Reducing the X.509 Attack Surface with DNSSEC's DANE

Eric Osterweil
Verisign Labs
eosterweil@verisign.com

Burt Kaliski
Verisign Labs
bkaliski@verisign.com

Matt Larson
Verisign Labs
mlarson@verisign.com

Danny McPherson
Verisign Labs
dmcpherson@verisign.com

***Abstract*—For the last decade, perhaps the most commonly used type of end-user security has been the HTTP Secure (HTTPS) protocol employed by web browsers (which runs over the Secure Sockets Layer, SSL or its successor, TLS). In HTTPS, any service (such as a website) may create its own cryptographic certificate to secure its communication channel, and clients use this certificate to verify data from, and transmit data to the server. This model has helped to secure online banking transactions, eCommerce websites, social networking websites, and more. However, two inherent complications to this approach are that clients must have a secure way to *learn* the authentic certificate for each website before they begin using this protocol, and they must be able to determine if they can *trust* the named entity that the certificate belongs to. These complications are conflated in today's security model, which is based on a list of prespecified trusted X.509 Certificate Authorities (CAs) that every client must know a priori, and a very ad-hoc approach to determining which of this list of CAs will vouch for any keys discovered.**

In this paper we first outline some of the fundamental problems that exist with today's CA model, problems that arise from its conflation of the two inherent complications, and some of the implications and attack vectors that these problems present to the security of this model's users. Then we introduce some of the relative benefits that can be gained from a new approach being standardized in the IETF called DNS-based Authentication of Named Entities (DANE), in which certificate credentials are verified by DNSSEC-enabled zones, rather than the CA model used today. We illustrate that the DNSSEC-verification model reduces the attack surface that users currently inherit, and show that this model opens avenues that have previously remained elusive (such as a usable S/MIME verification infrastructure).

I. Introduction

For over a decade now, Internet users have needed a way to securely visit data sources (websites, mail servers, etc.). However, busy data sources like these cannot, for example, proactively disseminate (or push) their crypto certificates to all of their users (how could websites like Google even know exactly who will visit them ahead of time). Moreover, sending a crypto key over the same channel that may be subject to a Man in the Middle (MitM) attack, would make in-line key learning vulnerable too. Yet, users need some way to "securely" learn crypto certificates for all the sources they will ever visit (even though they do not know who they may visit ahead of time), and determine if the named entities that hold those certificates are "trustworthy" (that is, a named entity may still act maliciously even if they have a valid certificate). Moreover, services like websites, email servers, etc. may want to periodically change their crypto certificates, so learning certificates is a moving target for clients.

Today, we rely on certain organizations called Certificate Authorities (CAs) to perform these tasks for us. That is, a relatively few organizations (compared to the millions of delegated organizations) who are trusted to vouch for both the *authenticity* and *"trustworthiness"* of *all* other organizations' (i.e. the named entities) certificates. Each of these CAs is associated with a certificate of its own, but unlike Google's certificate, all clients in the Internet are presumed to have securely learned the certificates of a list of "well known" CAs. Some of the details of this approach are discussed in [16]. Generally, large software vendors (like Mozilla Foundation, Apple, Microsoft, etc) configure the X.509 certificates for roughly 160 CAs for users when they are installed. These certificates are then treated as ground-truth for the trustworthiness of learned certificates by software ranging from web browsers, to mail readers, etc. In this model, software vendors have assumed the responsibility to choose and maintain CA lists, and these CAs are responsible for verifying all of the data sources a user may ever visit on the Internet. However, it has become increasingly well known that the CA model can present adversaries with several attack vectors that are both easy to implement and/or hard to detect. We discuss these in more detail, and give specific examples of known instances of compromise in Section II-A. One major liability of this approach is that after the mandatory DNS lookup that precedes certificate verification, there is no unambiguous and transparent way for service operators to inform Relying Parties (RPs) of how their certificate should be verified and trusted. That is, the DNS provides a clean way to unambiguously map a domain name to an Internet resource (generally an IP address), but SSL/TLS does not have equivalent semantics to allow the named entity who was looked up in the DNS to unambiguously inform RPs of the verification process. Rather, any certificate received over the TCP session is treated as the starting point for authentication. Therefore, if a service operator uses $CA1$ to verify their certificates, but a Man in the Middle (MitM) adversary inserts a falsified certificate from $CA2$, there is no way for an RP to detect this (specifically *because* the cryptographic verification will succeeded for the adversary's certificate). The term *named entity* is sometimes used when logically associating a domain name to the authoritative entity that controls it.

Recently, however, the DNS Security Extensions (DNSSEC) [6], [8], [7] have become an operationally relevant technology, and their deployment has been growing steadily for over six years [5]. This has opened the door for domain owners to explicitly manage their security in the same distributed database that users trust to find their service: DNS. Using DNSSEC to attest to TLS keys (for HTTPS and other TLS-based protocols) has already begun in the IETF's DNS-based Authentication of Named Entities (DANE) working group [1]. Moreover, some commercial products, such as Google's Chrome [10] have deployed native support for this and add-ons exist for browsers such as FireFox [3]. Perhaps one of the most fundamental observations to make about this approach is that Internet clients that act as RPs are already captive to DNS (for example, URLs are rarely formed with IP addresses). Thus, enabling certificate authentication within the same infrastructure that is already required for online transactions reduces the system's dependencies on additional systems and protocols, and thus reduces the overall attack surface.

