Observations from the DNSSEC Deployment*

Eric Osterweil
UCLA
eoster@cs.ucla.edu

Dan Massey
Colorado State University
massey@cs.colostate.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

Abstract

DNS Security Extensions have been developed to add cryptographic protection to the Internet name resolution service. In this paper we report the results from our monitoring effort with early DNSSEC deployment trials and the lessons learned.

1 Introduction

Cryptography can provide an effective step toward securing critical Internet protocols. The DNS Security Extensions (DNSSEC) [5, 7, 6] are one of the first attempts to deploy cryptography in a large-scale distributed system that is collectively operated by millions of autonomous administrative domains. In this paper, we present the first measurement study of the DNSSEC deployment by early adopters. Our study shows that the challenges in the DNSSEC deployment arise from difficulties in coordinating millions of diverse administrative domains as required by the cryptographic mechanisms, and from a lack of understanding and experience with cryptographic techniques. Our results also show the importance of a monitoring system to augment the DNSSEC deployment.

After a brief introduction to DNSSEC in Section 2, we report our measurement results in Section 3, and provide an analysis of the implications in Sections 4 and 5. We conclude the paper with a discussion of lessons learned.

2 Background

The Domain Name System (DNS) maps hostnames such as "www.ucla.edu" to IP addresses and provides a wide range of other mapping services ranging from email to geographic location. Virtually every Internet application relies on looking up certain DNS data. In this section we introduce a basic set of DNS terminology which is used throughout the text, including resource records (RRs), resource record sets (RRsets), and zones, followed by an overview of the DNS Security Extensions.

All DNS data is stored in the same data structure called *Resource Records* (RRs), and each RR has an associated name, class, and type. For example, an IPv4 address for www.ucla.edu is stored in an RR with name www.ucla.edu, class IN (Internet), and type A (IPv4 address). A host with several IPv4 addresses will have several RRs, each with the same name, class, and type but its own IPv4 address. The set of *all* resource records associated with the same name, class, and type is called an *Resource Record Set* (RRset). DNS resolvers query for RRsets. For example, a browser will query for ⟨www.ucla.edu, IN, A⟩, the reply will be the RRset for www.ucla.edu with all the IPv4 addresses for that name. Note that the smallest unit that can be requested in a query is an RRset, and all DNS actions including cryptographic signatures, discussed later, apply to RRsets instead of individual RRs.

The DNS is a distributed database organized in a tree structure. At the top of the tree, the root zone delegates authority to generic top level domains (such as com, net, org, and edu), and country code top level domains (such as uk, cn, and jp). The domain "com" then delegates authority to create google.com, "edu" delegates authority to create "ucla.edu", and so forth. In the resulting DNS tree structure, each node corresponds to a *zone*. Each zone is served by multiple *authoritative nameservers* to provide name resolution services for all the names in the zone. Every RRset in the DNS is owned by a particular zone and stored at the nameservers of that zone. For example, the RRset for ⟨www.ucla.edu, IN, A⟩ is owned by the ucla.edu zone and stored in the ucla.edu nameservers; while the RRset for ⟨www.colostate.edu, IN, A⟩ is owned by the colostate.edu zone and stored in the colostate.edu nameservers.

2.1 DNSSEC Overview

Security was not a primary objective when the DNS was designed in mid 80's and a number of well

*This work is partially supported by the National Science Foundation under Contract No. CNS-0524854. Opinions and findings expressed in this paper are those of the authors and do not necessarily reflect the views of NSF.

known vulnerabilities have been identified [1, 2]. DNSSEC provides a cryptographic solution to the problem, which seems pretty simple and intuitive. To prove that data in a DNS reply is authentic, each zone creates public/private key pairs. It then uses the private portions of its key(s) to sign data. Its public keys are stored in a new type of RR called "DNSKEY", and all the signatures are stored in another new type of RR called "RRSIG". In response to a query, an authoritative server returns both the requested data and its associated RRSIG RRset. A resolver that has learned the DNSKEY of the requested zone can verify the *origin authenticity* and integrity of the reply data. To resist replay attacks, each signature carries a definitive expiration time.

