Infoblox®
CONTROL YOUR NETWORK

WHITEPAPER

A Best Practices Architecture for DNSSEC

Cricket Liu, Vice President of Architecture

Background

The Domain Name System is the Internet's standard naming service. DNS is responsible for mapping domain names to addresses, addresses to domain names, and a good deal more. Virtually all non-trivial Internet applications—and this includes applications that run over TCP/IP-based corporate internets such as Web browsing, email, CRM, ERP, Active Directory, and others—rely on DNS. Without DNS, work on the Internet stops, just as surely as it does in the event of a cable cut or routing problem.

Unfortunately, DNS is now an old protocol. It dates to the early 1980s, when the Internet was in its infancy. And while DNS has been tremendously successful at accommodating the exponential growth of the Internet, it was not designed to operate in an environment as hostile as the Internet of today. Consequently, DNS lacks critical security features, such as authentication and integrity checking.

That DNS wasn't designed with security in mind is evident from the maladies that have afflicted it. Since 1997, when Eugene Kashpureff capitalized on a weakness in the implementation of the BIND name server to poison the caches of name servers around the Internet, we've seen an accelerating cycle: New vulnerabilities are found, new patches are released. The two latest vulnerabilities were found in the last two years, within 12 months of each other: In 2007, Amit Klein of Trusteer analyzed the source code of the then-current version of the BIND name server and discovered grave deficiencies in its randomization routines. Those routines are critical to making DNS messages difficult to spoof. And famously, in 2008, Dan Kaminsky developed a technique that could poison the caches of name servers in seconds—even name servers with enhanced randomization routines, patched against Klein's vulnerability.

It's difficult to overstate the seriousness of cache poisoning. A successful cache poisoning attack enables a hacker to inject whatever data he wants into the cache of an unsuspecting name server. This data may redirect users who think they're accessing their bank's web site, or the web site of an online merchant such as Amazon or eBay, to an exact replica of that web site. Those users may enter login, password and account information, which is then captured and later re-used. Or the spoofed data may redirect users' email through a hostile mail server, where that mail is copied or even modified before it's sent on to its real destination, with the sender and recipient unaware of the breach. Cache poisoning is also a major threat to the security of Internet-hosted applications (SaaS) and is a major impediment to the viability of cloud computing.

The Internet community has done a creditable job of addressing each vulnerability, deploying new or patched versions of name servers on an irregular but increasingly frequent basis, when the need arose. But no one seems capable of slowing the cycle, much less stopping it.

The Kaminsky vulnerability may seem like just the latest development in this ongoing cycle: yet another vector a hacker could use to poison the caches of name servers on the Internet. However, it's different in a very significant way: There is no patch.

The “Kaminsky vulnerability” is a weakness in the DNS protocol itself, not in any one implementation of a resolver or name server. While we’ve successfully mitigated the problem using a technique called source-port randomization, at best this has made the vulnerability more difficult to exploit, not impossible. In other words, we’ve bought ourselves time: A vulnerability that a hacker initially could have capitalized on in seconds now takes hours. But for sufficiently valuable targets, that effort may be well worth it. And as bandwidth and computational horsepower increase, the effort required to implement successful attacks diminishes.

Luckily, a permanent fix already exists: the DNS Security Extensions, or DNSSEC. Working Groups within the Internet Engineering Task Force have been working diligently on DNSSEC for over a decade now. If we deploy DNSSEC promptly, we may stave off a catastrophic compromise of the Internet’s DNS infrastructure.

What is DNSSEC?

In short, DNSSEC is a set of extensions to DNS that uses asymmetric cryptography to provide origin authentication and integrity checking for DNS data. But that’s a pretty dense statement, so let’s take it apart and examine what it really means.

Please note that this document is not intended to teach the reader the nuances of DNSSEC, but rather the general concepts behind DNSSEC and the state of DNSSEC’s adoption, and the implications these have for DNS architecture.

Several sources exist for detailed information on DNSSEC, including the RFCs that specify it, RFCs 4033 through 4045, available from www.rfc-archive.org. NIST Special Publication 800-81, Secure Domain Name System (DNS) Deployment Guide, also contains useful information. Finally, the book DNS and BIND, published by O’Reilly Media, contains a chapter (Chapter 11) about DNSSEC.

Extensions to DNS

When we say that DNSSEC is a set of “extensions to DNS,” we mean that DNSSEC is built on the “classic” Domain Name System, not a bottom-up redesign of the Internet’s naming service. DNSSEC introduces new resource record types to DNS and new administrative regimens, and requires upgrades to name servers to reap the benefits of DNSSEC’s increased security, but deploying DNSSEC will not break existing name servers or stub resolvers. That’s critical, since there are over 12 million name servers on the Internet and even more resolvers: Upgrading them all will take a long, long time.

