Electronic Communications of the EASST
Volume 080 (2021)

Conference on Networked Systems 2021
(NetSys 2021)

Implementing DNSSEC soft delegation for microservices

Andrés Marín-López, Patricia Arias-Cabarcos, Thorsten Strufe, Gabriel Barceló-Soteras,
Daniel Díaz-Sánchez, Florina Almenares-Mendoza

5 pages

Guest Editors: Andreas Blenk, Mathias Fischer, Stefan Fischer, Horst Hellbrueck, Oliver Hohlfeld, Andreas Kassler, Koojana Kuladinithi, Winfried Lamersdorf, Olaf Landsiedel, Andreas Timm-Giel, Alexey Vinel
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Implementing DNSSEC soft delegation for microservices

Andrés Marín-López[1 2*], **Patricia Arias-Cabarcos**[1], **Thorsten Strufe**[1†], **Gabriel Barceló-Soteras**[3], **Daniel Díaz-Sánchez**[2], **Florina Almenares-Mendoza**[2]

[1] Karlsruhe Institute of Technology, Germany
[2] Universidad Carlos III de Madrid, Spain
[3] Parliament of the Balearic Islands, Spain

Abstract: Securing DNS in Edge- and Fog computing, or other scenarios where microservices are offloaded, requires the provision of zone signing keys to the third parties who control the computing infrastructure. This fundamentally allows the infrastructure provider to create novel signatures at their discretion and even arbitrarily extend the certificate chain.
Based on our proposal on soft delegation for DNSSEC, which curtails this vulnerability, we report on our proof-of-concept: a C-implementation of chameleon hashes in OpenSSL, a server side implementation of the mechanism in the ldns server, and an offline client that validates the signed records, in this paper. We also discuss different approaches for generating DNSSEC RRSIG records, and the behavior of a resolver to verify the credentials and securely connect to an end point using TLS with SNI and DANE.

Keywords: DNSSEC; DANE; chameleon signatures; IoT; microservices; Fog computing

1 Introduction

Interconnection of microservices requires name resolution and security support. DNS and TLS are by far the most used protocols for this purpose. Local Certification Authorities (CAs) and self-issued certificates are supported in TLS, and suit in dynamic microservices scenarios. Consider a group of off-shore windmills with dense local connectivity but expensive connection to the mainland, which is coordinated and managed by a local set of edge-servers, as one example. From the core of the network, several microservices are selected to be instantiated at the nodes in the windmills. Once instantiated, the microservices will setup secure interconnections. Most connections must be resolved inside the fog, including name resolution and certificate verification. Our proposal in [DMAA19] was aimed at minimizing the intervention of the core of the network, especially the DNS at the core (from now on the DNScore) with a soft delegation to the DNS of the distant computing nodes, which we'll call DNSfog. We proposed using chameleon hashes [Kra00]. They allow an entity that possesses trapdoor information to create collisions – in

* Funded by the Comunidad de Madrid project CYNAMON P2018/TCS-4566 and co-financed by European Structural Funds (ESF and FEDER)
† Funded by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy – EXC 2050/1 – Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden.

other words to compute signatures that verify alternative, arbitrary inputs. Chameleon signatures are a lightweight solution compared to normal DNSSEC delegation. Signed patterns impose the limits of the delegation to the DNSfog to only matching RRs.

2 DNS Security

DNS handles objects called resource records (RRs) of different types (`A`/`AAAA`, `MX`, name server `NS`, etc.) DNS clients query for RRs matching a domain name (also known as owner field). DNSSEC adds integrity protection to DNS, to prevent cache poisoning and spoofing. DNSSEC clients are provided with signatures to verify the integrity of the resource records in the response. Validation requires the verification of the signatures, and DNSSEC introduces key chains for that matter. The chains originate at the root zone ".". The zone records on each level are signed by a zone signing key (ZSK). It's integrity is verified by a using local key signing keys (KSK), which in turn are signed by the ZSK of the parent zone in the DS records, basis for trust chain extension to a delegated zone.

