Why Bother Securing DNS?

Dieter Gollmann(✉)

Security in Distributed Applications,
Hamburg University of Technology, Hamburg, Germany
diego@tuhh.de

Abstract. The current state of DNS security is characterized by two opposing developments. DNSSEC introduces a PKI to support message authentication in the DNS protocol; DANE proposes to use this PKI also for provisioning TLS certificates. At the same time, PKIs are perceived as a major point of weakness; mechanisms like certificate pinning attempt to reduce the trust one needs to place in a PKI. We note that DNS provides rendezvous, identification, and introduction services and argue that this differentiation can reduce the impact of compromised trusted third parties.

Keywords: Domain Name System · TLS · DANE · Identification · Rendezvous services · Critical infrastructures

If it is trusted it can hurt you. [Robert Morris Sr.]

1 Introduction

It has become commonplace to note that critical infrastructures are increasingly relying on the internet, and that the internet has become a critical infrastructure itself. Complaints about the insecurity of the internet and demands for securing this critical infrastructure then quickly follow from such observations. With apparent inevitability, endeavours for securing the internet – a communications infrastructure after all – are drawn towards cryptography. We will follow this path in the case of the Domain Name System (DNS), a critical component within the internet. We will briefly reflect on current DNS security incidents, argue why reliance on "security solutions" that involve trusted third parties is bad for security, and put forward the case that security is not improved by deploying stronger security mechanisms but by reducing reliance on the infrastructure. Specifically, we observe that DNS serves more than one purpose. It provides a *rendezvous service* and an *introduction service*. *Identification services* are in the process of being added. Addressing these three aspects separately may be a way towards improving the security of applications using the internet. Separation of concerns is, of course, a well established security strategy.

B. Christianson et al. (Eds.): Security Protocols 2014, LNCS 8809, pp. 1–8, 2014.
DOI: 10.1007/978-3-319-12400-1_1

2 Domain Name System

The Domain Name System plays a crucial rôle in the internet, mapping host names to IP addresses. *Authoritative name servers* manage zones and make statements about the bindings between host names and IP addresses for hosts in their zone. All other participants trust their statements. Resolvers use a hierarchy of root servers and global top level domain servers to find authoritative name servers. This is the *rendezvous service* service provided: given a host name, *name resolution* returns its current IP address. Security is based on trust in the name servers and in a simple authentication of server responses.

The authentication mechanism originally specified for DNS uses a challenge-response pattern (*return routability*): queries for a host name contain a 16-bit query id; a resolver accepts the first response that contains this host name and the query id sent (and arriving at the expected port) as authoritative. This message authentication mechanism does not rely on any trusted third parties or shared secrets.

2.1 DNS Cache Poisoning Attacks

This authentication mechanism is relatively weak, leaving recursive name servers open to *cache poisoning attacks* [6]. Recursive name servers keep a cache of the bindings they have received. Queries for host names with cached bindings are served directly, without involving the authoritative name server. Cache entries expire based on a time-to-live set by the authoritative name server.

A cache poisoning attack triggers name resolution for a target host at the resolver and then floods the resolver with spoofed answers with guessed query ids and an IP address of the attacker's choice. The attack succeeds if a spoofed answer with correctly guessed query id arrives before the genuine answer.

The attacker's chances improve considerably if a resolver will run several name resolutions for a given host name in parallel. The attacker triggers several name resolutions and floods the resolver with spoofed answers. Now, one of the attacker's guesses has to match one of the resolver's query ids; the probability for the attack to succeed is related to the birthday paradox. Such a vulnerability had been reported for BIND 4 and BIND 8 in a security advisory[1] in 2002.

A DNS cache poisoning attack launched against the DNS server operated by the Chaos Computer Club (CCC), dnscache.berlin.ccc.de (213.73.91.35), followed the same pattern[2]. The CCC had been running *djbdns*, highly praised for its randomization algorithms, as its name resolution software. A birthday paradox vulnerability in *djbdns* had been known since 2009 [2], a patch for *djbdns* had been provided, but the CCC was still running an unpatched version. We are faced with a known instance of a known problem with a known remedy. In this respect, securing the infrastructure is a practical software security issue.

