Taking the DNS for a Walk; NSEC3 Prevalence and Recoverability

Harrison Mitchell
whitepapers@harrisonm.com

***Abstract*—Machines querying for non-existent names within the Domain Name System (DNS) are met with a Non-Existent Domain (`NXDOMAIN`) error. Within the context of DNS Security Extensions (DNSSEC), an authenticated negative answer is returned containing both the lexicographically prior and following existent names within the Next Secure (NSEC) DNS record. NSEC provides these values in plaintext allowing for linear DNS zone walking. NSEC3 instead hashes these neighbouring existent names in an attempt to limit DNS record disclosure. This paper presents a GPU-based attack on NSEC3 that recovered 44% of names for the internet's top 20,000 NSEC3-protected DNS zones, partially invalidating NSEC3's privacy and security goals.**

I. Introduction

Domain Name System (DNS) traffic is infamously unencrypted on UDP port 53, providing adversaries a means of tampering with DNS answers in transit. DNS Security Extensions (DNSSEC) [1]–[3] was proposed to cryptographically sign responses to provide origin authentication and integrity such that upon verification, tampered responses can be discarded.

Requesting non-existent records classically returns Non-Existent Domain (`NXDOMAIN`) errors. Given the undifferentiated nature of these responses, an attacker could replay cryptographically valid `NXDOMAIN` errors for any records regardless of existence. To prevent a "denial of existence" attack DNSSEC introduced a new record type: NSEC [4].

An NSEC response provides the two lexicographically neighbouring records of the non-existent record. A query for "`contact`" in the zone {`about`, `blog`, `forum`, `www`} would return "`blog NSEC forum A MX TXT`". Trivially daisy-chaining these ranges together reveals all available record names. Moreover, NSEC records return a list of record types that the preceding known-name contains. Ergo, from the example above it is known that "`blog`" has record types `A`, `MX` and `TXT`. In effect, DNSSEC zones that utilise NSEC become completely transparent with at most $O(n)$ requests.

In an effort to reduce information leakage and promote security and privacy, NSEC3 introduced hashing of neighbouring names [5]. Hashing however introduces larger response sizes and computational overhead. This leads to a schism in adoption of NSEC vs NSEC3 in the wild.

In this paper, I analyse the recoverability of hashed NSEC3 names to determine the efficacy of NSEC3's security measures against zone information leakage using consumer-grade hardware. After reviewing NSEC3 fundamentals (Section II), I survey the top 1,000,000 internet domains for their implementation of DNSSEC, NSEC and NSEC3 (Section III). Subsequently, I share a method for obtaining NSEC3 hashes (Section IV) followed by the introduction of four hash cracking methods (Section V) and a discussion on their recovery rates (Section VI). The goal of this paper is to ascertain the recoverability of NSEC3 zones to allow DNS operators to evaluate whether it can be relied upon as a privacy and/or security measure.

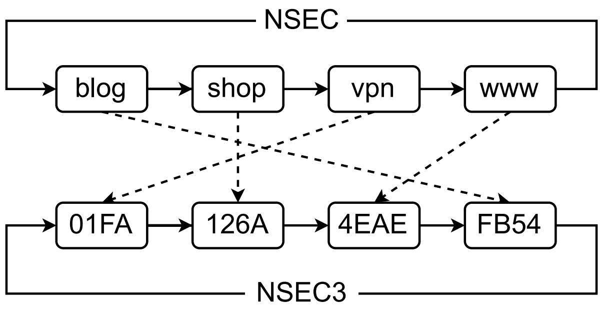

Fig. 1. NSEC's ordered loop of names compared to NSEC3's ordered loop of name hashes.

II. Authenticated Denial of Existence

Queries for non-existent DNSSEC domains need to be signed and authenticated to prevent "denial of existence" attacks by MitM, thus NSEC and NSEC3 were proposed. These mechanisms are designed to support offline signing. This process signs all DNS records prior to deployment, preventing delayed responses due to taxing cryptography at runtime. This provides the added benefit of allowing the offline signing secret keys to be air-gapped from the internet-facing DNS server.

Increasing NSEC3's security is an iteration variable. Iterations specify the number of times to re-apply the hashing function. Each iteration increases the complexity and resources necessary for cracking hashes at the cost of increasing resources necessary for initial signing. Salting hashes is also supported by NSEC3 but entirely ineffectual [6]. Non-disclosed salts enhance hash entropy, yet NSEC3 publicly provides them. Additionally, salts typically increase resistance to "rainbow table" attacks but NSEC3 hashes are inherently unique to their domain as the domain forms the hashed plaintext. NSEC3's hash function h per the RFC [5] is defined as:

$$h(n, s, 0) = f(n \parallel s)$$
$$h(n, s, i) = f(h(n, s, i-1) \parallel s), \text{for } i > 0$$

where n is name, s is a salt, f an arbitrary hash function and $\|$ representing concatenation. The function is called recursively $i+1$ times, colloquially, the number of iterations. While the choice of hashing function is arbitrary, currently the only standard-approved algorithm is SHA-1 despite the availability of securer alternatives.

