DNS For Network Defense

Using DNS to protect and observe

Klaus Möller

WP8-T1

Webinar, 3rd of December 2020

Public

www.geant.org
What we will cover today

• Protect
  - DNS Manipulation for good
    • Blackholing & Response Policy Zones
  - Useful Zones and Resecure Records (RRs)
    • Localhost, RFC 1918, etc.
    • RRs: TLSA, SSHFP, IPSECKEY, CAA, CERT

• Observe:
  - Query logging
  - Passive DNS monitoring

• Examples will use BIND 9 nameserver
Blacklisting SPAM

- Has been in use for a very long time (MAPS, Spamhaus, ...)
- MTA queries special SPAM Blacklist nameserver
  - I.e. SPAM BL is operated apart from normal zones and nameservers
- Nameservers serve zones with FQDNs of known spamming hosts
  - Answer is NXDOMAIN = Host is OK
  - Answer is 127.0.X:Y = Host is spamming,
  - X.Y tells which blacklist the host/domain/ip-address is on
    - Have to look this up for a given blacklist provider (e.g. Spamhaus)
- Usefulness has dimished over time, but is still SOP for most MTAs
BlackHoling DNS (BHDNS)

- Other names: Sinkhole DNS, DNS Firewall
  - List of blackholed names: DNS Blacklist (DNSBL) or Realtime Blacklist (RBL)
- Nameserver answers queries for “known bad” names “differently”
  - With NXDOMAIN or 127.0.0.1 for example
- What exactly is meant by “bad”? 
  - Malicious Stuff: Drive-by URLs, C&C servers, landing pages, black market, etc.
  - Others: SPAM, porn, shopping, betting, VPN, proxies, “critics” etc.
- Advantages for network administrators
  - Not limited to browsers (like WoT)
  - No client configuration, etc. (they likely use your nameserver anyway)
  - Names and IP-addresses from your network do not leak to the internet
How to operate a Blacklist

1. Policy should be maintained separately from the rest of the DNS
   - No fiddling with the original zone data
   - Entries would be spread all over the DNS

2. Automation
   - There will be 1000s of entries

• Response Policy Zones (RPZ)
  - RPZ zone files are syntactically normal zone files
  - But are treated differently by the nameserver
  - Can be maintained locally or obtained from provider
    • Zone transfer (AXFR, IXFR) or file transfer (wget, ftp, ...)
  - Supported by BIND, Knot DNS, PowerDNS, Unbound, ...
RPZ Schema

Source: https://www.switch.ch/dns-firewall/
How RPZs operate (1)

a) Tell the nameserver to use response policies

```plaintext
options {
    ...  
    response-policy {
        zone "rpz.local";
        zone "rpz.slave";
        zone "rpz.test" policy passthru;
    };
    ...
}
```

Override actions from zone file (i.e. for tests)
How RPZs operate (2)

b) Create zone files

```
$ORIGIN rpz.example.net.
...

nxdomain.example.com   CNAME   .     ; NXDOMAIN
nodata.example.com     CNAME   *     ; NODATA
bad.example.com        A       10.0.0.1
                       AAAA    2001:db8::1

*.azone.example.com    CNAME   garden.example.net.
ok.azone.example.com   CNAME   rpz-passthru.
24.0.2.0.192.rpz-ip    CNAME   .
32.1.2.0.192.rpz-ip    CNAME   rpz-passthru.
ns.example.com.rpz-nsdname CNAME   .
32.zz.db8.2001.rpz-nsip CNAME   .
25.128.2.0.192.rpz-ip  A       172.16.0.1
25.128.2.0.192.rpz-ip  MX      10 mx1.example.com
25.128.2.0.192.rpz-ip  TXT     "Your are infected."
```

No periods after Relative owner names

5th octet is subnet mask
RPZ Zone file rules

TRIGGER
RRSet Owner Name in zone file

1. **QNAME**: Match on domain name queried in requests and responses
2. **Client IP Address**: Match on querying Client IP Address if owner ends in `.rpz-client-ip`
3. **Response IP Address**: Match on IP addresses in the DNS response if owner ends in `.rpz-ip`
4. **NSDNAME**: Match nameserver names (NS records) if owner ends in `.rpz-nsdname`
5. **NSIP**: Match on name server IP addresses(A/AAAA) if owner ends in `.rpz-nsip`

→ ACTION
RRSET Target in zone file

1. **NXDOMAIN**: Return NXDOMAIN for targets ending in "CNAME ."
2. **NODATA**: Return NODATA for targets ending in "CNAME *."
3. **PASSTHRU**: Let response pass unaltered if target ends with CNAME rpz-passthru.
4. **DROP**: Drop query if targets ends with CNAME rpz-drop.
5. **TCP-Only**: Respond with if target ends with CNAME rpz-tcp-only.
6. **Local Data**: Respond with other data from zone file (arbitrary RR types)
RPZ: Sources