Asymmetric Cryptography and Key Pairs

Asymmetric cryptography is perhaps more commonly known as public-key cryptography. At the heart of asymmetric cryptography is the “key pair,” two mathematically related cryptographic keys that have very special properties:

- One decrypts data that the other has encrypted, and
- Knowing one of the keys doesn't reveal the other

In the most common scenario, one key is kept secret and used to encrypt data. This is called the "private key." The other key is made public and used to decrypt data that was encrypted with the private key. This one is called the "public key." Since the private key is a secret and not easily derivable from the public key, encrypted data that decrypts successfully using the public key is proven to have been encrypted—or "signed"—by someone in possession of the private key.

Origin Authentication

This brings us to origin authentication and integrity checking. In DNSSEC, each DNS zone is associated with one or more key pairs. (For simplicity's sake, we'll just discuss the case of a zone with a single key pair.) The private key is held by the administrator of the zone and kept secret. The public key is added to the zone in a new type of resource record, called a DNSKEY record.

The private key is used to sign the resource record sets in the zone[1]. The signature doesn't encrypt the records, but it provides a digital signature that can be used to validate the source of the records and ensure that they have not been modified by an attacker. Each signature is added to the zone in another new type of resource record, called an RRSIG record.

Integrity Checking

All of these DNS records can be cached on name servers that aren't authoritative for the signed zone, of course. A resolver or another name server may query one of those name servers and receive the cached records as a response. As when talking directly to the authoritative name servers for the signed zone, if the querying name server indicates the ability to validate DNSSEC-signed records, the caching name server will include the RRSIG records in the response. If the records—either the records in the answer or the RRSIG records—have been modified on the caching name server, the decryption (and hence the validation) will fail. Successful validation establishes not just the origin of the data, but also its integrity: It hasn't been modified since the zone's administrator signed it.

The Chain of Trust

There's one more aspect of DNSSEC that merits explanation: the chain of trust. A querying name server uses a zone's public key to validate signed data in the zone. But how does it validate the public key itself? Surely the public key isn't just signed by the zone's private key, which would mean that the public key would validate itself.

1. A resource record set is all of the resource records of a particular type attached to a particular domain name; for example, all of the A records attached to www.foo.example. They're signed as a unit because they're always returned together and validated as a group: Thus there's no need to sign them individually.

No, the public key's validity is established by the zone's parent. For example, if our signed zone is called infoblox.com, it's the com zone that vouches for the infoblox.com zone's public key.

After the infoblox.com zone has been signed, the administrator sends a copy of his public key to the administrator of the com zone, along with anything necessary to establish that he's the duly authorized administrator of the infoblox.com zone. Once the com zone's administrator is satisfied that he has the rightful public key for the infoblox.com zone, he adds a new record to the com zone, called a DS record. The DS record contains a "fingerprint" of the public key. Then the com zone's administrator signs the com zone, which adds an RRSIG record that proves this new DS record is authentic.

Now, when a name server resolving a domain name in infoblox.com queries one of the com name servers, it'll receive a referral to the infoblox.com name servers that includes the DS record. (Both the NS records in the referral and the DS record will be signed, of course.) When the querying name server later looks up the infoblox.com DNSKEY record to validate signed infoblox.com records, it will first validate the DNSKEY record from the fingerprint in the DS record.

And who signs the com zone's DNSKEY record? Why, the administrators of the root zone, of course. And the root zone's public key is widely known—even compiled in to name servers.

Unfortunately, this is how it should work: Today, the com zone isn't signed, nor is the root. But several top-level zones are signed, and there are plans to sign more —including com, net, and the root zone—over the next couple of years.

Issues with DNSSEC

It's natural to ask why—after over ten years of development—DNSSEC isn't more widely deployed. It obviously offers valuable functionality. So why aren't more zones signed?

First, DNSSEC has been a moving target. Depending on whom you ask and how he counts, we're either on our second or third version of DNSSEC. Despite the fact that earlier versions were never implemented on a large scale, theoretical misgivings about the nuances of their operation prompted the IETF to re-engineer the protocol—twice.

The latest version, which includes a new resource record type called NSEC3, looks like it addresses most remaining qualms with the protocol and could be successfully rolled out on the Internet at large. But name servers that support this latest version are brand new or just coming out.

