The Internet Dodged a Bullet

The vulnerability has been codenamed "KeyTrap", and if ever there was a need for responsible disclosure of a problem, this was that time. Fortunately, responsible disclosure is what it received.

What the German researchers who discovered this realized was that an aspect of the design of DNS, specifically the design of the secure capabilities of DNS, known as DNSSEC, could be used against any DNSSEC-capable DNS resolver to bring that resolver to its knees. The receipt of a single UDP DNS query packet could effectively take that DNS resolver offline for as many as 16 hours, pinning its processor at 100%, and effectively denying anyone else that server's DNS services. It would have therefore been possible to spray the Internet with these single packet DNS queries to effectively shutdown all DNS services across the entire globe. Servers could be rebooted once it was noticed that they had effectively hung, but if they again received another innocuous looking DNS query packet, which is their job after all, they would have gone down again. Eventually, of course, the world would have discovered what was bringing all of its DNS servers to its knees. But the outage could have been protracted and the damage to the world's economies could have been horrendous. So now you know why today's podcast is titled: *"The Internet Dodged a Bullet."* We should never underestimate how utterly dependent the world has become on the functioning of the Internet.

The detailed research paper describing this was just released publicly yesterday, though the problem has been known since late last year. Here's how the paper's Abstract describes what these German researchers discovered, to their horror:

> *Availability is a major concern in the design of DNSSEC. To ensure availability, DNSSEC follows Postel's Law [RFC1122], which reads: "Be liberal in what you accept, and conservative in what you send." Hence, nameservers should send not just one matching key for a record set, but all the relevant cryptographic material, in other words, all the keys for all the ciphers that they support and all the corresponding signatures. This ensures that validation succeeds, and hence availability, even if some of the DNSSEC keys are misconfigured, incorrect or correspond to unsupported ciphers.*
>
> ***We show that this design of DNSSEC is flawed.*** *By exploiting vulnerable recommendations in the DNSSEC standards, we develop a new class of DNSSEC-based algorithmic complexity attacks on DNS, which we dub KeyTrap attacks.* ***All popular DNS implementations and services are vulnerable.*** *With just a single DNS packet, the KeyTrap attacks lead to* ***a two million times*** *spike in CPU instruction count in vulnerable DNS resolvers, stalling some for as long as 16 hours. This devastating effect prompted major DNS vendors to refer to KeyTrap as* ***"the worst attack on DNS ever discovered".*** *Exploiting KeyTrap, an attacker could effectively disable Internet access in any system utilizing a DNSSEC-validating resolver.*
>
> *We disclosed KeyTrap to vendors and operators on November 2, 2023. We confidentially reported the vulnerabilities to a closed group of DNS experts, operators and developers from the industry. Since then, we have been working with all major vendors to mitigate KeyTrap, repeatedly discovering and assisting in closing weaknesses in proposed patches. Following our disclosure, an umbrella CVE has been assigned.*

So, believe it or not, all that just actually happened, as they say, behind closed doors.

Google's Public DNS and Cloudflare were both vulnerable, as was the very popular and widely deployed BIND 9 DNS implementation, and it was the one that could be stalled for as long as 16 hours.

So… what happened?

As the researchers wrote, months before this all came to light publicly, all major implementations of DNS had already been quietly updated because had this gotten out it could have been used to bring the entire Internet down. With Microsoft's release of patches which included mitigations for this, and after waiting a week for them to be deployed, the problem is mostly resolved (of you'll pardon the pun).

I say "largely" because this is not a bug in an implementation and apparently even now, some DNS servers will still have their processors pinned, but they will still be able to answer other DNS queries function. It sounds as though thread or process priorities have been changed to prevent the starvation of competing queries. Characterizing this as a "big mess" would not be an exaggeration.

KeyTrap exploits a fundamental flaw in the design of DNSSEC which makes it possible to deliberately create a set of essentially legal but ultimately malicious DNSSEC response records which the receiving DNS server will be quite hard pressed to untangle. Once the researchers had realized that they were onto something big, they began exploring all of the many various ways DNS servers could be stressed. So they created a number of different attack scenarios.

I want to avoid getting too far into the weeds of the design and operation of DNSSEC, but at the same time I suspect that **this** podcast's audience will appreciate seeing a bit more of the detail so that the nature of the problem can be better appreciated. The problem is rooted in DNSSEC's provisions for the resolution of "key-tag" collisions, multiple keys and multiple signatures.

