DNSSEC + DANE

Andres Marin
Colab. with Prof. Strufe
Securing DNS Cryptographically

- Securing DNS has different goals:
  - DNS transaction security
    - Peer/message authentication
  - DNS data security
    - Data origin authentication
    - Authenticated denial of existence
Transaction Authentication (TSIG)

- **Idea:**
  - Use signatures to secure data at zone transfer \texttt{master} \xrightarrow{\text{TSIG}} \texttt{slave}
  - Pre shared symmetric key at each entity
  - MD5 Hash used as signature

- **TSIG Resource Record:**
  
  \[
  \text{(Name, Type ("TSIG"), Class ("ANY"), TTL("0"), Length, Data(<signature>))}
  \]

- Possibility to authenticate, but very complex to administrate in large domains (manual pre-sharing of keys)
  - amount of keys required:

- Main application areas:
  - Secure communication between stub resolvers and security aware caching servers (?)
  - Zone transfers (master \xrightarrow{\text{TSIG}} \texttt{slave})
  - Combined with nsupdate in data centers, to update stale information in caches

[Vixie et. al: "RFC 2845: Secret Key Transaction Authentication for DNS"]
DNS Security (DNSSEC) – Objectives

- DNS security **objectives**:  
  - End-to-end zone data *origin authentication* and integrity  
  - *Detection* of data corruption and spoofing

- DNSSEC **does not** (want to) provide:  
  - DoS-Protection *(in fact, it facilitates DoS Attacks on DNS servers)*  
  - Data delivery guarantees (availability)  
  - Guarantee for correctness of data (only that it has been signed by some authoritative entity)

[Eastlake: „RFC 2535: Domain Name System Security Extensions“ (obsolete)]  
[Arends et. al: „RFC 4033: DNS Security Introduction and Requirements“]  
[RFCs:4033,4034,4035,4310,4641]
DNSSEC

- Usage of public key cryptography to allow for data origin authentication on a world wide scale

- **RRSets** (groups of RRs) signed with private key of authoritative entities
- **Public keys** (DNSKEYs) published using DNS

- Distinguish zone signing key (ZSK) and key signing key (KSK) (SEP-Secure Entry Point)
- Child zone keys are authenticated by parents and hence anchored trust chains established
- Only root zone key signing key (KSK) needed (manual distribution) to create complete trust hierarchy (in theory)
- How/Why shall we trust root zone key?
- Until then: islands of trust with manually shared anchor keys
- No key revocation 🥷 DNSSEC keys should have short expiration date (quick rollover)
DNSSEC – Targeted Threats

- Zone File
- Dynamic updates
- Auth Server
- Slaves
- Caching Server
- Resolver
- Cache Pollution
- Cache Impersonation
- Altered Zone Data
DNSSEC – Means of Securing RRSets

- **Goal:** authenticity and integrity of Resource Record Sets
- **Means:**
  - Public Key Cryptography (with Trust Chains)
  - Security integrated in DNS (new RRs)

- **New Resource Record Types:**
  - **RRSig:** signatures of RRs
  - **DNSKEY:** public keys
  - **DS:** for trust chaining (trust anchor signs key of child zone)
  - **NSEC:** pointer to next secure name in canonical order (authenticated denial for requested zone)
DNSSEC – New Resource Records: RRSIG

- Resource Record for transmission of **signatures**

- **RRSIG:**
  - **Name** – name of the signed RR
  - **Type** – RRSIG (46)
  - **Algorithm** – MD5(1), Diffie-Hellman(2), DSA (3)
  - **Labels** – number of labels in original RR (wildcard indication)
  - **TTL** – TTL at time of signature inception
  - **Signature Expiration** – End of validity period of signature
  - **Signature Inception** – Beginning of validity period of signature
  - **Key Tag** – ID of used key if signer owns multiple keys
  - **Signer’s Name** – Name of the signer
  - **Signature** – Actual Signature
RRSIG signature

- \( \text{signature} = \text{sign}(\text{RRSIG\_RDATA} | \text{RR}(1) | \text{RR}(2)... ) \)

- \( \text{RRSIG\_RDATA} = \text{all the fields but the signature} \)
  - Name | type | alg | labels | TTL | sig\_exp | sig\_inc | key tag | signer’s name

- \( \text{RR(i)} = \text{owner} | \text{type} | \text{class} | \text{TTL} | \text{RDATA length} | \text{RDATA} \)
DNSSEC – New Resource Records: DNSKEY

