Overview

In this document, the implementation of an iterative DNS resolver with support for validation using the DNSSEC protocol is described.

Implementation

Iterative DNS resolver

The implementation of the iterative DNS resolver is almost identical to the one in part A of the assignment. The only addition is that, for every DNS request, the response received from the DNS name server(s) is validated with the dnssec protocol using the *validate_response* method.

DNSSEC validation

The DNSSEC validation can be broadly broken into two parts.

Validating DNS records using RRSIGs

For every DNS record fetched from DNS nameservers, they need to be validated with their corresponding RRSIG records, to ensure that the signature matches, which proves that the DNS record RRSET has not been tampered with. This is done using the *dns.dnssec.validate* method in the dnspython library. For this method to work, all the relevant public keys(ZSKs and KSKs) of the DNS zones corresponding to the DNS request need to be present locally. This will be discussed more in the next section(s).

Managing public keys

We use DNSKEY requests to fetch the public ZSKs(Zone Signing Key) and KSKs(Key Signing Key) of each DNS zone in the path to resolving a DNS query. The ZSK is validated by using the *dns.dnssec.validate* method to match the DNSKEY record with the corresponding RRSIG in the response message to the DNSKEY request. Here, the RRSIG is decrypted using the KSK. Every time we validate a public DNSKEY record, we store it in a hash map with the corresponding DNS zone(for example, ‘com.’) which can be used to validate further DNS records with their RRSIG signatures.

Validating KSKs using DS records

DS records obtained from a nameserver in the parent domain contain a hash of the KSK of the child domain. The KSK can be converted to its corresponding DS record using the *dns.dnssec.make_ds* method in the dnspython library. The generated DS record is then compared with the DS record obtained from the parent nameserver. This way, we establish a ‘chain of trust’ from the root servers all the way to the authoritative name server.

Fetching DNSKEY records for the root server

Since we establish a chain of trust to verify the authenticity of DNSEY records, it is not possible to do the same for the public keys of the DNS root zone, as they do not have a parent zone. We work around this by fetching the DNSKEY records for the root zone at the start of the program and assuming that it is authentic. This seems like a reasonable assumption, as the root DNS keys are publicly available and can be verified, and doing the same programmatically seems like it is out of the scope of this assignment.

Deciding when to fetch DNSKEY records for a zone

Every time we receive a DS record, it corresponds to the next zone in the iterative DNS resolution. Therefore, we fetch and validate DNSKEY records for a zone whenever we receive its DS record.

Deciding when DNSSEC protocol is not supported

Whenever we have a case where a DS record is not present for a DNS zone, we assume that DNSSEC is not enabled in that zone, and stop the DNSSEC protocol.

Test cases

The DNSSEC implementation was tested with the following domains.

Domain	Result
verisigninc.com	DNSSEC enabled and validated successfully
dnssec-failed.org	DNSSEC enabled but validation failed when validating the zone’s KSK with the DS record
amazon.com	Did not get a DS record, so concluded that DNSSEC protocol is not enabled