The DNS security mess

D. J. Bernstein

University of Illinois at Chicago
The Domain Name System

fsf.org wants to see http://www.redhat.com.

Browser at fsf.org

“The web server www.redhat.com has IP address 96.6.144.112.”

Administrator at redhat.com

Now fsf.org retrieves web page from IP address 96.6.144.112.
Same for Internet mail.

fsf.org has mail to deliver to someone@redhat.com.

Mail client at fsf.org

“The mail server for redhat.com has IP address 66.187.233.32.”

Administrator at redhat.com

Now fsf.org delivers mail to IP address 66.187.233.32.
Forging DNS packets

fsf.org has mail to deliver to someone@redhat.com.

Mail client at fsf.org

“The mail server for redhat.com has IP address 157.22.245.20.”

Attacker anywhere on network

Now fsf.org delivers mail to IP address 157.22.245.20, actually the attacker’s machine.
Actually: Client sends query; attacker has to repeat some bits from the query.
Actually: Client sends query; attacker has to repeat some bits from the query.

Network probably has at least one attacker-controlled machine. That machine sniffs network, trivially forges DNS packets.
Actually: Client sends query; attacker has to repeat some bits from the query.

Network probably has at least one attacker-controlled machine. That machine sniffs network, trivially forges DNS packets.

“No sniffers on my network!”

... so a blind attacker guesses the bits to repeat, eventually gets lucky.
After analysis, optimization: blind forgery is about as easy as downloading a movie.
Amazing news

Tuesday 2 June 2009:

“.ORG becomes the first open TLD to sign their zone with DNSSEC . . . Today we reached a significant milestone in our effort to bolster online security for the .ORG community. We are the first open generic Top-Level Domain to successfully sign our zone with Domain Name Security Extensions (DNSSEC). To date, the .ORG zone is the largest domain registry to implement this needed security measure.”
“What does it mean that the .ORG Zone is ‘signed’?
Signing our zone is the first part of our DNSSEC test phase. We are now cryptographically signing the authoritative data within the .ORG zone file. This process adds new records to the zone, which allows verification of the origin authenticity and integrity of data.”
Cryptography! Authority!
Verification! Authenticity!
Integrity! Sounds great!
Cryptography! Authority! Verification! Authenticity! Integrity! Sounds great!

Now I simply configure the new .org public key into my DNS software. Because the .org servers are signing with DNSSEC, it is no longer possible for attackers to forge data from those servers!
Cryptography! Authority! Verification! Authenticity! Integrity! Sounds great!

Now I simply configure the new .org public key into my DNS software. Because the .org servers are signing with DNSSEC, it is no longer possible for attackers to forge data from those servers!

... or is it?
Let's find a .org server:

$ dig +short ns org
d0.org.afilias-nst.org.
b0.org.afilias-nst.org.
a0.org.afilias-nst.info.
c0.org.afilias-nst.info.
b2.org.afilias-nst.org.
a2.org.afilias-nst.info.

$ dig +short \b0.org.afilias-nst.org
199.19.54.1
Look up one of my domains:

$ dig \n    www.mwisc.org @199.19.54.1

Everything looks normal:

    ;; AUTHORITY SECTION:
    mwisc.org. 86400
        IN NS d.ns.mwisc.org.
    mwisc.org. 86400
        IN NS f.ns.mwisc.org.
    ;; ADDITIONAL SECTION:
    d.ns.mwisc.org. 86400
        IN A 131.193.36.21
    f.ns.mwisc.org. 86400
        IN A 131.193.36.24
Now ask for signatures:

$ dig +dnssec \n
www.mwisc.org @199.19.54.1

Same answer as before, *plus* four new records:

h9p7u7tr2u91d0v0ljs9l1gid
np90u3h.org. 86400 IN TYP E50 \#39 0101000104D399EA
AB148A77C7ACEFCBC55446032
B2D961CC5EB6821 EF2600072
2000000000290

h9p7u7tr2u91d0v0ljs9l1gid
np90u3h.org. 86400 IN RRS
IG TYPE50 7 2 86400 20090
70721303120090623203031 3
7493 org. 1kzaiDXNZExggNF
W3PFLNRNP8WPTECXUWH0tktDjX
thkE60pm6LoTOrRq TgfWK7NS
4GjN98rwqKH7iCfRr09CJ1BzC
XIdtWn5W0TOmtgwp413YF20 r
006RmDbXzbPcA5NXTsMk6b7fL
AHzRYEPBdBt1x3XJAZAPkrBPN
7dx2W w+g=
1b8fe79t5m6vkn6eo6s0n3gb7
mls aicq.org. 86400 IN TY
PE50 \\#38 0101000104D399E
AAB140ADEA6FED9985FAABFED
These .org signatures are 1024-bit RSA signatures.

2003: Shamir–Tromer et al. concluded that 1024-bit RSA was already breakable by large companies and botnets. $10 million: 1 key/year. $120 million: 1 key/month.

