A Security Evaluation of DNSSEC with NSEC3*

Jason Bau
Stanford University
Stanford, CA, USA
jbau@stanford.edu

John C. Mitchell
Stanford University
Stanford, CA, USA
mitchell@cs.stanford.edu

Abstract

Domain Name System Security Extensions (DNSSEC) and Hashed Authenticated Denial of Existence (NSEC3) are slated for adoption by important parts of the DNS hierarchy, including the root zone, as a solution to vulnerabilities such as "cache-poisoning" attacks. We study the security goals and operation of DNSSEC/NSEC3 using Murφ, a finite-state enumeration tool, to analyze security properties that may be relevant to various deployment scenarios. Our systematic study reveals several subtleties and potential pitfalls that can be avoided by proper configuration choices, including resource records that may remain valid after the expiration of relevant signatures and potential insertion of forged names into a DNSSEC-enabled domain via the opt-out option. We demonstrate the exploitability of DNSSEC opt-out options in an enterprise setting by constructing a browser cookie-stealing attack on a laboratory domain. Under recommended configuration settings, further Murφ model checking finds no vulnerabilities within our threat model, suggesting that DNSSEC with NSEC3 provides significant security benefits.

1 Introduction

Domain Name System Security Extensions, or DNSSEC [4, 5, 6], with Hashed Authenticated Denial-of-Existence (NSEC3) [8] is a security standard for DNS that has been in development since at least 1999 [1]. Briefly, DNSSEC adds cryptographic signatures to standard DNS records to provide origin authentication and cryptographic integrity, but not secrecy or improved availability, for those records. Recently, DNS security has garnered quite a lot of interest, due to the highly publicized DNS "cache-poisoning" vulnerability discovered by Dan Kaminsky [15, 22], and several actual exploits of this vulnerabilities on ISP-run DNS servers that resulted in the redirection of popular websites to attack sites for customers of these ISPs [18]. Though initial software patches were issued which made cache-poisoning attacks much less likely to succeed, DNSSEC is proposed as a long-term solution to DNS data integrity [17] against cache-poisoning as well as in-path "man-in-the-middle" attacks. As of August 2009, the operators of the .org, .com, and .net Top Level Domains (TLDs) as well as the operators of the DNS root zone have all announced plans to deploy DNSSEC/NSEC3 on their servers. Given the current interest in DNSSEC/NSEC3, we feel it worthwhile to perform a thorough security analysis of the protocol in order to understand its benfits and shortcomings.

During the course of this study, we found potentially problematic DNSSEC configuration options that were intentionally included in the protocol design to support incremental adoption and to minimize the performance impact of DNSSEC at the high-traffic top-level domains. We will highlight the DNSSEC/NSEC3 design trade-offs associated with these options, the resultant potential dangers, and recommend DNSSEC configuration choices for enterprise-level administrators considering DNSSEC adoption.

As background, we review standard DNS and Kaminsky-style cache-poisoning attacks. We then examine the security goals and limitations of DNSSEC/NSEC3, explain its operations, and consider its effectiveness against cache poisoning. We perform finite-state model checking of the DNSSEC/NSEC3 protocol against safety invariants derived from its stated security goals. By identifying the parts of DNSSEC packet content possessing cryptographic integrity, we define the capabilities of network attackers executing a man-in-the-middle attack on DNSSEC packets. The model checker identifies several security invariant violations, including resource records that remain valid after the expiration of signatures attesting to their validity and also protocol configurations that create an unsigned subspace in the DNSSEC namespace of a domain, allowing forged names to be inserted into an otherwise secure domain. We discuss how the first violation may prolong the vulnerability window if a private key is compro-

*This updated March 2, 2010 version corrects and supersedes a paper with the same title that appears in the NDSS'10 proceedings.

mised or signed records are successfully forged. Also, to demonstrate the exploitability of possible name-insertion, we implemented an actual attack on a realistic laboratory DNSSEC domain mimicking an enterprise deployment, exploiting the vulnerable configuration to steal user browser cookies. We then incorporate protocol configuration repairs into the Murφ model and verify that no exploitable vulnerabilities are detected. Based on our analysis, we provide recommendations to domain operators, DNSSEC software implementors, and website designers that maximize the security of DNSSEC implementation and deployment.

During the process of writing up this work, a presentation was given by Daniel Bernstein at WOOT '09 [12] pointing out possible vulnerabilities in DNSSEC. In comparison, our work further exposes the mechanisms behind the vulnerabilities and thereby provides configuration/operation advice to eliminate exposure to attacks. For the replay vulnerability caused by signature-expiration mismanagement reported by Bernstein, we provide simple operational guidelines that prevent possible attacks. One overlap with Bernstein's presentation is the relatively minor observation of forgeable glue NS and A records within DNSSEC response packets. While Bernstein correctly concludes this forgery raises security concerns, we explain why this forgery does not actually add any capabilities for the network attacker and thus does not create additional exploitable attacks. Beyond Bernstein's presentation, we found and experimentally confirmed an attack using NSEC3 opt-out that does not require cryptoanalysis. In fact, our entire work assumes unforgeable cryptographic signatures in order to study attacks possible even with adequate cryptography, complementing Bernstein's thoughts on breaking DNSSEC cryptography. A summary of the contributions of this work, in terms of security violations discovered and attack prevention advice, is listed in Table 1.

The remainder of this paper is organized as follows. Section 2 reviews standard DNS and cache-poisoning attacks. Section 3 gives an overview of the security limitations, goals, and mechanisms of DNSSEC/NSEC3 and demonstrates its effectiveness against cache-poisoning. Section 4 presents our finite-state model of DNSSEC/NSEC3, the network attacker model, and also the inconsistency in DNSSEC attestation chain temporal dependencies that we found. Section 5 presents the rest of the security violations reported by finite-state model checking as well as the configuration and implementation choices that eliminate them. Section 6 details our experiment confirming the exploitability of the name-insertion property, using insecure delegation and NSEC3 opt-out as illustrations. Finally, Section 7 presents our best-practice DNSSEC adoption and implementation advice and concludes.

2 Background: DNS Protocol

2.1 DNS Basics

We first review the relevant background information on DNS. Table 2 lists the relevant RFCs defining DNS. DNS is a hierarchical distributed database that translates alphanumeric domain names, such as www1.example.com, into (most commonly) IPv4 and IPv6 addresses. DNS lookups are ubiquitous as they must be performed before any network resource, such as a website or a mail server, is accessed by its alphanumeric domain name. The domain names may be thought of as database keys that are used to lookup a variety of values, called Resource Records (**RR**s), associated with the key. (The "key" domain name is called the RR's "owner name" in DNS parlance). The most common RRs are IPv4 addresses (the **A** RR), IPv6 addresses (the **AAAA** RR), mail servers associated with a domain (the **MX** RR), and name servers associated with a domain (the **NS** RR). The values associated with the MX and NS RRs are in name and not IP address form. The set of all RRs of the same type belonging to the same owner name, e.g. multiple NS or A RRs, is termed a **RRSet**.

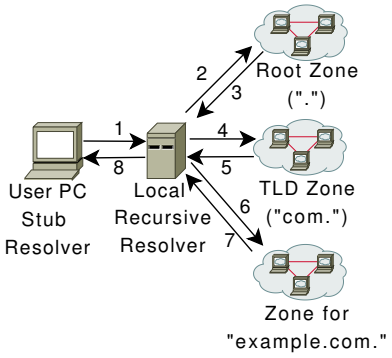
We now use the domain name www.example.com as an example to explain DNS terminology as well as its hierarchical operations. The name www.example.com has a canonical DNS form of **www.example.com.**. Each successive **label** ("www", "example", "com") in this form corresponds to a level within the DNS hierarchy (a **zone**), and the extra trailing dot (".") at the end of the canonical DNS form is inserted to signify the presence of the **root zone**, the top level of the DNS hierarchy.

A DNS zone is named by zero or more labels, e.g. "example.com." and consists of a set of RRs over which the zone is **authoritative**. The concept of authority is best illustrated by example. For instance, a zone is authoritative for all RRs whose owner name is the zone name – the .com zone is authoritative over the NS and MX records for .com. A zone server is also authoritative over RRs where 1) the owner name contain the zone name as a suffix and 2) no "longer suffix" of the RR's owner name is also a authoritative zone. For example, the example.com zone is usually authoritative over the A record for www.example.com., except when www.example.com is configured as its own zone, (possibly to support a domain name such as www1.www.example.com).

