

Domain Name System 'Security'



Contents

Breaking DNS:

- Denial of Service attack
- Rogue DNS servers
- Spoofing
- Legacy attacks

Fixing DNS:

- Preventive measures
- DNSSEC
- Other solutions



Denial of Service attack

Obvious attack:

- Flood the name server to take it down

Not really DNS specific.

Additional DNS weaknesses:

- Recursive lookup amplifies load per request
- Hierarchy of servers makes for single points of failure (use **redundancy**)
- Easy to trigger DNS queries from other locations

Rogue DNS servers

A DNS server asks ns.evilgenius.com for evilgenius.com

```
;; QUESTION SECTION:
;evilgenius.com.      IN A

;; ANSWER SECTION:
; no answer (look it up elsewhere)

;; AUTHORITY SECTION:
evilgenius.com.  3600  IN  NS  ns.vu.nl.

;; ADDITIONAL SECTION:
ns.vu.nl.        IN  A   90.90.90.90
google.com       IN  A   90.90.90.90
```

Only cache information from authoritative servers

Rogue DNS servers

A DNS server asks ns.evilgenius.com for evilgenius.com

```
;; QUESTION SECTION:  
;evilgenius.com.          IN  A  
  
;; ANSWER SECTION:  
evilgenius.com.          3600  IN  CNAME name1.evilgenius.com
```

```
;; QUESTION SECTION:  
;name1.evilgenius.com.    IN  A  
  
;; ANSWER SECTION:  
name1.evilgenius.com.    3600  IN  CNAME name2.evilgenius.com
```

```
;; QUESTION SECTION:  
;name2.evilgenius.com.    IN  A  
  
;; ANSWER SECTION:  
name2.evilgenius.com.    3600  IN  CNAME evilgenius.com
```

Rogue DNS servers

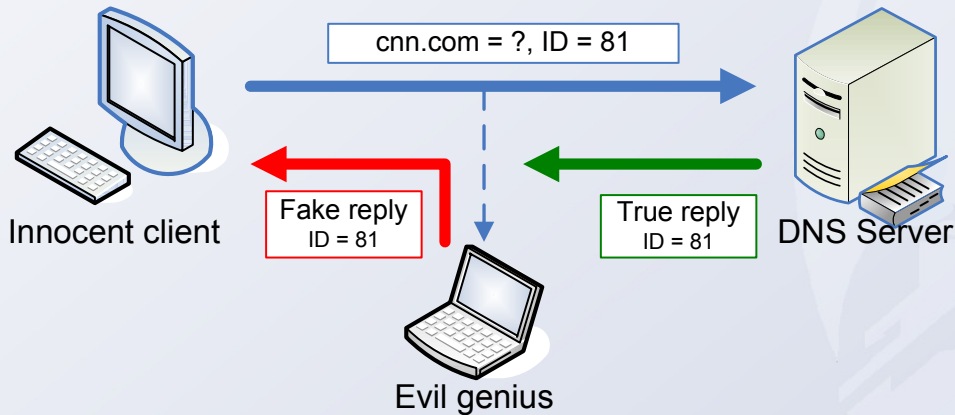
Other ways to annoy:

- Dynamic (push) updates
Master server pushes updates to slaves without authentication.
- Trigger many queries to ns.evilgenius.com, but don't reply
DNS Server needs to keep state for a while (DoS) and builds up a large list of unavailable transaction ID's (makes guessing easier)
- Redirect to arbitrary name servers
Causes long lookup chains

Local Spoofing

Man-in-the-middle attack:

- Inspect query
- Generate reply before the DNS server does



Remote Spoofing

Remote spoofing challenges:

- 65536 possible ID numbers in query/response
- 65536 possible ports used by client

but...

