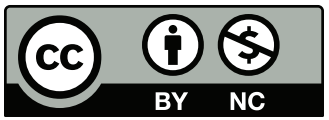
DNS Security



What is DNS Security?

- Bad guys play games with the DNS
- Traffic is sent to the enemy's machine instead
  - Enemy can see the traffic.
  - Enemy can easily modify the traffic.
  - Enemy can drop the traffic.
- Cryptography can mitigate the effects, but not stop them.
- Also: denial of service

History of DNS Security

- Inverse map attack known in the community, late 1980s
- Bellovin's "Security Problems in the TCP/IP Protocol Suite" (1989)
- Bellovin's cache contamination attacks, 1991 (but witheld until 1995)
- Vixie, 1995
- DNSSEC, late 1990s
- Kaminsky attack, 2008

What is the Domain Name System (DNS)?

- Tree-structured name space.
- Maps names to addresses, addresses to names.
- Control may be delegated at each level; levels represent administrative boundaries.
- Multiple servers handle each level.
- Servers need not be in the domain served.
- Local servers cache responses.
- Most hosts don't do any real name resolution; they ask a per-site caching resolver
- Queries usually use UDP, not TCP

Important DNS Record Types

Map name to address

```
ws1          IN  A   222.333.44.3
```

Route inbound email

```
ws1          IN  MX  10 mail1.ucia.gov
```

Delegate control

```
small.com.   IN  NS  server.foo.bar.
```

Inverse Mapping Tree

- Mapping addresses to host names uses a separate tree:

  ```
  3.44.33.222.in-addr.arpa.  IN  PTR  ws1.small.com.
  ```

- Inverse tree based on ownership of IP address blocks
- Note: IP address bytes are in reverse order. (Why?)
- DBMS-style inverse query not suitable — which server should you ask?
- Reversed order of address matches address space structure.
- No physical connection between the two trees.

DNS Queries and Answers

- Client may ask for particular record type, or all records for some name.
- Client may do *zone transfer*.
- Server gives answer, error indication, or `NS` record.
- When appropriate, server sends *addtional information* with answer (i.e., `A` record with `NS` record).

Attack: Subvert Name-based Authentication

- Some protocols use *name-based authentication*
- The connection reveals the client's IP address
- Use the DNS to map that IP address to a host name
- Compare the host name to the authorization list
- But — where does that host name really come from?

The Inverse Tree

- Remember that the address-to-name lookup is done via a tree based on IP addresses
- Delegation follows ownership of the IP address blocks
- The owner of the attacker's network controls that map
- That may be the attacker. . .
- So — the attacker creates a PTR record mapping his/her address to one of your trusted host names

Defending Against Inverse Tree Attacks

- Cross-check the returned domain name against the forward tree
- That is, after retrieving the host name, do a *forward query* to learn the real IP addresses for that name
- Compare the received address against that list
- Remember: though the attacker controls the address-to-name mapping, the defender controls the mapping from name to address
- Implemented: early 1990s

How DNS Uses Caches

- DNS transactions are rarely end-to-end
- A master site for a zone sends the full zone to several secondary servers. All are considered *authoritative*
- A typical host rarely talks directly to an authoritative server; it talks to its local caching resolver
- It may eventually talk to an authoritative server, but first it has to navigate a tree of `NS` records
- Those may be cached, too

Poisoning the Cache

- Caching resolvers receive their data over the wire
- If the data comes from non-authoritative sources, it may be false
- This can be used to introduce bogus data into a cache; this in turn can confuse users of that cache

The First Attack

- Query responses can contain *additional information*
- Example: if a host sends a query for an `MX` record, the server will send it back
- It may also (or may not) send the `A` record for the mail server
- They need not be in the same zone (see the example above for `cia.gov`)
- If I can make you query my zone, I can poison your cache for other zones

Footnote on Disclosure

- At the time, no real fix was feasible
- I did not publish my paper; instead, I shared it only within the security community
- About four years later, it showed up on a (convicted) hacker's server
- Witholding the paper did no good. . .

Sequence Number Attacks (Vixie; Kaminsky)

- DNS queries contain a sequence number, to match up queries with replies
- The sequence number is only 16 bits long
- It is not possible to make it very unpredictable
- Attack: send out fake responses; one will match the query number
- (It's actually somewhat more complex than that)

What Kaminsky Did

- Kaminsky's attack wasn't new, but it generated a lot of publicity
- Was the publicity justified? Yes!
- Kaminsky took two known-but-impractical attacks, and did the hard engineering work to combine them
- Without that, the attacks weren't practical in most situations. They are now.

Footnote on Analysis

- The two attacks he combined were documented in an RFC
- I (among others) reviewed that RFC before publication
- Neither I nor anyone else spotted the combination

Fake NS Attack

- Resolvers locate the proper authoritative name server by following a chain of NS records
- If you can change the NS record received, you can send a resolver to the wrong site
- It will then receive bogus data in the *answer* section
- Note: simply hacking *any* authoritative name server at a higher level will suffice to feed bad data to many resolvers

Attacking the Registration System

- Where do `NS` in `.com` come from?
- Sites contact a *DNS registrar* and supply the data
- Attack: steal a site's credentials for the registrar's web site, and change the data
- Note: all technical components of the DNS will function correctly — but wil give the wrong answer
- Example: `http://voices.washingtonpost.com/securityfix/2008/12/hackers_hijacked_large_e-bill.html` describes an attack on CheckFree, a payment site

The Root of the Problem

- The DNS is not an end-to-end protocol
- Simple transmission security will not suffice
- Even if it would, registration system attacks are outside of the DNS protocol
- (Perhaps we need a new definition of "end"?)