Administrative Burden

Another impediment to DNSSEC's broader adoption is the burden it places on DNS administrators. With the most commonly available tools, generating key pairs and signing a zone for the first time is a clumsy, complicated process, requiring lots of work at the command line. Each time a zone's administrator edits the zone's records, he must re-sign the entire zone. Periodically, keys must be "rolled over" to foil cryptanalysis attacks. But the administrator can't just replace the old public key with a new one: The old one must remain in the zone until all records that were signed with the old private key age out of caches around the Internet. And, perhaps worst of all, most of this complexity is not well documented, so administrators are left to stumble through these processes using trial-and-error.

The tools, thankfully, are improving, and automate many of the most common and most complex tasks. Commercial products, including Infoblox's, offer graphical interfaces to manage zone data and insulate the administrator from low-level, command-line operations.

Overhead

DNSSEC also requires more of name servers. DNSSEC-signed zones are substantially larger than their unsigned equivalents, so name servers authoritative for signed zones generally use several times more memory after DNSSEC is implemented. The name servers, on average, also send larger responses, since the responses to DNSSEC-smart queriers must include more records—for example, signatures and public keys in the form of RRSIG and DNSKEY records—and these records are much larger than the usual resource records.

For recursive name servers, the overhead can be even greater. Authoritative name servers don't need to worry about encrypting or decrypting data[2]: A separate signing program adds RRSIG records to the zone before the name server loads it. But recursive name servers must decrypt each RRSIG record received in order to validate it, and in the process of resolving a single domain name may need to validate many RRSIG records. That requires considerably more processor time than plain vanilla name resolution.

What DNSSEC Doesn't Do

On top of the actual issues with deploying DNSSEC, many people misunderstand what DNSSEC does. We've discussed what DNSSEC does, namely providing origin authentication and integrity checking of DNS data. Here's a list of things DNSSEC doesn't do (but is commonly believed to do):

- Provide privacy of DNS data. While the signatures contain encrypted data, the records they sign are still sent in the clear.
- Protect DNS against DDoS attacks. There are no mechanisms in DNSSEC to guard against distributed denial of service attacks. In fact, in some ways DNSSEC actually makes DDoS attacks easier to mount, as it makes responses larger (and those responses are easy to misdirect to a target).
- Protect your name server against most attacks. Many—even most—attacks that target name servers are more mundane than cache poisoning. For example, they use implementation flaws such as buffer overruns to crash a name server or execute code. DNSSEC provides no protection against these attacks, other than making it hard for a hacker who successfully breaches your name server to replace your signed data.

What You Should Do

Despite the many threats that DNSSEC does not combat, the consequences of a successful cache poisoning attack are so dire that it's worth implementing DNSSEC purely to address that risk.

Adoption of DNSSEC will proceed on the Internet much more quickly than on internal, corporate networks. Internet-facing zones are generally more critical, more exposed to attackers, as well as smaller and more static than their internal counterparts.

Consequently, your deployment of DNSSEC should begin by signing your Internet-facing zones and configuring validation of signed zones on the Internet. Later, as DNSSEC becomes more broadly accepted and deployed, you can implement DNSSEC on an end-to-end basis, signing and validating internal zone data.

Phase 1

In Phase 1 of your implementation, you'll enable DNSSEC on your external name servers, sign your external zones and distribute those zones' public keys to the administrators of name servers that need to verify your zone data. You'll also configure your forwarders with the "trust anchors" they need to validate signed data in other zones.

Phase 2

In Phase 2, you'll sign your internal zones, too, and configure trust anchors, as necessary, to enable your internal name servers to validate your internal signed data.

Phase 1 is much more pressing than Phase 2, because most threats to DNS originate from the Internet, not internal networks, and because most data that a name server can validate is hosted on the Internet. Consequently, many organizations may adopt a "wait-and-see" approach toward Phase 2, and defer broader implementation until DNSSEC's value is proven and they're comfortable managing DNSSEC key pairs and signed zones, and have integrated DNSSEC into their internal zone-administration regimen.

This document, then, will concentrate on describing an architecture for the first phase of DNSSEC implementation.

Please note that this architecture builds upon the best practices architecture described in the Infoblox white paper titled DNS Appliances Architecture: Domain Name System Best Practices, available from the Library section of www.infoblox.com. This document doesn't cover aspects of the architecture unrelated to DNSSEC; for that, please see the earlier white paper.

Authoritative Architecture for DNSSEC

If you're following best practices, you're already running a hidden primary name server for your external zone data, and implementing this architecture will be trivial. If not, it's now even more critical that you deploy a hidden primary: With DNSSEC, the security of your primary name server becomes even more essential, because the key pairs for your zones will likely be stored on the primary. A hacker who gains access to your primary name server can simply modify your zone data and re-sign it using your zones' private keys, subverting any security DNSSEC provides. Using a hidden primary configuration allows you to run the primary name server for your external zones inside your firewall, where it will be better protected—not to mention easier to manage.

