DNSSEC
Domain Name System Security Extensions

Jelte Jansen
Radboud University, June 2nd, 2015
DNSSEC
Rebuilding the airplane in-flight since 1995

Jelte Jansen
Radboud University, June 2nd, 2015
SIDN

- Domain name registry for .nl ccTLD

- > 5.5 million domain names

- 2.4 million domain names secured with DNSSEC
SIDN Labs

• R&D team SIDN

• Improve services of SIDN

• Center of expertise

• Increase security of the Internet (mostly in .nl)

• Facilitate external research
DNSSEC Recap in 27.4 seconds

- www.sidn.nl?
- 2001:7b8:c05::80:5
- Root
- .nl
- sidn.nl
- Resolver
- www.sidn.nl?
Addresses are just one type of DNS data

- A (IPv4 address)
- AAAA (IPv6 address)
- MX (mail server name)
- NS (name server name)
- TXT (arbitrary text data)
- CNAME (canonical name pointer)
- GPOS (geographical position)

and many, many more
DNS is used in...

- web
- mail
- firewalls
- network configuration
- voip
- games
- ip-television
- p2p
- streaming
- etc.
- etc.
- etc.

With very few exceptions, nearly *everything* uses DNS.
DNS is used in...

• web
• mail
• firewall
• network
• voip
• games
• ip-television
• p2p
• streaming
• etc.
• etc.
• etc.

With very few exceptions, nearly *everything* uses DNS.
DNS is old

Defined in RFC 1034 / RFC 1035 (1986)
DNS is extensive
DNS is extensive
DNS is scalable

• Over 250 million second-level domain names
• Distributed at delegation points
• You can run your own dns for your own zone
• Answers can be easily cached
DNS is insecure as %?)!

- Mostly UDP so no handshake/source verification
- Originally, only 16 bits of data to match answer to question (qid section)
- Easy to spoof answers
DNS is insecure as %?

The Flaw at the Heart of the Internet

Dan Kaminsky discovered a fundamental security problem in the Internet and got people to care in time to fix it. It’s a dramatic story with a happy ending—but we were lucky this time.

By JERICK NAGLE

As Kaminsky and his colleagues, most notably Peter Eckersley, were working on the design of a new secure DNS system around 2002, they received a call from a colleague who had discovered a large-scale sniffing attack on their network. This attack was not just a single incident but a new class of threat—domain name system (DNS) poisoning.

DNS is a fundamental part of the Internet infrastructure, used to translate domain names into IP addresses. Kaminsky’s expertise is in the Internet’s distributed naming system, which he has studied extensively. He was one of the first to recognize the potential for DNS poisoning, a type of attack that can be used to redirect traffic to malicious sites.

The attack involved a flaw in the way DNS servers handle requests for domain names. Instead of using a secure channel to verify the authenticity of requests, DNS servers simply rely on the integrity of the domain name itself. This makes it easy for an attacker to create a fake name resolution and redirect traffic to a malicious site.

Kaminsky and his team worked to develop a secure alternative to DNS, called DNS Security Extensions (DNSSEC). This solution involves adding digital signatures to DNS messages, allowing users to verify the authenticity of domain names.

The deployment of DNSSEC has been gradual, but it has played a critical role in improving the security of the Internet. Kaminsky’s work on DNSSEC helped to raise awareness of the importance of secure communication on the Internet and has contributed to the development of more secure systems.

Kaminsky’s research has also had implications beyond DNS. It has highlighted the need for secure communication on the Internet, and has inspired the development of new technologies to protect against similar types of attacks.

Kaminsky’s work has been widely recognized, and he has received numerous awards for his contributions to the field of network security. His research has had a profound impact on the Internet and has helped to make it a more secure and reliable platform.

Kaminsky’s insight into the vulnerabilities of DNS was a wake-up call for the Internet community. It highlighted the importance of security in the design of network technologies and raised awareness of the potential for attacks on the Internet.

Kaminsky’s work on DNSSEC has helped to improve the security of the Internet and has had a lasting impact on the field of network security. His research has helped to make the Internet a more secure and reliable platform for communication.
Choices of protection:

- Protect the channel (DNSCurve, DNSCrypt)

- Protect the data (DNSSEC)
A short history of DNSSEC

- Flaws identified around 1990 (and probably earlier)

- Flaws publicized in 1995

- Attempt 1 for DNSSEC in 1997 (RFC 2065)
  - Too hard to implement

- Attempt 2 for DNSSEC in 1999 (RFC 2535)
  - Did not scale

- Attempt 3 for DNSSEC ('dnssec-bis'...) in 2005 (RFC 4033-4035)
  - Additional features later added to 3rd version (e.g. NSEC3 in 2008)

- Note: Kaminsky's exploit was published in 2008
Attacks are real

The Flaw at the Heart of the Internet

DAN KAMINSKY DISCOVERED A FUNDAMENTAL SECURITY PROBLEM IN THE INTERNET AND GOT PEOPLE TO CARE IN TIME TO FIX IT. IT'S A DRAMATIC STORY WITH A HAPPY ENDING ... BUT WE WERE LUCKY THIS TIME.

By JERICK NAONE

D an Kaminsky, an uncharacteristically quiet dude, was looking for bugs earlier this year when he happened upon a flaw at the core of the Internet. The security researcher was using his knowledge of Internet infrastructure to come up with a better way to stream videos to users. Kaminsky's expertise is in the Internet's domain name system (DNS), the protocol responsible for matching websites and URLs with the numeric addresses of the servers that host them. The same content can be hosted by multiple servers with several addresses, and Kaminsky thought he had a great trick for directing users to the server best able to handle their requests at any given moment.