In addition to authoritative RRs, a zone may also store **glue records** that aid in delegation. Glue records are RRs, typically A and NS, under the authority of child zones but copied to parent zones for the purpose of "gluing" together a delegation. For instance, the .com server may

Security Property Violations	Prevent At	Prevention Advice	Section
Resource Record remains valid in local resolver cache after expiration of signatures or key rollover (revocation) higher in attestation chain	Resolver Software ISP	Resolver software sets RR TTL to depend on all signatures in attestation chain to trust anchor Resolver software imposes an independent (from authoritative zone values) cap on TTL and signature validity periods	4.5.1
Glue records may be forged to direct next recursive query to attack DNS server	Domain Operator Resolver Software	Use all secure delegations If forgery is suspected, query supposed authoritative zone to obtain signed version of glue records. (Even if no action is taken, this violation does not result in acceptance of forged RR as final query answer. See paper section.)	5.1.1
NSEC3 opt-out may be used to prepend falsified owner name in domain, as stated in RFC 5155, resulting in vulnerability to cookie-theft and pharming	Domain Operator Website Designer	Do not set NSEC3 opt-out flag Do not use overly coarse cookie "domain" setting	5.1.3
Replay of still valid A+RRSIG after IP-address move (Bernstein [12])	Domain Operator	Do not relinquish IP-address until all A+RRSIGs have expired	5.1.4
Inter-operation with standard-DNS child zones means insecure answer returned by DNSSEC resolver	Domain Operator	Adoption of DNSSEC; Do not interoperate DNSSEC with DNS	5.1.2
Lack of end-user software indicator of secure vs. insecure DNSSEC query result exposes end-user to exploitable insecure DNSSEC query result	End-User Software (Browser/OS)	Support DNSSEC by providing lookup security indicators using DNSSEC AD Bit	3.1.2
Network attacker can arbitrarily manipulate DNSSEC reply header and status bits	Resolver Software ISP or OS	Do not trust header bits. Resolver validates only using internal state and signed RRs. Cannot trust remote DNSSEC validation without secure channel. Provide secure channel or validate all DNSSEC RRs locally	5.3.1
Network attacker can add recorded RRs / subtract RRs / mangle bits in RRs in DNSSEC reply packet	Resolver Software	Build *attested cache* for answering user queries using only authoritative signed RRs contained in DNSSEC replies.	5.3.1

Table 1. Summary of Recommendations



Reply	RRSets in DNS Reply	RRSets added by DNSSEC
3	*"com. NS a.gtld.net."* *"a.gtld.net. A 192.5.6.30"*	"com. DS" "RRSIG(DS) by ."
5	*"example.com. NS a.iana.net."* *"a.iana.net. A 192.0.34.43"*	"com. DNSKEY" "RRSIG(DNSKEY) by com." "example.com. DS" "RRSIG(DS) by com."
7	"www.example.com. A 1.2.3.4"	"example.com. DNSKEY" "RRSIG(DNSKEY) by example.com." "RRSIG(A) by example.com."
8	"www.example.com. A 1.2.3.4"	

Figure 1. DNS(SEC) name resolution sequence for query "www.example.com A?" resolving to IP address "1.2.3.4". Authoritative RRSets are in plain text and glue RRSets are in *italic*. The stub resolver is not expected to handle DNSSEC RRs, so none are sent to it.

DNS		DNSSEC	
RFC	Relevance	RFC	Relevance
1034, 1035	DNS Definition	4033, 4034, 4035	DNSSEC Definition (NSEC)
2671	EDNS0 longer packets (used by DNSSEC)	5155	NSEC3 Definition
3833	Threat analysis of DNS	4641	DNSSEC operational guidlines
2845	TSIG Channel Security	2535	DNSSEC initial proposal (AD, CD header bits)
2931	SIG(0) Channel Security	3757	Key Signing Keys (KSKs) and Zone Signing Keys (ZSKs)

Table 2. Relevant DNS and DNSSEC RFCs

store both the the NS record for example.com, with a value of ns.example.com, and the A record for ns.example.com, so that a single query response may contain all the information needed to follow a delegation. However, the .com zone would not be authoritative over either glue record; the glue records fall under the authority of the example.com server.

Figure 1 illustrates a typical DNS lookup process, which involves two types of DNS resolvers, a **stub resolver** and a **recursive resolver**. Consider the name resolution process that occurs after a user types www.example.com into the browser address bar. This triggers the DNS resolution process of the stub resolver on the user's PC, which then issues a query ("www.example.com A?") to the local ISP-run DNS server. This server now becomes a local DNS recursive resolver: it first queries the DNS root server for the A RR of www.example.com. The root server is not authoritative for this information, so it issues a delegation response, pointing the local recursive resolver towards the authoritative server for the .com zone. This query/response pair occurs again between the recursive resolver and the .com authoritative server, which leads to the resolver obtaining the address of the authoritative DNS server of example.com. When the recursive resolver queries the authoritative DNS server for example.com, it finally obtains an answer to "www.example.com A?", which it can then pass back to DNS stub resolver on the user's PC that initiated this entire process.

To reduce DNS network traffic, each DNS server caches RRs to keep from issuing redundant requests. DNS replies include the specified caching period (TTL) of a returned RR, set by the authoritative zone. As an example, suppose that the set of queries and responses in Figure 1 has occurred recently, so that the TTL of caches records has not yet expired. When another user of same local recursive resolver requests "mail.example.com A?", the local resolver will be able to bypass steps 2-5 due to caching and directly query the "example.com." authoritative server with "mail.example.com A?".

2.2 DNS Packet Format

We will now briefly describe the DNS packet format and transmission characteristics and subsequently discuss "cache-poisoning" attacks on DNS, conducted via both "man-in-the-middle" and "out-of-path" means.

The format of a DNS packet is illustrated in Figure 2 (DNSSEC packets are completely identical). DNS queries and responses are usually contained in a single small packet, less than 512 bytes, and are usually sent over UDP. This makes it fairly simple for network attackers to spoof DNS responses. The only protection that the DNS packet format provides against spoofing is in the 16-bit TXID (transaction ID) field. A DNS resolver will accept as valid the first response packet containing a TXID matching the TXID of an outstanding query. This creates a race condition for attackers: their spoofed responses to the DNS resolver must match an outstanding TXID before the actual response returns.

2.3 Cache-Poisoning Attack

In a cache poisoning attack, the attacker spoofs a DNS response packet so that a DNS resolver accepts and caches data "poisoned" by the attacker, such as an A RR of a valid owner name pointing at the IP address of an attacking server. The resolver then provides this poisoned data to the end user, redirecting common domain name requests (such as www.google.com) away from the legitimate server to attacking servers [18].

2.3.1 Man-in-the-Middle

Man-in-the-middle attackers are attackers who have read and write access to network packets belonging to the victim. In this scenario an attacker can overhear the queries made by the local recursive resolver to the remote DNS zones and

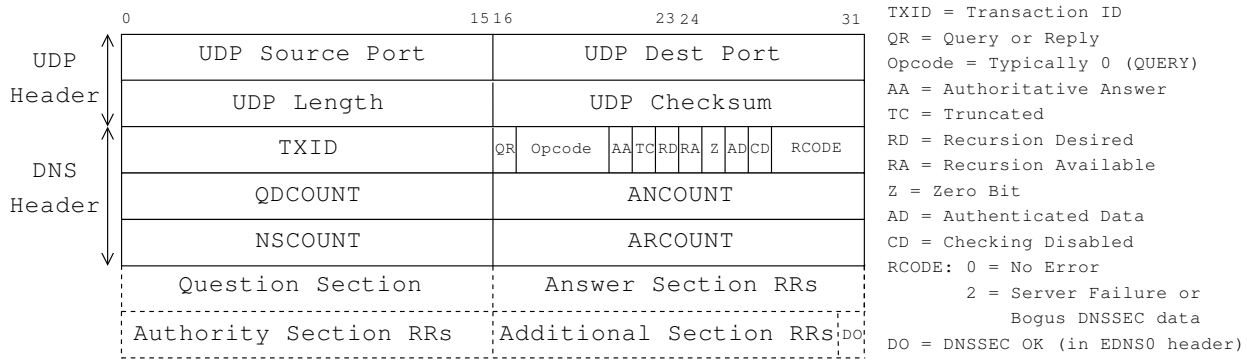


Figure 2. DNS and DNSSEC packet format. DO bit is in EDNS0 Header in Additional Section RR

inject faux replies from the remote zones. As DNS is unencrypted, it is trivial for the man-in-the-middle attacker to copy the correct TXID to generate an acceptable spoofed DNS reply, which will then poison the cache of the recursive resolver.

2.3.2 Out-of-Path Attack

We will now discuss out-of-path DNS cache-poisoning attacks, of which the recent work publicized by Dan Kaminsky [15] is the most infamous. In a Kaminsky attack, the attacker does not require the ability to overhear the outgoing DNS requests generated by the local recursive resolver. Instead, the ingenuity of the Kaminsky attack involves increasing the number of valid outstanding TXIDs, thus increasing the probability that a randomly generated spoof TXID will match an outstanding one.