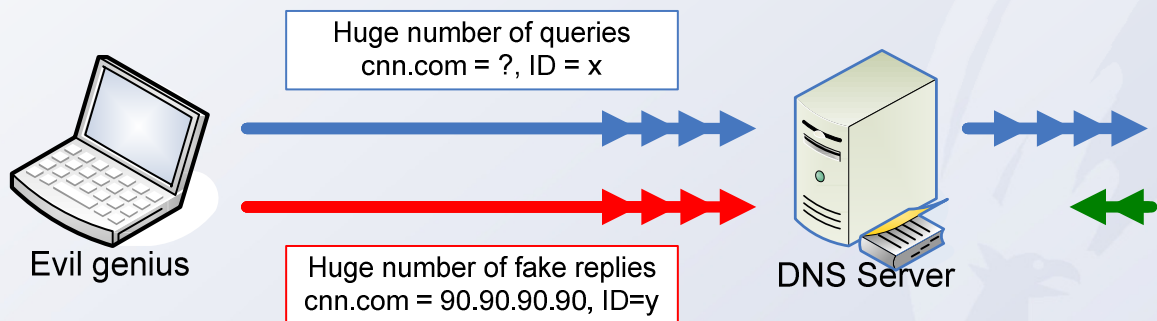
- UDP makes IP spoofing easy
- Port selection is usually predictable
- Easy to trigger recursive DNS lookup
- Birthday attack



Remote Spoofing

Remote spoofing:

- Trigger large number of queries
- Send large number of fake responses



Probability of success 50% for 300 handled messages

BIND9 Spoofing

300+ messages is still a bit much

but...

- ID numbers may not be very random either
- Pseudo-random number generators can be predictable
- Pseudo-random number generators can be forged (their state reconstructed)

BIND9 was found to be predictable to 10 choices

BIND9 was found to be forgeable using 13-15 messages

Intermission: BIND9 PRNG

```
uint generate(uint state, uint tap) {
    if(state & 1)
        return (state >> 1) ^ tap;
    else
        return (state >> 1);
}
uint skipgenerate(uint state, uint tap, uint skip) {
    if(skip)
        state = generate(state, tap);

    return generate(state, tap);
}
skip1  = statel & 1;
skip2  = state2 & 1;
statel = skipgenerate(statel, tap1, skip2);
state2 = skipgenerate(state2, tap2, skip1);
trxid  = (statel ^ state2) & 0xFFFF;
```

BIND9 Spoofing

Basic attack: Predict next transaction ID

Trigger recursive lookup for name1.evilgenius.com

```
if transID & 1 == 0
```

```
    send(ID=any, "name<i>.evilgenius.com IN CNAME vu.nl")
```

```
    nextID[] = prepareNextIdCandidates(transID)
```

```
    for i = 0 to 10
```

```
        send(ID=nextID[i], "vu.nl IN A 90.90.90.90")
```

```
else
```

```
    send(ID=any, "name[i].evilgenius.com IN CNAME name[i+1]...")
```

BIND9 Spoofing

Advanced attack: Forge state (make fully predictable)

Trigger recursive lookup for name1.evilgenius.com

if(first request)

 for(i = 0 to 256) push(i)

 prevID = transID

else

 for i in candidate

 state1 = generate(candidate[i])

 state2 = generate(candidate[i] ^ prevID)

 if(transID & checkBits != state1 ^ state2) remove(candidate[i])

 else push(1 + state1)

 push(0 + state1)

 prevID = transID

DNS Dependencies

So what? Just install the latest version.

DNS names depend on more than one name server.