- Resource Record containing *public keys* for distribution
- **DNSKEY**: (Label, Class, Type, Flags, Protocol, Algorithm, Key)

  - **Label** – Name of key owner
  - **Class** – Always: IN (3)
  - **Type** – DNSKEY
  - **Flags** – key types: Key Signing Key (257) or Zone Signing Key (256)
  - **Protocol** – Always DNSSEC (3)
  - **Key** – Actual key
DNSSEC – New RRs: Delegation Signer (DS)

- DS contains *hash-value of DNSKEY* of the name server of a sub zone

- Together with NS Resource Record, DS is used for trust chaining

**DS**: (Name, Type, Key Tag, Algorithm, Digest Type, Digest)

- **Name** – Name of the chained sub zone
- **Type** – DS
- **Key Tag** – Identification of the hashed key
- **Algorithm** – RSA/MD5(1), Diffie-Hellman(2), DSA(3) (of referred DNSKEY)
- **Digest Type** – SHA-1(1), SHA-256(2)
- **Digest** – Actual value of hashed DNSKEY
DNS – Authority Delegation and Trust Chaining

- Data can be trusted if signed by a ZSK
- ZSK can be trusted if signed by a KSK
- KSK can be trusted if pointed to by trusted DS
- DS record can be trusted if
  - Signed by parents ZSK
  - Signed by locally configured trusted key
DNS – Authority Delegation and Trust Chaining (Example)

Trusted Key (locally configured)

Child Zone

Parent Zone

ns NS child
ns.child DS

<KSK-id>

RRSIG DS

(...) parent.

child

@NS

<DNSKEY> (...)<KSK-id>

RRSIG dnskey (...)<KSK-id>

parent.

<KSK-id>

RRSIG dnskey (...)<ZSK-id>

child.parent.

ns A 10.5.1.2

RRSIG A (...)<ZSK-id>

child.parent.

www A 10.5.1.3

RRSIG A (...)<ZSK-id>

child.parent.
DNSSEC – New Resource Records: NSEC

- Next Secure (NSEC) gives information about the next zone / sub domain in canonical order (last entry points to first entry for the construction of a closed ring)
- Gives the ability to prove the non-existence of a DNS entry: Authenticated Denial

**NSEC** (Name, Type, Next Domain)
- Name – Name of the signed RR
- Type – NSEC (47)
- Next Domain – Name of the next domain in alphabetical order

- Allows adversary to crawl entire name zone ("zone walking")
DNSSEC – New RRs: NSEC3 (1)

- Successor to NSEC: NSEC3 and NSEC3PARAM
- Uses hashed domain names to make zone walking more difficult
- Hashing based on salt and multiple iterations to make dictionary attacks more difficult

NSEC3

- **Name** – Name of the signed RR
- **Type** – NSEC3 (50)
- **Hash Algorithm** – SHA-1 (1)
- **Flags** – To Opt-Out unsigned names
- **Iterations** – Number of iterations of Hash Algorithm
- **Salt Length** – Length of the salt value
- **Salt** – Actual salt value
- **Hash Length** – Output length of hash function
- **Next Hashed Owner Name** – Next Hash of domain name in alphabetical order
DNSSEC – New RRs: NSEC3 (2)

- Potential advantage: Salting and hashing does not allow for easily deducing hostnames from zone walks

- Problem:
  - Hostnames usually have very low entropy (to remember them)
  - Easy dictionary attacks - despite the usage of salts & iterations
  - But not used heavily anyways:
    - .: Uses NSEC
    - .com: No salt, No iterations
    - .de: Static salt BA5EBA11, 15 Iterations
DNSSEC: NSEC5 / Record Type Denial

- Provide authenticated denial of existence without leaking names requires online signing.
- Providers do not want to trust the DNS servers with keys...

- Cloudflare Record Type Denial
  - Send positive response but deny requested record type
DNSSEC Issues

Pro’s:
- DNSSEC allows to prevent unauthorized/spoofed DNS records

Con’s:
- Added complexity (signing, checking, key distribution) eases DoS attacks on DNS servers
- Zones need to be signed completely (performance challenge for large companies or registries)
- Authenticated denial with NSEC gives the possibility to “walk” the chain of NSEC and to gain knowledge on the full zone content (all zones/ sub domains) in $O(N)$ $\Rightarrow$ NSEC3, ...
- Distribution of anchor keys still a manual task (allows for human error, social engineering)