The Kaminsky attack works with delegation responses rather than authoritative answers. The attacker issues many DNS queries to a DNS recursive resolver for nonexistent names sharing a common suffix zone, e.g. aNotExist.example.com, bNotExist.example.com, etc. (The queries may also be coerced from a user, for instance by an attacker-crafted web page containing these names in <img>tags). This creates many valid outstanding TXIDs at the recursive DNS resolver. Since all of these queries contain a common suffix zone ("example.com"), all responses coming from the ".com" zone will include NS and A glue records for the name server of "example.com". The attacker thus has many chances to poison the RRs for the "example.com" name server at the resolver, by sending many spoofed delegation responses (packet 5 from Figure 1) with different TXIDs containing altered NS and A glue records. Since the resolver will accept and cache the glue records upon finding a match, this creates an instance of the "birthday problem" [11] from probability that can lead to success after only seconds of attacking the 16-bit TXID field. After a successful match, the resolver will query the attacking server to resolve any RRs with owner name ending in "example.com", essentially giving the attacker full control of the "example.com" zone for users of this resolver. After the successful attack, all users of a poisoned DNS resolver that attempt to access "example.com" will be directed to a server of the attacker's choosing.

In order to address this vulnerability, Kaminsky worked with DNS software vendors to randomize the UDP source port of DNS queries [9, 13]; these random ports become the destination ports for DNS response packets. This effectively adds 10-11 bits of entropy for the attacker, making the expected success time of an attack several tens of minutes rather than seconds. However, the mitigation does not fundamentally prevent a spoofing packet success; it only lowers the probability of such an event. This mitigation also provides no defense against man-in-the-middle attackers. Therefore, many researchers, including Kaminsky himself [9, 17], have been actively supporting DNSSEC as a long-term solution to DNS security vulnerabilities, including cache-poisoning.

3 DNSSEC Protocol

3.1 Stated Security Goals and Limitations

DNSSEC, as the name implies, consists of a set of security extensions to the DNS protocol (see Table 2 for the relevant RFCs). DNSSEC introduces additional security-related resource records with each reply, for the purpose of providing cryptographically signed integrity to the original DNS resource records. This makes DNSSEC effective against both types of cache-poisoning attacks described in Section 2.3. DNSSEC does not guarantee delivery of resource records and does not provide integrity for unsigned

portions of packets. Its security goals are described in RFC 4033 as follows:

"The Domain Name System (DNS) security extensions provide origin authentication and integrity assurance services for DNS data, including mechanisms for authenticated denial of existence of DNS data."

In RFC 4033, the authors explicitly distinguish DNSSEC data (RR) security from channel security. DNSSEC packets, containing resource records carrying encoded-binary cryptographic material, are typically carried in the clear over UDP. Thus, the current paper is largely about the implications of the DNSSEC design decision to provide data (RR) security rather than channel security. We will first discuss the limitations of DNSSEC, and then consider in turn the three component DNSSEC data integrity goals from above: origin authentication, data integrity assurance, and authenticated denial-of-existence, and detail how the DNSSEC protocol attempts to reach these goals, even with in-the-clear communications.

3.1.1 "Last-Hop" Limitations

RFC 4033 specifically states the "last-hop" between stub resolver and recursive resolver (1 and 8 in Figure 1) may be out-of-scope for DNSSEC, to be protected via DNS channel security means such as SIG(0) [3] or TSIG [2]. This is because in anticipated DNSSEC deployment, cryptographic signatures are expected to flow from authoritative servers only to local recursive resolvers, with stub resolvers on end-user PCs not equipped to handle signature verification.

As our finite-state analysis is focused on the DNSSEC protocol, we consider last-hop security out-of-scope and designate the recursive resolver as the trusted end-point for name resolution in the analysis. However, we emphasize that the channel security of this last hop is critically important to end-to-end DNSSEC integrity. For example, the recursive resolver marks the difference between two types of responses to the stub resolver: verifiably secure answers and insecure answers, with a single "Authenticated Data" (AD) bit. Thus, attackers able to manipulate DNS replies over this last hop may forge *secure* answers simply by setting the AD bit. In usage scenarios where last-hop security is absent, such as unencrypted wireless hotspots, DNSSEC cannot guarantee domain-name lookup integrity to the end user.

3.1.2 Interoperability with DNS Limitations

Under current specifications, any inter-operation with standard DNS zones exposes the end-user of a DNSSEC recursive resolver to forgeable query results. When interoperating with a standard DNS zone, a DNSSEC recursive resolver cannot verify the integrity of remote zone data due to the lack of cryptographic signatures. For compatibility, the recursive resolver still returns any responses from the zone to the stub resolver, but without setting the AD security indicator bit. Thus, whenever a DNSSEC recursive resolver must query a standard DNS zone, the recursive resolver is forced to provide an answer without security guarantees to the stub resolver. As of this writing, end-user software accepts both secure and insecure results from the stub resolver, without any user-interface elements to indicate the security of the lookup result. Thus, the current end-user cannot trust the security of DNS lookups even if a DNSSEC recursive resolver with last-hop channel security is utilized. While this is the fault of end-user software, not DNSSEC/NSEC3, this is still an issue that enterprise network administrators (and application developers) should recognize.

3.2 Origin Authentication

The need for origin authentication is possibly best understood in the context of preventing cache-poisoning attacks. As we described above, these attacks are possible because the DNS recursive resolver will accept DNS data sent to it by any computer connected to Internet (possibly with a falsified source IP address) as long as the destination port/TXID fields match. There is no mechanism within DNS, aside from source IP address, that verifies the data originates from an authoritative server for a particular zone. To solve this issue, DNSSEC provides a form of hierarchical public key infrastructure (PKI) which allows resolvers to securely obtain the public key for a DNSSEC zone and to use this for authenticating signed data belonging to the zone.

DNSSEC introduces three new RRs to support this PKI: DNSKEY, RRSIG (RR Signature), and DS (Delegation Signer). The **DNSKEY** RR contains the binary-text-encoded public key along with relevant key parameters such as the encryption algorithm used. The zone uses the corresponding private key to sign all of the RRSets over which it is authoritative. Each signature over an RRSet is recorded in a **RRSIG** RR. The **DS** verification RR contains a cryptographic digest of a DNSKEY belonging to a child zone in a delegation. The DS RR is considered under the authority of the parent zone and can thus be signed by the parent zone (with a corresponding RRSIG). It is returned by the

parent side of a delegation as an authenticated pointer to a DNSKEY in the child zone. This [Parent DNSKEY $\overset{signs}{\rightarrow}$ Parent DS $\overset{signs}{\rightarrow}$ Child DNSKEY] sequence forms a link in an extensible attestation chain that can impart trust to any public key obtained via the chain, so long as the chain begins at a **trust anchor**. In the DNSSEC PKI, a trust anchor is any DNSKEY or DS RR confirmed as trustworthy via out-of-band means and configured in the resolver as trustworthy. With the recent announcement of root zone signing, this is expected to be the root DNSKEY.

The operation of the DNSSEC PKI is illustrated in Figure 1, which lists the DNSSEC packet contents for the name resolution "www.example.com A?", for which the DNSKEY of the "example.com." zone are needed. Starting with the DNSKEY of the root zone as the trust anchor, Reply 3 provides the DS to attest to the DNSKEYs of "com.". Reply 5 adds the DNSKEY of "com." and the DS to attest to the DNSKEY "example.com.", which is provided by Reply 7.

3.2.1 Origin Authentication with Regular DNS

In order to inter-operate with non-SEC DNS implementations, DNSSEC must also provide for cases where a DNSSEC zone has a non-DNSSEC parent or child zone. In the insecure parent zone case, since the trust chain cannot be established all the way back to the DNS root, either the DNSKEY of the secure zone or a DS generated from the DNSKEY must be manually configured as a trust anchor at the recursive resolver. When there is no such manually configured trust anchor, no attestation chain can impart trust to the DNSKEY of the secure zone. In this case, no records from the secure zone are verifiable by the recursive resolver and all records ostensibly from the zone will passed on to the stub resolver as an insecure answer.

In the case of an insecure child zone of a secure zone, an insecure delegation will be created with no DS record within the secure zone pointing at the child zone. We will see that this has significant security consequences.

3.3 Integrity Assurance

Given the hierarchical PKI provided by DNSSEC, it is straightforward for a zone to provide "integrity assurance" for its existent data. The zone signs all the RRSets over which it is authoritative and transmits the RRSIG along with the RRSet in its replies. For example, when responding to the "www.example.com A?" query, the example.com authoritative server will transmit both the A record and the RRSIG containing the signature over the A record, as Reply 7 in Figure 1 demonstrates.

DNSSEC allows a zone only to sign RRs over which it is authoritative. This means that any glue records included in a delegation response are unsigned, as illustrated in Replies 3 and 5 from Figure 1. As Bernstein has noted and as we will explain, these glue records may be forged, causing the local resolver to query an attacking server in its recursive next step. We show in Section 3.7, however, that this redirection does not allow the network attacker to influence to end result of name resolution.

3.4 Authenticated Denial of Existence

Thus far, we have discussed how DNSSEC provides integrity assurance for existent RRs. Authentication and integrity is also required for responses denying the existence of any RRs matching a query. If authentication mechanisms did not exist, for example, an attacker may be able to forge a response packet denying the existence of an existent domain name and have this response cached at the local resolver for long periods, creating a directed denial-of-service attack.