Name servers for vu.nl:

```
;; AUTHORITY SECTION:
vu.nl.      86400    IN       NS       ns1.surfnet.nl.
vu.nl.      86400    IN       NS       star.cs.vu.nl.
vu.nl.      86400    IN       NS       ns.vu.nl.
```

IP of ns1.surfnet.nl is determined by name server of surfnet.nl

DNS Dependencies

Name servers for surfnet.nl

```
;; AUTHORITY SECTION:
surfnet.nl.      75686   IN      NS      ns0.ja.net.
surfnet.nl.      75686   IN      NS      ns1.surfnet.nl.
surfnet.nl.      75686   IN      NS      ns2.surfnet.nl.
surfnet.nl.      75686   IN      NS      ns3.surfnet.nl.
```

Name servers for ja.net:

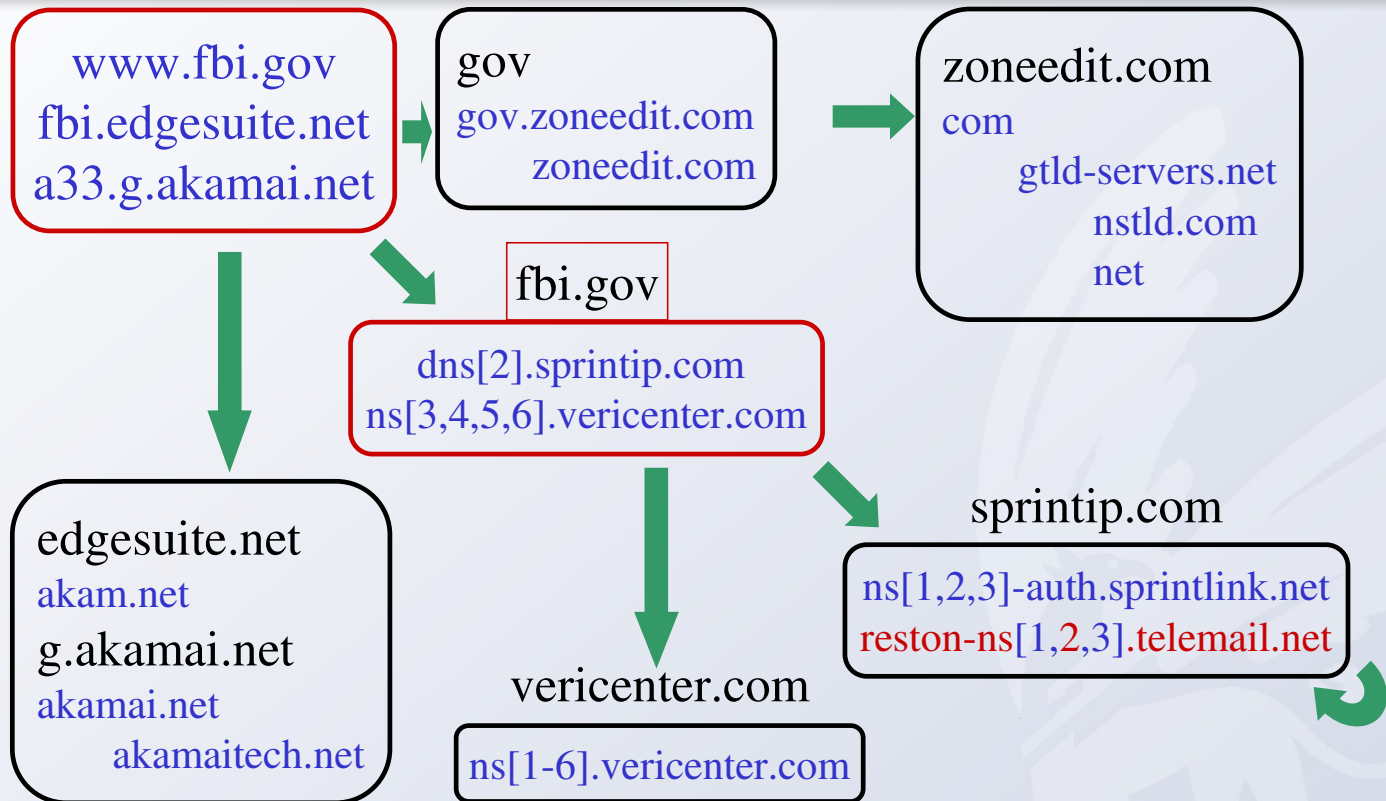
```
;; AUTHORITY SECTION:
ja.net.          79262   IN      NS      ns1.cs.ucl.ac.uk.
ja.net.          79262   IN      NS      ns1.surfnet.nl.
ja.net.          79262   IN      NS      ns4.ja.net.
ja.net.          79262   IN      NS      ns0.ja.net.
```