Deployment:
- http://www.secsfinder.net/islands.html
- https://blog.apnic.net/2017/12/06/dnssec-deployment-remains-low/
- https://stats.labs.apnic.net/dnssec/XA
TLS authentication

- Many applications use the certificate-based authentication in Transport Layer Security (TLS)
  - allow clients to authenticate server.
  - allow server and client to agree upon acceptable ciphersuite

- Typically, authentication is based on PKIX certificate chains rooted in certificate authorities (CAs)

- What are the challenges in PKIX?
  - trust roots are configured out of band (depend on vendors)
  - DoS attacks to block certificate status verification
  - trusted CAs may be attacked and misbehave
TLS authentication

- Authentication is often based on PKIX certificate chains rooted in certificate authorities (CAs)

**PKIX challenges**
- trust roots are configured out of band (depend on vendors)
- DoS attacks to block certificate status verification
- verification path building
- trusted CAs may be attacked and misbehave
- ...
Certificates (privkey)stolen

- Adobe, Microsoft developer, ...

Adobe Secure Software Engineering Team (ASSET) Blog / Inappropriate Use of Adobe Code Signing Certificate

We recently received two malicious utilities that appeared to be digitally signed using a valid Adobe code signing certificate. The discovery of these utilities was isolated to a single source. As soon as we verified the signatures, we immediately decommissioned the existing Adobe code signing infrastructure and initiated a forensics investigation to determine how these signatures were created. We have identified a compromised build server with access to the Adobe code signing infrastructure. We are proceeding with plans to revoke the certificate and publish updates for existing Adobe software signed using the impacted certificate. This only affects the Adobe software signed with the impacted certificate that runs on the Windows platform and three Adobe AIR applications* that run on both Windows and Macintosh. The revocation does not impact any other Adobe software for Macintosh or other platforms.
CAs attacked

- Comodo: fraud certs to mail.google.com, login.skype.com, addons.mozilla.org

<table>
<thead>
<tr>
<th>Date</th>
<th>Incident</th>
<th>Target</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 25, 2011</td>
<td>Stuxnet driver is discovered to be signed with a valid certificate belonging to Realtek Semiconductor Corps. On July 16, 2011, Verisign revokes Realtek Semiconductor Corps certificate.</td>
<td>Stuxnet</td>
<td>Stolen Certificate</td>
</tr>
<tr>
<td>Mar 24, 2011</td>
<td>As a revenge for Stuxnet, an Iranian hacker forges fake certificates for google email services.</td>
<td>Comodo</td>
<td>Vulnerability in Enrollment</td>
</tr>
<tr>
<td>Aug 29, 2011</td>
<td>A user named finds a certificate warning about a revoked SSL Certificate Google services. The certificate was issued on July 10th by Dutch DigiNotar. The fake certificate was forged by Comodo Hacker, and revoked immediately.</td>
<td>DigiNotar</td>
<td>Non Disclosed Web</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Incident</th>
<th>Target</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 6, 2011</td>
<td>The real extent of the DigiNotar breach becomes clear: 531 bogus certificates issued including Google, CIA, Mossad, Tor, Comodo Hacker also claims to own four more CAs, among which GlobalSign which precariously suspends issuance of certificates. Another one StartCom was able to avoid the hack since its CEO was sitting in front of HSM, although the attacker claims to own emails, DB and Customer data.</td>
<td>DigiNotar, Globalsign and StartCom</td>
<td>N/A</td>
</tr>
</tbody>
</table>
TLS authentication and DNSSEC

▪ Remember DNSSEC:
  ○ links a key to a domain name
  ○ allows online access to signed keys
  ○ keys associated to a domain must be signed by a key in the parent domain

▪ DNS-Based Authentication of Named Entities (DANE) supports TLS using DNSSEC
  ○ DANE provides information about the cryptographic credentials associated with a domain
  ○ Clients can increase the level of assurance they receive from the TLS handshake process
  ○ Not only https but any application
  ○ TLS server name
  ○ verification path easier to build
  ○ hierarchical control
DANE certificate usages

- Let Alice be the
  - operator of a TLS-protected application service on the host h.alice.com, and
  - the administrator of the corresponding DNS zone.

- Let Bob be a client connecting to h.alice.com.

- Let Charlie be a well-known CA that issues certificates with domain names as identifiers.