For performance reasons, development of an off-line method was a strong design consideration for authenticated denial of existence, preferable by top-level domain operators over on-line methods such as that proposed by RFC 4470 [7]. The initial DNSSEC scheme for this creates RRs, named Next Secure (**NSEC**), that list all of the existent RRs belonging to an owner name within an authoritative zone, so that a resolver can verify the non-existence of an RR against the RR list of its owner name. Each NSEC RR also contains the next existent owner name in canonical form, so that the non-existence of an owner name within a zone may be shown by returning a **covering** NSEC, whose owner and next existent names bracket the queried name. As an undesirable consequence, the entire contents of a zone may be trivially enumerated by following NSEC records and making appropriate queries.

An alternative scheme for hashed authenticated denial of existence, named **NSEC3** [8], is nearly equivalent to NSEC except that all owner names are cryptographically hashed and not available in cleartext. The canonical order of existent names in NSEC3 is the hashed order. Under NSEC3, zone enumeration of hashed names remains trivial, but the attacker must expend computational resources in a dictionary attack to learn the zone contents in cleartext. A salt string is appended to each owner name, the same for each name in the domain, in order to eliminate *pre*-computation of dictionaries by exploding their size. However, since the salt is available publically via a RR query, NSEC3 is still

vulnerable to the leakage of RR owner names after few days of *post*-query computation [12].

With NSEC, all owner names within the zone, including names only associated with NS records used for delegation, form the NSEC "next owner" chain. In NSEC3, such an owner name may "opt-out" of the chain via a bit in the NSEC3 RR. When the "opt-out" bit is set in an NSEC3 record, one or more unsigned delegations may exist with owner names that hashes to a value between the two hashed names in the NSEC3 RR. Opt-out allows top-level zones to support incremental adoption of DNSSEC at the enterprise level by excluding delegations to enterprise zones not supporting DNSSEC from the NSEC3 chain. This saves the TLDs the computational cost of NSEC3-hashing the names of non-DNSSEC child domains.

When a resolver receives a signed opt-out NSEC3 RR covering its queried name, it must still consult unsigned information, such as glue records indicating a delegation, to decide whether the query answer exists lower in the DNS hierarchy. The NSEC3 opt-out option allows insecure delegations and thus any RRs to be inserted by an attacker into the "span" of the NSEC3 record [8]. This NSEC3 characteristic, posing dangers to enterprise-level zones because of trust implied by domain membership, forms one instance for the demonstrated attack that we will detail later in this paper.

3.5 Temporal Specifications

Under DNS, a RR in a DNS reply packet included a specification of TTL as the time, starting from reply reception, that the resolver may validly cache the RR. This specification of TTL relative to packet reception makes DNS reply packets susceptible to replay attacks. To avoid replay vulnerability, DNSSEC introduces absolute-time temporal specifications for its signatures. Each RRSIG RR has a signature validity period, stated as absolute start and end times. This introduces a dependency of TTL times upon signature validity times at the resolver, as TTLs for RRs must not remain valid for longer than the valid periods of signatures attesting to these RRs. The absolute timing eliminates the possibly of replay after the expiration of the corresponding RRSIG.

3.6 Packet Format & Attacker Capabilities

Because DNSSEC operates solely by adding RRs to regular DNS, its packet format is essentially unchanged from DNS (see Figure 2). The security-related DNSSEC RRs are carried alongside the original DNS RRs in the same packet (see Figure 1). DNSSEC does introduces a single enable bit, DNSSEC OK (DO), located in the EDNS0 header contained in the Additional Section of DNS packets. It also defines two bit in the DNS header: Authenticated Data (AD), which indicates that the sending server has validated the RRs in the packet, and Checking Disabled (CD), which tell upstream servers to not perform RR validation.

The DNSSEC signature scheme only allows for individual RRSets to be signed by an associated RRSIG record. Thus, the integrity provided by DNSSEC is at individual RRSet+RRSIG granularity. Essentially, the only guarantee of DNSSEC is that it is impossible, short of private key compromise, for a network attacker to create a RRSet and RRSIG pair containing manipulated data validly signed by the originating zone. We thus incorporate capabilities for manipulating all other aspects of DNSSEC packets into our attacker model, including stripping RRSIGs from RRSets, changing header bits, inserting and deleting recorded RRs, etc. See Section 4.4 for a detailed description.

3.7 Robustness Against Cache Poisoning

DNSSEC is effective against the cache-poisoning attacks described in Section 2.3. In the presence of the attacker capabilities listed in Section 4.4, which model a man-in-the-middle network attacker, our finite-state model checking results in Section 5.2 demonstrate that signed DNSSEC records obtained using only secure delegations are not vulnerable to forgery. An end-user trusting only secure query responses is thus safe from such a network attacker.

We will also now detail how DNSSEC successfully protects against out-of-path (Kaminsky) cache-poisoning. Recall that the Kaminsky attack works by redirecting the IP addresses associated with glue NS and A records, causing the recursive resolver to query a DNS server controlled by an attacker. As noted by Bernstein, redirection of the child zone query to an attacking DNSSEC server is still possible under DNSSEC, since glue records are unsigned and forgeable. However, with the DNSSEC protocol, a DS record with RRSIG will also be sent in a secure delegation response. The authenticity of this signed DS record is verifiable by the recursive resolver via the attestation chain (it should not follow delegation responses without a signed and attested DS), thus giving the recursive resolver a way to verify the public key of the child zone.

With a trusted public key for the child zone, the resolver can validate whether a RR contained in a response sent by the attacking server is properly signed by the child zone. Short of key compromise, the attacking server therefore cannot falsify any signed RRSets in this child zone, including DS records for further secure delegation. Since the ultimate

RRs requested by name resolution, usually A or MX, are available in signed form in their authoritative zone, a resolver never has to rely on an unsigned record as its final answer. Thus, as long as a DNSSEC resolver accepts only RRSets appropriately signed by their authoritative zone as final query answers, the response packets may come from any server, redirected or not, without allowing the attacker to violate the ultimate integrity of a DNSSEC name resolution.

In fact, server redirection does not increase the packet forgery capabilities of the network attacker. Once an attacker has caused a recursive resolver to query its attacking DNSSEC server, it can form any type of response to the resolver that it chooses except create a valid RRSet and RRSIG pair signed with the zone's private key. These are exactly the same capabilities that we ascribed to the man-in-the-middle network attacker in Section 3.6, albeit made more convenient for the attacker by eliminating the race with a legitimate DNSSEC server. Thus, glue record forgery does not present any additional security threat to DNSSEC beyond the normal capabilities of a network attacker, though it may allow the attacker to more easily inhibit DNSSEC performance with rogue packets that, for example, consume resolver CPU time.

4 Finite State Model Checking

In order to evaluate the security of the DNSSEC protocol, we performed a finite-state "rational reconstruction" of DNSSEC using Murφ [14], a Nondeterministic Finite Automaton (NFA) enumerator to check its operations against safety invariants derived from its stated security goals. In the rational reconstruction process, decribed in [21], the most basic parts of the protocol messages are modeled and executed in the model checker, to see if any safety invariants are violated. When invariants are violated, more protocol components are added until the invariants pass or cannot be passed. The entire process thus aids in understanding the component design of the protocol and ensures that the properties expressed in the invariants test the functionality of each protocol component.

Furthermore, since Murφ tries all possible combinations of modeled attacker capabilities, when the reconstructed protocol runs to completion without violating any invariants, we may draw the conclusion that the protocol preserves the expressed safety invariants against the attacker described in the model. In this section, we will detail our reconstruction of the protocol, the network attacker mode, and the security invariants. We will also report on an inconsistency in the temporal dependencies of the DNSSEC attestation chain found by our modeling.

4.1 Overview of Murφ Model

Our model is based on a typical usage scenario of the DNSSEC service. Table 3 summarizes this finite-state model. We model three layers of the DNSSEC hierarchy, representing root zone servers ("."), TLD zone servers ("com."), and an authoritative server for a single zone ("example.com."). The root zone DNSKEY is our modeled trust anchor. In the state machine, these zone servers are simply modeled as a set of transition rules on network state; they do not introduce any additional state themselves. We also model a local recursive resolver, representing ISP-run DNSSEC resolvers, as a set of transition rules on network state as well as local state, representing name resolution status and knowledge of the DNSSEC hierarchy, such as zone keys, DS RRs, and server addresses. The network is simply a set of modifiable packet state structures. The final aspect of our model is the attacker model, which consists both of transition rules modifying network state and additional state representing packet knowledge recorded by the attacker.

4.2 Root, TLD, and Authoritative Servers Model

The behaviors of the root, TLD, and authoritative zone servers require no server state and are entirely described by network state transition rules. Our modeled root and TLD behaviors are quite simple. They respond to network state containing a query packet addressed to them and will write a response to the network containing either a secure delegation, with DS and RRSIG authoritative RRs and NS and A glue RRs.

The modeled behavior of the authoritative "example.com." server is more complex, as it covers the entire set of zone responses to an RR query. The full set is enumerated in Figure 4.3.1. Response 1 represents the simple case where the query matches an RR existent in the zone. Responses 2-4 represent when the query matches an existent delegation point instead of an RR in the zone. Response 2 is the secure delegation case. Responses 3 and 4 represent the options for an insecure delegation: a NS glue record used for insecure delegation in DNSSEC may either be recorded by the NSEC3 chain (response 3), or unrecorded, with the covering NSEC3 setting opt-out instead (response 4).