2003: RSA Laboratories recommended a transition to 2048-bit keys “over the remainder of this decade.” 2007: NIST made the same recommendation.
Will be a few years before 1024-bit RSA is breakable by academics in small labs. They’re finishing RSA-768 now.
Will be a few years before 1024-bit RSA is breakable by academics in small labs. They’re finishing RSA-768 now.

“RSA-1024: still secure against honest attackers.”
Will be a few years before 1024-bit RSA is breakable by academics in small labs. They’re finishing RSA-768 now.

“RSA-1024: still secure against honest attackers.”

What about serious attackers using many more computers? e.g. botnet operators?

I say:
Using RSA-1024 is irresponsible.
But that’s not the biggest problem with the DNSSEC signatures in .org.
But that’s not the biggest problem with the DNSSEC signatures in .org.

Suppose an attacker forges a DNS packet from .org, including exactly the same DNSSEC signatures but changing the NS+A records to point to the attacker’s servers.
But that’s not the biggest problem with the DNSSEC signatures in .org.

Suppose an attacker forges a DNS packet from .org, including exactly the same DNSSEC signatures but changing the NS+A records to point to the attacker’s servers.

Fact: DNSSEC “verification” won’t notice the change. The signatures say nothing about the NS+A records. The forgery will be accepted.
What did .org sign?

The signature for mwisc.org says “.org might have data with hashes between 1b39ggevfp3b72r9r901o1osqddn4ben and 1bfadvmpj1fqlfvdv8eksiokfheo7km9 but has not signed any of it.”

mwisc.org has a hash in that range.

.org now has thousands of these useless signatures. This is .org “implementing” a “needed security measure.”
The Internet has about 78000000 *.com names.
The Internet has about 78000000 * .com names.

Surveys by DNSSEC developers, last updated 2009.06.24, have found 241 * .com names with DNSSEC signatures. 116 on 2008.08.20; 241 > 116.
The Internet has about 78000000 *.com names.

Surveys by DNSSEC developers, last updated 2009.06.24, have found 241 *.com names with DNSSEC signatures. 116 on 2008.08.20; 241 > 116.

“DNSSEC:
Fifteen years of development. Millions of dollars of U.S. government grants (DISA, NSF, DHS, etc.). Hundreds of users.”
What went wrong?

Some of the Internet’s DNS servers are extremely busy: e.g., the root servers, the .com servers, the google.com servers.

Can they afford crypto? Hmmm. DNSSEC tries to minimize server-side costs by precomputing signatures of DNS records.

“No per-query crypto.”

Signature is computed once; saved; sent to many clients. Hopefully the server can afford to sign each DNS record once.
Clients don’t share the work of verifying a signature.

DNSSEC tries to reduce client-side costs through choice of crypto primitive.

Many DNSSEC crypto options: 640-bit RSA, original specs; 768-bit RSA, many docs; 1024-bit RSA, current RFCs (for “leaf nodes in the DNS”); DSA, “10 to 40 times as slow for verification” but faster for signatures.
DNSSEC made breakable choices such as 640-bit RSA for no reason other than fear of server overload.

DNSSEC needed more options to survive the inevitable breaks. Profusion of options made DNSSEC crypto complicated, hard to review for bugs.

2009: Emergency BIND upgrade. Minor software bug meant that DNSSEC DSA signatures had always been trivial to forge.
My main point today: DNSSEC’s fear of overload forced DNSSEC down a path of unreliability, insecurity, and unusability. This is why DNSSEC has been a failure.
My main point today: DNSSEC’s fear of overload forced DNSSEC down a path of unreliability, insecurity, and unusability. This is why DNSSEC has been a failure.

My main point Saturday: Per-query crypto leads to a much simpler protocol with much higher reliability, much higher security, and much higher usability.
My main point today: DNSSEC’s fear of overload forced DNSSEC down a path of unreliability, insecurity, and unusability. This is why DNSSEC has been a failure.

My main point Saturday: Per-query crypto leads to a much simpler protocol with much higher reliability, much higher security, and much higher usability. Can still handle the loads using state-of-the-art crypto.
DNS architecture

Browser pulls data from DNS cache at fsf.org:

Administrator at redhat.com

“'The web server www.redhat.com has IP address 96.6.144.112.”'

Cache pulls data from administrator if it doesn’t already have the data.
Administrator pushes data through local database into .redhat.com DNS server:

- **Browser** at fsf.org
- **DNS cache**
- **.redhat.com DNS server**
- **.redhat.com database**
- **Administrator** at redhat.com

"The web server www.redhat.com has IP address 96.6.144.112."

at redhat.com
DNS cache learns location of .redhat.com DNS server from .com DNS server:

at fsf.org

"The DNS server for .redhat.com is ns2 with IP address 209.132.183.2."

at redhat.com
DNS server software listed in Wikipedia: BIND, Microsoft DNS, djbdns, Dnsmasq, Simple DNS Plus, NSD, PowerDNS, MaraDNS, ANS, Posadis, Secure64 DNS.