Normally, DNS is reliable but not nimble. When a computer requests DNS to translate the numeric address associated with a given URL, it receives the answer for a period of time known as “time to live,” which can be anywhere from seconds to days. This helps reduce the number of requests the server makes. Kaminsky's idea was to bypass this time by allowing the server to get a fresh answer every time it wanted to know a site's address. Consequently, traffic on Comcast's network would be seen to be sent to a single address at every moment, rather than to whatever address had already been served. Kaminsky was sure that the strategy could significantly speed up content distribution.

It was only later, after telling casually about the idea with a friend, that Kaminsky realized his “trick” could completely break the security of the domain name system and, therefore, of the Internet itself. The time to live, it turns out, was at the core of DNS security: being able to bypass it allowed for a wide variety of attacks. Kaminsky wrote a little code to make sure the situation was as bad as he thought it was. “Once I saw it work, my stomach dropped,” he says. “I thought, “What do I do about this?” This affects everything.”

Kaminsky's technique could be used to direct Web surfers to any Web page an attacker chose. The most obvious use is to send people to phishing sites (websites designed to trick people into entering banking information and other personal information, allowing an attacker to steal their identities) or other false versions of Web pages. But he danger is even worse: protocols such as those used to deliver e-mail or for secure communication over the Internet ultimately rely on DNS. A creative attacker could use Kaminsky’s technique to intercept sensitive e-mail or to create forged email addresses in the certificates that ensure secure transactions between users and banking websites. “Every day I find another domino,” Kaminsky says. “Another system that relies on DNS is ... it means, literally, you look around and see everything that's using a network: anything that's using a network— everything is probably using DNS.”

Kaminsky called Paul Vixie, president of the Internet Systems Consortium, a nonprofit organization that supports several aspects of Internet infrastructure, including the software most commonly used in the domain name system. “Paul, if somebody wants to report a problem, you can say that it's going to take a lot of time for them to explain it—a week, maybe a week to a month. It's probably months before you can fix it.”

Kaminsky says. “In this case, it took 20 seconds for him to explain the problem, and another 15 seconds for him to answer my objections. Altogether, I said: ‘Paul, I'm speaking to you over an insecure cell phone. Please do not call my home phone, what you just said made over an insecure cell phone. Again!’

Frustrated, Vixie said that the vulnerability wasn't located in any particular hardware or software but in the design of the DNS protocol itself. It wasn't clear how to fix it. In secret, Kaminsky and Vixie gathered together some of the top DNS experts in the world for people from the U.S. government and...
Attacks are real

Report Claims DNS Cache Poisoning Attack Against Brazilian Bank and ISP

By Larry Selter | Posted 2009-04-22  

OPINION: Attack shows the potential for serious spoofing attacks that could leave end users helpless. The only real solution is DNSSEC, which will take years to implement under the best of circumstances.

DnsCachePoisoning.png

As Kaminsky succinctly stated, he was not looking for bugs earlier this year when he happen upon a flaw at the core of the Internet. The security researcher was using his knowledge of Internet infrastructure to come up with a better way to stream videos to users. Kaminsky’s expertise is in the Internet’s domain name system (DNS), the protocol responsible for matching websites’ URLs with the numeric addresses of the servers that host their data. The same system can be abused by multiple servers with several addresses, and Kaminsky thought he had a great trick for directing users to the servers best able to handle their requests at any given moment.

Normally, DNS is reliable but not reliable. When a computer says, “I need to look up this address,” it consults a local cache of names and addresses. If the address is not found there, the computer sends a request to a large central server. At the central server, the request is sent to another server, and so on until the answer is found. This is the way DNS is supposed to work, but it is also how attackers have been able to abuse the system.

Kaminsky’s trick was to advertise a location of a server that was not actually there. When a computer asked for the address, it would get a fresh answer every time it wanted to know a site’s address. Consequently, traffic for a common website would be sent to the optimal address at every moment, rather than to the true address it had already been served. Kaminsky was sure that the strategy could significantly speedup content distribution.

It was only later, after talking casually about the idea with a friend, that Kaminsky realized his “trick” could completely break the security of the domain name system and, therefore, of the Internet itself. The time to live, it turns out, was at the core of DNS security. By being able to bypass it allowed for a wide variety of attacks, Kaminsky wrote. A little code to make sure the situation was as bad as he thought it was. “Once I saw it work, my stomach dropped,” he said. “I thought, ‘What the heck am I going to do about this?’ This affects everything.”

Kaminsky’s technique could be used to direct Web traffic to any Web page an attacker chooses. The most obvious use is to send people to phishing sites (websites designed to trick people into entering banking information and other personal information, allowing an attacker to read their identity) or other false versions of Web pages. But a more dangerous worm protocol tool is how used to deliver e-mail or secure communications over the Internet ultimately rely on DNS. A creative attacker could use Kaminsky’s technique to intercept sensitive e-mail or to create forged versions of the certificates that ensure secure transactions between users and banking websites. “Every day I find another domain,” Kaminsky says. “Another phishing site I find DNS in... I mean, literally, you look around and see anything that’s using a network, anything that’s using a network—and it’s probably using DNS.”

Kaminsky called Paul Vicki, president of the Internet Security Consortium, a nonprofit organization that supports several aspects of Internet infrastructure, including the software most commonly used in the domain name system. “Usually, if somebody wants to report a problem, people think its going to take a lot of time for them to explain it—maybe a whiteboard, maybe a Word document ten rows,” Vicki says. “In this case, it took 20 seconds for him to explain the problem, and another 20 seconds for him to answer my objections.”

If Kaminsky and others gathered together some of the top DNS experts in the world and people from the U.S. government and
Attacks are real

Report Claims DNS Cache Poisoning Attack Against Brazilian Bank and ISP

By Larry Seltzer | Posted 2009-04-22   Email   Print

**Title**

OPINION: Attack shows the potential for serious spoofing attacks that could leave end users vulnerable. The only real solution is DNSSEC, which will take years to implement under the best of circumstances.