In order to "advertise" your signed, external zones to name servers on the Internet, you'll need to run one or more name servers that are accessible from outside your firewall. In most cases, these name servers will be located on DMZ networks or at a collocation facility, as shown in Figure 1, on the next page. Configured as secondaries for your external zones, they'll retrieve the zone data via zone transfers from your hidden primary, safely inside the firewall.

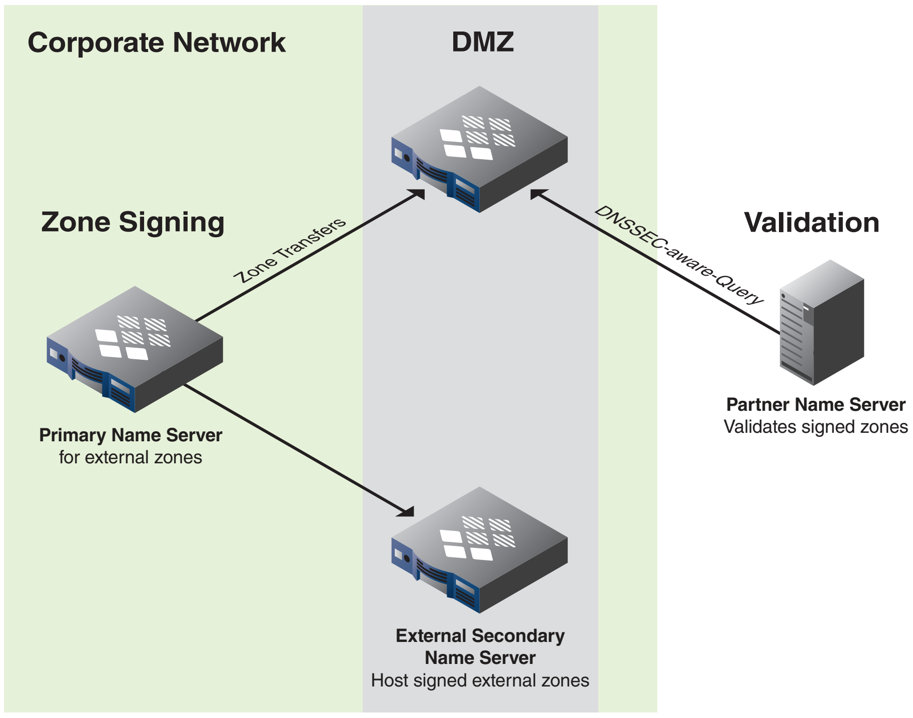


Figure 1: A DNSSEC Architecture for Authoritative Name Service.

Disconnected Signers

Some security-minded organizations may prefer a deployment option with no risk of compromising private keys. The only way to achieve this level of security is to store the private keys from your key pairs on a computer not connected to the rest of your network. In order to sign your external zones, you'll need copies of your zone data files on this computer, as well as signing software (such as DNSSEC-signzone from the BIND 9 distribution). When you want to modify an external zone, you'll edit the zone data file on the disconnected computer, re-sign the zone using the signing software, then copy the signed zone data file to the primary name server (using a USB flash drive, for example).

Distributing Trust Anchors

Given the poor adoption rates of DNSSEC on the Internet, most of your external zones won't have the benefit of a signed parent. If that's the case, you'll need to distribute your zones' public keys directly to the administrators of name servers that need to validate your signed data. Those administrators can add your zones' public keys to their name servers' configurations as "trust anchors," thereby giving their name servers the ability to validate data in your signed zones, or in any of their subzones.[3]

How you distribute your zones' public keys is up to you—there's no built-in mechanism to handle it. Just make sure you choose a secure means of distribution, such as PGP-signed email.

Recursive Architecture for DNSSEC

Most large organizations have deployed forwarding architectures for recursive name resolution: Internal name servers are configured to relay queries they can't answer to one or more forwarders, which then resolve Internet domain names on behalf of the internal name servers. The forwarders, naturally, are allowed to query name servers on the Internet directly.

Since initially you won't sign your internal zones, the forwarders will be able to perform all DNSSEC validation. This means that you'll only need to enable DNSSEC and configure trust anchors on your forwarders. (In fact, your internal name servers won't need any DNSSEC configuration at all—they won't even need to support DNSSEC!) The forwarders will discard any DNSSEC-signed records that fail validation. This is shown in Figure 2, below.