- Where Do we get lists of “bad names”?
  - Abuse.ch URLhaus
  - **SWITCH DNS Firewall**
  - SURBL securityZONES
  - FarsightSecurity NOD
  - More examples in the references

- Caveat emptor!
  - Quality varies
  - Availability varies
  - Price varies
However

• With great power comes great responsibility
• BHDNS and RPZ are great tools for censorship too
• Check with your legal advice (liability anybody?)
  – Are you allowed to block at all?
  – What has to be done to block in a legally conforming way?
• And check with your users and bosses too
  – A policy will have to be drafted, discussed, etc.
• Much more additional work
  – Configuring RPZs in a nameserver is trivial
  – Using them in a responsible and acceptable way is hard
Useful Zones to serve

• Why?
  – Would be forwarded to root nameservers
  – Information leak (internal names, IP-addresses)
  – Unnecessary traffic/burden on the root NS

• localhost, .example, .example.net, .example.org
  – May sometimes be seen on the net
  – Usually misconfigurations (samples copied literally)
  – .local will break Bonjour!

• RFC 1918 et al.
  – Also for IPv6 and other networks, see RFC 6890 & RFC 8190
Web Proxy Auto-Discovery Protocol (WPAD) Entries

- Browsers search for hosts named `wpad` in their domains to retrieve a URL for proxy auto-configuration
- For `host.sub.dom.tld` it would look for
  - `wpad.sub.dom.tld`
  - `wpad.dom.tld`
  - `wpad.tld`
- The URL tried will be: `http://wpad ... /wpad.dat`
- `wpad.dat` is a JavaScript file doing proxy auto configuration (e.g. `proxy.pac`)
- If the host/URL does not serve a file, the next host on the list will be tried
- Information gotten from DHCP (WPAD option) takes precedence
  - But only within IPv4
- Better turn off “detect proxy setting automatically” (aka WPAD) (`network.proxy.enable_wpad_over_dhcp: false`)
Useful Entries:

- **use-application-dns.net.**
  - “Canary domain” queried by Firefox (and maybe other Mozilla products)
  - Use case: turn off DNS over HTTPS (DoH), if
    - Negative result (NXDOMAIN, SERVFAIL)
    - or empty answer (no A or AAAA RR)
  - Just put an empty zone file into your nameserver

- **Google Chrom* captive portal detection domains**
  - Chrom* browsers make DNS queries to three random (8-15 characters) domains
  - If at least two of them resolve to the same IP-address, a DNS captive portal is assumed
  - In turn, chrome does no try to interpret single words as hostnames
  - The same goes for requests to http://clients3.google.com/generate_204
Useful RR Types

- SSHFP – SSH Host Key Fingerprints
- TLSA – Binding of a X.509 Certificate to a service
- CAA – Certification Authority Authorization – Who may issue certificates for a domain
- More RRs, very little use so far
  - IPSECKEY – IPSEC Public Key
  - OPENPGPKEY – Bind PGP Keys to an e-mail address
SSHFP RRs

- Puts SSH Host Fingerprints into DNS
  - So you don’t have to distribute files from `/etc/ssh/host_key*` to `/etc/ssh/known_hosts` or `~/.ssh/known_hosts`

- Example:

<table>
<thead>
<tr>
<th>Hostname</th>
<th>Type</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Public Key Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>alice.example.edu</td>
<td>SSHFP</td>
<td>4</td>
<td>2</td>
<td>AAAAC3NzaC1lZDI1NTE5AAI...</td>
</tr>
<tr>
<td>alice.example.edu</td>
<td>SSHFP</td>
<td>1</td>
<td>1</td>
<td>AAAAB3NzaC1yc2EAAAAADAQAB...</td>
</tr>
<tr>
<td>alice.example.edu</td>
<td>A</td>
<td>10.1.2.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```plaintext
alice.example.edu. IN A 10.1.2.3
alice.example.edu. IN SSHFP 4 2 AAAAC3NzaC1lZDI1NTE5AAI...
alice.example.edu. IN SSHFP 1 1 AAAAB3NzaC1yc2EAAAAADAQAB...
```
SSHFP Example

• Generation with ssh-keygen (from /etc/ssh/host_key_<pub key algorithm>.pub)