[1] http://www.rnp.br/cais/alertas/2002/cais-ALR-19112002a.html

[2] https://www.fehcom.net/diary/2014/20140212.html, http://www.heise.de/newsticker/meldung/DNS-Server-des-CCC-Anfaellig-wegen-veralteter-Software-2112171.html

However, powerful attacks are possible even when query ids are chosen at random and when the search space is enlarged with further randomizations, e.g. the choice of port number and mixing upper and lower case characters in the spelling of the host name. Dan Kaminsky had shown an attack that exploits *additional resource records*, another performance optimization. A DNS response may contain an *additional section* where the authoritative name server includes bindings for hosts that resolver had not asked for but might want to resolve in the near future. For example, the response for a query for www.example.com might also include a record for mail.example.com. Resolvers do not blindly trust authoritative name servers on additional resource records but perform *bailiwick checking*. Only records for hosts in the same domain ("in the bailiwick") of the host the query has been issued for are cached.

The attack asks to resolve a random host name in the bailiwick of the target. This random host name has most likely no entry in the resolver cache, so name resolution is triggered, and most likely the host does not exist, so the authoritative name server would send a NXDOMAIN response. The attacker's spoofed responses contain a binding for the target in their additional section. If the attacker's response wins the race the cache entry for the target entry gets poisoned; if the attacker loses the race a new race for another random host name is started immediately. This attack convinced the DNS community that it was high time to move to cryptographic message authentication in the DNS protocol.

2.2 DNS Rebinding Attacks

Cryptographic message authentication strengthens defences against "outsiders" impersonating authoritative name servers. It does not stop authoritative name servers from exploiting the trust placed in them. In *DNS rebinding attacks* [3–5], an unsporting authoritative name server maps a host in the attacker's zone to an IP address of a host that is not. In this way, the attacker may, e.g., circumvent the *same origin policy* enforced by browsers and use a client as a proxy to access hosts outside the attacker's zone (but believed by the client to be in the zone).

Same origin policies regulate, e.g., where a script executed in the browser may connect to. To enforce this policy, the browser has to know the *origin* of the script (authentication of origin is not our concern here) and the IP addresses corresponding to that origin. The bindings issued by authoritative name servers can thus be viewed as policy rules in an access control system, which are evaluated in the browser. In the language of access control, authoritative name servers act as *Policy Information Points*.

Authoritative name servers are, by design, authoritative for binding hosts in their own zone to IP addresses. They thereby become authoritative for binding IP addresses to hosts in their zone, but without any restrictions on the IP addresses they may issue bindings for. They can thus hijack arbitrary IP addresses for their zone. This is a serious construction flaw in an access control system. The defence suggests itself: send a query to the IP address to check whether it "speaks for" the given host name. There are strong parallels to the defences against bombing attacks in networks with node mobility as discussed in [1].

3 DNS and Public Key Cryptography

Deploying cryptographic authentication in the internet is at its heart a key management challenge. Parties need to be provided with correctly attributed public verification keys. Certificates create cryptographically protected bindings between hosts and their public keys. The issuers of those certificates (a.k.a. certification authorities) become trusted third parties. We will look at the way this key management challenge has been addressed and, in particular, at the trust placed into certificate issuers.

3.1 DNSSEC

The attacks on DNS had re-ignited interest in cryptographic authentication based on digital signatures (DNSSEC, RFC 4033) as a replacement for the weak authentication mechanism mentioned above. This kind of authentication relies on a Public Key Infrastructure. The PKI for DNSSEC, by and large, mirrors the hierarchical structure of the DNS and is gradually becoming operational. The signed root zone, implemented by ICANN and Verisign with input from the U.S. Department of Commerce, exists since July 2010[3]. At the time of writing (2014-05-29), 403 of the 589 top level domains are signed, 395 have trust anchors published in the root zone[4]. Top level domains and the root zone are "roots of trust" in this PKI, but they are roots of trust in DNS anyway as far as name resolution is concerned.

A PKI needs a secure way of distributing public verification keys to the relevant parties. Accepted methods for public key delivery are listed in the *DNSSEC Practice Statement for the Root Zone KSK Operator*[5]. The internet draft on *DNSSEC Trust Anchor Publication for the Root Zone*[6] covers the same topic.