I'm going to quote three paragraphs from their research paper. But just sort of let it wash over you so that you'll get some sense for what's going on without worrying about understanding it in any detail. You will not be tested on your understanding of this. Okay, so they explain:

> *We find that the flaws in the DNSSEC specification are rooted in the interaction of a number of recommendations that in combination can be exploited as a powerful attack vector:*
>
> ***Key-tag collisions:*** *First, DNSSEC allows for multiple cryptographic keys in a given DNS zone, for example during key-rollover or for multi-algorithm support. Consequently, when validating DNSSEC, DNS resolvers are required to identify a suitable cryptographic key to use for signature verification. DNSSEC uses key-tag values to differentiate between the keys, even if they are of the same zone and use the same cryptographic algorithm. The triple of (zone name, algorithm, key-tag) is added to each respective signature to ensure efficiency in key-signature matching. When validating a signature, resolvers check the signature header and select the key with the matching triple for validation. However, the triple is not necessarily unique: multiple different DNS keys can have an identical triple. This can be explained by the calculation of the values in the triple. The algorithm identifier results directly from the cipher*

used to create the signature and is identical for all keys generated with a given algorithm. DNSSEC mandates all keys used for validating signatures in a zone to be identified by the zone name. Consequently, all DNSSEC keys that may be considered for validation trivially share the same name. Since the collisions in algorithm-id and key name pairs are common, the key-tag is calculated with a pseudo-random arithmetic function over the key bits to provide a means to distinguish same-algorithm, same-name keys. Using an arithmetic function instead of a manually chosen identifier eases distributed key management for multiple parties in the same DNS zone; instead of coordinating key-tags to ensure uniqueness, the key-tag is automatically calculated. However, the space of potential tags is limited by the 16 bits in the keytag field. Key-tag collisions, while unlikely, can thus naturally occur in DNSSEC. This is explicitly stated in RFC4034, emphasizing that key-tags are not unique identifiers. As we show, colliding key-tags can be exploited to cause a resolver not to be able to identify a suitable key efficiently but to have to perform validations with all the available keys, inflicting computational effort during signature validation.

To interrupt for a second, cryptographic keys are identified by tags, and those tags are automatically assigned to those keys. Work on DNSSEC began way back in the 1990's when the Internet's designers were still counting bits and were assigning only as many bits to any given field as would conceivably be necessary. Consequently, the tags being assigned to keys were, and are today, only 16 bits long. Since these very economical tags only have 16 bits, thus one of 64K possible values, inter-tag collisions, while unlikely, are possible. If DNSSEC were being designed today, tags would be the output of a collision-free cryptographic hash function and there would be no provision for resolving tag collisions because there would be none. The paragraph I just read said *"the key-tag is calculated with a pseudo-random arithmetic function"*, in other words, something simple from the '90's that scrambles and mixes the bits around but doesn't do much more. Consequently, servers need to consider that key-tags are not unique.

Okay, on to the next problem:

Multiple keys: *The DNSSEC specification mandates that a resolver must try all colliding keys until it finds a key that successfully validates the signature or all keys have been tried. This requirement is meant to ensure availability. Even if colliding keys occur, such that some keys may result in failed validation, the resolver must try validating with all the keys until a key is found that results in a successful validation. This ensures that the signed record remains valid and the corresponding resource therefore remains available. However, this "eager validation" can lead to heavy computational effort for the validating resolver, since the number of validations grows linearly with the number of colliding keys. For example, if a signature has ten colliding keys, all with identical algorithm identifiers, the resolver must conduct ten signature validations before concluding the signature is invalid. While colliding keys are rare in real-world operation, we show that records created to deliberately contain multiple colliding keys can be efficiently crafted by an adversary, imposing heavy computation upon a victim resolver.*

And, finally, the third and final problematic design feature:

Multiple signatures: *The philosophy of trying all of the cryptographic material available to ensure that the validation succeeds also applies to the validation of signatures.*

Creating multiple signatures for a given DNS record can happen, for example, during a key-rollover. The DNS server adds a signature with the new key, while retaining the old signature to ensure that some signature remains valid for all resolvers until the new key has propagated. Thus, parallel to the case of colliding keys, the RFCs specify that in the case of multiple signatures on the same record, a resolver should try all the signatures it received until it finds a valid signature or until all signatures have been tried.

Okay. So we have the essential design features which were put into the DNSSEC specification in a sane way with the purpose of never failing to find a valid key and signature for a zone record. Their term for this is "eager validation", and they write:

We combine these requirements for the eager validation of signatures and of keys, along with the colliding key-tags to develop powerful DNSSEC-based algorithmic complexity attacks on validating DNS resolvers. Our attacks allow ***a low-resource adversary*** *to fully DoS a DNS resolver for up to 16 hours* ***with a single DNS request.***

Members from the 31-participant task force of major operators, vendors and developers of DNS & DNSSEC, to which we disclosed our research, dubbed our attack: "the most devastating vulnerability ever found in DNSSEC."