In order to learn the DNSKEY for a given zone, the resolver constructs a *chain of trust* that follows the DNS hierarchy down from a trusted root. For example, the DNS root public key would be used to authenticate the public key of *edu*; the public key of *edu* would be used to authenticate the public key of *ucla.edu*; and so forth. To build the chain of trust with its parent, one key at the zone must correspond to a matching record, called the DS RR, stored at its parent. The DS record is signed by the parent's private key to create an authentication link from the parent to child zone. It is the child's responsibility to request an update to the DS RR anytime the child's public key changes. Since the child zone must request a DS RR update at its parent zone every time the child's public key changes, the child zone can gain added flexibility by using two public keys. A *key signing key (KSK)* is a public key that matches the DS RR stored at the parent. It is then used to sign one or more public keys called the zone signing keys (ZSK). These ZSKs can then be changed locally without notifying the parent.

3 Monitoring DNSSEC Deployment

The latest DNSSEC specifications were published in March 2005 and our monitoring project, SecSpider, began collecting data a few months later in October 2005. The first challenge is to find zones that have deployed DNSSEC. Once a secure zone is found, SecSpider performs periodic checks on the zone and records the details about how DNSSEC is being managed at that zone. Our data collection provides both a glimpse at how widely DNSSEC is being deployed as well as insights into design issues, cryptographic assumptions versus practices, and open challenges for both DNSSEC design and operations.

3.1 Tracking the Number of Secure Zones

Our first objective is to monitor how many zones have deployed DNSSEC and observe the deployment trends over time. The total number of DNS zones is vast (tens of millions) and is constantly changing. Any of these zones may deploy (or disable) DNSSEC at any time. SecSpider uses a combination of DNS crawling (searching the DNS namespace) and user submissions to find zones that have deployed DNSSEC. SecSpider started after initially crawling over 18 million zones and continually expands its search. However due to the vast size of DNS, some DNSSEC zones can take a long time to be found.

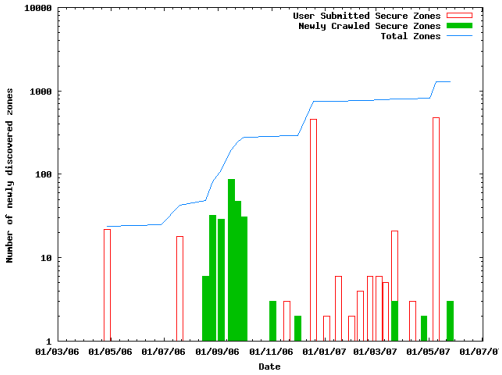

Figure 1. Number of Secure Zones

At this early stage of deployment, user submissions are the most effective vehicle for learning new DNSSEC zones. SecSpider has been prominently featured in the community with links on several websites and mailing lists. In addition to submitting their own secure zones, users may also submit the name of any zone that is important to them (even insecure zones such as google.com). SecSpider visits all the submitted zones frequently, so when they choose to deploy DNSSEC, they will immediately benefit from the additional monitoring discussed below. SecSpider also automatically finds and monitors the parent of each secure zone, to verify the authentication chain correctness if the parent zone is secure. If the parent zone is not secure, SecSpider still tracks it and looks for the moment when the parent enables DNSSEC.

Figure 1 shows the number of zones that have deployed DNSSEC. The bottom bars show whether the newly found secure zones were found by the crawler or submitted by users. Currently, SecSpider monitors 1,767 zones. Of these, 871 are classified as secure. The crawler continues to search for additional secure zones and users continue to submit new (both secure and not yet secure) zones for monitoring. Although the number of DNSSEC-enabled zones is small and represents no meaningful percentage of the DNS at this time, some promising

trends are emerging. Several country code top level domains (ccTLDs) have deployed DNSSEC and are attracting more production quality zones in those regions. At the time this data was compiled, the ccTLD *bg* has secure delegations to 69 secure zones under it and the ccTLD *se* has secure delegations to 40 secure zones.