3. Note that this transitivity means that you only need to distribute public keys for your "apex" zones. In other words, if you've signed your corp.example zone as well as your labs.corp.example zone, you only need to send out the corp.example zone's public key, since a name server with that key can also validate signed data in labs.corp.example.

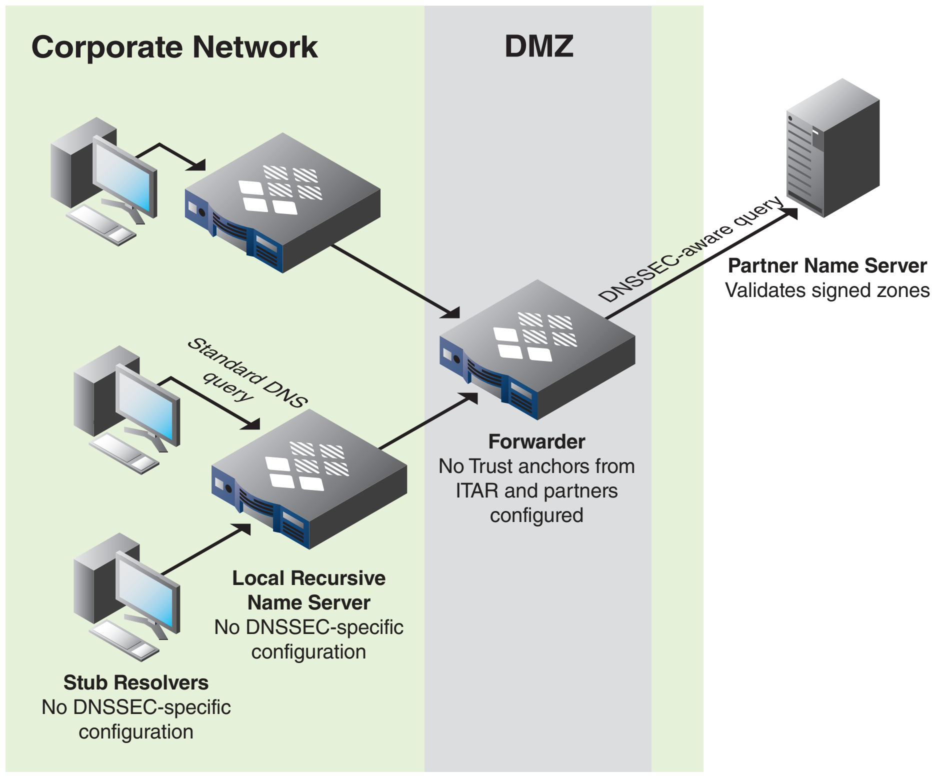

Corporate Network
DMZ
DNSSEC-aware query
Partner Name Server
Validates signed zones
Standard DNS query
Forwarder
No Trust anchors from ITAR and partners configured
Local Recursive Name Server
No DNSSEC-specific configuration
Stub Resolvers
No DNSSEC-specific configuration

Figure 2: A DNSSEC Architecture for Recursive Name Service.

Many of the trust anchors you'll configure on your forwarders will come from your peer DNS administrators at suppliers, partners and customers. In order to validate as many records as possible, however, you may want to configure your forwarders with the trust anchors listed in the Interim Trust Anchor Repository at https://itar.iana.org. The Internet Assigned Numbers Authority maintains the ITAR as a clearinghouse for the public keys of signed top-level zones. As of this writing, it contains trust anchors for 18 top-level zones.

Considerations

The top-level .gov zone uses NSEC3 records, a special DNSSEC record type introduced only recently. Consequently, only the newest BIND name servers, from BIND 9.6.0 forward, support validation of NSEC3 records. If you intend to validate any signed data under gov, you'll need to upgrade your forwarders to BIND 9.6.0 or better.

Deploying DNSSEC in Your Organization

Every organization is different and as such each must develop their own plan for deployment. That said, there are a few basic guidelines that every organization should follow in order to endure a timely and effective implementation:

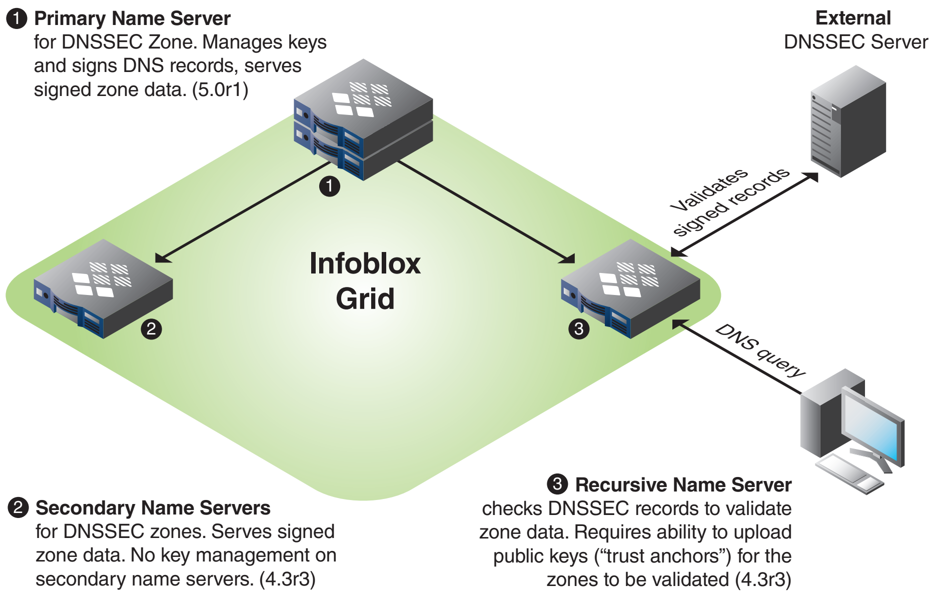
- Establish a security policy for DNSSEC
    - Determine which zones need to be signed. Most organizations will begin their DNSSEC deployment by signing only Internet-facing zones.
    - Decide whicah servers will serve signed zones. In most cases, supporting DNSSEC will require the latest version of your name server software.
    - Establish client-side policy. Some name servers allow the administrator to accept expired signatures, for example. During an initial rollout, that may be necessary.
    - Establish key generation and management procedures. Where will the private keys be stored and how will they be protected? How, and how frequently, will keys be rolled over?
    - Set cryptographic standards. What cryptographic algorithm will you use, and what key length? How long will your signatures remain valid?
- Design the DNSSEC implementation (using best practices)
- Assess your current infrastructure and upgrade or configure equipment to perform DNSSEC as needed
- Recommendations:
    - Start with a pilot or trial, and test. You might start by creating a shadow of an existing, production zone and signing it. If you make it a subdomain of an existing, production zone, you can configure trust anchors for it and test validation.
    - Sign your authoritative, production zones first, test, then configure validation on clients.

The Infoblox Solution for DNSSEC

The latest shipping version of Infoblox NIOS software has built-in support for DNSSEC and allows you to become compliant with the OMB mandate.

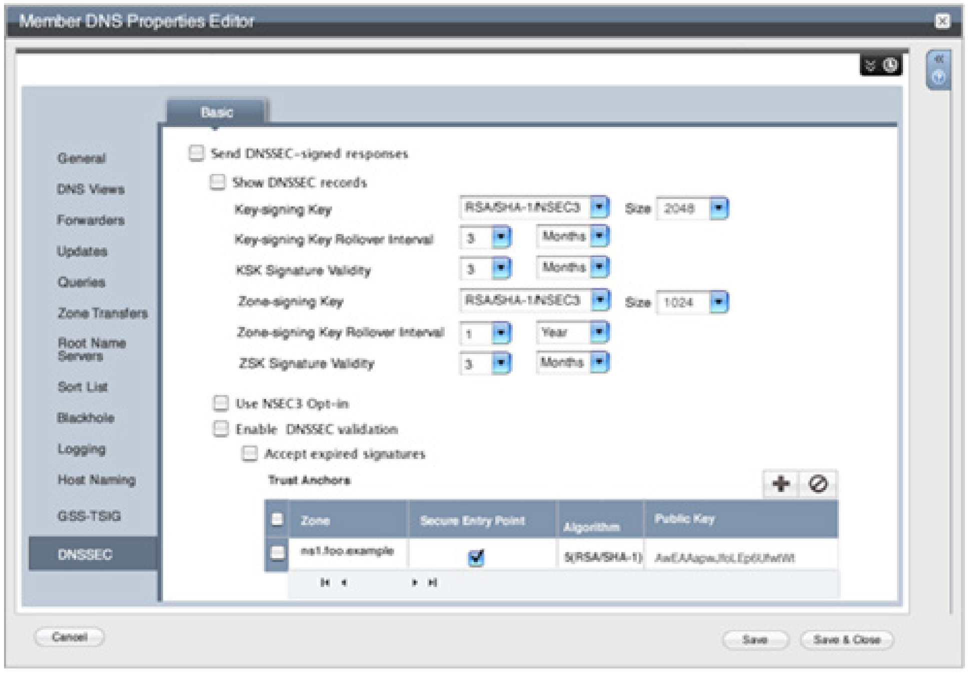
Transparent to end users, DNSSEC by Infoblox can be configured with single clicks, and delivers automated, on-the-fly key generation and management using the latest technology and protocol features (BIND 9.6.1 with NSEC3 support). The result is faster adoption at a much lower cost requiring less in-house training and expertise.