**Summary:** The DNS cache poisoning attack is a real threat. Arbor Networks have provided more details in their "Attack Activity" analysis, SANS confirmed HD Moore's statement on DNS cache poisoning and independent sources are starting to see evidence of attempts on their local networks, in what appears to be an attempt to take advantage of the "recent" DNS cache poisoning vulnerability. The DNS server is NOT overtly vulnerable, however, it may be subtly vulnerable in certain circumstances.

By Dancho Danchev for Zero Day | July 29, 2006 -- 03:24 GMT (04:24 PDT)
Attacks are real

Report Claims DNS Cache Poisoning Attack Against Brazilian Bank and ISP

By Larry Seltzer | Posted 2009-04-22  

OPINION: Attack shows the potential for serious spoofing attacks that could leave end users exposed. The only real solution is DNSSEC, which will take years to implement under the best of circumstances.

DNS cache poisoning attacks exploited in the wild

GOOGLE’S MALAYSIAN DOMAINS HIT WITH DNS CACHE POISONING ATTACK

Google’s Malaysian domains google.com.my and google.my were hijacked, redirecting users to a webpage that announced the attack was perpetrated by a Pakistani group called Madleets. MYNIC, the sole administrator for web addresses in Malaysia confirmed the attack in a statement.

“We can confirm there was unauthorised redirection of www.google.com.my and www.google.my to a malicious page.
Attacks are real

Report Claims DNS Cache Poisoning Attack Against Brazilian Bank and ISP

By Larry Seltzer | Posted 2009-04-22  

OPINION: Attack shows the potential for serious spoofing attacks that could leave end users without a patch. The only real solution is DNSSEC, which will take years to implement under the best of circumstances.

Update: Arbor Networks have provided more details in their press release. HD Moore’s statement in DNS cache poisoning attacks are starting to see evidence of a “recent attempt to take down a client.”

DNS cache poisoning attack exploited in the wild

DNS poisoning slams web traffic from millions in China into the wrong hole

ISP blames unspecified attack for morning outage

By John Leyden, 21 Jan 2014
DNSSEC: Requirements

- Spoof protection (duh)
- Replay protection (outside of signature expiry)
- Backwards compatibility with non-DNSSEC-aware resolvers
- Support for offline signing (private keys not on nameservers)
- No global revocation list of public keys
- Crypto algorithm agility

More requirements defined in RFC 4033
DNSSEC: OK let's just sign some stuff!

But what do you sign?

• Every packet?

Problem: answers not in fixed form, additional data, and depend on query.

• Every Resource Record (RR)?

Problem: a signature for *every* RR is too much data
DNSSEC: OK let's just sign some stuff!

But what do you sign?

A lookup is a combination of (name, class, type)

So that becomes the minimum data set: the RRset

A Resource Record Set (RRset), is the set of records of a given type for a given domain.

<table>
<thead>
<tr>
<th>name</th>
<th>TTL</th>
<th>CLASS</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>tjeb.nl.</td>
<td>3600</td>
<td>IN</td>
<td>NS</td>
<td>ns.tjeb.nl.</td>
</tr>
<tr>
<td>tjeb.nl.</td>
<td>3600</td>
<td>IN</td>
<td>NS</td>
<td>ext.ns.whyscream.net.</td>
</tr>
<tr>
<td>tjeb.nl.</td>
<td>3600</td>
<td>IN</td>
<td>NS</td>
<td>ns-ext.nlnetlabs.nl.</td>
</tr>
<tr>
<td>tjeb.nl.</td>
<td>3600</td>
<td>IN</td>
<td>TXT</td>
<td>&quot;This is my zone&quot;</td>
</tr>
<tr>
<td>tjeb.nl.</td>
<td>3600</td>
<td>IN</td>
<td>TXT</td>
<td>&quot;There are many like it&quot;</td>
</tr>
<tr>
<td>tjeb.nl.</td>
<td>3600</td>
<td>IN</td>
<td>TXT</td>
<td>&quot;But this one is mine&quot;</td>
</tr>
<tr>
<td>tjeb.nl.</td>
<td>3600</td>
<td>IN</td>
<td>A</td>
<td>178.18.82.80</td>
</tr>
<tr>
<td><a href="http://www.tjeb.nl">www.tjeb.nl</a></td>
<td>600</td>
<td>IN</td>
<td>A</td>
<td>178.18.82.80</td>
</tr>
</tbody>
</table>

(Q: How many RRsets in above zone snippet?)
DNSSEC: Normal query

> dig NS tjeb.nl @ns.tjeb.nl

<snip>

;; ANSWER SECTION:
tjeb.nl. 3600 IN NS ns-ext.nlnetlabs.nl.
tjeb.nl. 3600 IN NS ext.ns.whyscream.net.
tjeb.nl. 3600 IN NS ns.tjeb.nl.

<snip>
DNSSEC: DNSSEC query

> dig +dnssec NS tjeb.nl @ns.tjeb.nl

<snip>

;; ANSWER SECTION:
tjeb.nl. 3600 IN NS ns.tjeb.nl.
tjeb.nl. 3600 IN NS ns-ext.nlnetlabs.nl.
tjeb.nl. 3600 IN NS ext.ns.whyscream.net.
tjeb.nl. 3600 IN RRSIG NS 8 2 3600 20150613023750 20150514021455 11499 tjeb.nl.