3.2 Secure Zones Monitoring

Once a zone name is identified by either crawling or user submission, it is relatively straightforward to detect whether the zone has deployed DNSSEC. First, a secure zone must have at least one DNSKEY RR for its name, to hold its public key. By simply querying for RRset ⟨zonename, IN, DNSKEY⟩, one can determine if DNSSEC is being used. The DNSKEY RRset holds the set of public keys and all other RRsets in the zone must be signed by one of the corresponding private keys. If a resolver signals support for DNSSEC, an RRset returned by the zone's nameservers should be accompanied by corresponding RRSIG resource record(s). The combination of DNSKEY (public key) and RRSIG (signature by the private key) allows the resolver to authenticate the RRset data.

In addition to the DNSKEY and RRSIG records, a secure zone also contains NSEC records. To prove that a particular name or RRset does not exist, DNSSEC introduces a canonical ordering of all names in a zone and uses the NSEC RRs to enumerate them. An NSEC record specifies the "next" conically ordered name that occurs in the zone. One unintended side effect of the controversial NSEC RRs is that they make zone monitoring much easier. By starting with the zone name and using the NSEC record, one can find the first name in the zone, then use the NSEC record associated with the first name to find the second name in the zone and so forth. This allows one to "walk the zone" to discover all the records, including all delegated zones. The use of NSEC records raises privacy concerns which are beyond the scope of this paper, we simply point out that the "zone walking" can be exploited both for finding other secure zones and for tracking changes as discussed later in this section.

By querying for a DNSKEY RR and then using NSEC RRs to enumerate secure zones, SecSpider can find all records and check for proper signatures and responses for all nameservers associated with a zone. SecSpider monitors secure zones once a day and posts the summary information on the SecSpider website *http://secspider.cs.ucla.edu/*. The monitoring data has enabled zone administrators to detect and correct a number of configuration errors. In the remainder of this paper, we focus on key management practices and challenges seen by SecSpider.

4 Findings from Monitoring

Up to now the main focus of DNSSEC tool-set development has centered around signing zones. A number of tool-sets exist to help administrators: generate key pairs, sign zones, insert public keys into DNSKEY RRs, insert NSEC RRs, and sign all the RRsets with RRSIGs. Most recent DNS servers load the resulting signed zone and apply the DNSSEC rules when answering queries. DNSSEC-aware resolvers thus receive signed responses.

However the data collected by SecSpider shows that the challenge in DNSSEC deployment goes far beyond signing a zone. In this section we briefly describe three major issues we have identified.

4.1 Islands of Security

The DNSSEC design leverages DNS' tree structure to build a public key hierarchy. Although the concept is simple, its implementation turns out to be challenging in practice. Secure delegations involve coordination across different *administrative domains*, each of which may have different goals and priorities regarding DNSSEC deployment. When a zone enables DNSSEC, it is often the case that its parent zone still does not support DNSSEC. Therefore there is no authentication chain leading down from the root, resulting in an "island of security".

More formally, an island of security is a subtree where every zone in the subtree is signed; if this subtree contains more than one zone, every zone below the subtree's root has also established an authentication chain with its parent. Islands of security play an important role in DNSSEC deployment because client resolvers need to be configured with the public keys belonging to the roots of *all* of the islands of security in order to authenticate DNSSEC replies to queries. If DNSSEC were fully deployed, the entire DNS would form a single island of security and resolvers would only need to be configured with the root zone's public key and could then build an authentication chain to any zone in the DNS.

However the root zone and most top level domains have not deployed DNSSEC. As a result, the current DNSSEC deployment consists of individual islands of security. When SecSpider began monitoring, very few zones had either a secure parent or child zone, thus essentially every zone made its own island of security. Over time, bigger islands began to occur, and a few islands began to grow noticeably at the beginning of this year. Figure 2 shows the size distribution of the current islands of security: the x-axis indicates the number of zones in each island, and the y-axis shows how many islands have that size. Currently, the ccTLD *bg.* is the largest known island with 69 zones; a few other islands have a size less than 10, but the vast majority of islands still

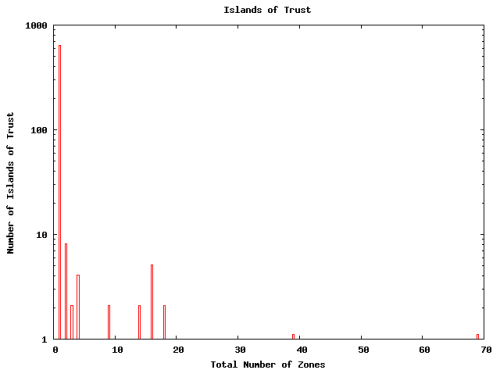

Figure 2. Islands of Security.