DNSSEC responses include the signatures (RRSIG) of the RRs. As in DNS, the answer may optionally include (1) the authoritative part including the name servers (NS) and corresponding RRSIGs, and (2) the additional part, the prefetching mechanism of DNS, with the RRs the server anticipates will be required by the client, for instance the A or AAAA records of the NS in the authoritative part, and corresponding RRSIGs.

The DNS-Based Authentication of Named Entities (DANE) specification [RFC6698] introduces the TLSA RR type. TLSA associates a certificate or a public key of an end-entity or a trusted issuing authority with the TLS or DTLS transport endpoint. DANE TLSA records validated by DNSSEC can be used to augment or replace the use of trusted public CAs. TLS clients can use the service name indication (SNI) when negotiating the connection with a TLS server, and DANE supports SNI by TLSA records which can hold: a credential for the server (PKIX-EE); a domain name Trust Anchor (TA) linking the authorized issuer CA for the domain (PKIX-TA); a local CA entitled to issue trusted certificates for the domain (DANE-TA); and even to indicate an ad-hoc key for the TLS server (DANE- EE).

3 DNSSEC soft delegation

Considering the usual dynamics in microservice deployments, DNSSEC would require the server to provide signatures for every local change. We hence propose soft delegation, allowing the DNScore to delegate signing of the microservices using chameleon signatures to the DNSfog.

Let us illustrate how the chameleon signatures work using discrete logarithms [AM05]:

1. Let DNSfog selects q a large prime, $p = kq+1$, and g, a generator of $\mathbb{Z}_p^*$.
   Let $SK_f = x \in (1, q-1)$ be the trapdoor, with corresponding public key $PK_f = y = g^x \bmod q$.
   Let (PK$_S$,SK$_S$) be the public and private keys of DNScore (the signer).
2. DNSfog sends the DNScore two random parameters a and b. The DNScore takes message M and computes $e = \text{SHA}_{256}(M, b)$,
   $\text{CHAM-HASH}_f(M, a, b) = a - (y^e g^b \bmod p) \bmod q$.
3. The DNScore signs it: $sig = SIG_S(RRSIG\_RDATA|a|b|\text{CHAM-HASH}_f(M, a, b))$.

4. The signature can be verified with $(sig, M, a, b, p, g, q, y, PK_S)$.
5. When DNSfog wants to modify the message M', it picks a new random l,computes the collision for the chameleon hash, the hash of M', and the corresponding new b':

$$a' = \text{CHAM-HASH}_f(M, a, b) + (g^l \bmod p) \bmod q \tag{1}$$

$$e' = \text{SHA}_{256}(M', a') \tag{2}$$

$$b' = k - e'x \bmod q \tag{3}$$

The new message signature can be verified with $(sig, a, b, M', a', b', p, g, q, y, PK_S)$, which means that the signature sig still holds for a modified M'. Note that to verify the signature both the original parameters (a,b) and the new parameters (a’,b’) are required. That is how the DNScore limits the delegation to DNSfog by only allowing to generate modifications M' from message M.

M is obtained by hashing the serialization of a specially crafted RR, where the owner is modified to contain a placeholder (`<T>`). We call this RR a pattern. An A pattern looks like:
`<T>.example.com. 3600 IN A rnd`
The pattern is serialized as:
`RRpattern=(owner|type|class|ttl|len(RDATA)|RDATA), RDATA=<T>.example.com|rnd.`
And the message is computed as M= `sha256(RRPattern).`

In our soft delegation approach, the DNScore will generate the signature
`sig=sign(RRSIG_RDATA|a|b|CHAM-HASH(M,a,b)).`
The signature is thus bound to the (a,b) parameters and to the chameleon hash of the pattern, which includes the DNSfog public key `y`. The public parameters `y,p,g,q` are in a *DS* record with private OID at the DNScore.