1lS4C3055edHcTXag7F+R1Kb61FTHNfQc2M6WEMGFD+yG+YGaKPCIZnD / NTzmhOa+j8APVOVYSEyffxruyTt+bc6hLmnD3hGV9ScGxe3yg4mMeczm2tTVh35RLg1KaeJsFpGNwSkts9V8oSANtvx2/51ZwOqfy15o3b7PldywWwE=

<snip>

RRSIG = Signature over an RRset
### DNSSEC: RRSIG

<table>
<thead>
<tr>
<th>Type Covered</th>
<th>Algorithm</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original TTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signature Expiration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signature Inception</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key Tag</td>
<td>Signer's Name</td>
<td></td>
</tr>
<tr>
<td>Signature</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- tjeb.nl. 3600 IN RRSIG NS 8 2 3600 20150613023750 20150514021455 11499 tjeb.nl.
- 11S4C3055edHcTXag7F+R1Kb61FTHNfQc2M6WEMGFD+yG+YGaKPCIZnD / NTzmhOa+j8APVOWYSeyffxruyTt+bc6hLmnD3hGV9ScGxe3yg4mMecz m2tTVh35RSLg1KaeJsFpGNwSkt9V8oSAnTvm2/51ZwOqfy15o3b7PLDY WwE=
**DNSSEC: RRSIG**

<table>
<thead>
<tr>
<th>Type Covered</th>
<th>Algorithm</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original TTL</td>
<td>Signature Expiration</td>
<td>Signature Inception</td>
</tr>
<tr>
<td>Key Tag</td>
<td>Signer's Name</td>
<td></td>
</tr>
</tbody>
</table>

| tjeb.nl. | 3600 | IN RRSIG NS 8 2 3600 20150613023750 20150514021455 11499 tjeb.nl. 11S4C3055edHcTXag7F+R1Kb61FTHNfQc2M6WEMGFD+yG+YGaKPCIZnD / NTzmhOa+j8APVOWYSe yffxruyTt+bc6hLmnD3hGV9ScGxe3yg4mMecz m2tTVh35RSLg1KaeJsFpGNwSkt9V8oSAnTvm2/51ZwOqfy15o3b7PLDY WwE= |
**DNSSEC: RRSIG**

<table>
<thead>
<tr>
<th>Type Covered</th>
<th>Algorithm</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNSSEC: RRSIG</td>
<td>NS</td>
<td>3600</td>
</tr>
<tr>
<td>Original TTL</td>
<td>20150613023750</td>
<td>11499 tjeb.nl.</td>
</tr>
<tr>
<td>Signature Expiration</td>
<td>20150514021455</td>
<td>11499 tjeb.nl.</td>
</tr>
<tr>
<td>Signature Inception</td>
<td>20150514021455</td>
<td>11499 tjeb.nl.</td>
</tr>
<tr>
<td>Key Tag</td>
<td>11S4C3055edHcTCPa7F+R1Kb61FTHNFQc2M6WEMGFD+yG+YGaKPCIZnD / NTzmhOa+j8APVOWYSeYffxruyTt+bc6hLmnD3hGV9ScGxe3yg4mMecz m2tTVh35RSLg1KaeJSpGNwSkt9V8oSAntVnm2/51ZwOqfy15o3b7PLDY WwE=</td>
<td></td>
</tr>
</tbody>
</table>
### DNSSEC: RRSIG

<table>
<thead>
<tr>
<th>Type Covered</th>
<th>Algorithm</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original TTL</td>
<td>Signature Expiration</td>
<td>Signature Inception</td>
</tr>
<tr>
<td>Signature Inception</td>
<td>Key Tag</td>
<td>Signer's Name</td>
</tr>
<tr>
<td>Signature</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
tjeb.nl. 3600 IN RRSIG NS 8 2 3600 20150613023750 20150514021455 11499 tjeb.nl.
11S4C3055edHcTXag7F+R1Kb61FTHNfQc2M6WEMGFD+yG+YGaKPCIZnD /
NTzmhOa+j8APVOWYseyffxruyTt+bc6hLmnD3hGV9ScGxe3yg4mMecz
m2tTVh35RSLg1KaeJsFpGNwSkt9V8oSAnTvm2/51ZwOqfy15o3b7PLDY
WwE=
```
DNSSEC: RRSIG

tjeb.nl.  3600  IN  RRSIG  NS  8  2  3600  20150613023750
20150514021455  11499  tjeb.nl.
11S4C3055edHcTXag7F+R1Kb61FTHNfQc2M6WEMGFD+yG+YGaKPCIZnD /
NTzmhOa+j8APVOWYSeyffxruyTt+bc6hLmnD3hGV9ScGxe3yg4mMecz
m2tTVh35RSLg1KaeJsFpGNgwSkt9V8oSAnTvm2/51ZwOqfy15o3b7PLDY
WwE=
### DNSSEC: RRSIG

<table>
<thead>
<tr>
<th>Type Covered</th>
<th>Algorithm</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNSSEC: RRSIG</td>
<td>IN</td>
<td>tjeb.nl.</td>
</tr>
</tbody>
</table>

#### Key Details:
- **tjeb.nl.** 3600 IN RRSIG NS 8 2 3600 20150613023750
- **Signature Expiration:** 20150514021455 11499 tjeb.nl.
- **Signature Inception:** 20150514021455 11499
- **Key Tag:** 1lS4C3055edHcTXag7F+R1Kb61FTHNfQc2M6WEMGFD+yG+YGaKPCIZnD
- **Signature:** /NTzmhOa+j8APVOWYSeyffxruyTt+bc6hLmnD3hGV9ScGxe3yg4mMecz
- **Signer's Name:** m2tTVh35RSLg1KaEJsFpGNwSkt9V8oSAnTvm2/51ZwOqfy15o3b7PLDY
- **WwE=**
DNSSEC: RRSIG

<table>
<thead>
<tr>
<th>Type Covered</th>
<th>Algorithm</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original TTL</td>
<td>Signature Expiration</td>
<td>Signature Inception</td>
</tr>
<tr>
<td>Signature Inception</td>
<td>Key Tag</td>
<td>Signer's Name</td>
</tr>
<tr>
<td>Signature</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

tjeb.nl. 3600 IN RRSIG NS 8 2 3600 20150613023750 20150514021455 11499 tjeb.nl.
1S4C3055edHcTXag7F+R1Kb61FTHNfQc2M6WEMGFD+yG+YGaKPCIZnD / NTzmhOa+j8APVOWYSeYffxruyenTt+bc6hLmnD3hGV9ScGxe3yg4mMecz m2tTVh35RSLg1KaeJsFpGNwSk9vV8oSANvTvm2/51ZwOqfy15o3b7PLDYWwE=
DNSSEC: RRSIG

<table>
<thead>
<tr>
<th>Type Covered</th>
<th>Algorithm</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original TTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signature Expiration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signature Inception</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key Tag</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signer's Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signature</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

tjeb.nl. 3600 IN RRSIG NS 8 2 3600 20150613023750 20150514021455 11499 tjeb.nl. 11S4C3055edHcTXag7f+R1Kb61FTHNfQc2M6WEMGFD+yG+YGaKPCIZnD / NTzmhOa+j8APVOWYSeyffxruyTt+bc6hLmnD3hGV9ScGxe3yg4mMecz m2tTVh35RSLg1KaeJsFpGNwSkt9V8oSAnTvm2/51ZwOqfy15o3b7PLDY WwE=
**DNSSEC: RRSIG**

<table>
<thead>
<tr>
<th>Type Covered</th>
<th>Algorithm</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original TTL</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Signature Expiration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Signature Inception</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Key Tag</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Signer's Name</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Signature</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
tjeb.nl.  3600  IN  RRSIG  NS  8  2  3600  20150613023750
          20150514021455  11499  tjeb.nl.
11S4C3055edHcTXag7F+R1kb61FTHNfQc2M6WEMGFD+yG+YGaKPCIZnD /
  NTzmhOa+j8APVOWYseyffxruyTt+bc6hLmnD3hGV9ScGxe3yg4mMecz
  m2tTVh35RSLg1KaeJsFpGNwSkt9V8oSAnTv52/51ZwOqfy15o3b7PLDY
  WwE=
```
DNSSEC: RRSIG