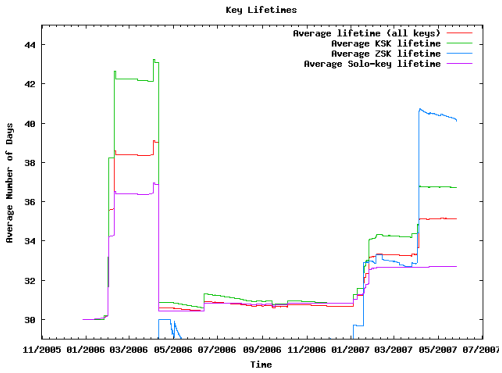

Figure 3. Average Key Lifetimes.

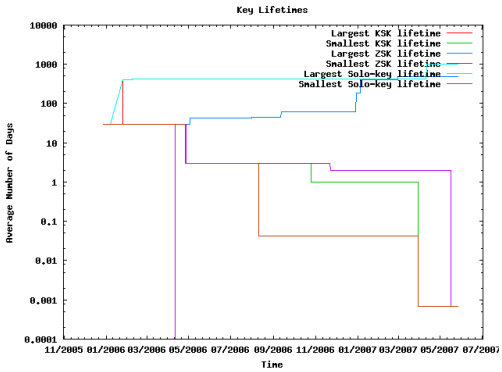

Figure 4. Maximum and Minimum Key Lifetimes

have a size of 1.

4.2 Key Management Practices

For those zones that do have a secure parent, they need to maintain the corresponding authentication chain. For this, the child zone must request a DS RR update at its parent zone every time the child's key changes, and the parent must sign the new DS RR. As described in Section 2.1, a child zone can gain added flexibility by using a *key signing key (KSK)* to sign one or more zone signing keys (ZSK), which in turn are used to sign data RRs in the zone. The KSK matches the DS RR stored at the parent zone to maintain the chain of trust, and ZSKs can be changed locally without notifying the parent. Even zones that don't have a secure parent (i.e. islands of security) benefit by using a KSK and ZSK: their KSKs need to be configured into secure resolvers, thus would be difficult to change; however they can freely change the ZSKs. For zones using both KSK and ZSK, one would expect the former to have a much longer lifetime than the latter.

Of the 871 secure zones currently known by SecSpider, 711 of them use both a KSK and ZSK. Figure 3 show the average key lifetime graphed over time. Most of the DNSSEC zone signing tools use a default 30-day key lifetime. Prior to early 2006, the average lifetime for both KSK and ZSK public keys was 30 days. Figure 4 shows the maximum observed KSK (and ZSK) lifetime and minimum observed KSK (and ZSK) lifetime. The graph shows the very first early adopters selected the default key lifetimes and variations became apparent starting in early 2006. However, the largest ZSK lifetime still closely tracks the largest KSK lifetime, defeating the purpose of having a long lasting KSK key that requires external coordination and shorter lived ZSK whose change requires no external coordination.

4.3 Lifetimes and Replay Attacks

The Lifetime of signatures plays an important role in DNSSEC because there is no revocation mechanism. Although recent work [3] has suggested mechanisms to address this need, the absence of any accepted practice underscores the importance of DNSSEC's signature dates. In DNSSEC, each signature over an RRset contains inception and expiration dates. These dates are often on the order of months, and 30 days is a typical default value. By contrast, the time to live (TTL) value used for DNS caching is on the order of minutes or hours. If an RRset happens to change before its signature expires, the old RRset and its corresponding signature (RRSIG) is replaced by a new RRset and new RRSIG at the authoritative server. Resolvers who have the old copy cached will discard the old value after the TTL expires and fetch a new copy from the authoritative server if the RRset is needed again.