DNS Dependencies

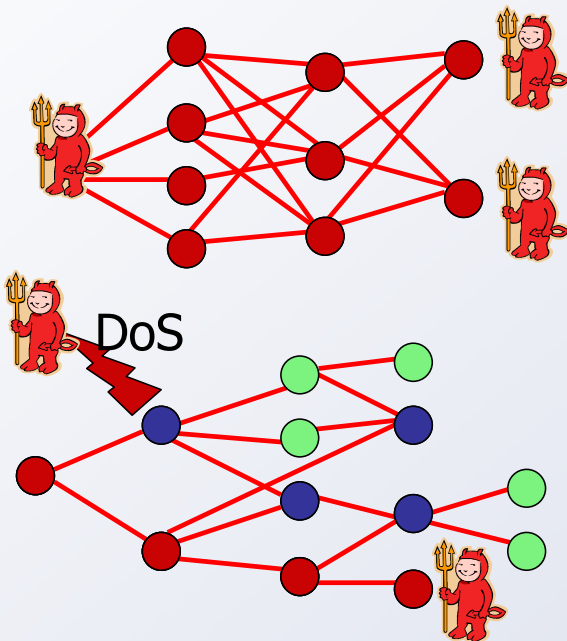
Survey:

- Surveyed BIND versions of names in DMOZ & Alexa Top 500
 - Queried public version numbers
 - 40% response rate
- Compared against database of known vulnerabilities
 - Many have well-known exploit scripts available
- Examined the dependency graphs to determine how vulnerable names are

DNS Dependencies



DNS Dependencies



An attacker can compromise a name completely if it can acquire a graph cut

If a full cut is not vulnerable, attacker must combine compromise with DoS

DNS Dependencies

Survey results:

- 17% of servers have well-known flaws
- 30% of names vulnerable directly
- 84% of names vulnerable through 2x DoS
- Almost all names vulnerable through 8x DoS

Well in theory...

In practice 'glue records' coincidentally hold things together

'Fixing' DNS



Preventive measures

Band-aid solutions

- Only cache information from authoritative servers
- Cross-check IP \leftrightarrow DNS mappings (not always possible)
- Transaction signatures for zone transfer, dynamic updates
- Split-split strategy:
 - Split: Advertising name server for DNS servers (iterative)
 - No cache to poison
 - More resilient to DoS
 - Split: Resolving name server for DNS clients (recursive)
 - Only allow internal traffic

DNSSec

Add authentication and integrity to DNS:

- Key distribution
- Origin authentication
- Transaction authentication

Built into DNS through new Resource Records:

- KEY (now DNSKEY) – public key
- SIG (now RRSIG) – signature for RR set
- NXT (now NSEC) – proof of non-existence
- New: DS – proof of delegation

DNSSec

Problems

- Deployment 0% after 13 years
- Security is optional
- Poor performance
- Protocol changes all the time
- Legacy problems again:
 - We won't be secure until all dependencies are
- Applications cannot rely on (non-)security
- Compromise of top-level keys is a global threat
- DoS threat increases

Other solutions

X509 Certificates

- Hierarchical certificates match hierarchical domains
- Inefficient

CoDoNS (Beehive)

- Peer-to-peer DNS with $O(1)$ avg. lookup complexity
- Backwards compatible
- No secure routing
- Doesn't match administrative model

Conclusion

- DNS is, was and will be broken
- Legacy makes DNS more and more vulnerable
- DNS should never be used for authentication
- DNSSEC resolves some of the issues

- DNSSEC has failed?
- What to do?