<table>
<thead>
<tr>
<th>Type Covered</th>
<th>Algorithm</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original TTL</td>
<td>Signature Expiration</td>
<td>Signature Inception</td>
</tr>
<tr>
<td>Key Tag</td>
<td>Signer's Name</td>
<td></td>
</tr>
</tbody>
</table>

Signature

tjieb.nl. 3600 IN RRSIG NS 8 2 3600 20150613023750 20150514021455 11499 tjieb.nl.
1134C3055edHcTXag7F+R1Kb61FTHNfQc2M6WEMGFD+yG+YGaKPC1ZmD_/NTzmhOa+j8APVOWYSeyffxrutyTt+bc6hLmnD3hGV9ScGxe3yg4mMecz/m2tTVh35RSLg1KaeJsFpGNwSkt9V8oSAnTvm2/51ZwOqfy15o3b7PLDYWwE=
DNSSEC: Signature content

Signature = \texttt{sign(RRSIG\_RDATA | RR(1) | RR(2)... )}

RRSIG\_RDATA is the wire format of the RRSIG RDATA fields with the Signer's Name field in canonical form and the Signature field excluded.

\texttt{RR(i) = owner | type | class | TTL | RDATA length | RDATA}
DNSSEC: DNSSEC query (again)

```plaintext
> dig +dnssec NS tjeb.nl @ns.tjeb.nl

;; ANSWER SECTION:
tjeb.nl. 3600 IN NS ns.tjeb.nl.
tjeb.nl. 3600 IN NS ns-ext.nl.netlabs.nl.
tjeb.nl. 3600 IN NS ext.ns.whyscream.net.
tjeb.nl. 3600 IN RRSIG NS 8 2 3600 20150613023750
20150514021455 11499 tjeb.nl.
11S4C3055edHcTXag7F+R1Kb61FTHNfQc2M6WEMGFD+yG+YGaKPCIZnD /
NTzmhOa+j8APVOWYseyffxruryTt+bc6hLmnD3hGV9ScGxe3yg4mMecz
m2tTVh35RSLg1KaeJsFpGNwSkt9V8oSAnTvm2/51ZwOqfy15o3b7PLDY
WwE=
```

<snip>
DNSSEC: Get public key for signature

> dig DNSKEY tjeb.nl @ns.tjeb.nl

<snip>

;; ANSWER SECTION:
tjeb.nl. 3600 IN DNSKEY 256 3 8
AwEAAee4BKqSMI/wEKdLXQyn+TzOjEMWG5IXy+WRGw+6MiKrbLit60eJ
xNXszf/zR55UUtMqP761AFkFwZgpmUs6ac3pYOTUYRVFjjG1/hnUF1/t
hd9uZLe1E3gwa5m6dcOHaspG5xYsJ2wEBmYj1z1xTh70892PwxVR9R9G
MKh4YyNt
tjeb.nl. 3600 IN DNSKEY 257 3 8
AwEAAcHR47QfC0d1PEQkAsKRh3VYFvUK1ierSdlT7HBS3/NOQ6ghVs9u
Yskdbs2pLSRbu4CSu6X0MgKZO01xoJhi6FqBa330c0Mmp/dd6AW4pNdZ
a4icP6fKT+HcPbLU9dUsrjDo13iXgUy3gs5BLG9KnTaLzW9KmxTInB
UHFLjZa70F1+ILNfJ/e1D6eX3C104nmGSWpO6OB+nQDz46ra23eGJ7E
NAu1/uhPcqeqXg3HWKjQHTzQW5XxVymhdXx/ILC3SZhsqNq1kKZjmHbg
7V1+iograUg1XEaxaOE25W9jrzvQnMxlZT8I9LTyyi1YArvxMCTcGkNW
Ri4Ca4/HEDs=