This assumes the resolver will fetch the new copy from a valid authoritative server. If an attacker can replay the old value, resolvers will continue to consider the old data to be valid until the signa-

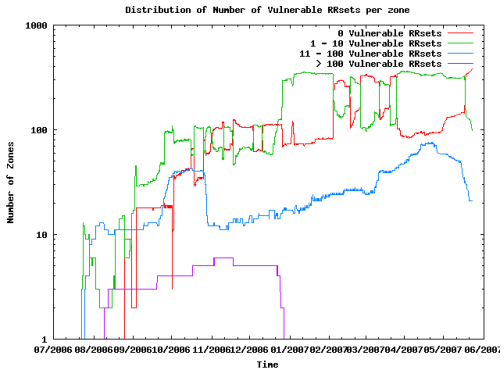

Figure 5. RRsets Vulnerable to Replay Attack

ture expires (days or months!). This incongruity between signature lifetimes of days or months and cache lifetimes of minutes or hours is largely due to the overhead and complexity associated with signing records. The net result is a potentially very long replay window in which an attacker can replay old records and resolvers believe the records to be authentic.

We say an RRset is *vulnerable* if the RRset currently stored at the authoritative server has changed, but the signature on the previous RRset has yet to expire. An attacker can replay this old data until the signature expires, effectively blocking any intended change from occurring. For example, a web server under DDoS attack may change its IP address. However an attacker could replay the RRset with the old address and old signature, falsely proving to resolvers that web server *did not* change its address. Note it is trivial for the attacker to retain old data since (s)he can simply not delete the old RRset when the TTL expires.

Figure 5 shows that for a protracted period in the past, there were a significant number of zones whose signing behaviors left over 100 vulnerable RRsets at any given time. However, in the more recent past we can see that zones have increasingly chosen non-default signature lifetimes. We believe this partly due to the vulnerable RRset tracking provided by SecSpider; some zone administrators have also discussed vulnerable RRset issues with us.

5 Key Management Revisited

The DNSSEC standard has undergone major revisions and one of the major changes involved key management between parent and child zones. This revision introduced the DS RR to help decouple parent and child key management and it is believed the protocol is now stable. However our monitoring results suggest that much work has yet to be done. These remaining challenges and open issues are primarily operational in nature. Although they may not require a change in the protocol, they do require new operational practices at a minimum and ideally new supporting mechanisms and protocols to make the system truly effective.

5.1 Experience with Cryptography

DNSSEC and cryptographic techniques are a new addition to DNS administration. Our results show that the very early adopters followed the default rules settings, almost without any modification. Nearly all zones use both the KSK and ZSK and nearly all keys had the default lifetime on their associated signatures. Recently, more divergence has occurred, but some fundamental concerns remain. For example, how effective is the use of KSK and ZSK if both keys have identical lifetimes and expire at the same time? How well are signature lifetimes being chosen for data records? Are the implications of these choices well understood? Our results gave mixed reports. It is clear that the operators for the majority of the zones being monitored are yet to master the practice of cryptographic techniques. On the other hand, zones are beginning to make their own choices instead of copying the defaults, and there does appear to be improvement in vulnerable RRsets.

5.2 Parent and Child Coordination

To maintain a secure delegation, the DS record at the parent must match the DNSKEY record at the child or the authentication chain will fail. The analogous procedure in DNS is a parent and child must coordinate nameserver (NS) records. Ideally the NS RRset at the parent should match the child's, however our previous work [4] shows that mismatches are common. This is largely due to the facts that: parent and child zones belong to different organizations, operator coordination is required every time the child zone makes changes, and manual configurations inevitably introduce human errors. Fortunately DNS is resilient to such errors as long as at least one NS record at the parent matches an NS at the child.

Unfortunately cryptographic checking is much less tolerant to errors. Ideally, the parent zone should have only one DS record that exactly matches the DNSKEY RR at the child. However given that DNS has not shown a history of strong coordination between parent and child, it is likely that human errors will lead to DNSSEC key mismatches, which can result in secure resolvers discarding all answers from the child zone, since they fail the DNSSEC authentication checks. Relying on manual coordination is a recipe for disaster. We believe that protocols to automate the mainte-

nance of DS/DNSKEY consistency and monitoring efforts such as SecSpider are *essential components* in DNSSEC deployment, in order to catch errors before end users report loss of service. Furthermore, the degree to which a resolver requires perfect authentication needs to be reconsidered. Rigid application of authentication checks provides clear and provable statements about whether data will or will not be accepted, but may reject valid data due to inevitable configuration errors in practice. More flexible choices can tolerate such inevitable errors, but can also open doors for attacks.