Finally, responses 5 and 6 represent cases when the query matches neither existent RR nor delegation. The zone must then indicate non-existence of the queried RR by returning the covering NSEC3, which may happen to have opt-out set (response 5), or not (response 6). Our modeled authoritative zone has RR content that will elicit each of these six re-

States	Transition Rules
Local Resolver State Knowledge of TLD and Authoritative Zone Address (and validity) DS (and validity) DNSKEY (and validity) Names to Resolve (name1-name6) Answer (and validity) **Network** Set of Packets **Attacker Knowledge** Set of Packets	**Local Resolver** Query Generation $LocalResolverState \rightarrow Network$ Reply Handling $Network \rightarrow LocalResolverState$ TTL and Signature Expiration $LocalResolverState \rightarrow LocalResolverState$ **Root, TLD, and Authoritative Zone Servers** Query Response $Network \rightarrow Network$ **Attackers** Learning Legitimate Replies $Network \rightarrow AttackerKnowledge$ Forgery Generation $AttackerKnowledge, Network \rightarrow Network$

Table 3. Overview of DNSSEC Murφ Model. Arrows denote *StatesRead* $\rightarrow$ *StatesWritten*.

sponses when queries for name1 through name6 are sent by the resolver, allowing Murφ to enumerate all possible states of an authoritative zone responding to a query.

4.3 Local Recursive Resolver Model

The modeled local recursive resolver tries to resolve the set of six names that elicit the full range of DNSSEC response behavior as described in the previous section. These six names also form the basis of our invariants, as we check that the information associated with the names in the authoritative zone matches the understanding the resolver learns from replies. The resolver state records the answer supplied by the authoritative zone to each of the six query targets, along with the temporal validity of the answer. When any answer is in the expired state, the resolver will try to resolve the corresponding name by writing a query packet to the authoritative zone server, provided its knowledge of authoritative server address is valid. For the purpose of querying and authenticating replies, the local resolver state also maintains TLD and authoritative zone address, DNSKEY, and DS, and will appropriately query when these expire. (The root server address and DNKSEY are not modeled as resolver state because they are hard-coded in a resolver implementation).

4.3.1 TTL and Signature Expiration

We model validity expiration for all query answers and all server addresses, DNSKEYs, and DSs. In DNSSEC, all of this information is stored as a RR. RRs have an associated TTL and, if signed, also a signature validity period for the corresponding RRSIG. As per RFC 4033, TTLs for RRs must expire when the corresponding RRSIGs expire; this is strictly enforced in our model by combining RR TTL and RRSIG validity into a single entity. Also, all modeled validity states initialize to 'expired', and transition rules exist for each record that change a 'valid' state to an 'expired' state.

4.3.2 Reply Validation Logic

The local resolver model also contains logic that validates the contents of a reply packet and decides what actions to take with regards to the query based on received information. This logic is of utmost importance to the security of a DNSSEC implementation. For example, incorrect resolver validation behavior that accepts unsigned RRs from an expected DNSSEC zone opens up a downgrade path for attackers to exploit. We distilled the guidance of RFC 4035 into our model.

In particular, our modeled resolver places RRSets contained in replies into two separate entities for use in this logic: an *Attested Cache*, whose contents are secure RRSets that have a full attestation chain back to the trust anchor, and a *Non-Attested Cache*, whose contents are RRSets the zone expects to be insecure, such as glue records or data from regular DNS zones. The attested cache consists of zone DNSKEYs and DSes as well as signed A and NSEC3 RRs from reply packets. For instance, to include an A record signed by the authoritative zone in the attested cache, the resolver's TLD and authoritative zone DSes and DNSKEYs must all be valid. The unattested cache consists of zone addresses and glue records from reply packets. RRSets determined to be bogus, such as those with invalid signatures, or indeterminate, such as those with incomplete attestation chains, are discarded by validation logic. We believe that

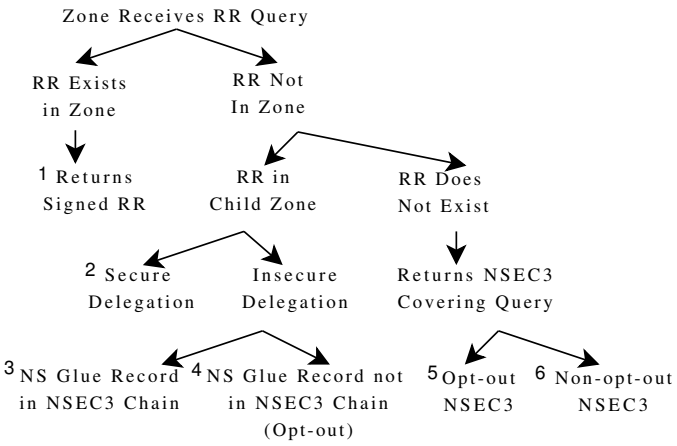


Figure 3. Zone Response Behavior to an RR Query

Matching RRSets in Attested Cache	Matching RRSets in Non-Attested Cache	Action
A		Answer
DS	NS, A	Secure Delegation
NSEC3 (owner name matches query, shows glue NS exists)	NS, A	Insecure Delegation
NSEC3 (covering query, opt-out)	NS, A	Insecure Delegation
NSEC3 (covering query, opt-out)		Denial-of-Existence
NSEC3 (covering query, no opt-out)		Authenticated Denial-of-Existence

Figure 4. Modeled Resolver Action Logic, Depending on Resolver Cache Contents Matching Query

the attested/non-attested cache distinction may be useful to future DNSSEC implementers.

The resolver decides what actions to take on behalf of each query based on the contents of the attested and non-attested caches. Table 4 summarizes these logical rules.

4.4 Modeled Attacker Capabilities

Our Murφ model checks DNSSEC in the presence of a network attacker possessing all reply packet manipulation capabilities short of key compromise. The attacker's ultimate goal is to induce the resolver to accept a corrupted query answer. This is a standard attacker model, used by many previous studies including [19, 21]. The full list of attacker capabilities in our finite-state model is summarized here. Due to the nature of a non-deterministic finite automaton (NFA), all attacks involving any combination of the capabilities are exercised. To prevent state-space explosion in Murφ, only hostnames recognized by the modeled resolver, i.e., name1 through name6, are used by the attacker model.

1. Attacker may overhear any packets intended for the authoritative, TLD, or root server.
2. Attacker may record any reply packets to the resolver.
3. Attacker may modify any recorded reply packet and resend them to the resolver. However, the attacker may not compromise cryptography, thus limiting its packet modification capabilities to the following:
   (a) Attacker may modify any header bits
   (b) Attacker may modify the Question section.
   (c) Attacker may strip any number of RRs from a reply, including RRSIGs for other RRs.
   (d) Attacker may add any number of recorded A, NSEC3, DS, NS, or RRSIG RRs to a reply, so long as the added RRs were not modified by the attacker.
   (e) Attacker may create authoritative A, NSEC3, and DS RRs with corresponding RRSIGs signed by the attacker's own key, and add them to a reply.
   (f) Attacker may modify the contents of any A or NS glue record.

4.5 Security Invariants

We run the DNSSEC model in Murφ to check if any reachable state violates any security invariants. These invariants, which characterize the intended security properties of DNSSEC, are all logical expressions based on the state of the local recursive resolver. The first set of invariants checks that the local resolver has not recorded an spoofed answer for one of the queried names. Thus, if the answer to name[1-6] is valid, its answer must be the value intended by the authoritative server: An A RR with the correct address for name 1, an indication of secure delegation with the proper DS for name 2, indications of insecure delegation with the correct child zone address for names 3 and 4, and indications of non-existence of names 5 and 6.

Another invariant checks that no key other than the correct TLD and authoritative zone keys become accepted in resolvers attested cache. The next invariants check that the local resolver's knowledge of the addresses of the TLD and authoritative servers have not been spoofed.

The last invariant checks for the integrity of the attestation chain. We feel that it is a desirable property that a record be

considered valid at the local recursive resolver for only as long as all of the other records that form this record's attestation chain back to the trust anchor remain valid. For example, for an A RR with owner name www.example.com., the RR attestation chain is [1 ". DNSKEY" $\stackrel{signs}{\rightarrow}$ 2 "com. DS+RRSIG" $\stackrel{signs}{\rightarrow}$ 3 "com. DNSKEY" $\stackrel{signs}{\rightarrow}$ 4 "example.com. DS" $\stackrel{signs}{\rightarrow}$ 5 "example.com. DNSKEY" $\stackrel{signs}{\rightarrow}$ 6 "www.example.com. A+RRSIG"]. In our model, this maps to the invariant that while any of the query answers at the resolver are valid (representing 6), none of auth_DNSKEY (5), auth_DS (4), tld_DNSKEY (3), or tld_DS (2) should expire, since this would break the attestation chain to the trust anchor.