<snip>

**DNSKEY** = Public key of keyset
DNSSEC: Get public key for signature

tjeb.nl. 3600 IN DNSKEY 256 3 8
AwEAAee4BKqSMI/wEKdLXQyn+TzOjEMWG5IXy+WWRgw+6MiKrbLit60eJ
xNXszf/zR55UUtMqP761AFkFwZgpmUs6ac3pYOTUYRFjjG1/hnUF1/t
hd9uZLe1E3gwa5m6dcOHaspG5xYsJ2wEBmYj1z1xTh70892PwxVR9R9G
MKh4YyNt
DNSSEC: Get public key for signature

```
tjeb.nl. 3600 IN DNSKEY 256 3 8
AwEAAee4BKqSMI/wEKdLXQyn+TzOjEMWG5IXy+WRGw+6MiKrbLit60eJ
xNXszf/zR55UUtMqP761AFkFwZgpmUs6ac3pYOTUYRVFjjG1/hnUF1/t
hd9uZLe1E3gwa5m6dcOHaspG5xYsJ2wEBmYj1z1xTh70892PwxVR9R9G
MKh4YyNt
```
DNSSEC: Get public key for signature

tjeb.nl. 3600 IN DNSKEY 256 3 3
AwEAAee4BKqSMI/wEKdLXQyn+TzOjEMWG51Xy+WRGw+6MiKrbLit60eJ
xNXszf/zR55UUtMqP761AFkFwZgpmUs6ac3pYOTUYRVFjjG1/hnUF1/t
hd9uZLe1E3gwa5m6dcOHaspG5xYsJ2wEBmYj1z1xTh70892PwxVR9R9G
MKh4YyNt
DNSSEC: Get public key for signature

tjieb.nl. 3600 IN DNSKEY 256 3 8
AwEAAee4BKqSMI/wEKdLXQyn+TzOjEMWG5IXy+WRGw+6MiKrbLit60eJ
xNXsf/zR55UUtMqP761AFkFwZgpmUs6ac3pYOTUYRVFjjG1/hnUF1/t
hd9uZLe1E3gwa5m6dcOHaspG5xYsJ2wEBmYj1z1xTh70892PwxVR9R9G
MKh4YyNt
DNSSEC: Get public key for signature

tjeb.nl.   3600  IN  DNSKEY  256 3 8
AwEAAee4BKqSMI/wEKdLXQyn+TzOjEMWG5IXy+WRGw+6MiKrbLit60eJ
xNXszf/zR55UUtMqP761AFkFwZgpmUs6ac3pYOTUYRVFjjG1/hnUF1/t
hd9uZLe1E3gwa5m6dcOHaspG5xYsJ2wEBmYj1z1xTh70892PwxVR9R9G
MKh4YyNt
DNSSEC: Have RRSIG and DNSKEY, now what

- Validated signature with key(s)
- Validated keyset with (key-signing) key(s)
- Need to validate that key-signing key too...
- Ask parent
DNSSEC: Get digest of key at parent

> dig DS tjeb.nl @ns1.dns.nl

;; ANSWER SECTION:
tjeb.nl. 7200 IN DS 17992 8 2
764501411DE58E8618945054A3F620B36202E115D015A7773F4B78E0F952CECA

DS = Hash of a child's public key
**DNSSEC: Get digest of key at parent**

<table>
<thead>
<tr>
<th>Key Tag</th>
<th>Algorithm</th>
<th>Digest Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
> dig DS tjeb.nl @ns1.dns.nl
<snip>
;; ANSWER SECTION:
tjeb.nl. 7200 IN DS 17992 8 2
764501411DE58E8618945054A3F620B36202E115D015A7773F4B78E0F952CECA
<snip>
```
DNSSEC: Get digest of key at parent

```bash
> dig DS tjeb.nl @ns1.dns.nl
<snip>
;; ANSWER SECTION:
tjeb.nl. 7200 IN DS 17992 8 2
764501411DE58E8618945054A3F620B36202E115D015A7773F4B78E0F952CECA
<snip>
```
DNSSEC: Get digest of key at parent

> dig DS tjeb.nl @ns1.dns.nl
<snip>
;; ANSWER SECTION:
tjeb.nl. 7200 IN DS 17992 8 2
764501411DE58E8618945054A3F620B36202E115D015A7773F4B78E0F952CECA
<snip>
DNSSEC: Get digest of key at parent

> dig DS tjeb.nl @ns1.dns.nl
<snip>
;; ANSWER SECTION:
tjeb.nl.    7200  IN  DS 17992 8 2
764501411DE58E8618945054A3F620B36202E115D015A7773F4B78E0F952CECA
<snip>
DNSSEC: Get digest of key at parent