5.3 Incremental Deployment

Our monitoring thus far shows that DNSSEC deployment does not follow the DNS tree hierarchy. Parent and child zones have different interests for enabling DNSSEC, and experience different levels of difficulty doing so. Fundamentally, DNS is a distributed system and a child zone cannot make its parent to turn on DNSSEC. The result is a large number of small islands of security. In the current scenario, a secure resolver would need to be manually configured with the public keys for all the islands of security in order to securely resolve data from all the secure DNS zones. It would also need to update its set of public keys whenever KSKs change, and our results show the average KSK lifetime is 34 days. At best, this presents a major configuration challenge for resolvers.

Efforts to secure the root and other top level domains are important and being pursued, however the different interests between parent and child zones suggest that the large number of islands of security will remain a reality in the foreseeable future. Hence we must be able to successfully roll out DNSSEC in the face of this reality.

6 Discussion and Future Work

SecSpider has been running for over a year and we have learned a great deal from its monitoring results. Our major observations can be summarized as follows. First, our measurement showed the lack of understanding in cryptographic management, such as the implication of key lifetime and signature lifetime. Since the TCP/IP specification publication in 1981, we have accumulated 26 years of operational experience with running the Internet. However the experience using cryptographic techniques is much shorter and limited. If one aims to quickly roll cryptographic protection into the Internet's operations, the design must work well without requiring all operators to be cryptographically literate.

Second, DNSSEC deployment requires coordination across administrative domain boundaries, errors will inevitably occur in this process, as seen by observing DNS operations [4]. The damage from NS RRset mismatches between parent and child zones mostly leads to degraded performance, however the damage by DS RR mismatches can lead to unavailability. Since it is impossible to eliminate human errors, we feel strongly that monitoring should be an *integral* part of the DNSSEC deployment. Our measurements also show that the DNSSEC operators have reacted to our measurement results and changed their practice accordingly. This is an encouraging sign showing that a monitoring system can provide effective feedback to correct or improve the operations, further underscoring the important role of a monitoring system as a first-class component of the DNSSEC deployment.

Finally, our measurements also highlight one fundamental need in the DNSSEC rollout: supporting DNSSEC deployment by isolated islands. Due to the distributed nature of the DNS system, a child cannot make its parent turn on DNSSEC when the child needs its key validation. Thus to enable everyone who needs DNSSEC, we must provide key verification for isolated islands of security.

We are extending SecSpider to be a distributed monitoring system, that will probe DNSSEC zones from multiple locations. This helps to substantially reduce the impact of (1) network failures on our monitoring results, and (2) potential damage to our monitoring results due to man-in-middle (MIM) attacks, as it would be difficult, if not impossible, for MIM attackers to insert themselves into the paths of all our monitor probes located in different continents and ASes. As SecSpider becomes a distributed monitoring system and its monitoring results become more trustworthy, we plan to evolve it towards a key lookup system, as a viable solution to support isolated islands of DNSSEC deployment.

References

[1] S. M. Bellovin. Using the domain name system for system break-ins. pages 199–208.

[2] D. A. D. Atkins. Threat Analysis of the Domain Name System (DNS). RFC 3833, August 2004.

[3] E. Osterweil, V. Pappas, D. Massey, and L. Zhang. Zone state revocation for dnssec. In *LSAD '07: Proceedings of ACM Sigcomm Workshop on Large Scale Attack Defenses*, 2007.

[4] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang. Impact of Configuration Errors on DNS Robustness. In *ACM SIGCOMM*, 2004.

[5] M. L. D. M. S. R. R. Arends, R. Austein. DNS Security Introduction and Requirement. RFC 4033, March 2005.

[6] M. L. D. M. S. R. R. Arends, R. Austein. Protocol Modifications for the DNS Security Extensions. RFC 4035, March 2005.

[7] M. L. D. M. S. R. R. Arends, R. Austein. Resource Records for the DNS Security Extensions. RFC 4034, March 2005.