4.5.1 Temporal Inconsistency Discovered

The attestation chain temporal integrity invariant is in fact violated during our run of Murφ. The DNSSEC protocol only specifies temporal constraints between TTL of a RR and the signature validity period of its corresponding RRSIG; there are no constraints between the TTL of an RR and the validity period of another signature in its attestation chain. Thus, a signature within the attestation chain may expire before the RR to which it is suppose to attest, since there are no further constraints specified by the RFCs on a valid attestation chain for the data once it enters the cache. A fully trusted attestation chain *does* exists at the time that an signed RR is received and validated by the local recursive resolver; this invariant violation is thus significant in admittedly extra-ordinary scenarios where data from a once-trusted authoritative zone is later found untrustworthy, such as key compromise.

Using the example from the previous section, consider the case where the key of "example.com." is compromised, leading to a signed "example.com." DS+RRSIG that validates a key controlled by the attacker. If the TTLs of RRs under the authority of "example.com.", such as the A RR for "www.example.com.", depended upon the validity of all of the signatures tracing back to the trust anchor, this period of compromise would at least be bound by the expiration of "example.com." DS during the routine key rollover for "example.com.". However, if RRs for "example.com." depend only on the expiration of their associated RRSIG, then the attacker may create RRSIGs with arbitrarily long validity periods, extending the period of compromise for RRs under the authority of "example.com." indefinitely, even past key rollover.

One potential remedy for this invariant violation requires resolver logic be strengthened beyond RFC 4033's recommendations. The resolver cache may specify that a RRSet may not have TTL expiration time after the expiration time of ANY signatures that form its attestation chain, not just the RRSIG directly associated with the RRSet. However, this potentially synchronizes multiple TTL expirations for RRs in lower zones and across multiple resolvers on a RRSIG lifetime in an ancestor zone. Synchronicity in TTLs may mean potentially unacceptable increases in query traffic. Thus, we favor an alternative solution: since the problem arises when data from a supposed trusted zone is later repudiated, resolvers should cap TTL and signature lifetimes even when authoritative zone says they should be longer. While no protocol specification exists for an artificial resolver cap on the TTL and signature lifetime specified by an authoritative zone, this is prudent practice for resolvers to limit the exposure period of their customers to harmful data, in an extra-ordinary circumstance.

5 DNSSEC Security Invariant Violations and Guarantees

5.1 Inherent Security Violations

Our Murφ model checking found several security property violations in the DNSSEC protocol, some of them exploitable by a network attacker. The violations are described here and also summarized in Table 1.

5.1.1 Glue Record Redirection Violation

The first violation occurs due to the forgeability of glue records used in delegations, making all delegations vulnerable to redirection. Since attackers may modify unsigned glue records, Murφ found invariant violations resulting from the attacker changing TLD server and authoritative server addresses stored in local recursive resolver state. However, even with this *server* redirection, since the TLD and authoritative zones in our model are reached by secure delegations, Murφ did not find forgery of any signed query answers from the authoritative *zone* at the recursive resolver. See Section 5.2 for details of the model checking result. The mechanism for this protection was previously described in Section 3.7, which also notes how this redirection allows the attacker to more easily hinder resolver performance.

Glue record manipulation by the attacker also led to the violation of invariants checking the integrity of insecure delegations returned by the authoritative zone. Redirection of an insecure delegation, which always points to a standard DNS child zone, is the exact mechanism of the Kaminsky attack. Data served by the attacking server is accepted and

cached at the recursive resolver without validation, exposing the end-user to cache poisoning. Such an attack can only be prevented by the adoption of DNSSEC by the child zone, which secures the delegation.

5.1.2 Inter-operation with DNS

To generalize the consequences of inter-operation with standard DNS zones, we note that a DNSSEC local recursive resolver cannot provide secure answers to the stub resolver unless the resolution process queries only DNSSEC zones starting at the trust anchor. An intervening standard DNS zone requires an insecure delegation, meaning the local DNSSEC resolver will not be able to form the full attestation chain required to verify the final answer from the trust anchor. Since it precludes verification at the recursive resolver, any DNS-DNSSEC inter-operation causes an insecure, forgeable answer to be passed to the stub resolver. Since users are not informed of insecure query results due to the current absence of software interface indicators, inter-operation with DNS effectively exposes users trusting in DNSSEC resolvers to attacker exploitation.

Insecure delegations caused by DNS-DNSSEC inter-operation, and also NSEC3 opt-out, as described in the next sub-section, are instances of configurations whereby an insecure sub-namespace is created in a DNSSEC namespace. These types of configurations form the basis of our laboratory cookie-theft experiment in Section 6. Zone operators desiring the best DNSSEC security, especially at the enterprise level, should take care to avoid them.

5.1.3 NSEC3 Opt-out Violation

The next class of security invariant violations results from the attacker being able to change the content of a DNSSEC reply packet by subtracting or adding RRs. We found that the attacker was able to convert an insecure delegation to a unauthenticated denial-of-existence and vice-versa. To understand this, recall from Figure 4 that an insecure delegation using opt-out requires the presence of an authoritative NSEC3 record with opt-out, its associated RRSIG, and A and NS glue records, and that an unauthenticated denial of existence requires an authoritative opt-out NSEC3 record and its associated RRSIG. The network attacker has the capability to convert between these two response types simply by adding or subtracting the glue records.

The conversion from insecure delegation to denial-of-existence is useful for an attacker as a denial-of-service attack that may linger on the local resolver due to its caching of denial-of-existence responses. On the other hand, the ability to insert an insecure delegation may be used by an attacker to insert any arbitrary RR with an owner name that hashes between the names on the NSEC3 RR. This security violation confirms the property of NSEC3 opt-out as described in RFC5155 [8].

For example, an attacker may insert an A RR for spoof.example.com using an opt-out NSEC3 with owner name www.example.com and next name mail.example.com, as long as 'spoof' hashes between 'www' and 'mail'. We will show that this property is exploitable by experimentally carrying out a browser cookie-stealing attack detailed in Section 6. Attacks of this nature may only be prevented by the domain operator of "example.com." not using opt-out and including all owner names into the NSEC3 chain.

5.1.4 Mismanaged Signature Expiration

In this sub-section, we provide mitigation advice for a vulnerability, first mentioned by Bernstein [12], which is actually a consequence of the lack of signature revocation in DNSSEC. The vulnerability occurs when the signature expiration of A RRSets and associated RRSIGs is mismanaged. RRSIGs have a 30-day validity period according to the default settings in BIND, and DNSSEC lacks a revocation mechanism that can hasten the expiration date. Suppose that a domain owner decides to relinquish one set of IP addresses in favor of another and creates new A RRSets and RRSIGs. During the period when the RRSIGs associated with the old A RRSet are still valid, if attackers gain control of any IP address relinquished by the domain owner, they will be able to replay a completely valid DNSSEC response pointing an A RR at an attack server. This attack can be completely mitigated by domain owners not relinquishing IP addresses until they are certain all RRSIGs for RRs pointing to these IP addresses have expired.

5.2 DNSSEC Guarantees from Model Checking Completion

After removing the invariant that checked the integrity of zone server addresses, the invariant that checked the integrity of the denial-of-existence expressed by NSEC3 with opt-out, and the invariant that check the integrity of insecure delegations, our Murφ model ran to completion, exhausting all possible network attack combinations, without violating another invariant. The completion of execution implies that the modified protocol, as modeled, not containing opt-out NSEC3 or insecure delegations, contains no further vulnerabilities short of cryptographic compromise. This means that, when acquired by the resolver using a full chain of secure delegations, signed existent DNSSEC RRs and signed

non-opt-out NSEC3 denials-of-existence are safe against forgery by the network attacker described in our model, which is incapable of key compromise.

5.3 Faulty Resolver Logic Vulnerabilities

DNSSEC security depends on correct implementation of appropriate resolver logic. Section 5.1 described DNSSEC security violations found even with correct resolver validation logic, i.e. inherent to the DNSSEC protocol. To demonstrate the importance of resolver logic to DNSSEC implementation security, we will discuss some common attack paths that become exploitable vulnerabilities with faulty resolver logic. We begin with the attack paths and then discuss how to prevent them with correct validation logic.

Attackers may arbitrarily modify headers and add or subtract individual RRs from DNSSEC replies, opening up downgrade paths to DNS. For instance, an attacker that strips all RRSIG, DS, and NSEC3 RRs from a DNSSEC response packet will create a valid DNS packet. Also, an attacker may modify unsigned packet contents to introduce inconsistent information into reply packets. For example, attackers may set the AD (Authenticated Data) in a reply packet containing a forged RR with an invalid RRSIG, in an attempt to cause the resolver to accept the indicated success of remote validation and forgo its own validation. Finally, as previously stated, attackers may modify unsigned RRs contained in the reply packet, such as the glue A and NS RRs contained within the "additional" packet section.