```bash
> dig DS tjeb.nl @ns1.dns.nl
<snip>
;; ANSWER SECTION:
tjeb.nl. 7200 IN DS 17992 8 2
764501411DE58E8618945054A3F620B36202E115D015A7773F4B78E0F952CECA
<snip>
```
Turtles all the way up

To check RR: check RRSIG
To check RRSIG: check DNSKEY
To check DNSKEY: check DS at parent
To check DS: check RRSIG of DS
To check RRSIG of DS, check DNSKEY
Etc.

That is, up to a 'trust anchor': a preconfigured trusted public key

Chain of trust: the set of signed elements from an anchor to the data
DNSSEC validation

In practice a validator goes top-down:

- **Trust Anchor**
- **parent zone**
  - key signing key
  - zone signing key
  - signature with Key Signing Key
- **child zone**
  - DS pointing to child
  - signature with Zone Signing Key
  - key signing key
  - zone signing key
  - signature with Key Signing Key
  - TXT Resource
  - signature with Zone Signing Key
DNSSEC on DNS recap

- www.sidn.nl?
- 2001:7b8:c05::80:5

Resolver

Root

.nl

.sidn.nl at 2001:610:0:800d::5

2001:7b8:c05::80:5

www.sidn.nl?
DNSSEC on DNS recap

- www.sidn.nl?
- 2001:7b8:c05::80:5
- Resolver

- Root
- .nl
- .nl at 2001:610:0:800d::5
- sidn.nl
- sidn.nl at 2001:610:0:800d::5
- www.sidn.nl?
- www.sidn.nl?
DNSSEC on DNS recap

www.sidn.nl?

2001:7b8:c05::80:5

www.sidn.nl?

.nl at 2001:7b8:606::85

www.sidn.nl?

sidn.nl at 2001:610:0:800d::5

www.sidn.nl?

2001:7b8:c05::80:5

Resolver
DNSSEC on DNS recap

- www.sidn.nl?
- 2001:7b8:c05::80:5

Resolver

www.sidn.nl?

www.sidn.nl at 2001:610:0:800d::5

sidn.nl at 2001:610:0:800d::5

Root

.nl

.nl
DNSSEC on DNS recap

- www.sidn.nl?
- www.sidn.nl?
  - Root
  - .nl
  - sidn.nl

2001:7b8:c05::80:5

Resolver

sidn.nl at 2001:610:0:800d::5
DNSSEC: Denial of existence

Problem: How to sign data that does not exist?

Normally you'd sign the RRsets of data, but if the name doesn't exist (NXDOMAIN) or there is no data of the queried type (NOERROR/NODATA), what can you sign?

Already decided that signing the packets was not an option.

Need to sign the 'holes' in the data.
DNSSEC: NSEC

NSEC records 'cover' the spots between existing names in a zone, by claiming 'there is no data between these names'.

NSEC records also have a list of types that do exist at their exact name

Say you have a zone with the names

a.zone. (with an address RR)
b.zone. (with a TXT RR)
c.zone. (with a TXT RR)

When signing, add nsec records

a.zone. NSEC b.zone. A
b.zone. NSEC c.zone. TXT
c.zone. NSEC a.zone. TXT
DNSSEC: Get digest of key at parent

```
> dig +dnssec nosuchdomain.tjeb.nl. @ns.tjeb.nl
<snip>
;; ANSWER SECTION:
newsubdomain.tjeb.nl. 3600 IN NSEC ns.tjeb.nl. NS RRSIG NSEC
<snip>
```

**NSEC: Shows what data exists and what does not**
DNSSEC: NSEC example

<table>
<thead>
<tr>
<th>Next Domain Name</th>
<th>Type Bit Maps</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns.tjeb.nl.</td>
<td>NS</td>
</tr>
<tr>
<td>RRSIG</td>
<td>NSEC</td>
</tr>
</tbody>
</table>

newsubdomain.tjeb.nl. 3600 IN NSEC ns.tjeb.nl. NS RRSIG NSEC
DNSSEC: NSEC example

newsubdomain.tjeb.nl. 3600 IN NSEC ns.tjeb.nl. NS RRSIG NSEC
DNSSEC: NSEC example

newsubdomain.tjeb.nl. 3600 IN NSEC ns.tjeb.nl. NS RRSIG NSEC
DNSSEC: NSEC gives new problem

If you have a record that shows you next name, you can 'walk' the zone, and get all its contents:

```bash
> ldns-walk example.nl.
exa...n_ example.nl. A NS SOA TXT AAAA RRSIG NSEC DNSKEY TYPE65534
*.example.nl. MX TXT RRSIG NSEC
*.domainkey.example.nl. TXT RRSIG NSEC
alpha.example.nl. TXT RRSIG NSEC
bravo.example.nl. TXT RRSIG NSEC
delta.example.nl. TXT RRSIG NSEC
echo.example.nl. TXT RRSIG NSEC
www.example.nl. A AAAA RRSIG NSEC
```
DNSSEC: NSEC3

NSEC3 (RFC 5155) was defined in 2008 to add two new things:

• Protection from zone walking

• Optional signing of delegation

NSEC3 records hash their names and their next names, resolvers hash the queried name and check if the hash falls between the hashed names.
DNSSEC: NSEC3

> dig +dnssec nosuchdomain.nsec3.tjeb.nl @ns.tjeb.nl

5S0HQ9TNQB8MSGGPOSJRBOSEVKRKSJUA1.nsec3.tjeb.nl. 18000 IN NSEC3 1 0 5 BEEF BQPT8AJ8K019893M2B4HKPU9Q14HV6S7 A NS SOA MX TXT RRSIG DNSKEY NSEC3PARAM TYPE65534
**DNSSEC: NSEC3**

<table>
<thead>
<tr>
<th>Hash Alg.</th>
<th>Flags</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salt Length</td>
<td>Salt</td>
<td></td>
</tr>
<tr>
<td>Hash Length</td>
<td>Next Hashed Owner Name</td>
<td></td>
</tr>
<tr>
<td>Type Bit Maps</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5S0HQ9TNQB8MSGGPOSJRBSEVKRKSJUA1.nsec3.tjeb.nl. 18000 IN NSEC3 1 0 5 BEEF BQPT8AJ8K019893M2B4HKPU9Q14HV6S7 A NS SOA MX TXT RRSIG DNSKEY NSEC3PARAM TYPE65534
<table>
<thead>
<tr>
<th>Hash Alg.</th>
<th>Flags</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salt Length</td>
<td>Salt</td>
<td></td>
</tr>
<tr>
<td>Hash Length</td>
<td>Next Hashed Owner Name</td>
<td></td>
</tr>
<tr>
<td>Type Bit Maps</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5S0HQ81NQB8MSGGPOSJRBSEVKRKSJUA1.nsec3.tjeb.nl. 18000 IN NSEC3 1 0 5 BEEF BQPT8AJ8K019893M2B4HKPU9Q14HV6S7 A NS SOA MX TXT RRSIG DNSKEY NSEC3PARAM TYPE65534
DNSSEC: NSEC3

<table>
<thead>
<tr>
<th>Hash Alg.</th>
<th>Flags</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salt Length</td>
<td>Salt</td>
<td>Next Hashed Owner Name</td>
</tr>
<tr>
<td>Hash Length</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5S0HQ9TNQ9B8MSGGPOSJRBSEVKRKJSJUA1.nsec3.tjeb.nl. 18000 IN NSEC3 1 0 5 BEEF BQPT8AJ8K019893M2B4HKPU9Q14HV6S7 A NS SOA MX TXT RRS1G DNSKEY NSEC3PARAM TYPE65534
# DNSSEC: NSEC3

<table>
<thead>
<tr>
<th>Hash Alg.</th>
<th>Flags</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salt Length</td>
<td>Salt</td>
<td>Next Hashed Owner Name</td>
</tr>
<tr>
<td>Hash Length</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
5S0HQ9TNQP5MSGGPOSJRBSEVKRKSJUA1.nsec3.tjeb.nl. 18000 IN NSEC3 1 0 5 BEEF BQPT8AJ8K019893M2B4HKPU9Q14HV6S7 A NS SOA MX TXT RRSIG DNSKEY NSEC3PARAM TYPE65534
```
### DNSSEC: NSEC3