3.2 TLS

There exists a second – already widely used – PKI for the internet, created for facilitating access to secure web sites via *https*. This PKI puts certification authorities (CAs) in a very powerful position. They can issue certificates for any host in the web. Once a CA is included in the list of trusted roots on a client it becomes authoritative for the entire web (for that client).

An attacker who has compromised a certification authority can thus issue bogus but nevertheless valid certificates for arbitrary hosts. A well reported case is that of the Dutch CA DigiNotar. Google had noted in 2011 a DigiNotar issued certificate for google.com not contained in Google's own list of certificates for

[3] http://www.root-dnssec.org/
[4] http://stats.research.icann.org/dns/tld_report/
[5] https://www.iana.org/dnssec/icann-dps.txt
[6] http://tools.ietf.org/html/draft-jabley-dnssec-trust-anchor-07

google.com. The incident had serious impacts on IT services offered by the Dutch government[7] and on DigiNotar, which filed for bankruptcy.

Is the list of trusted certificates (public keys) at the client a solution to this problem? Clients could for important hosts define a set of authorized CAs (introduced as *certificate pinning* in Chrome, although CA pinning would be more accurate), thus moving those hosts out of the reach of all other CAs. Certificate pinning is also used to describe solutions where the certificate is hard coded in a client application and the certificate received in a TLS handshake is compared against this "pinned" certificate. In this case, the CA authorized for the application is fixed.

Is the list of trusted certificates (public keys) at the client a part of the problem? Consider Mikko Hyppönen's post[8] from April 2013 on finding that the US DoD certification authority is pre-installed on various Apple devices:

- *My phone carries a root certificate for a military.*
- *From one country.*
- *And it's not my country.*
- *And I can't remove it.*
- Issuer: C=US, O=U.S. Government, OU=DoD, OU=PKI, CN=DoD CLASS 3 Root CA

The average user is hardly in a position to judge the trustworthiness of a trusted CA, e.g., its proximity to the government of the country it is operating in.

3.3 DANE

The wish to restrict the impact of corrupted CAs in TLS takes us back to DNS. The idea of certificate (CA) pinning could be extended. CAs could be authorized to issue certificates only for a limited scope of hosts. We would then have to define a policy that states which CA is authoritative for which set of hosts.

In DNS, authoritative name servers are already trusted on mapping host names to IP addresses. With the introduction of DNSSEC, they are also trusted to sign resource records, to confirm the public keys of sub-authorities, and to protect their own private keys. It is then a plausible next step to build a PKI for *https* on the basis of DNS and let authoritative name servers (or registrars) issue certificates for hosts in their domain, but only for hosts in their domain. Such a PKI has been specified as *DNS-Based Authentication of Named Entities* (DANE, RFC 6698). Corrupted CAs can only affect their own zone.

On the other hand, compromise of an authoritative name server now lets the attacker not only provide a wrong IP address for a host (attack at the network layer) but also a wrong public encryption key (attack at the application layer). Have we improved security or made matters worse? This brings us to the main question of this discussion paper:

[7] http://www.onderzoeksraad.nl/uploads/items-docs/1833/Rapport_Diginotar_EN_summary.pdf

[8] https://twitter.com/mikko/status/327170802673917952

Have we been walking in the wrong direction by putting too many requirements on the Domain Name System, which made us rely more heavily on DNS, forcing us in turn to look for stronger security mechanisms?

4 Splitting Services

Alternatively, we might treat DNS just as a *rendezvous service* providing the current IP address of a host without any pretence of delivering authentication.

- Addresses can change in space and time.
- It matters when no rendezvous service is available.
- It does not matter when wrong information is provided as long as alternative services can be consulted.

Cryptographic protection may have a rôle at the network layer but not in the rendezvous service itself. We do not want to trust rendezvous services in the first place. DNS would just make a best effort to provide an IP address for a host. Failure to provide a correct IP address is then an availability issue, not an authentication issue, to be addressed with methods for improving availability.

A further service needs to confirm that the host a client is looking for is residing at the address obtained. We need such a service anyway because authoritative name serves may lie. The fact that, with DNSSEC, their answers are signed does not imply that they are true. The service used in the case where a host is already known to the client can be different from the service used when there has been no previous interaction.