The record generated by the DNScore will be like:
`<T>.example.com 3600 IN RRSIG A 254 3 ttl d1 d2 32478 example.com. B64(sig,a,b)`
The algorithm value 254 (`PRIVATE_OID`) indicates that the verification is different from standard. The (a,b) parameters are concatenated to the signature so that they can be recovered for verification.

To create a new RR from the pattern, the DNSfog computes M' as the sha256 of the serialization of the new RR, computes the parameters (a’,b’) to have a collision of the chameleon hash, and concatenates the (a’,b’) parameters to the signature.

When the verifier gets the RRSIG of the new RR, it will also query for the corresponding pattern RR and RRSIG. It will compute the chameleon hash of the RRPattern using the (a,b) parameters from the RRSIG of the pattern. Next it will check the signature validity using the public key of the DNScore. Then it checks that the chameleon hash of the new RR with (a’,b’) holds: `CHAM-HASH(M,a,b)=CHAM-HASH(M',a',b')`

Finally, the verifier also has to check if the owner, type and TTL match the ones in the pattern.

4 Implementation and Evaluation

The DNSfog has two alternatives to compute the collisions:
Approach 1: compute a single RRSIG for the whole RRset of all the created microservices, or
Approach 2: multiple RRSIGs: one per each of the created microservices.
In approach 1, the signature is performed on the hash of the whole RR set. Since no new RRSIG

are added, there is no need to recalculate the NSEC/NSEC3 with each RR added. But this approach does not protect against domain enumeration, since the responses MUST include ALL the RRs to verify the signature of the RR set, disclosing thus all the existing A/TLSA records. Besides responses size may exceed the limit of UDP.

In approach 2, each microservice will have its own RRSIG, so equal number of NSEC/NSEC3 RRs have to be added. This approach provides domain enumeration protection but requires computing an extra collision for the NSEC/NSEC3 records, on the other hand responses are shorter, since only the required domain name A/TLSA are returned with the corresponding RRSIG. We have opted in our implementation for approach 2.

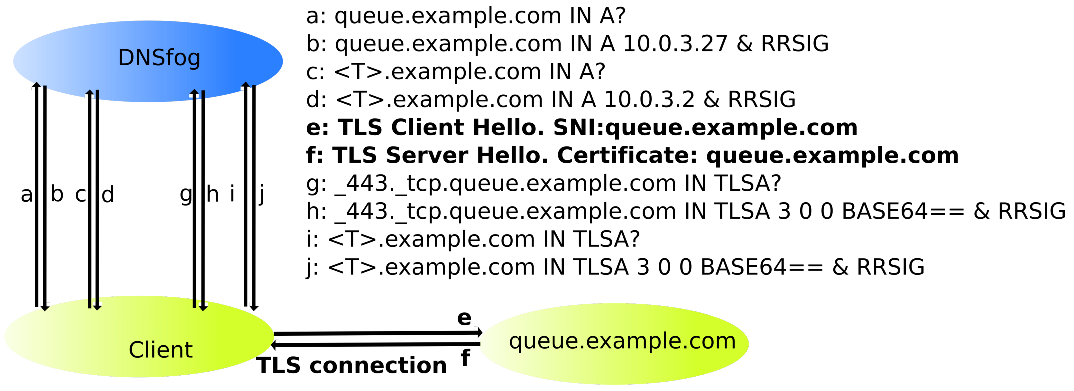
a: queue.example.com IN A?
b: queue.example.com IN A 10.0.3.27 & RRSIG
c: <T>.example.com IN A?
d: <T>.example.com IN A 10.0.3.2 & RRSIG
e: TLS Client Hello. SNI:queue.example.com
f: TLS Server Hello. Certificate: queue.example.com
g: _443._tcp.queue.example.com IN TLSA?
h: _443._tcp.queue.example.com IN TLSA 3 0 0 BASE64== & RRSIG
i: <T>.example.com IN TLSA?
j: <T>.example.com IN TLSA 3 0 0 BASE64== & RRSIG
DNSfog
a
b
c
d
g
h
i
j
Client
e
TLS connection
f
queue.example.com

