DNSSEC: Trust along the chain

Article by Alexander Venedyukhin, Lead Analyst at TCI

Anti-spoofing mechanisms built into the “classic” domain name services are tied to the properties of the DNS exchange protocol, and therefore prevent only the most primitive attacks. For example, these mechanisms do not allow for counteracting active packet spoofing at the network transport layer, which means that any intermediate node can completely replace the addressing picture seen through DNS. The wide variety of threats in today’s global network calls for a universal security tool that protects name and address information regardless of the low-level properties of DNS and network transport. Such an add-on has been available for a long time – it is based on a digital signature and is called DNSSEC.

Signature valid

The main logic feature of DNSSEC is that this technology works at a different layer than the DNS protocols. Yes, DNSSEC keys and signatures are published in the same way as other DNS resource records, but the received data is authenticated outside the domain system.

This means that authorized and authenticated addressing data within a given zone do not even need to be obtained through DNS. Most important is that they are accompanied by correct DNSSEC signatures, and the corresponding records can be received via any channel, not only a trusted one. This is a significant advantage when viewed from an application standpoint. For an application that can verify DNSSEC signatures by building a chain of trust from a local copy of the root key, it becomes less important whether an available source of DNS records, such as an access provider's resolver, is trusted. The same can be said about any intermediate node that could spoof information on the way from a server to a client: DNSSEC detects such spoofing.

In the “classic” DNS, there are also safeguards based on similar algorithms (this is, first of all, TSIG), but they are available only to those nodes that have previously agreed on a shared secret. It is clear that this option is not suitable for public and open use, when an arbitrary node can contact DNS, since this node cannot know the secret key in advance. DNSSEC allows an "indefinite range" of client programs to effectively use the well-known public-key hierarchy scheme in which signature chains are built and the required keys can be securely obtained over an insecure channel.

So, DNSSEC solves the problem of ensuring data integrity: a party that received data from DNS can check that this data has not been changed on the way from the authoritative server, and that this information is published in the domain zone by operator – someone who has access to the corresponding secret key(s) and thus can generate DNSSEC signatures that authenticate DNS records. In most practical cases, this is the domain zone registry.

Algorithms

Let's take a look at how DNSSEC works using a simple example. Suppose the client needs to determine an IP address corresponding to the test.ru name. The test.ru zone is safe, that is, the name servers to which it is delegated support DNSSEC. DNSSEC is also supported in the higher zones (.ru and root zone). In order to obtain an IP address, one needs to perform a DNS search for an A Record (address Mapping record). The client resolver that supports DNSSEC will receive from DNS not only an A Record for test.ru, but also a number of signature values that certify the transferred values' authenticity. The signatures are sent in additional DNS records called RRSIG. Public keys are required to verify signatures. The relying party obtains keys (except for the root key) from DNS as well. The public key sets correspond to domain zones. So, in the case of test.ru, the resolver must receive keys for the test.ru zone, for the .ru zone, and for the root zone. The required keys are passed in the DNSKEY records (DNSSEC has two types of keys: KSK for "key-signing key" and ZSK for "zone-signing key." Further, for simplicity, these types are not distinguished, except for the root KSK). Key value records also get signed. This is critically important: a mechanism is needed to verify the authenticity of the key values.

Authentication requires the transfer of trust across the entire hierarchical chain of the zones, which means that the .ru zone must contain data that make it possible to authenticate the keys received for the test.ru zone. Technically, the .ru zone in our example delegates administration rights to the test.ru zone. In DNSSEC, such delegation must be accompanied by trust transfer. By the way, if the delegation in this particular case did not occur, then the response to the request for the A Record for test.ru, along with all signatures, would have been sent by the .ru zone servers (of course, only if such a name is available in the zone). The same arrangement works for names like www.test.ru if they are located directly in the test.ru zone.

Delegating in DNSSEC is just as important as in the "classic" DNS, but is arranged in a slightly different way. As a reminder, in DNS, the delegation is implemented by sending, in response to a request from a resolver (client), a list of authoritative server names that correspond to a given DNS zone. In the case of DNSSEC, the DNS delegating response is saved and works the same way. However, the list of servers in the delegating response is not certified by a signature, since trust is

built in DNSSEC through cryptographic keys. Each standalone domain zone has its own set of DNSSEC signing keys because different registries are not required to use the same keys. Since the signatures within the zones are also isolated, the task arises of checking that the signatures that the client sees correspond to the domain zone registry's keys, and have not been replaced on their way to the client.

