with DNSSEC

- Query name: $n$
- Response: $(r_1, r_2)$ with $r_1 < n < r_2$
Zone Enumeration

- Copy DNSSEC zone by crawling NSEC records
- Nominet (.uk) position:
  
  „Zone file enumeration is a show-stopper for us that will prevent us from fully implementing DNSSEC.“

- DENIC (.de) position:
  
  „In conflict with Germany’s Federal Data Protection Act“
Secure Denial of Existence with DNSSEC

Query: test?

Not found

ftp
mail
ns1
www

NSEC
Secure Denial of Existence with DNSSEC

- Hash query name: $h(n)$
Secure Denial of Existence with DNSSEC

Query: test?

78a1 NSEC3 8e5d

Not found

- Hash query name: \( h(n) \)
- Response: \((h(r_1), h(r_2))\) with \( h(r_1) < h(n) < h(r_2) \)
Reasons for and against NSEC3

- Privacy
- “Opt-out” feature
- CPU load
- Message size

⇒ How well does NSEC3 prevent zone enumeration?
NSEC3 Definition

- Repeated SHA1 computation
- Salt is identical for all names in a DNSSEC zone

\[ h(n, s, 0) = SHA1(n\|s) \]
\[ h(n, s, i) = SHA1(h(n,s,i-1)\|s), \text{ for } i > 0 \]
## Top–Level Domain Survey

<table>
<thead>
<tr>
<th>TLD</th>
<th>i</th>
<th>Salt</th>
</tr>
</thead>
<tbody>
<tr>
<td>at.</td>
<td>5</td>
<td>b4c3ee8e123154f8</td>
</tr>
<tr>
<td>ch.</td>
<td>2</td>
<td>e7bd8151</td>
</tr>
<tr>
<td>cn.</td>
<td>10</td>
<td>aef123ab</td>
</tr>
<tr>
<td>com.</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>de.</td>
<td>15</td>
<td>ba5eba11</td>
</tr>
<tr>
<td>eu.</td>
<td>1</td>
<td>5ca1able</td>
</tr>
<tr>
<td>fr.</td>
<td>1</td>
<td>41b69d30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TLD</th>
<th>i</th>
<th>Salt</th>
</tr>
</thead>
<tbody>
<tr>
<td>info.</td>
<td>1</td>
<td>d399eaab</td>
</tr>
<tr>
<td>jp.</td>
<td>8</td>
<td>f5435b71d7</td>
</tr>
<tr>
<td>net.</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>nl.</td>
<td>5</td>
<td>c9545f07a61d0d33</td>
</tr>
<tr>
<td>org.</td>
<td>1</td>
<td>d399eaab</td>
</tr>
<tr>
<td>ru.</td>
<td>3</td>
<td>00ff</td>
</tr>
<tr>
<td>uk.</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

- 335 use NSEC3, 53 use NSEC, 185 not signed
- 60% of TLDs with NSEC3 use i=1
- 42% of TLDs with NSEC3 change the salt
ATTACKING NSEC3

1. Collect hashes
2. Reverse hashes
Why AMD Graphic Cards?

![Graph showing total time vs. NSEC3 iterations for 4-core CPU, NVIDIA GTX 690, and AMD HD 7970]
ATTACKING NSEC3

1. Collect hashes
2. Reverse hashes
Hash Crawling

- Retrieve NSEC3 hashes from name server
  - Send queries for random non-existing names

M8eZcl8.com?

78a1  NSEC3  8e5d

Not found

- Check hash before sending query

| NSEC3 | gap | NSEC3 | gap | NSEC3 | gap |
Crawling 345,000 Hashes from .com
ATTACKING NSEC3

1. Collect hashes
2. Reverse hashes
GPU–based NSEC3 Hash Breaking

- Iterate through cleartext candidate names
  - Compute NSEC3 hash, compare with known hashes
- Global memory access on GPU is expensive
  - Bloom filter for 300% speedup
Brute–Force Attack

Number of names found

1 to 8 characters

9 characters

Time in hours

Brute force
Dictionary Attack

- Effectiveness depends on dictionary quality
  - Good public lists (Alexa, Quantcast)
  - List of all IPv4 reverse mappings (not exhaustive!)

- 7.1 mio candidate names

- Generate more candidates by inserting strings
  - Insert most common 200,000 n-grams (with $1 \leq n \leq 15$)

`foo.com.`

\[\uparrow \uparrow \uparrow \uparrow \uparrow \]

- `e`
Dictionary Attack

![Graph showing number of names found over time for Bruteforce and Wordlist.](image-url)
Effectiveness of Insertion Wordlist

![Graph showing the effectiveness of insertion wordlist]

- **Y-axis:** Number of names found
- **X-axis:** Index of n-grams, sorted by frequency in dictionary

Universität Duisburg–Essen
Verteilte Systeme
Matthäus Wander
Markov Attack

- Some strings are more probable than others
  - ’th‘ more common in English than ’tx‘
  - Model language with 1st order Markov chains

- Markov attack requires training
  - Use hits from brute-force and dictionary attacks
  - Markov model derived from names found

- Enumerate most probable names
  - Omit others (with given time frame)
Markov Attack

Number of names found vs Time in hours for Bruteforce, Markov, and Wordlist methods.
Discussion

- Reversed 65% .com domains after 5 days
- Server operator pays for NSEC3 with CPU load
- Adjust iterations to increase CPU workload
Discussion

- Reversed 65% .com domains after 5 days
- Server operator pays for NSEC3 with CPU load
- Adjust iterations to increase CPU workload
Discussion

- Reversed 65% .com domains after 5 days
- Server operator pays for NSEC3 with CPU load
- Adjust iterations to increase CPU workload
Efficiency of CPU vs. GPU

![Graph showing the efficiency of CPU vs. GPU.]
Conclusions

- NSEC3 returns negative DNSSEC responses without disclosing existing domain names
- GPUs very efficient for breaking NSEC3 hashes
- Reversed 65% .com domains after 5 days
- Assess whether weak privacy justifies the costs
- Future work:
  - GPU-based NSEC3 accelerators for DNSSEC servers