Zone State Revocation (ZSR) for DNSSEC

Eric Osterweil (UCLA)
Vasileios Pappas (IBM Research)
Dan Massey (Colorado State Univ.)
Lixia Zhang (UCLA)
Outline

- What are DNS & DNSSEC
- Key Revocation Problem
- Threat Model
- ZSR Approach
- ZSR design
- Conclusion
DNS

- Global hierarchical namespaces (zones)
  - ucla.edu is a zone
  - www is a record
- Largest globally distributed database
  - Too large for standard management approaches
- Zones use **SOA serial numbers** to indicate changes
- Nameservers serve zone data
Caching in DNS

- 10’s of millions of zones
- Caching needed to scale
- *Caching resolvers* are between clients and nameservers
- Caching resolvers walk DNS tree, not client machines
  - 3 types of machines
Why DNSSEC?

- Caching is vulnerable
- *Eve* can insert her own answer if she responds first
- DNS has no way to know what data is authentic
- Clients will get values from their cache and believe them
DNSSEC

- DNSSEC is a PKI
  - Public/private keys
- Parents vouch for children
- DNSKEY records
  - Public keys
- Uses pre-generated signatures
  - No “online signing”
  - Signatures valid for definitive period (inception to expiration)
Problem

- DNS is one of the largest-scale systems
- The zones in DNS are all independently run
- This mandates a very simple protocol
- Coordination is very difficult
- DNS can tolerate slight misconfigurations and slow coordination
- DNSSEC has stricter requirements
Problem(2)

- Normally, to change keys one must transition
  - Due to caching, zones must serve old and new keys
- What about an unplanned emergency?
  - i.e. a private key has been compromised!
- Need a way to flush millions of remote caches
If Eve can create records and insert them, caching resolvers will use valid keys to “verify” her records.
Attack Vectors

- **Spoofing attack**
  - *Eve* replies before the real nameserver does

- **Poisoning attack**
  - *Eve* tricks caches into taking data ahead of time

- **Man in the Middle (1)**
  - *Eve* intercepts traffic to a \( n \) of \( m \) nameservers

- **Man in the Middle (2)**
  - *Eve* intercepts traffic to \( m \) of \( m \) nameservers
ZSR’s Approach

- Signature lifetimes are temporal
- But emergencies are unplanned
  - Orthogonal to temporal lifetimes
- In ZSR, zones can override lifetimes
- ZSR can notify millions of caches to flush compromised records
ZSR Requirements

- Designed to be incrementally deployable
- ZSR must be able to perform 3 operations to be robust against Eve:
  1. Prove a key is compromised
  2. Revoke data
  3. Notify resolvers of revocations
ZSR’s Mechanism

- ZSR augments signatures with **lease** periods
  - Lease: uses zones’ *state* (serial number) + lease period
- Signatures are valid while zone’s serial number is less than a lease
- Leases are broken by increasing serial numbers
- ZSR introduces a highly-scalable cache update protocol into modern DNS
Proving Key Compromise

- After suspecting a zone state change, key revocation must be proven

- REVKEY is a self-certified revocation certificate of a DNSKEY
Revoking Data

- RRSIGs include current inception/expiration dates + zone lease
  - Lease is serial # that invalidates sig
  - Lease is current serial number + $L$
  - $2^{31}$ based on evaluation

<table>
<thead>
<tr>
<th>&lt;Various Data&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception Time: 20070101000000</td>
</tr>
<tr>
<td>Expiration Time: 20070108000000</td>
</tr>
<tr>
<td>Signature Body</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&lt;Various Data&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception Time: 20070101000000</td>
</tr>
<tr>
<td>Expiration Time: 20070108000000</td>
</tr>
<tr>
<td>Lease: 2007010101 + $2^{31}$</td>
</tr>
<tr>
<td>Signature Body</td>
</tr>
</tbody>
</table>
Example

Just changing the zone’s state breaks leases, automatically
Notifying Resolvers

- After data is cached zones may need to revoke
- Zones notify by embedding the serial number in every DNS response
  - <Zone name, Serial #, timestamp, signature>
- Once a zone has broken leases, all cached records are flushed
Example

- Any query to the zone allows caches to flush revoked signatures
- Even for different records
Evaluation

- From 2.5 million zones 50,000 were randomly chosen and monitored
- DNS data observed during May, 2006
- Query patterns taken from North American University
  - 821 unique stub-resolvers
  - 117,540 DNS names
  - 55,632 unique zones
Feasibility

- DNSSEC data shows vulnerability period is a significant concern
- Our evaluation shows overloading the SOA serial number is unlikely to impair its current usage
- Sample usage pattern shows ZSR can significantly reduce zone vulnerability
Conclusion

- Emergencies will happen
- Lack of a revocation mechanism is a serious liability in DNSSEC
- ZSR is a scalable protocol to flush revoked keys at Internet scales
- ZSR uses existing mechanisms and can be incrementally deployed
Thank You

Questions?
Backup
Spoofing

- *Eve* can spoof data, but she needs:
  - To get the DNS query sequence #
  - Spoof src/dst IP/port info
  - Be faster than the real nameserver
  - And must (likely) be on the local subnet
Poisoning

- *Eve* can poison caches
- If a cache asks *Eve’s* zone for anything, it may store everything she responds with
- She can add [www.target.sec](http://www.target.sec) data with her own zone’s data
Man in the Middle (1)

- If *Eve* can intercept all traffic to some of target.sec’s nameservers
- She can snoop, reply to, drop, etc.
- However, this will not be true for traffic to other nameservers
Man in the Middle (2)

- *Eve* has the same capacity as in the Man in the Middle (1) vector, but for all nameservers.
- This does not imply that *Eve* can intercept all Internet traffic for C.
DNSSEC Signature Lifetimes

- DS records (secure delegation records)
  - 3 - 30 signature lifetimes
  - Average 17.03 days
- DNSKEY signature lifetimes
  - 3 - 30 days
  - Average 26.45 days
- Without ZSR, zones must (potentially) wait this long for caching effects
SOA Serial Number

- 80% of monitored zones did not change their serial number
- Of remaining 20%, the period was 13.5 hours
- 99.2% seem to mishandle serial numbers
- Incorrect padding leads to ± 2.2^8 oscillations
- ZSR can choose 2^{31} as its lease-breaking value and stand out
Zone Access Patterns

- Window of vulnerability defined by query patterns
- Without ZSR, vulnerability independent of attack
- ZSR reduces window
- But unpopular zones skew results
  - Only queried once during sample period
Query-Based Performance

- Using query rates

- Zone is only vulnerable if users are querying for its data

- This figure shows the number of queries vulnerable to each type of attack