III. Internet Survey

I probed the top 1 million most trafficked sites as provided by Alexa[1]. Querying a random 32 character subdomain sufficiently guaranteed hitting a non-existent record, allowing for the analysis of the prevalence of DNSSEC, NSEC and NSEC3. Of the domains, only 4% (40,223) employed DNSSEC split relatively evenly between NSEC and NSEC3 with 46.8% (18,827) and 53.2% (21,396) respectively. Figure 2 plots observed NSEC3 salt lengths and iterations.

IV. Hash Collection

Crawling NSEC records involves linear querying of names canonically until the collected (prior, following) name ranges form a closed loop. As NSEC3 is instead hash based, random candidate names are generated and hashed locally (as the plaintext, salt, iterations and algorithm are known) until a hash lies within a yet-discovered range based on a technique proposed by Bernstein [7]. At this point, the nameservers are queried with this candidate, returning the enclosing range. This process is repeated until a closed loop of hashes is obtained for offline cracking. To automate this crawling process I have written a Python tool which is available for public download [8]. Crawling the collected 21,396 NSEC3-signed domains returned 889,599 NSEC3 hashes for offline cracking.

V. Hash Cracking

With the NSEC3 hashes in hand, an attempt was made to "reverse" the hashing process. Given the one-way nature of hash functions, I iterate through a pool of candidates and check if hashing with the NSEC3 provided parameters matches any crawled hashes. By design this is an extremely computationally expensive process. Four "NVIDIA GTX 1070 Ti" GPUs were able to provide an aggregate 3 million hash attempts per second. However the choice of candidate pool can drastically alter the odds of success. The following four candidate pools were devised.

A. Bruteforce

Bruteforcing involves testing every combination of characters up to a specified length e.g. for the set of characters $\{a, b \dots z\}$, the set of candidates of length four consists of $\{aaaa, aaab \dots zzzz\}$. This guarantees recovery of hashed values to that length. Whilst the DNS standard applies no limits to the legal characters of a subdomain [9], the set $|\{\text{'-'}, \text{'\_'}, 0 \dots 9, a \dots z\}| = 39$ was chosen, as other characters are seldom used in practice. Of note is the inclusion of the period character as NSEC3 hashes can represent nested subdomains e.g. "`test.vpn`". Whilst bruteforcing is comprehensive, each additional character adds exponential workload $O(l^{39})$. Considering the available resources, a maximum bruteforce length of seven characters was chosen providing $\sum_{n=1}^{7} 39^n \approx 141$ billion candidates.

[1]http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

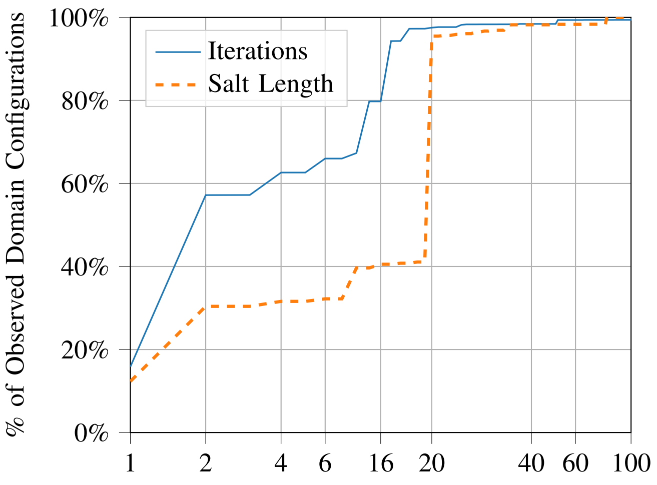

Fig. 2. Logarithmic view of observed NSEC3 iterations and salt lengths.

B. Dictionary Attack of Known Values

Pre-compiled lists of known subdomains are freely available online. These are likely to have a large overlap with the domains associated with the collected NSEC3 hashes as many domains share common subdomain values e.g. "`www`". Three popular lists[2] were merged and bolstered with subdomains obtained from crawling the 21,396 NSEC domains identified above. This resulted in a list of over 41,000,000 candidate names.

C. Mutated Dictionary Attack of Known Values

Dictionary attacks often have high yields, but can only provide as many results as it has entries. These pre-compiled lists are largely representative, but common mutations can be applied to increase coverage. DNS operators are known to prepend and append values such as "`test`" to subdomains. While many common such mutations may already exist within the wordlist, by applying a mutation rule, they can be applied to all dictionary items. The top 20 most frequently occurring subdomain mutations were selected in addition to the 5 years either side of 2022 to combat stale entries within the dictionary. Finally technology acronyms[3] such as "`vpn`" were added to form a mutation list of 60 entries which was prepended and appended to each dictionary item.