```
ssh-keygen -r alice.example.edu
```

```
alice.example.edu IN SSHFP 1 1 cd169ea783f92777390f9f61830fe8d6ee52398f
alice.example.edu IN SSHFP 1 2 dd4f605d871df00b52ca112a216eed55717b6315c4b023ad86668d96f58cc0e5
alice.example.edu IN SSHFP 2 1 2206541d5e37ccbab4729b2dac8d648bb029f97e
alice.example.edu IN SSHFP 2 2 b0a01592759417e5f4ad576cf282148d33e5b578a4fb3e9bd56d541e94a8bbf
alice.example.edu IN SSHFP 3 1 3611fec195ef2e31281490b93e2d697d1f3ecc61
alice.example.edu IN SSHFP 3 2 e4ff16f41711a56349c8a60099e10e9a19fd67b8276c33de568815cc05009d
alice.example.edu IN SSHFP 4 1 a4eb77ccca51d06c4d3660c6919c7090bde4a3ab
alice.example.edu IN SSHFP 4 2 83bb62496bb293f2b628891a28bd3db5da3c135880bce31df74066db7a904d90
```

• SSH invocation to verify:

```
ssh -o VerifyHostKeyDNS=ask alice.example.edu
```

• In ~/.ssh/config

```
Host alice.example.edu
  VerifyHostKeyDNS ask
```
**TLSA RR**

- Binds a X.509 certificate to a server (protocol, port) and FQDN
- Prevents stolen X.509 keys to be used on other names or IP addresses
- Part of **DANE** = DNS-Based **A**uthentication of **N**amed **E**ntities
- Without DNSSEC, TLSA verification will always fail
- Example/Format:

  ```bash
  > dig +multi tlsa _443._tcp.bob.dom.example.edu
  ... 
  _443._tcp.bob.dom.example.edu. 10 IN TLSA 3 1 1 ( E3C9F ... 74D2 )
  ```
TLSA RR Fields

• Certificate Usage: Certificate data presented by the service must match
  – 0: against a public CA certificate (“CA restraint”)
  – 1: End Entity (EE) match validated by public CA (“Service certificate restraint”)
  – 2: against a private CA certificate (“Trust anchor assertion”)
  – 3: against only the certificate without any CAs (“Domain issued certificate”)

• Selector - which part of the servers TLS certificate will be matched against
  the certificate association data
  – 0: Certificate Association Data field is based on the full certificate data
  – 1: Certificate Association Data field is based on the public key only

• Matching Type: how the certificate association data is presented
  – 0: Certificate Association Data field contains the full certificate
  – 1: Certificate Association Data field contains a SHA-256 hash
  – 2: Certificate Association Data field contains a SHA-512 hash
CAA RRs

- Problem it solves: What Certification Authority (CA) may issue certificates for a given domain “say example.net”
- For use by CAs when issuing certificates
- May be set for any level within the DNS
- Records are evaluated from left to right, first match

<table>
<thead>
<tr>
<th>example.org</th>
<th>IN CAA 0 issue &quot;pki.dfn.de&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub.example.org</td>
<td>IN CAA 0 issue &quot;example-pki.org&quot;</td>
</tr>
</tbody>
</table>
CAA RR Structure

\(<\text{domain}>\) IN CAA \(<\text{Flag}>\) \(<\text{Tag}>\) \(<\text{Value}>\)

- **Flag:**
  - Currently 0 or 1 (*issuer critical*)
  - If set to 1, the tag/value pair must be understood (and followed) or no certificates may be issued

- **Tag:**
  - **issue**: CA under "value" is allowed to issue certificates for the domain
  - **issuewild**: CA under "value" is allowed to issue wildcard (*) certificates for the domain
  - **iodef**: value is an URL to report certificate misuse
DNS Query logging

- Log DNS queries at central (caching) nameservers
  - Look for queries to “bad” domains
  - Take action (i.e. clean host)

- Easy to enable
  - Just type `rndc querylog` to turn on in BIND9
  - Logs to Syslog (usually ends up in `/var/log/messages`)

- Check with your lawyers & privacy officers first!

- Tells
  - Who made the query (the IP address)
  - When the query was made (the timestamp)
  - What the query was asking for (i.e. RR type and domain)

- Does not tell what the answer was
DNS Query logging: Nameserver Config

- Two parts in BIND9

```
# cat /etc/named.conf
... 
options {
    querylog yes;
}
```

```
# cat /etc/named.conf
...
logging {
    channel querylog {
        file "/var/log/querylog";
        print-time yes;
        print-category yes;
        print-severity yes;
        severity debug 3;
    }