5.3.1 Eliminating Vulnerabilities By *Attested Cache* Resolver Design

The resolver must thus be scrupulously designed to minimize susceptibility to attack by only trusting the validly signed content of reply packets. A resolver must not accept valid DNS responses where DNSSEC responses are expected, to eliminate downgrade attacks. Resolver logic must also not trust header fields. As a consequence, each resolver must perform its own verification of RR data in reply packets and not rely on upstream servers to indicate validation and query success/failure.

In effect, to answer user RR queries for a particular zone, the local recursive resolver must build an *attested cache* containing both RRs authoritative to that zone and a full attestation chain from the trust anchor to the zone, using only validly signed RRs contained in reply packets. Glue records may only be used as guides for which DNSSEC server to query next in a delegation and cannot be accepted into the attested cache. (The resolver logic we outlined in Section 4.3.2 is an instance of this attested cache implementation style.)

The importance of properly treating the unsigned records in a reply was anecdotally demonstrated during the time that this paper was being written, as a vulnerability was discovered where BIND incorrectly added unsigned RRs from the "additional" sections of DNSSEC responses to its cache [10]. The vulnerability was deemed a severe risk for DNSSEC users of BIND.

Resolvers must only securely answer the user's query when all of the information necessary to answer queried RR with integrity guarantee is contained within this attested cache, for example when a matching RR with valid RRSIG along with its full attestation chain exists or when the non-existence of the queried RR can be proven using NSEC3 RRs with valid RRSIGs and full attestation chains. Secure answers provided strictly from resolver attested cache are guaranteed against forgery, short of attacker compromise of zone keys, and end users may trust the integrity of resolver answers indicating such authentication via the AD bit, if received over a secure channel.

However, we again note that even a completely correct resolver cannot excise the inherent DNSSEC security property violations listed in Section 5.1.

5.4 NSEC3 Salt *Post*-Computation

As an aside which was previously mentioned, the cryptographic hashing of names in NSEC3 takes a salt input which, if sufficiently long, eliminates the possibility of *Pre*-computed dictionaries. However, as RFC 5155 specifies that the value of the salt is publicly available and identical for each NSEC3 RR in the domain, attackers still may obtain the full set of NSEC3 RRs for use in a dictionary attack *post*-acquisition. The identical salt does not increase the size of the required dictionary in this post-computation attack on the domain names. Also, any names in the domain revealed by the attack would remain valid past a re-salting unless they were also changed. Because of Bernstein's success in enumerating most of a domain in a matter of days [12], the effectiveness of NSEC3 for hiding names over an extended period of time is open to debate.

6 Implemented Attack Experiment

In this section, we will describe how we utilized configurations that create an insecure sub-namespace in a DNSSEC

Exploit	Legitimate Reply		Forged Reply	
	Signed RRs	Unsigned Glue RRs	Signed RRs	Unsigned Glue RRs
NSEC3 Opt-out	Opt-out NSEC3 covering "attack1.bank.com"		Opt-out NSEC3 covering "attack1.bank.com"	"attack1.bank.com. NS ns.atk.com" "ns.atk.com A 5.6.7.8"
Insecure Delegation		"attack2.bank.com NS ns.a2.bank.com" "ns.a2.bank.com A 1.2.3.4"		"attack2.bank.com NS ns.a2.bank.com" "ns.a2.bank.com A 5.6.7.8"

Table 4. Forged reply packets from "bank.com." zone used in cookie theft attack. 5.6.7.8 is an IP-address owned by attackers.

namespace, i.e. NSEC3 opt-out and also insecure delegations, to insert forged names into an experimental DNSSEC zone and steal browser-cookies. This experiment demonstrates that it is possible for a realistic attacker to exploit the security property violations we observed to subvert security policies that rely on membership in a particular domain. While we recognize that a man-in-the-middle network attacker may steal browser-cookies via means other than DNSSEC, we exploit DNSSEC for cookie theft primarily as a convenient demonstration that our observed security invariant violations allow an attacker to successfully subvert a DNSSEC domain and fool existing stub resolver and end-user software, raising security implications discussed in Section 6.3.

For our experiment, we set up a server running a BIND 9.6 instance for the hypothetical authoritative DNSSEC zone "bank.com.", containing an A record for the name "www.bank.com", an opt-out NSEC3 that covers the name "attack1.bank.com", and an insecure delegation to a zone "attack2.bank.com", so named for illustration purposes. This server also hosts a legitimate web page at "www.bank.com", which sets secure and insecure cookies with domain equal to "bank.com", as well as a third-party web page containing <img>tags linked to the zones "attack1.bank.com" and "attack2.bank.com". We also set up a user machine running web browsers, the OS stub resolver, and another BIND instance operating as the recursive DNSSEC resolver, that communicates with the stub resolver over the loopback interface. Finally, we set up an attacker machine which can overhear and inject DNS packet traffic between the recursive resolver and the zone server. While this scenario places the experimental attacker as a man-in-the-middle, the only information used by the attacker from the overheard DNSSEC request packet is the TXID. Thus, it is also possible to mount this attack as a via Kaminsky-style out-of-path means.

6.1 Attacking Name Insertion

In the first exploit step, our attacker attempts to poison the local recursive DNSSEC resolver by inserting an A record pointed at the attacker server with owner name containing the suffix "bank.com". This may be done using both the the opt-out NSEC3 RR (creating an A RR for "attack1.bank.com") and the insecure delegation ("attack2.bank.com"). In either case, the attacker must first get the user to initiate recursive resolution for the attacking A RR on the local DNSSEC resolver. In our experiment, this was initiated by two means: the user actually typing the name into the browser address bar, and the user accessing a third-party page with an image hosted at "attack[12].bank.com". In the wild the resolution may be initiated via by phishing email, <img>tags on third-party sites, or other means. Then, while the local recursive resolver queries the legitimate "bank.com." DNSSEC server, our attacking server sends a forged DNSSEC reply packet with TXID matching the query to the local resolver, in a race with the legitimate reply. Table 4 summarizes the forged reply packets.

Table 4 also demonstrates how our attack is feasible for an out-of-path attacker. The signed RRs used in the forged reply are public and available to the attacker by simply querying the "bank.com" DNSSEC zone. Thus, the only "secret" information copied from request to forged reply is the TXID. The attacker needs only to guess the TXID to execute this attack without man-in-the-middle capabilities. This implies an out-of-path attacker may also mount a Kaminsky-style attack that requests many bogus sub-names of "attack[12].bank.com" to create a birthday-problem instance that matches TXID.

In our experiment, the name-insertion attack succeeds whenever the forged reply packet arrives at the local resolver ahead of the legitimate reply, as the TXID is copied from request to spoofed response. In both cases, an insecure delegation is created that causes the resolver to query the attacking server and accept the forged "attack[12].bank.com"

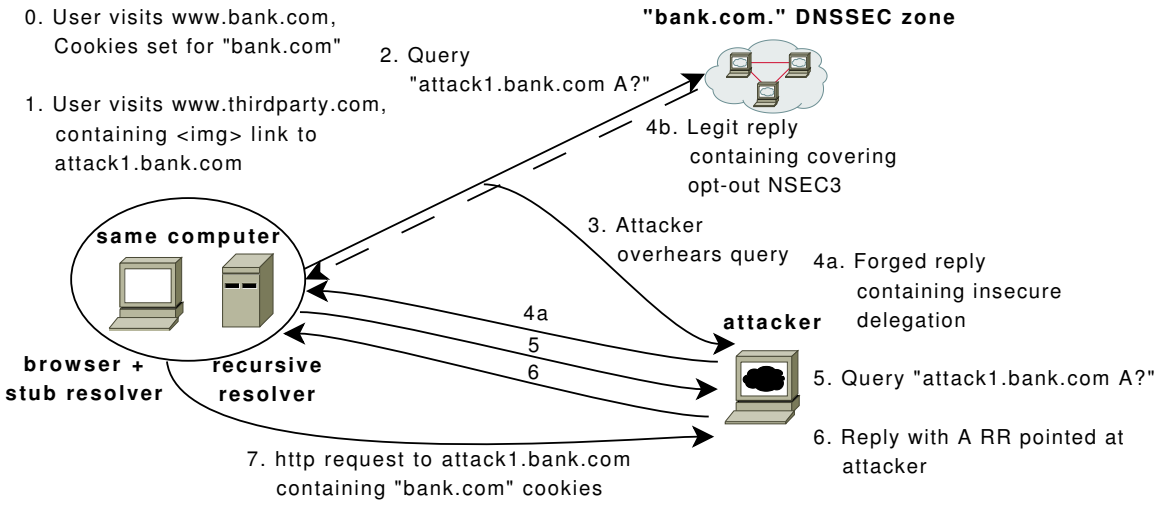


Figure 5. Illustration of NSEC3 Cookie Theft Attack. Packet 4a wins the race against packet 4b.

A RR in its reply. This forged A RR also poisons the cache of the local server, so that subsequent DNSSEC queries for "attack[12].bank.com" by users of the local resolver return the attack site address without requiring more injected attack packets.