Indeed, suppose a malicious user intercepts all of the client's DNS requests and can send arbitrary responses. In this case, the malicious user could generate their own sets of keys for test.ru, identify the signatures from these keys, and send the required data set, along with the correct signatures, to a client under attack. This, however, is a well-known problem inherent in almost all mass digital signature schemes. The DNS construction logic requires that the keys corresponding to a specific domain zone be published in it. So, to counter the above attack, DNSSEC uses a secure delegation mechanism: a fingerprint of the trusted public key of the delegated zone is placed in the delegating zone; a special DS record is used for this purpose. The key fingerprint value is signed using the delegating zone keys, which makes it possible to ensure that the key has not been tampered with. (The fingerprint is calculated using a cryptographic hash function, so tampering with the key and having the fingerprints match as a result will not work.)

So, a signature is placed in the delegating zone certifying the delegated zone key fingerprint. The chain of trust is built as follows: having received a key from a particular zone, the resolver calculates its fingerprint and matches the value with the fingerprint received from the higher-level zone, which is certified by the signature. The process continues until it reaches the root key (KSK). Let's go back to the example with test.ru and the A Record. The resolver receives a set of keys corresponding to the test.ru zone and checks to see that at least one of them matches the DS record for test.ru, which is located in the .ru zone (that is, in the zone that is one level higher). For the .ru zone, the steps are similar, but the DS record is already requested from the DNS root. The presence of DS records makes it possible to build a hierarchical chain of keys and verify the signatures. The DNS root key starts this chain and is used to authenticate the set of keys for the root zone. The root public key must be obtained through a trusted channel other than DNS. Typically, this key is built into the distribution package of the client application that performs the DNSSEC validation.

Trusted confirmation of the absence of records is of particular interest. Suppose a malicious user wants to spoof responses about the test.ru zone by intercepting

traffic. In that case, they could simulate the lack of DNSSEC support for the test.ru zone by substituting the name servers' responses related to DNSSEC, namely, a response with a DS record (either for the ru zone or for test.ru). If a zone does not support DNSSEC, then spoofing the answers related to it is not a problem: no signatures will need to be forged. In other words, the possibility of such an attack negates all the DNSSEC benefits: a malicious user only needs to simulate the absence of DNSSEC.

Of course, DNSSEC has built-in protection mechanisms against this attack: the fact that a certain record is missing in the zone is also confirmed with the use of a digital signature. To do so, special schemes are used to sign the “intervals” between possible names and types of records confirming the fact that these intervals are “empty” indeed. In particular, signed records from a higher zone are required for a DNSSEC validating resolver to declare a zone “unsafe” (that is, without DNSSEC support).

So, DNSSEC makes it possible to publish tamper-proof data to DNS. A DNSSEC-capable client can verify the authenticity of the addressing information, regardless of the channel which was used to receive the information.

Key management

The modern version of DNSSEC accommodates the use of various digital signature algorithms, including ECDSA (the elliptic curve algorithm) and RSA. The global DNS root key still uses RSA, even though the asymmetric cryptography algorithm is now considered obsolete and ECDSA a better option.

ECDSA uses keys derived from elliptic curve cryptography. ECDSA keys are much shorter than RSA keys, which means data packets required to transmit DNS responses are also smaller. In some cases, ECDSA offers ways for additional optimization of server-side operations (which, again, is due to shorter keys compared to RSA). There are also Russian digital signature algorithms using elliptic curves, known collectively as GOST Signature (GOST 34.10 family standards).

GOST Signature for DNSSEC was proposed even before ECDSA, since for some time, it was almost the only elliptic curve cryptosystem that had the status of a

state standard, with minimum potential problems related to patent protection. The ECDSA cryptosystem and the implementation of a number of cryptographic operations were potentially covered by the numerous Certicom patents (until 2014-2018; now the most restrictive patents have expired). Problems arising from the protection of exclusive rights to the methods used for the implementation of cryptographic operations were definitely a major obstacle to using open standards.

Therefore, GOST Signature has been supported in the DNSSEC protocol for a long time. However, the corresponding Russian standard has changed over the time that has passed since it was implemented. The old version of the signature, GOST R 34.10-2001, is no longer recommended for use. That standard has been replaced by a new one, GOST R 34.10-2012, and its use requires an RFC update. On the Technical Center of Internet side, Dmitry Belyavsky leads the work on new standards for IETF. The corresponding draft RFC, developed by Russian specialists, is now under discussion at IETF working groups.

It should be noted that DNSSEC allows several different cryptosystems to be used to protect the same domain zone. This means Russia's GOST cryptography can be used both separately and together with ECDSA or RSA.