<table>
<thead>
<tr>
<th>Hash Alg.</th>
<th>Flags</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salt Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hash Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next Hashed Owner Name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Salt**

```plaintext
5S0HQ9TNQB8MSCCPOSJRBSEVKRKSJUA1.nsec3.tjeb.nl. 18000 IN
NSEC3 1 0 5 BEEF BQPT8AJ8K019893M2B4HKPU9Q14HV6S7 A NS SOA MX
TXT RRSIG DNSKEY NSEC3PARAM TYPE65534
```
<table>
<thead>
<tr>
<th>Hash Alg.</th>
<th>Flags</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salt Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hash Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next Hashed Owner Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type Bit Maps</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DNSSEC: NSEC3

5S0HQ9TNQB8MSGGPOSJRBSFVKPSJSJUA1.nsec3.tjeb.nl. 18000 IN NSEC3 1 0 5 BEEF ROPT8AJ8K019893M2B4HKPU9Q14HV6S7 A NS SOA MX TXT RRSIG DNSKEY NSEC3PARAM TYPE65534
### DNSSEC: NSEC3

<table>
<thead>
<tr>
<th>Hash Alg.</th>
<th>Flags</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salt Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hash Length</td>
<td></td>
<td>Next Hashed Owner Name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Type Bit Maps**

```
5S0HQ9TNQB8MSGGPOSJRBSEVKRSJUA1.nsec3.tjeb.nl. 18000 IN
NSEC3 1 0 5 BEEF BQPT8AJ8K019893M2B4HKPU9Q14HV6S
TXT RRSIG DNSKEY NSEC3PARAM TYPE65534
```
DNSSEC: NSEC3

Define $H(x)$ to be the hash of $x$ using the Hash Algorithm selected by the NSEC3 RR, $k$ to be the number of Iterations, and $||$ to indicate concatenation. Then define:

$$IH(salt, x, 0) = H(x || salt)$$

and

$$IH(salt, x, k) = H(IH(salt, x, k-1) || salt), \text{ if } k > 0$$

Then the calculated hash of an owner name is

$$IH(salt, \text{ owner name, iterations})$$
DNSSEC: NSEC3

Apply that to example:
nosuchdomain.tjeb.nl with 5 iterations of SHA-1 with salt BEEF gives

aql5q4nkstddqs1e3fb5us22lu8bnfc9

We got:

5S0HQ9TNQB8MSGGPOSJRBSEVKRKSJUA1.nsec3.tjeb.nl. 18000 IN
NSEC3 1 0 5 BEEF BQPT8AJ8K019893M2B4HKPU9Q14HV6S7 A NS SOA MX
TXT RRSIG DNSKEY NSEC3PARAM TYPE65534

So we need to check whether

5S0HQ9TNQB8MSGGPOSJRBSEVKRKSJUA1 < aql5q4nkstddqs1e3fb5us22lu8bnfc9 < BQPT8AJ8K019893M2B4HKPU9Q14HV6S7

(case-insensitive)

**Conclusion: Existence denied.**
And that is DNSSEC!

- Pre-defined trust anchor(s)
- Signatures in RRSIGs
- Public keys in DNSKEY sets
- Key(s) of child in signed DS sets
- Authenticated denial of existence done through NSEC or NSEC3
Bonus content
(if there is time)
DNSSEC as base for other security protocols

• DANE (RFC 6698): connects X.509 (known mostly from https) to DNS(SEC)
  • adds to CA trust
  • allows for self-signed certificates

• In browser (with plugin; no native support)

• Mail Transfer Agents
  • native support in Postfix (2.11)
  • Experimental support in Exim (4.85)
DNSSEC as base for other security protocols

• SSHFP (RFC 4255): can be used as host key check in SSH

• Native support in OpenSSH
  • but disabled by default
DNSSEC in .nl: some numbers
DNSSEC Information (dutch)

- http://www.dnssec.nl
- http://www.dnsseccursus.nl
Questions?

Jelte Jansen
Research Engineer
Jelte.jansen@sidn.nl

@twitjeb

www.sidnlabs.nl