In the first case, the client would not ask the host "who are you" but "are you the one I want to connect to". The client could remember from a previous visit how to recognize this host, and the host could answer this question by providing evidence that it is the same as at the client's last visit. We call this an *identification service*. Such a service confirms that a host is the same as last time and not someone else pretending to be that host. This follows Pekka Nikander's argument[9] that etymologically identity, stemming from Latin *idem et idem*, means "the same as before". We are well aware that identification has also other meanings in the field of IT security.

For identification a pinned public key of the host or a shared secret would do. Current developments towards certificate pinning have been noted in Sect. 3.2. In (our usage of) identification the client needs a local name for the host to connect to. Identification does not need trusted third parties.

For hosts not known to the client, a service is needed that equips the client with the means to authenticate the host. We call this an *introduction service*. The TLS PKI is such a service. Host names have to be globally unique. The introducer acts as a trusted third party. Asking several independent parties reduces the impact of a compromised introducer.

[9] http://tools.ietf.org/html/draft-nikander-ram-ilse-00

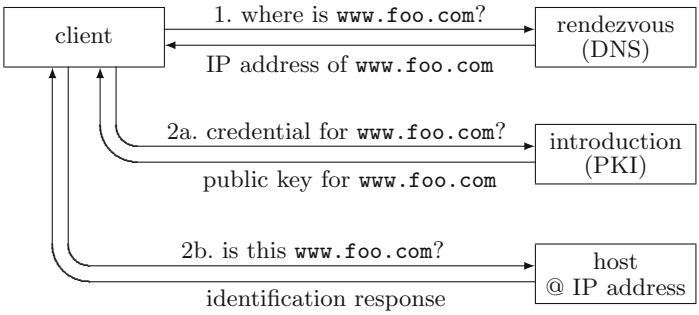

Fig. 1. Interplay of rendezvous, introduction, and identification services when connecting to a host.

Figure 1 describes how a client connects to www.foo.com. Step 1 retrieves an untrusted IP address for the host. An optional step 2a gets a credential from an introductions service; in the case of a PKI the credential is a public key. Step 2b is a run of an identification protocol run between the client and the host at the IP address obtained in step 1. The response is verified using either the credential from step 2 or a credential pre-installed at the client.

5 Conclusion

Infrastructures are critical because of critical applications using the infrastructure. In the first instance, it is not the critical infrastructure that needs to be secured but the applications that had turned the infrastructure critical. This in turn may point to security services the infrastructure should provide. The fewer parties these services have to trust the better.

We may be taking a wrong turn when we ask DNS to provide additional services and rely on cryptography for securing DNS. Authentication does not protect against lying insiders, i.e., against the very entities providing these services. Relying on DNS introduces an awful lot of trusted third parties, albeit with certain limits to the damage they can cause. Arguably, this does not secure the infrastructure but increases the attack surface.

DNS can be viewed as a rendezvous service returning the current IP address of a host; it needs an infrastructure of name servers such as the one we have got today, but it should not be necessary to trust this infrastructure. Certificate pinning is adding an identification service to TLS. Identification needs no trusted third party at all. Introduction services are, by definition, trusted third parties. The principle of *divide et impera* suggests that there may be benefits in splitting rendezvous from introduction services.

Acknowledgements. The author thanks Daniel Thomas for a constructive criticism of this paper.

References

1. Aura, T., Roe, M., Arkko, J.: Security of internet location management. In: Proceedings of the 18th Annual Computer Security Applications Conference, pp. 78–87, December 2002
2. Day, K.: Rapid DNS poisoning in djbdns, February 2009. http://www.your.org/dnscache/djbdns.pdf. Accessed 5 June 2014
3. Dean, D., Felten, E.W., Wallach, D.S.: Java security: from HotJava to Netscape and beyond. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, pp. 190–200 (1996)
4. Jackson, C., Barth, A., Bortz, A., Shao, W., Boneh, D.: Protecting browsers from DNS rebinding attacks. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 421–431 (2007)
5. Johns, M.: (Somewhat) breaking the same-origin policy by undermining DNS pinning. Posting to the Bug Traq mailing list, August 2006. http://www.securityfocus.com/archive/107/443429/30/180/threaded. Accessed 5 June 2014
6. Schuba, C.: Addressing weaknesses in the domain name system protocol. Ph.D. thesis, Purdue University (1993)