Since the DNS root zone was signed by DNSSEC in 2010, IANA is the operator of the key signing key for the DNS root zone (or "an organization performing the IANA naming function"). It makes sense because IANA is technically responsible for management of the DNS root zone. The RZ ZSK operator is Verisign, performing the function of generating the zone signing keys (ZSKs) on behalf of IANA and ICANN. Verisign signs the root zone file using the ZSKs, which in turn must be signed with the root key-signing key (the root KSK is the master key stored outside the DNS). Copies of the root KSK are stored with a high degree of protection, and access to them is strictly regulated. They have to be accessed whenever the root ZSK is replaced.

Initially, the plan was for new ZSKs to be signed in a special face-to-face ceremony with trusted persons arriving at a secure vault from different parts of the world, including crypto officers who hold parts of the secret access keys, as well as observers. This procedure ruled out the possibility of unauthorized use of the keys, and generally enhanced confidence in the entire system.

However, in 2020, face-to-face interactions had to be suspended due to the global situation, and a decision was made to generate several ZSK sets to create a

reserve for the future. Instead of holding the next meeting offline, the previously generated keys were transferred to Verisign according to the ZSK rotation schedule. Perhaps face-to-face meetings will resume later, although the changed practice has questioned their actual importance and even necessity.

The DNSSEC key hierarchy is similar to that of TLS certificates used by web browsers. The main difference is that DNSSEC now uses a single root, a single root key that logically matches the single root of the domain name system itself. With TLS, there are many root certificates built into web browsers: at least one root certificate and root key for each of the dozens of known and trusted root CAs. TLS certificates are also associated with domain names, since the purpose of a certificate is to make sure that a certain server key matches a certain domain name (less often, a certain IP address). It might seem that TLS certificates and the TLS protocol implement the same security mechanism as DNSSEC.

Indeed, if a client (web browser) connects to a website whose IP address has been spoofed at the DNS level, the client will still be able to authenticate the server's TLS certificate and the name match. A CA signs the certificate, certifying that they have verified that it belongs to the owners of the domain name, meaning it validates the identity of the host. This is all true, but the scope of DNSSEC is much bigger.

First, the DNSSEC protocol is applied in all other cases where DNS is used, not just when it comes to the web and web browsers. Second, even with a website and its A record (IP address), DNSSEC can detect spoofing (and thereby detect an attack) earlier than TLS. This greatly reduces the actual attack surface because the client, having detected the spoofing, will not even connect to the attacker's host. This is very important because if the attacker achieves a connection attempt, they can use more methods to succeed. Those include non-technology methods such as social engineering: for example, in a spear-phishing scam targeting a corporate system, users receive mail from “technical support” informing them about an alleged “TLS certificate replacement” on the website and asking them to open the corporate portal page and “agree with the browser warning.”

Worse still, an attacker can somehow obtain a valid certificate for the name they are trying to attack. True, in DNSSEC, a leak of private keys is also possible; or, they can be replaced by intercepting the administrative control of the domain zone. But one has to agree that spoofing both TLS and DNSSEC is more difficult. Moreover, if an attacker obtains DNS control with sufficient rights to replace

DNSSEC keys, then most likely they will be able to obtain a valid TLS certificate for the domain name as well (because the control mechanisms use DNS). But the opposite situation – the attacker obtaining the private key from the web server's TSL certificate – won't allow DNS spoofing.

Two other technologies are often compared to DNSSEC – DNS-over-TLS (DoT) and DNS-over-HTTPS (DoH). Both DoT and DoH are primarily aimed at hiding the content of DNS requests and authenticating nodes. These technologies actually have nothing to do with DNS records proper. In contrast, DNSSEC is not only based on publishing DNS records, but it also protects the data itself, not the channels to access it. DoH, and especially DoT, is a good complement to DNSSEC, but on the downside, it does not allow any additional information to be placed in DNS in a trusted way. Meanwhile, this feature of DNSSEC is exactly what significantly expands the capabilities of DNS.

Prospects

DNSSEC provides DNS with an efficient authentication mechanism for all records. This means it allows for the publication of data used by various applications in a trusted way. These could be cryptographic key fingerprints used in other protocols, the description of policies for issuing TLS certificates for the corresponding zone, etc. Naturally, the relevant records (SSHFP, TLSA, CAA and others) can be placed in DNS without using DNSSEC, but this feature helps avoid the spoofing of identifiers and parameters in these records.

Unfortunately, DNSSEC support is not widespread yet. As of June 2021, .ru has DS records for some 5,500 domain zones, which is 0.1 percent. This is mainly due to the fact that the introduction of DNSSEC requires the modernization of the authoritative name servers and management tools, and very few are willing to perform the updates.