DNS database-management tools listed by 2008 Salomon: BPP, DNS Boss, DNStool, gencidrzone, h2n, makezones, NSC, nsupdate, SENDS, updatehosts, Utah Tools, webdns, zsu. Plus hundreds of homegrown tools written by DNS registrars etc.
DNSSEC changes everything

DNSSEC requires new code in every DNS-management tool.

Whenever a tool adds or changes a DNS record, also has to precompute and store a DNSSEC signature for the new record.

Often considerable effort for the tool programmers.

Example: Signing 2GB database can produce 10GB database (2005 NIST study).

Tool reading database into RAM probably has to be reengineered.
Administrator also has to send public key to .com.

The .com server and database software and web interface need to be updated to accept these public keys and to sign everything.

DNS cache needs new software to fetch keys, fetch signatures, and verify signatures.
Tons of pain for implementors. Still many gaping holes after fifteen years of work.

Example: .org has no way to receive DNSSEC public keys from *.org users (via, e.g., joker.com).

Example: .org software can’t manage signatures for millions of .org records. Much too slow, much too big.
Replay attacks

Attacker inspects DNSSEC signatures from redhat.com.

redhat.com changes location, acquires new IP addresses, changes DNS records.
Replay attacks

Attacker inspects DNSSEC signatures from redhat.com.

redhat.com changes location, acquires new IP addresses, changes DNS records.

Attacker buys the old addresses, forges DNS responses with the old DNS records and the old signatures. Passes signature verification. Successfully steals mail!
DNSSEC has a partial defense. Signature has an expiration date, normally signing date $+$ 30 days.

Not very good security: replay attack continues to work for up to 30 days.
DNSSEC has a partial defense. Signature has an expiration date, normally signing date + 30 days.

Not very good security: replay attack continues to work for up to 30 days.

Plus extra code: re-sign before old signatures expire.

Any mistakes destroy your domain ("DNSSEC suicide"). 2009: This happened to all ISC DLV DNSSEC users. UCLA admin: “The solution in all cases was to disable DNSSEC validation.”
Another type of replay:

www.redhat.com is actually published by Akamai.


Attacker replays same response to user in Berlin. User expected fast, reliable connection to a nearby server; receives slow, unreliable connection across the ocean.

Expiration dates don’t help.
Query espionage

RFC 4033: “Due to a deliberate design choice, DNSSEC does not provide confidentiality.”
Query espionage

RFC 4033: “Due to a deliberate design choice, DNSSEC does not provide confidentiality.”

http://dnscurve.org/espionage.html has a simple dnsoutloud script combining tcpdump, text2wave, and play.
Query espionage

RFC 4033: “Due to a deliberate design choice, DNSSEC does not provide confidentiality.”

http://dnscurve.org/espionage.html has a simple dnsoutloud script combining tcpdump, text2wave, and play.

Would any DNSSEC proponent like to run dnsoutloud in a busy Internet cafe with the volume turned up?
Database espionage

Privacy-violating speed:
\[ \approx 2^{29} \text{ noisy guesses/day:} \]
DNS today.

\[ > 2^{40} \text{ silent guesses/day,} \]
many more with large botnet:
Current DNSSEC (NSEC3).

Instantaneous: Old DNSSEC,
or DNS with public AXFR.
DDoS amplification

dig +bufsize=4096 +dnssec any se @a.ns.se

To Sweden: 31-byte UDP packet.
From Sweden:
3974-byte UDP packet.

dig +bufsize=4096 +dnssec any br @a.dns.br

To Brazil: 31-byte UDP packet.
From Brazil:
1621-byte UDP packet.
This is crazy!

Imagine an “HTTPSEC” that works like DNSSEC.
This is crazy!

Imagine an “HTTPSEC” that works like DNSSEC.

Store a signature next to every web page. Recompute and store signature for every minor wiki edit, and again every 30 days. Any failure: HTTPSEC suicide. Dynamic content? Give up.
This is crazy!

Imagine an “HTTPSEC” that works like DNSSEC.

Store a signature next to every web page. Recompute and store signature for every minor wiki edit, and again every 30 days. Any failure: HTTPSEC suicide.

Dynamic content? Give up.

Replay attacks work for 30 days. Filename guessing is much faster. Nothing is encrypted. Denial of service is trivial.
Security review

Confidentiality: Bad today. With DNSSEC, even worse.
Security review

Confidentiality: Bad today.
With DNSSEC, even worse.

Integrity: Bad today.
With DNSSEC, better,
but (1) still not great
and (2) only after incredible
amounts of implementor pain.
Security review

Confidentiality: Bad today.
With DNSSEC, even worse.

Integrity: Bad today.
With DNSSEC, better, but (1) still not great and (2) only after incredible amounts of implementor pain.

Availability: Bad today.
With DNSSEC, much worse.