The researchers devised a total of four different server-side resource exhaustion attacks and I was a little tempted to title today's podcast after three of them. Had I done so today's podcast would have been titled: "SigJam, LockCram & HashTrap." And while I certainly acknowledge that have been fun, I really didn't want to lose sight of the fact that the entire global Internet really did just dodge a bullet. And we don't know which foreign or domestic cyber intelligence services may today be silently saying *"Darnit!! They found it... that was one we were keeping in our back pocket for a rainy day, while keeping all of our foreign DNS server targeting packages updated."*

So what are these four different attacks?

SigJam utilizes an attack with one key and many signatures. They write:

The RFC advises that a resolver should try all signatures until a signature is found that can be validated with the DNSKEY(s). This can be exploited to construct an attack using many signatures that all point to the same DNSSEC key. Using the most impactful algorithm, an attacker can fit 340 signatures into a single DNS response, thereby causing 340 expensive cryptographic signature validation operations in the resolver until the resolution finally terminates by returning a SERVFAIL response to the client. The SigJam attack is thus constructed by leading the resolver to validate many invalid signatures on a DNS record using one DNS key.

The LockCram attack does the reverse, using many keys and a single signature. They write:

Following the design of SigJam we develop an attack vector we dub LockCram. It exploits the fact that resolvers are mandated to try all keys available for a signature until one validates or all have been tried. The LockCram attack is thus constructed by leading a resolver to validate one signature over a DNS record using many keys. For this, the attacker places multiple DNS keys in the zone which are all referenced by signature records using the same triple of (name, algorithm, key tag). This is not trivial, as resolvers can de-duplicate identical DNSKEY records

*and their key tags need to be equal. A resolver that tries to authenticate a DNS record from the zone attempts to validate its signature. To achieve that, the resolver identifies all the DNSSEC keys for validating the signature, which, if correctly constructed, conform to the same key tag. An RFC-compliant resolver must try **all** the keys referred to by the invalid signature before concluding the signature is invalid, leading to numerous expensive public key cryptography operations in the resolver.*

The next attack "KeySigTrap" combines the two previous attacks by using multiple key and multiple signatures. They explain:

*The KeySigTrap attack combines the many signatures of SigJam with the many colliding DNSKEYs of LockCram, creating an attack that leads to **a quadratic increase** in the number of validations compared to the previous two attacks. The attacker creates a zone with many colliding keys and many signatures matching those keys. When the attacker now triggers resolution of the DNS record with the many signatures, the resolver will try the first key to validate all signatures. After all signatures have been tried, the resolver will move to the next key and again attempt validation with all signatures. This continues until all pairs of keys and signatures have been tried. Only after attempting validation on all possible combinations does the resolver conclude that the record cannot be validated and returns a SERVFAIL to the client.*

Elsewhere in their report when going into more detail about this KeySigTrap, they explain that they were able to set up a zonefile containing 582 colliding DNSSEC key and the same 340 signatures that we saw in SigJam. Since the poor DNS resolver that receives this response will be forced to test every key against every signature, that 582 times 340 or 197,880 expensive and slow public key cryptographic signature tests. And that was caused by sending that DNS server a single DNS query packet.

Interestingly, the researchers discovered that not all DNS servers were similarly affected. For some reason, several took the request much harder. Upon investigating they discovered why.

The Unbound DNS server is DoSed approximately 6 times longer than the other resolvers. The reason is the default re-query behavior of Unbound. In a default configuration, Unbound attempts to re-query the nameserver five times after failed validation of all signatures. Therefore, Unbound validates all attacker signatures 6 times before returning a SERVFAIL to the client. Essentially, Unbound is being penalized for being a good citizen. Disabling default re-queries brings Unbound back to parity with the other.

But BIND was the worst with the 16-hour DoS from a single packet. They explained:

*Investigating the cause for this observation, we identified an inefficiency in the code, triggered by a large number of colliding DNSSEC keys. The routine responsible for identifying the next DNSSEC key to try against a signature does **not** implement an efficient algorithm to select the next key from the remaining keys. Instead, it re-parses all keys again until it finds a key that has not been tried yet. This algorithm does not lead to inefficiencies in normal operation where there might be a small amount of colliding keys. But when many keys collide, the resolver spends a large amount of time parsing the keys to select the next key, which extends the total response duration of the DoS to 16h.*

What all of this should make clear is that these potential problems arise due to DNSSEC's deliberate "eager to validate" design. DNS servers really want to attempt all variations, which is exactly what gets them into trouble. The only solution will be something heuristic. We've talked about heuristics in the past. They can be thought of as a rule of thumb, and they usually appear when exact solutions are not available – which is exactly the case here.

