Introduction

This document describes the project plan for OpenDNSSEC. The OpenDNSSEC Daemon (ODD) will be a daemon, responsible for enabling and managing DNSSEC in an existing DNS infrastructure. The target audience for ODD is reflected in its default settings. The mission is that this software can seamlessly integrate into an existing deployment scenario, without the necessity to overhaul the entire existing infrastructure, or to be closely acquainted with the DNSSEC protocol. The different possible configurable settings reflect a wide variety of possible deployment scenarios.

Two phased approach

The first phase will deliver a proof of concept. This will be various pieces of software (coded and scripted) bundled together to perform the desired tasks. It will only contain bare operating minimals. The second phase will deliver a production version. It is somewhat richer in features, but its performance will be better and much more stable.

Secure key storage

Securely storing a DNSKEY is a requirement for ODD. The most secure way to store a key is to use a Hardware Security Module (HSM). HSMs can use a wide variety of API's, though the most supported API is PKCS11. The ODD will use PKCS11 to outsource public key related, cryptographic functions. To be able to support an environment without the need for an HSM, ODD will be shipped with a virtual HSM that implements the PKCS11 standard.

Continuous signing

Continuous signing is a requirement because of zones that are so frequently updated, that the time between updates is smaller than the time to re-sign the zone.

Support multiple zones

The ODD needs to be able to massively scale. It should not only support a large amount of zones, it should also support a large number of DNSKEYs. This is to satisfy the need for large hosting outfits like ISPs. Note that in order to support a large number of zones, each zone might have its own, distinct set of secondaries.

Key and Signing Policy Enforcer

The Key and Signing Policy (KASP) Enforcer is a stateful engine which instructs (either active or passive) the signer which keys to use, revoke, schedule, etc at what time.

Protocol Compliance

The ODD will comply to RFC4033/4/5 and 5155 for DNSSEC. Furthermore, ODD will understand notify, axfr and ixfr. The ability to respond to general queries is an explicit non-requirement.

Security

Next to the DNSSEC protocol extensions, ODD will be able to handle TSIG and ACL based controls.

A-synchronicity.

The ODD will be able to handle incoming zonetransfers, or dynamic updates independent of outgoing transfers. In general, incremental changes to an unsigned zone leads to larger incremental changes to signed zones. In order to better manage the outgoing stream of updates, either allow a more flattened stream of updates, or a sporadic burst of bulk transfers, the ODD will have a fine tuned outgoing transfer mechanism independent of incoming transfers. Incoming transfers will be handled as they come in.

Architecture

Introduction

The architecture of ODD consists of several modules, working together as one. The core of the ODD is the signer engine, which interacts with several modules and internal databases.

The Signer Engine

The internal signer engine is the heart of the ODD. It receives changes to unsigned zones, after which it interacts with the NSEC-ifier to maintain the circular NSEC or NSEC3 record list. New records are given to the RRset Signer. The signed zone is then updated with removed and new records. The Signer Engine is controlled by KASP, which keeps tracks of the various keys and their states.
The main task of the engine is to make sure that signatures never expires and to resign the zone when a new key is used, and to manage the soa serial update.

The NSEC-ifier

The NSEC-ifier keeps an nsec or nsec3 chain fully linked. Whenever there is a change (either adding or removing records) in the zone data, the NSEC-ifier is called to update the linked list. Changed, added or removed NSEC(3) records then gets pushed back into the signer engine.
The NSEC-ifier can also be instructed to create a new NSEC3 chain, when a new hash algorithm, new iterations, or a new salt is used.

The RRset Signer

The RRset Signer takes an incoming RRset, prepares an RRsig record with the proper rdata, and instructs the HSM to create the signature. This signature is then added to the RRSIG rdata. The RRSIG is returned to the signer engine.

(quick notes)
updates the signed zones database.
The signer engine informs the NSEC-ifier that the NSEC(3) chain is altered.
The NSEC-ifier requests the signer engine to change (add/remove) and sign the NSEC(3) records.
The signer engine streams the signature candidates to the RRset signer.
The signer engine updates the signed zone database.

Inbound adapter

The ODD has an inbound adapter which can receive changes to zones using AXFR, IXFR, DDNS, or re-reading a zonefile from disk. This adapter will feed the internal system with changes to existing zones.

Outbound adapter

The ODD has an outbound adapter which can emit changes to zones using AXFR, IXFR, or writing a zonefile to disk. This adapter will feed the external system with changes to existing zones.