Defenses

- Heuristics
- Randomization
- UDP port usage
- DNSSEC

Heuristics

- Original defense against cache contamination
- Make sure answer make sense for a given query
- Example: `A` records don't normally need additional information
- Reject additional information pertaining to other zones
- Use additional information only for that query, i.e., keep the additional information cache with the original query, not the general data

Randomization

- Sequence numbers in queries were originally sequential
- Instead, use a PRNG to pick them
- As noted, you can't do a great job in only 16 bits

UDP Port Usage

- Kaminsky's idea: use many port numbers
- That is, use 128 different UDP ports for DNS queries
- Effectively extends the sequence number space by 6 bits
- But — it causes problems for some firewalls, requires lots of file descriptors, etc.

DNSSEC — Digital Signatures for the DNS

- The right solution
- Very hard to get right
- The process is itself worth a talk
- The code now exists — but we have to get it deployed

The Obvious Solution

- As noted, we can't use transmission security
- Obvious answer — treat DNS answers as email; Let authoritative servers digitally sign the records they hand out
- Use `RRSIG` records
- It doesn't quite work:
  - CPU load
  - Where do certificates come from?

CPU Load

- Some authoritative DNS servers handle a *lot* of traffic
- (How many queries/second for `www.google.com`?)
- Worse yet — what about a deliberate query storm, as a DoS attack?
- Easy: `<img src="http://www.target.com/some_image.jpg">`
- (Record need not even exist)
- Better yet — use Javascript on some web page to request `<img src="http://www%04d.target.com/some_image.jpg">`, in a loop

Advance Signing

- Solution: DNS records are signed offline; signed zone is uploaded to server
- Also protects signing key — need not be on accessible machine
- But — what about TTL?
- But — what about requests for non-existent hosts?

DNS Time-to-Live (TTL)

- When do DNS records expire?
- *Not* at an absolute time — the assumption is that machines do not have synchronized clocks
- A 1984 assumption — is it still valid?
- Instead, answers from authoritative servers have a time-to-live field (TTL) to govern lifetime in caches
- TTL is decremented by caching resolvers — can't be protected by the signature
- Is there a danger of replays? Yes — but that can't be fixed, given other design considerations

Negative Answers

- What if someone asks for `erchuvrai.yourdomain.com`?
- Next request: `etaneapdu.yourdomain.com`
- PRNGs are very cheap. . .
- Should return `NXDOMAIN` — but how can those be authenticated?
- They can't be digitally signed!
- Answer: `NSEC` records

NSEC Records

- Suppose that there are valid records for ABC, DEF, and WWW in your domain, and nothing else
- Create a NSEC record for ABC pointing to DEF
- If someone asks for BBB, return the NSEC record for ABC, which proves that it can't exist
- Similarly, DEF has an NSEC record that points to WWW
- (WWW has an NSEC record pointing to the zone itself. Similarly, the zone has a record pointing to ABC — complex, and not discussed further here.)

NSEC Records Break Privacy

- As mentioned, many zones prohibit zone transfers, in order to hide hostnames
- But — NSEC permits "walking" the zone
- Solution: NSEC3 records
- Like NSEC, except that it's based on the order of *hashed* DNS names
- (Considerably more complex than that. . .)

Where are the Certificates?

- What key is used to sign a zone?
- More precisely, where is the public key, and why should it be trusted?
- Answer: put it in the parent zone
- But — causes problems for key roll-over, especially timing of new zone key versus new `KEY` record in parent zone
- Solution: `DS` (Delegation Signer) record, to add a level of indirection
- (Again, complex)

Other Issues

- Deployability
- Legacy resolvers (use AD bit)
- Wild cards (`*.columbia.edu`)
- Zone cuts
- Record size — digital signatures are long, and DNS records are limited to 512 bytes (use EDNS0 signaling)

Deployability

- Suppose I sign `myowndomain.com`
- `.com` isn't signed — where are the `KEY` and `DS` records?
- Suppose Verisign wants to sign `.com` — where are its `KEY` and `DS` records?
- Who signs the root? ICANN? The U.S. Department of Commerce? The ITU? The Secretary-General of the UN?
- Use `DLV` records, to put the verifying key in the zone — but why should that key be trusted?
- What if an attacker deletes the DLV record and signature records in responses? Should the zone have been signed?
- Must have out-of-band infomration that a zone should be trusted

A Lot More Complexity

- I've presented a complex architecture for DNSSEC
- The reality is far more complex. . .
- Two problems: the DNS was not designed to be authenticated, and the original designers of DNSSEC didn't understand all of the fine points of DNS
- Lessons: (a) build in security from the beginning; (b) security designers have to work closely with base protocol designers

Semester Summary

Major Topics

- Cryptography
- Application security (web, email, storage)
- Protocols (IPsec, SSL, SSH, SIP)
- Firewalls and IDS
- Infrastructure (Routing, DNS)
- Attacks
- *Complexity*

Cryptography

- Symmetric algorithms; modes of operation
- Hash functions
- Public key (asymmetric) algorithms; certificates; PKI

Application Security

- Different applications need very different security solutions
- Transmission security vs. object security
- Trust anchors

Protocols

- Different needs at different layers
- Each layer has its own security properties — and vulnerabilities
- Security is generally needed at multiple layers

Firewalls and IDS

- Site-wide defenses
- Imperfect — but often better than nothing
- Increasingly rich connectivity problematic

Infrastructure

- True network security
- Requires cooperation across the Internet, not just local behavior
- Not yet deployed

Attacks

- Many attacks
- Many bad guys
- A variety of motives (especially money), but in general, attacks have gotten very sophisticated

Complexity

- The problems are complex
- The solutions are even more complex (and this course just skimmed the surface)
- Security works best when designed in from the beginning
- You can't just bolt on crypto
- *Know* the application domain, and work closely with the application programmers — they're the reason that the systems will exist