As we might expect, the researchers have an extensive section of their paper devoted to "what to do about this mess," and they worked very closely for months with all of the major DNS system maintainers to best answer that question. I'll skip most of the blow-by-blow, this but give a sample solution. Under the subhead "Limiting all validations" they wrote:

> *The first working patch capable of protecting against all variants of our attack was implemented by Akamai. In addition to limiting key collisions to 4, and limiting cryptographic failures to 16, the patch also limits total validations in ANY requests to 8.*
>
> *Evaluating the efficacy of the patch, we find the patched resolver does not lose any benign request even under attack with > 10 attacking requests per second. The load on the resolver does not increase to problematic levels under the ANY type attack with 10 req/s, and the resolver does not lose any benign traffic. It thus appears that the patch successfully protects against all variations of KeyTrap attacks.*
>
> *Nevertheless, although these patches prevent packet loss, they still do not fully mitigate the increase in CPU instruction load during the attack. The reason that the mitigations do not fully prevent the effects of the KeyTrap attacks is rooted in the design philosophy of DNSSEC. Notice however that we are still closely working with the developers on testing the patches and their performance during attack and during normal operation.*

The disclosure timeline for this is extra-interesting since it provides a good sense for the participants and the nature of their efforts and interactions:

In the following, we describe the timeline of disclosure to indicate how the vulnerability was reported and how we worked with the experts from industries to find solutions for the problems that we discovered.

- 02.11.2023: Initial disclosure to key figures in the DNS community. Both confirm that KeyTrap is a severe vulnerability, requiring a group of experts from industry to handle.

- 07.11.2023: Confidential disclosure to representatives of the largest DNS deployments and resolver implementations, including Quad9, Google Public DNS, BIND9, Unbound, PowerDNS, Knot, and Akamai. The group of experts agrees that this is a severe vulnerability that has the potential to cut off internet access to large parts of the Internet in case of malicious exploitation.

  A confidential chat-group is established with stakeholders from the DNS community, including developers, deployment specialists and the authors. The group is continuously expanded with additional experts from the industry to ensure every relevant party is included in the disclosure. Potential mitigations are discussed in the group.

- 09.11.2023: We share KeyTrap zonefiles to enable developers to reproduce the attack locally, facilitating the development of mitigations

- 13.11.2023: Akamai presents the first potential mitigation of KeyTrap by limiting the total number of validation failures to 32. (That doesn't work)

- 23.11.2023: Unbound presents its first patch, limiting cryptographic failures to a maximum of 16, without limiting collisions.

- 24.11.2023: BIND9 presents the first iteration of a patch that forbids any validation failures.

- 08.12.2023: An umbrella CVE-2023-50387, is assigned to the KeyTrap attacks.

- 02.01.2024: After discussions with developers, we find some have problems recreating the attack in a local setup. We thus provide them an updated environment with a DNS server to ease local setup and further facilitate testing of patches. (In other words, the researchers actually put this live on the Internet so that the DNS developers would DoS their own DNS servers.

- 03.01.2024: BIND9 presents the second iteration of a patch, limiting validation failures.

- 03.01.2024: Our newly implemented DS-hashing attack proves successful against all mitigations not limiting key collisions, including BIND9 and Unbound, and is disclosed to the group.

- 16.01.2024: Our ANYTYPE attack circumvents the protection from limiting colliding keys and limiting cryptographic failures.

- 24.01.2024: The first working patch is presented by Akamai. Other resolvers are implementing derivations of the countermeasures to protect against the attacks.

And so now we get to their final conclusions. They write:

*Our work revealed a fundamental design problem with DNS and DNSSEC: Strictly applying Postel's Law to the design of DNSSEC introduced a major and devastating vulnerability in virtually **all** DNS implementations. With just **one** maliciously crafted DNS packet an attacker could stall almost any resolver, for example the most popular one, BIND9, for as long as 16 hours.*

The impact of KeyTrap is far reaching. DNS evolved into a fundamental system in the Internet that underlies a wide range of applications and facilitates new and emerging technologies. Measurements by APNIC show that in December 2023, 31.47% of web clients worldwide were using DNSSEC-validating resolvers. Therefore, our KeyTrap attacks have effects not only on DNS but also on any application using it. An unavailability of DNS may not only prevent access to content, but risks also disabling security mechanisms, like anti-spam defenses, Public Key Infrastructure (PKI), or even inter-domain routing security like RPKI or rover.

Since the initial disclosure of the vulnerabilities, we've been working with all major vendors on mitigating the problems in their implementations, but it seems that completely preventing the attacks will require fundamentally reconsidering the underlying design philosophy of DNSSEC, in other words, to revise the DNSSEC standards.

https://www.athene-center.de/fileadmin/content/PDF/Technical_Report_KeyTrap.pdf

So, as we can see, as I titled this podcast, The Internet really did dodge a bullet.

It is also terrific to see that at the technical and operational level we have the ability to quickly gather and work together to resolve any serious trouble that may arise.

Fortunately, that doesn't happen often.