Leverages Infoblox Grid Technology



1 Primary Name Server
for DNSSEC Zone. Manages keys and signs DNS records, serves signed zone data. (5.0r1)
External DNSSEC Server
Validates signed records
Infoblox Grid
1
2
3
DNS query
2 Secondary Name Servers
for DNSSEC zones. Serves signed zone data. No key management on secondary name servers. (4.3r3)
3 Recursive Name Server
checks DNSSEC records to validate zone data. Requires ability to upload public keys ("trust anchors") for the zones to be validated (4.3r3)

The Infoblox DNSSEC solution leverages Infoblox Grid technology, which makes the management and configuration of DNSSEC easier across the DNS infrastructure, often with single mouse clicks. This level of DNSSEC automation, management and integration is unprecedented.



☐ Send DNSSEC-signed responses

☐ Show DNSSEC records

Key-signing Key	RSA/SHA-1/NSEC3	Size	2048
Key-signing Key Rollover Interval	3	Months	
KSK Signature Validity	3	Months	
Zone-signing Key	RSA/SHA-1/NSEC3	Size	1024
Zone-signing Key Rollover Interval	1	Year	
ZSK Signature Validity	3	Months	

☐ Use NSEC3 Opt-in

☐ Enable DNSSEC validation

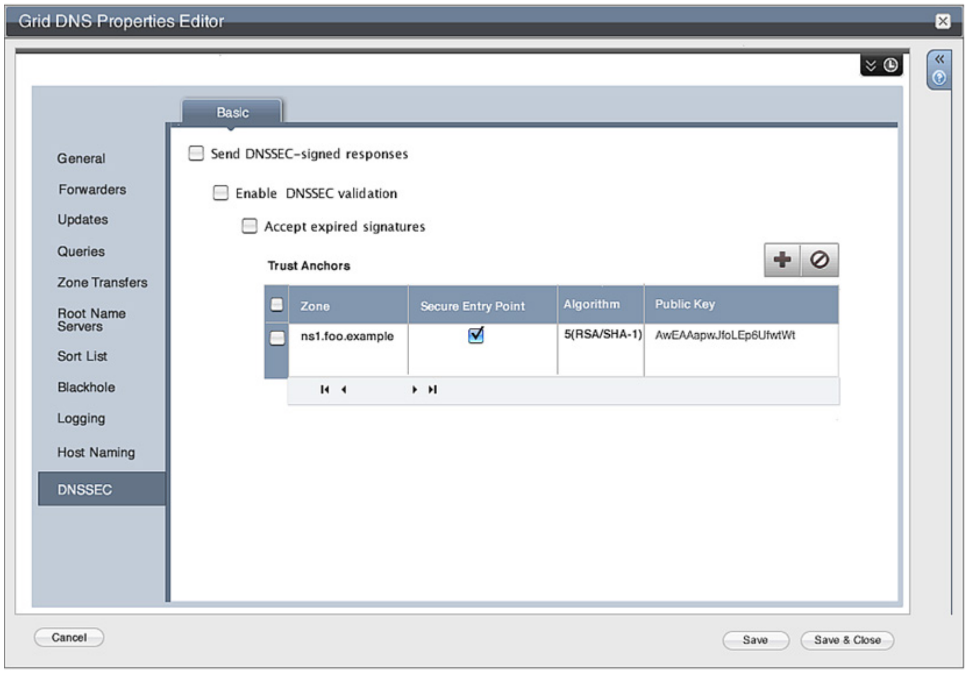
☐ Accept expired signatures

Trust Anchors

Zone	Secure Entry Point	Algorithm	Public Key
ns1.foo.example	✓	5(RSA/SHA-1)	AwEAAapwJfoLEp6UfwWM

Automated Management and Configuration

DNSSEC by Infoblox offers central configuration of all DNSSEC parameters, enforces standards by configuring DNSSEC parameters at a Grid level (default key type, size and validity period (based on NIST-800-81 and RFC 4641 standards) and includes NSEC and NSEC3 support.