};
```

- Performance impact can be huge
  - Use seperate server for caching resolver, log there
Passive DNS (PDNS) Monitoring

- Invented 2004 by Florian Weimer (then at RUS-CERT)
- Sensor monitors incoming DNS responses (and sometimes queries)
- Logs data in a "standard" format
  - Timestamped (for the history)
  - De-duplicated (resolvers sent several queries in parallel)
- Example of a “standard” format: `dnstap` (binary)
  - Also the name of a PDNS monitoring tool
- Data from many sensors can be combined in a shared database
- Little impact on privacy when logging only responses to caching resolvers
  - But personally identifiable information when combined with internal data
- **Again: Check with your lawyers & privacy officers first!**
Importance of PDNS Monitoring

- Historical DNS data is the point (e.g. past DNS responses)
- Think threat hunting, i.e. you have a C&C server hostname
- But firewalls, NAT, VPN, NetFlows log IP-addresses, not FQDNs
- Lookup of name in PDNS DB gives closes the gap
  - Timestamps also give further hints
  - Frequency of address changes might hint at Fast-Flux DNS networks
- Hints for other names for a given IP (multiple SPAM domains)
- Looking responses that are typos of your domain
  - e.g. dfm-cert instead dfn-cert
  - Needs PDNS DB that supports Soundex or fuzzy matching
- Detecting cache poisoning by querying external PDNS DBs
PDNS Sensors

- Primitive sensor: `tshark -i <if> "udp and src port 53"`
  - Pair with PacketQ für SQL queries against .pcap files
- Use your recursive/caching Nameservers as sensors
  - Format & Tool: `dnstap`
  - Supported by: BIND, CoreDNS, Dnsdist, Knot, NSD, PowerDNS, Unbound, ...
- NIDS (Snort, Suricata, OSSEC, etc.)
  - Have to write rules for that
  - Bro: `https://github.com/JustinAzoff/bro-pdns`
- Firewalls can act as Sensors (Palo Alto, Cisco, Watchguard, etc.)
- Option for Outsourced DNS services (OpenDNS, etc.)
- Sensors on endpoints (Red Canary, etc.)
PDNS (public) Databases

• Companies:
  – VirusTotal (aka Google):
  – SecurityTrails: https://securitytrails.com/dns-trails

• CERTs:
  – CIRCL Passive DNS: https://www.circl.lu/services/passive-dns/
  – CERT-EE: ?
  – BFK: Down, see https://www.bfk.de/bfk_dnslogger_en.html
    • Non public service may be still active: https://portal.bfk.de/
What have you learned?

• How to use DNS Response policy zones to blackhole traffic to/from malicious hosts/domains
• Utilize DNS to distribute and verify public key information
• Monitor DNS traffic for malicious activity

What has been left out?

• All this would not be secure if the integrity of the DNS itself can’t be ascertained
• How do we do that? → DNSSEC, see you in the next module
Thank you

Any questions?

Next module: **DNSSEC**, 7\(^{th}\) of December 2020

www.geant.org
References:

- BlackHole DNS for Spyware
  https://www.malwaredomains.com/bhdns.html
- Vixie et al.: DNS Response Policy Zones (RPZ),
- Building DNS Firewalls with Response Policy Zones (RPZ)
  https://kb.isc.org/docs/aa-00525
- Windows DNS Server Sinkhole Domains Tool,
  https://www.sans.org/blog/windows-dns-server-sinkhole-domains-tool/
- DANE Testsites: https://www.huque.com/dane/testsite/
- Hash-slinger - Generate and verify various DNS records such as SSHFP, TLSA and OPENPGPKEY:
  https://github.com/letoams/hash-slinger
Response Policy Zone Providers (Examples)

- CleanBrowsing [https://cleanbrowsing.org/filters](https://cleanbrowsing.org/filters)
- Deteque: [https://www.deteque.com/dns-firewall/](https://www.deteque.com/dns-firewall/)
- FarsightSecurity NOD: [https://www.farsightsecurity.com/Services/NOD/](https://www.farsightsecurity.com/Services/NOD/)
- RiskAnalytics Malwaredomains: [https://www.malwaredomains.com/](https://www.malwaredomains.com/)
- SURBL securityZONES: [http://www.surbl.org/df](http://www.surbl.org/df)
- **SWITCH DNS Firewall** [https://swit.ch/dnsfirewall](https://swit.ch/dnsfirewall)
References:

- SANS InfoSec Handlers Diary Blog: “Internet Choke Points: Concentration of Authoritative Name Servers”, https://isc.sans.edu/forums/diary/Internet+Choke+Points+Concentration+of+Authoritative+Name+Servers/26428/
- Domain Name System Operations Analysis and Research (DNS OARC), also maintains PacketQ, https://www.dns-oarc.net/
- Query multiple PDNS databases: Passive::DNS Client: https://github.com/tresni/passivedns-client
- dnstap: https://dnstap.info/
Requests For Comments (RFCs):