- Given those actors, let’s review DANE certificate usages:
  - CA constraints (PKIX-TA)
  - Service certificate constraints (PKIX-EE)
  - Trust anchor (DANE-TA)
  - Domain-issued certificates (DANE-EE)
PKIX-TA

root zone
  com
    alice.com

h.alice.com
x.x.x.x

RRSIG
DANE reg:
CA constraint:
Charly_CA

TLS handshake:
cert(h.alice.com)
cert(Charly_CA)

x.x.x.x

cert_req
h.alice.com

h.alice.com?
CA constraints: PKIX-TA

- Alice has a cert issued by Charly to h.alice.com
  - Alice fears that an attacker gets a cert issued by another well known CA to h.alice.com
    - Clients would accept it since it is valid
  - Alice wants all the clients to accept only Charly’s issued certs for h.alice.com
  - In the TLS handshake
    - the server includes Charlie's cert in the server Certificate message's certificate_list
  - Charly should also check the CA Constraint in Alice domain prior to issue the cert
PKIX-EE

root zone
com
alice.com
h.alice.com

h.alice.com?

x.x.x.x
RRSIG
DANE reg:
PKIX-EE
cert(h.alice.com)

tls handshake:
cert(h.alice.com)

h.alice.com
x.x.x.x
cert_req
h.alice.com
Service Certificate Constraints: PKIX-EE

- Alice has a cert issued by Charly to h.alice.com
  - Alice fears that an attacker gets another cert issued by Charly to h.alice.com
    - Clients would accept it since it is valid
  - Alice wants all the clients to accept only the present cert she had been issued by Charly
    - in the TLS handshake
      - the server includes the cert issued by Charlie as the first in the certificate_list

- Similar as in CA Constraints, a successful attacker would need to
  - take control of DNS zone
  - tamper with the dnssec records
  - have a valid cert issued by Charly
  - modify the DANE records accordingly
DANE-TA

root zone

com

alice.com

cert_issue
h.alice.com

x.x.x.x
RRSIG
DANE reg:
CA constraint:
Alice_CA

TLS handshake:
cert(h.alice.com)
cert(Alice_CA)

h.alice.com
Trust Anchor Assertion (DANE-TA) and Domain-Issued Certificates (DANE-EE)

- Alice runs her own CA to issue certificates to applications and hosts in her domain
  - Alice wants all the clients to accept only the certificates issued by her in the TLS handshake
    - the server includes Alice’s self-signed CA cert as the first in the certificate_list
  - Besides adding the self-signed cert as a trust anchor, Alice can add it as CA Constraints
    - This way clients will only accept Alice issued certificates for the domain

- Such a trust anchor can be also used in the previous scenarios as a prerequisite for Charly to issue a cert to h.alice.com
  - The CA can check if the cryptographic key linked to the domain has been used to sign the certificate request or can be used to validate the signing key.

- How this relates to the use case where Alice wants to use a little known certificate authority?
Delegated Services

- Suppose Oscar operates h.alice.com on behalf of Alice.
- Oscar has control over certificates to present in TLS handshakes for h.alice.com.

a. Alice has the A/AAAA records in her DNS and can sign them along with the DANE record, Oscar and Alice need tight coordination if the addresses and/or the certificates change.

a. Alice delegates a sub-domain name to Oscar, and has no control over the A/AAAA, DANE, or any other pieces under Oscar's control.

a. Alice can put DANE records into her DNS server but delegate the address records to Oscar's DNS server.
   - Alice controls the usage of certificates
   - Oscar is free to move the servers around as needed
   - Coordination only needed when the certificates change (Always?)
TLSA record

- DANE performs its functions defining a new DNSSEC Resource Record named the TLSA
- The TLSA record gives information about a host in the domain:
  a. the certificate usage: PKIX-TA (0), PKIX-EE(1), DANE-TA(2), DANE-EE(3)
  b. the selector: the full cert (0) or just the public key info (1)
  c. the matching type: Full (0), SHA2-256 (1), SHA2-512 (2)
  d. data: full value or digest of the certificate or subject public key as determined by the matching type and selector

Example of PKIX-TA CERT SHA2-512:

_443._tcp.h.alice.com. TLSA 0 0 2 {blob}
Other proposals for DANE

- DANE can also be used for other purposes:
  a. Distributing OpenPGP public keys [RFC 7929]
  b. Associate Certificates with Domain Names for S/MIME [RFC 8162]
  c. SMTP transport security [RFC 7672]

- Other resources:
  a. https://weberblog.net/how-to-use-danetlsa/
  b. https://weberblog.net/pgp-key-distribution-via-dnssec-openpgpkey/