Figure 1: Verification of microservice queue.example.com

Figure 1 shows a microservice named `queue.example.com` instantiated by the orchestrator at IP 10.0.3.27 with a self-signed certificate entirely included in the TLSA record (DANE-EE). The figure illustrates the whole process, asuming the client has already the public keys PK_S which is the zsk `DNSKEY` of the DNScore and the public parameters `p,g,q,y` which are in the DS record with private OID signed by the DNScore, i.e. the corresponding delegated public key of the DNSfog.

5 Discussion and Conclusions

DNSSEC delegation is performed by signing a DS. Normal delegation gives the DNSfog authority over a whole subdomain. An attacker may extract the delegated private key, if the DNSfog is in a remote site only visited for maintenance, as in the wind mill scenario. Our proposal limits the delegation to a set of types (patterns), in the same way as we limit most verification traffic to the fog. In our proposal if the DNSfog is attacked, the attacker can only use the private key to alter RRs according to the pattern, and only locally at that fog.

Online signing requires the DNScore signing a key for each DNSfog and the key to be available at the DNSfog. This implies the above mentioned risk for remote unattended sites. In [DMAA19] we show that the collision computation time is negligible when compared with the signing time with RSA or ECDSA, while the time of computing the chameleon hash is 7 times slower than RSA signature, if advanced factorization is used. This increased load is pushed from the DNScore to the DNSfog, allowing the schema to scale.

The delegation increases the trust chain so that resolvers must also trust in the DNSfog. Nevertheless, this is the same as in normal delegation scheme or in the PKI trust model, where the trust is rooted at the top CAs, but in the end you have to trust also the final entity presenting a certificate. At least for the time it takes a private security incident to be noticed and notified, and the final entity certificate to be revoked and the CRL to be published or pushed to the OCSP server.

Trust is bounded by the patterns signed by the DNScore. They limit the delegation to the DNSfog, which has the trapdoor to add, delete or modify RRs according to the patterns. We call this a soft delegation since it is limited to the signed patterns: only RRs matching the delegated patterns can be modified by the DNSfog. The soft delegation alleviates the DNScore from the necessity of signing the whole zone with every new change in the microservices deployment in the Fog. The DNScore only has to sign the delegated patterns using the chameleon hash. The DNSfog will compute the collision when a new RR is required using eqs. 1-3. We expect delegation to patterns of types A, AAAA and TLSA records.

In an extended version of this article we will address more details of the implementation. Our results from the server-side integration of the proposed soft delegation mechanism in the open source ldns dnssec server show that the Chameleon hashes and the collisions are computed faster than in [DMAA19]. The proof of concept server side implementation is available at https://gitlab.pervasive.it.uc3m.es/100416942/ldns-tools-chamhash. Besides we have developed an offline verifier which takes as input a JSON file containing all the required records to validate the signatures of DNSSEC records. We have verified that common cases of requests in microservice environments are protected through the use of the offline verifier together with some scripts and a python wrapper to invoke ldns. This can be found at https://gitlab.pervasive.it.uc3m.es/100416942/core-fog-scripts.

Bibliography

[AM05] G. Ateniese, B. de Medeiros. On the Key Exposure Problem in Chameleon Hashes. In Blundo and Cimato (eds.), *Security in Communication Networks*. Pp. 165–179. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[DMAA19] D. Díaz-Sánchez, A. Marín-Lopez, F. Almenárez Mendoza, P. Arias Cabarcos. DNS/DANE collision-based distributed and dynamic authentication for microservices in IoT. *Sensors* 19(15):3292, 2019.

[Kra00] H. R. Krawczyk. T: Chameleon signatures. In *Proceedings of NDSS*. Volume 2000, pp. 143–154. 2000.