6.2 Cookie Theft

To steal user cookies once the false names have been inserted on the local DNSSEC resolver, the attacker utilizes browser policy governing the cookie "domain" setting. The policy specifies that non-secure cookies be sent in all http requests made to sites which are sub-domains of the cookies' "domain" setting. In our experiment, the attack web site at "http://attack[12].bank.com", hosted on the attack server, receives in http requests all legitimate non-secure cookies set with domain equal to "bank.com". The coarse-grain setting for cookie domain required for this attack reflects a common practice. For example, all of the cookies for PayPal are set with domain equal to "paypal.com", even when the actual web pages are served from the address "www.paypal.com".

After the name insertion on the local DNSSEC resolver, the cookie theft succeeds in our experiment any time the user has active cookies set by "http://www.bank.com" and subsequently makes a http request for any object (images, web pages, etc.) in the "attack[12].bank.com" domain. Even if the name insertion has not yet occurred, the http request to "http://attack[12].bank.com" itself generates a predicate DNSSEC lookup that creates an opportunity for the spoofed name insertion. Both the name insertion and the cookie theft occur automatically after the single originating user action of visiting the attack site or a third-party site linking to attack site. The cookie theft is also very difficult for the user to detect, since the stolen payload is carried by in a *request* to the attacker, allowing the attacker to return a visually benign object or make no response at all. Figure 5 illustrates the entire attack using NSEC3 opt-out.

In order to steal secure cookies, the user must open "https://attack[12].bank.com", as browser policy will only send secure cookies over secure https. This makes the attack slightly more difficult, since the attacker should not possess Certificate Authority-validated credentials for encrypting the https connection. In our experiment this limitation was bypassed by the user clicking through a browser warning dialog stating incorrect credentials, for Opera and older version of Firefox and Internet Explorer. The attacker in the wild may also use one of the CA-spoofing methods detailed during BlackHat USA 2009 [20, 16], where attackers obtains CA-validated credentials for a domain name containing a null character, such as 'bank.com\0.attacker.com', that become valid for the domain name expressed before the null character due to faulty browser implementation. Using these certificates, stealing secure cookies becomes as simple as stealing non-secure ones.

6.3 Implications

We have experimentally demonstrated how a network attacker can exploit NSEC3 opt-out and also insecure delegations to insert an illegitimate name into a DNSSEC zone. We have also shown the feasibility of such name-insertion via Kaminsky-style out-of-path means. Illegitimate name insertion may be used for cookie theft, as we have demonstrated. Pharming attacks, which are a form of phishing

where attacker page is shown at an address that legitimately belongs to the victim domain, are also made possible by this name-insertion characteristic.

7 Security Advice and Conclusion

DNSSEC is a complex system, containing many options reflective of efforts to balance requirements at disparate zones and also to support incremental adoption. While DNSSEC is the same protocol at every level of the DNS hierarchy, the deployment considerations and trade-offs are different for domains at different levels. Options designed to support TLDs may in turn represent dangerous configurations for enterprises. Furthermore, DNSSEC must be operated by many participants, such as domain administrators, software implementers, and ISPs. Thus, in the interest of clarity, we summarize here the requirements for DNSSEC to be fully secure from end-user lookup to end-user reply:

1. DNSSEC adoption by authoritative zone
2. Authoritative zone to not use NSEC3 opt-out and to have no insecure delegations
3. All ancestor zones (root and TLD) to adopt DNSSEC and guarantee secure delegations at every step from trust anchor authoritative zone
4. DNSSEC adoption by local recursive resolver
5. Secure channel in the last-hop between stub and recursive resolvers

In addition, to support incremental adoption, DNSSEC also requires indicators of DNS lookup security to be implemented in end-user interfaces.

It is clear that many parts of the DNS ecosystem need to participate in DNSSEC in order for anyone to benefit, perhaps dampening the enthusiasm for DNSSEC early-adoption. We hope the planned DNSSEC deployment of the root and TLD zones generates sufficient momentum towards adopting end-to-end DNS security.

As we observed, several DNSSEC configurations of interest, such as NSEC zone enumeration and NSEC3 opt-out, result from protocol design trade-offs that support higher-performance off-line authenticated denial-of-existence. We believe that cryptographic research regarding authenticated denial-of-existence, possibly toward efficient off-line techniques that do not reveal any information about extant DNS resources, may be valuable.

To conclude this study of DNSSEC security, we offer the following advice, also summarized in Table 1, to the various operators and users of DNSSEC to eliminate exploitability of the security property violations described in this study.

- For administrators running an DNSSEC server authoritative over a domain such as 'bank.com.', we advise that all NSEC3 records NOT use opt-out. We also advise that any insecure delegations from this zone be made secure with the adoption of DNSSEC by the delegation-target zone, to eliminate mechanisms for falsified name insertion and DNS-DNSSEC interoperation.

  To eliminate replay attacks, domain owners should not relinquish IP addresses until they are certain all RRSIGs for RRs pointing to these IP addresses have expired.

- For website designers, we urge a fine-grained cookie "domain" setting. Coarse-grained cookie "domain" setting, as we have shown, can be utilized as an avenue for cookie theft via DNS name insertion. In our experiment, if the cookie domains were set to a finer-grain that covers only the web pages that actually require these cookies, the attack scenario in Section 6 would have been prevented under DNSSEC, since it is impossible to forge records that prepend a subdomain to an existent name such as "www.bank.com".

- For DNSSEC software implementers, we emphasize the importance of resolver software logic to the security of DNSSEC. Our collected resolver software recommendations are:

  - Bound RR TTL lifetime on the signature validity period of all records forming the attestation chain to the trust anchor, not just the single RRSIG covering the RR.
  - Do not trust the header bits of DNSSEC reply packets. As a consequence, all resolvers must validate the content of DNSSEC reply packets themselves.
  - Build an *attested cache* only containing signed RRs with full attestation chain to the trust anchor. Answers to user queries are only secure when formed entirely from contents of this attested cache.
  - Use glue records only as indications of delegation and pointers to child zone server address, but not as data that can enter the attested cache.

- For ISPs, local recursive resolvers must request all DNSSEC RRs to be included in packets to prove RR integrity at the closest recursive resolver to the end user. A secure channel between this recursive resolver and the end user's stub resolver is required to guarantee DNSSEC integrity all the way to the end-user.

- For end user software vendors, especially browsers, we urge the development of user-interface elements indi-

cating the security/insecurity of a DNSSEC lookup.

We hope that this study will further DNSSEC adoption by identifying its security properties, and that the advice offered will lead to better DNSSEC security when it is deployed.

Acknowledgments We thank Roy Arends, Olafur Gudmundsson, and Ben Laurie for very helpful comments and discussion of DNSSEC and NSEC3.

References

[1] RFC 2535. Domain Name System Security Extensions.

[2] RFC 2845. Secret Key Transaction Authentication for DNS (TSIG).

[3] RFC 2931. DNS Request and Transaction Signatures (SIG(0)s).

[4] RFC 4033. DNS Security Introduction and Requirements.

[5] RFC 4034. Resource Records for the DNS Security Extensions.

[6] RFC 4035. Protocol Modifications for the DNS Security Extensions.

[7] RFC 4470. Minimally Covering NSEC Records and DNSSEC On-line Signing.

[8] RFC 5155. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence.

[9] BIND Security Advisory. DNS Cache Poisoning Issue ('Kaminsky bug'). `https://www.isc.org/sw/bind/forgery-resilience.php`, 07/08/2008.

[10] BIND Security Advisory. BIND 9 Cache Update from Additional Section. `https://www.isc.org/node/504`, 11/23/09.

[11] Wikipedia Article. Birthday Problem. `http://en.wikipedia.org/wiki/Birthday_problem`.

[12] Daniel Bernstein. Breaking DNSSEC. 3rd Usenix Workshop on Offensive Technologies, August 2009.

[13] Microsoft Security Bulletin. Vulnerabilities in DNS Could Allow Spoofing (953230). `http://www.microsoft.com/technet/security/Bulletin/ms08-037.mspx`, 07/08/2008.

[14] David Dill. The Murφ Verification System. Computer Aided Verification, 8th International Conference, 1996.

[15] Dan Kaminsky. It's the End of the Cache as We Know It. BlackHat USA, Auguest 2008.

[16] Dan Kaminsky. Black Ops of PKI. BlackHat USA, August 2009.

[17] Dan Kaminsky. DNS 2008 and the New (old) Nature of Critical Infrastructure. BlackHat DC, February 2009.

[18] Robert Lemos. Poisoned DNS Servers Pop Up as ISPs Patch. `http://www.securityfocus.com/news/11529`.

[19] Gavin Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol using CSP and FDR. In *2nd International Workshop on Tools and Algorithms for the Constructions and Analysis of Systems*, 1996.

[20] Moxie Marlinspike. More Tricks For Defeating SSL. BlackHat USA, August 2009.

[21] John C. Mitchell, Vitaly Shmatikov, and Ulrich Stern. Finite-State Analysis of SSL 3.0. In *Seventh USENIX Security Symposium*, pages 201–216, 1998.

[22] Erica Naone. The Flaw at the Heart of the Internet. *Technology Review*, November/December 2008.