Send DNSSEC-signed responses

Enable DNSSEC validation

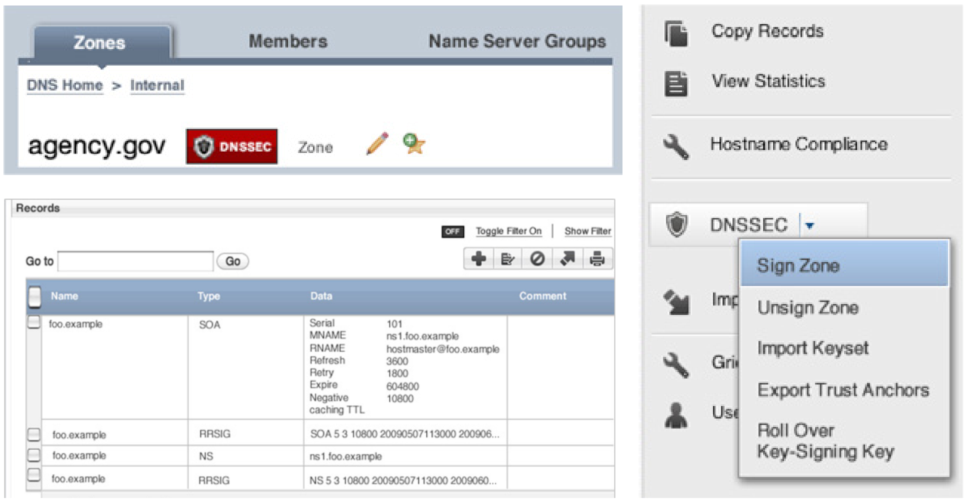
Accept expired signatures

Trust Anchors

Zone	Secure Entry Point	Algorithm	Public Key
ns1.foo.example	✓	5(RSA/SHA-1)	AwEAAapwJfoLEp6UfwtWt

Configuring a secondary and/or recursive name server for DNSSEC can be accomplished with a single click, including enabling sending DNSSEC records as a secondary, enabling validation of DNSSEC for an external zone and easy importing of trust anchors.

Any zone can be signed with a single click by using the “Sign Zone” toolbar button. Keys are generated on the fly and records are automatically signed; all associated DNSSEC records are auto-created. Signed zones are automatically maintained. All ZSK key expiration and resigning are handled automatically. When new records are added DNSSEC zones are automatically resigned.



Name	Type	Data	Comment
foo.example	SOA	Serial 101 MNAME ns1.foo.example RNAME hostmaster@foo.example Refresh 3600 Retry 1800 Expire 604800 Negative caching TTL 10800	
foo.example	RRSIG	SOA 5 3 10800 20090507113000 200906...	
foo.example	NS	ns1.foo.example	
foo.example	RRSIG	NS 5 3 10800 20090507113000 2009060...	

Signed zones are then easily identifiable with the DNSSEC icon. DNSKEY, RRSIG, DS, NSEC, NSEC3, NSEC3PARAM record types are all supported. New zone signing keys are automatically generated before the current keys expire. Key rollover is transparent to the admin and admins are automatically notified in the GUI before keys expire.

Conclusion

While viewed for years by many as a complex solution in search of a problem, DNSSEC is poised for explosive growth as it represents the only viable technology that addresses the imminent threat of DNS cache poisoning. DNSSEC technology has been extensively reviewed and, with proper tools, can be deployed with minimal operational overhead. Several top level domains (TLDs) have been signed, including .gov and .org and the country level domains for 6 countries. The US Office of Management and Budget (OMB) has mandated the used of DNSSEC by US government agencies before the end of 2009. Notably, ICANN and VeriSign have announced plans to sign the root zone by the end of 2009. It's clear that DNSSEC will move from the lab to extensive production deployment from the end of 2009 through 2010.

Infoblox is shipping DNSSEC-compliant appliances today and is delivering the only fully automated, “one click” DNSSEC solution for full production deployment in 2009. Every organization needs to assess its DNSSEC implementation drivers and readiness and develop a DNSSEC policy and implementation plan. Infoblox has extensive DNSSEC expertise and is available to our customers and prospects to define and implement plans for deploying this critical technology.

About Infoblox

Infoblox (NYSE:BLOX) helps customers control their networks. Infoblox solutions help businesses automate complex network control functions to reduce costs and increase security and uptime. Our technology enables automatic discovery, real-time configuration and change management and compliance for network infrastructure, as well as critical network control functions such as DNS, DHCP and IP Address Management (IPAM) for applications and endpoint devices. Infoblox solutions help over 6,500 enterprises and service providers in 25 countries control their networks.

CORPORATE HEADQUARTERS:

+1.408.986.4000

+1.866.463.6256

(toll-free, U.S. and Canada)

info@infoblox.com

www.infoblox.com

EMEA HEADQUARTERS:

+32.3.259.04.30

info-emea@infoblox.com

APAC HEADQUARTERS:

+852.3793.3428

sales-apac@infoblox.com