[2]https://wordlists-cdn.assetnote.io/data/manual/best-dns-wordlist.txt
https://gist.github.com/jhaddix/f64c97d0863a78454e44c2f7119c2a6a
https://mega.nz/file/UsJXzITR#TMIFTXnW1zABHfZUIKMPMvA-c8UvzGWeAd4mg4gGCtQ

[3]https://en.wikipedia.org/wiki/List_of_information_technology_initialisms

D. Dictionary Attack of Plausible Names

DNS names by design are intended to be human created, remembered and entered. It naturally follows that DNS operators will use common terms and words from spoken language to form names. These common words were also augmented using the same type of mutations as C above. The dictionary was further expanded by creating pairs of every entry (delimited by a hyphen, period and null) to capture plausible names that are not included in public dictionaries such as ”`alumnivpn`” and ”`research.files`”.

VI. Discussion

Across the four cracking methods, 44% of unique hashes were recovered within 2.5 days. This is not too dissimilar to previous research achieving a recovery rate of 64% [10]. The key distinction lies within the fact that prior research had been performed on domain names within a TLD zone (e.g. ____`.com`) rather than subdomains within domains themselves (e.g. ____`.example.com`). Not only are domains limited to a smaller set of characters but they are less likely to be highly entropic and irrecoverable. Subdomains within crawled plaintext NSEC data highlight that many subdomains are highly nested and entropic e.g. ”`fd-paypal-201703-2048._domainkey`”. These subdomains are required for purposes such as email security and proving domain ownership. Domain names do not need to cater for these purposes and are thus inherently more recoverable. It was clear that public subdomain dictionaries are of high quality, recovering the most hashes within the shortest period of time. The bruteforce attack recovered an additional 5.4% of unique hashes followed by 2.1% by the dictionary mutations and 1.6% by the custom created plausible dictionary.

VII. Conclusion

I presented an attack on sampled NSEC3 records from the internet (via my open source tool [8]) to discover how recoverable DNS record names are, and how this impacts the security and privacy NSEC3 intended to improve. To the best of my knowledge, this paper presents the first analysis of cracking NSEC3 subdomain hashes. It was concluded that 44% of NSEC3 hashes could be recovered within 2.5 days of GPU-based hash cracking. These findings suggest that DNS operators with a need to keep records private should consider the recommendations made by RFC 9276 [11] or implement NSEC ”black lies” [12].

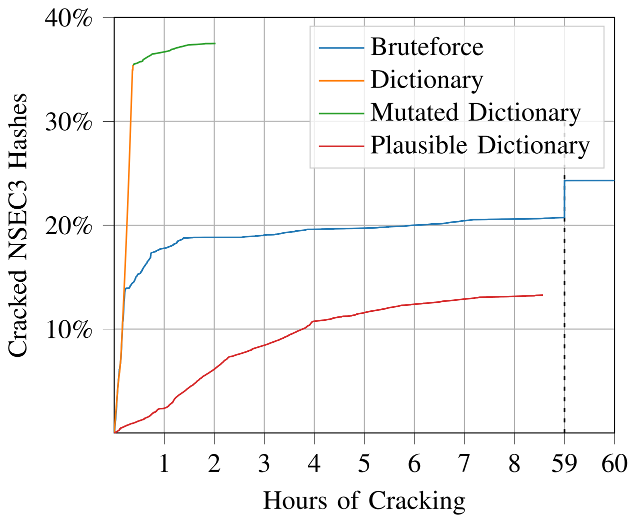

Fig. 3. Names cracked over time.

References

[1] S. Rose, M. Larson, D. Massey, R. Austein, and R. Arends, “Dns security introduction and requirements,” no. 4033, 03 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4033

[2] ——, “Resource records for the dns security extensions,” no. 4034, 03 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4034

[3] ——, “Protocol modifications for the dns security extensions,” no. 4035, 03 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4035

[4] J. Schlyter, “Dns security (dnssec) nextsecure (nsec) rdata format,” no. 3845, 08 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3845

[5] R. Arends, G. Sisson, D. Blacka, and B. Laurie, “Dns security (dnssec) hashed authenticated denial of existence,” no. 5155, 03 2008. [Online]. Available: https://www.rfc-editor.org/info/rfc5155

[6] *A security evaluation of DNSSEC with NSEC3.*, 01 2010. [Online]. Available: https://theory.stanford.edu/ jcm/papers/dnssec_ndss10.pdf

[7] D. Bernstein, “Breaking dnssec,” 2009. [Online]. Available: http://cr.yp.to/talks/2009.08.10/slides.pdf

[8] H. Mitchell, “Nsec(3) walker,” 09 2022. [Online]. Available: https://github.com/Harrison-Mitchell/NSEC-3-Walker

[9] R. Elz and R. Bush, “Clarifications to the dns specification,” no. 2181, 07 1997. [Online]. Available: https://www.rfc-editor.org/info/rfc2181

[10] *GPU-Based NSEC3 hash breaking*, 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6924218?arnumber=6924218

[11] W. Hardaker and V. Dukhovni, “Guidance for NSEC3 Parameter Settings,” RFC 9276, Aug. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9276

[12] D. Grant, “Economical with the truth: Making dnssec answers cheap,” The Cloudflare Blog, 06 2016. [Online]. Available: https://blog.cloudflare.com/black-lies/