Based on observations that more could be possible with the general approach of verifying certificate authenticity using the now-secure DNS, work has begun towards addressing the needs to make S/MIME's global deployment a possibility [9]. With an approach like the one DANE is advocating for TLS, users' will be able to secure their email inboxes (inbound and outbound) by configuring and learning crypto keys from DNS. In fact, this has been discussed for over 13 years [4], and while approaches like PGP's web of trust [12] have been attempted to bridge this gap through user-based multilateration, they have failed to gain significant traction and provide convincing security assurances. Recently, other work [19] has begun to investigate some alternative ways to do this.

Many well designed security protocols have stagnated in the Internet's operational setting because there has not been a globally secure/operationally deployed system that allows for Internet-scale cryptographic key learning and verification. However, with the operational deployment of DNSSEC, a new climate is emerging in Internet operations.

The rest of this paper is organized as follows: First, Section II details how the current CA model works. Then, Section III describes the DANE model and compares its security assurances against the current CA model's. Finally, we conclude with a discussion of future directions in Section IV.

II. The CA / X.509 Landscape

Without X.509 and certificate authorities, Internet users are at risk from having their TCP connections (HTTP, SMTP, etc) hijacked. Figure 1 shows how even if DNS is secured with DNSSEC, hijacking a TCP session can be quite straightforward over a narrowly focused attack surface.

CAs have "secured" various channels for us for over a decade. Every time a user's web browser contacts an HTTPS website, every time their mail client (the Mail User Agent, MUA) connects to their secure mail server, every time their operating system checks for (or authenticates) new software packages their computer checks the requested data service's certificate against a list of CAs configured on their machine. In this model, the notion of authenticity and trustworthiness are conflated into a single cryptographic verification.

The specific process is surprisingly flexible: a user's operating system, or a software provider (such as Mozilla) maintains a list of globally "trustable" CAs on every user's computer.[1] Thus, whenever there is a software update of (say) FireFox, or OS X, a current list of CAs are pushed to a user's computer. This allows software packages such as Thunderbird (an MUA), FireFox, Opera, etc. to verify the authenticity of X.509 certificates that are learned when users connect to services they encounter online, and infer the trustworthiness of these certificates. For example, when a user clicks on an HTTPS link they find in a Google search page, their browser receives an X.509 certificate from the remote web server. Their browser then performs a number of sanity checks (that include date checks, possibly OCSP/CRL checks, other blacklisting checks, possibly a usage type check, etc.), then looks in that certificate to see what CA it claims to be verified by and then looks in its own local CA list to see if it has that certificate. If so, then the CA's certificate is used to verify the signature that the remote web server's certificate claims belongs to that certificate. If there is a match, then the certificate can be used to secure the communications between the two parties.

This model has several notable benefits. Perhaps the clearest is the fact that any data source in the Internet (web servers, mail servers, etc) can choose any of the most well known CAs to verify them. For example, when a web site wants to deploy HTTPS, it can create its own Certificate Signing Request (CSR), pass this to a CA of their choice (say PiousCA), then take the returned certificate (with a verifying signature from PiousCA) and configure it to be served from their webservers. Thus, any web browser that has the PiousCA CA certificate configured can verify the TLS transaction, and that transaction verifies the web server's content. This model allows services to determine whom they would like to trust, and allows clients to have flexibility in when learning keys from a diverse set of services. If an attacker launches a Man-in-the-Middle (MitM) attack against a client, they must provide a certificate that was issued by a valid CA and which reports to belong to the service in question, or it will be rejected by the client.

[1] It is common for the list of CAs to be approximately 160 entries in size, and there is no definitive source that prescribes what CAs should be in the lists managed by different parties. Thus, different software may have different lists of CAs

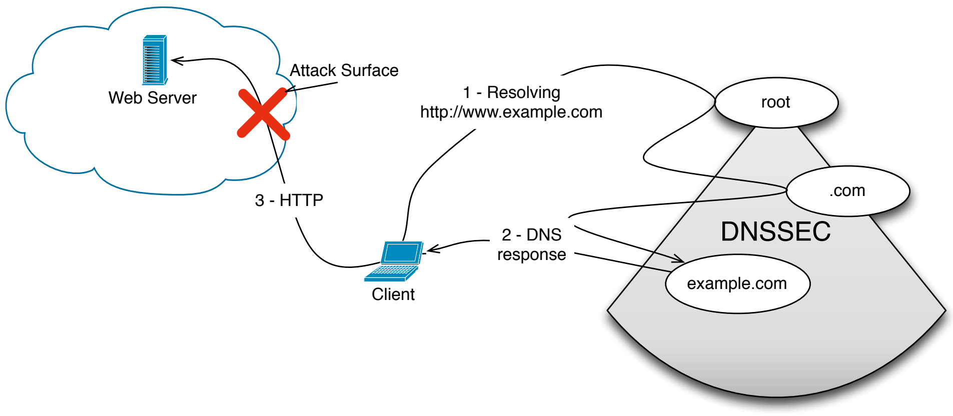

Fig. 1. The vulnerability of an HTTP session.

While it is easy for data services and clients to use this certificate learning system, is *easy* really the right metric to judge its success? Moreover, if any CA can vouch for any service at any time (even if the service owner doesn't want it or doesn't know about it), how much trustworthiness is actually imparted during the authenticity verification and how much operational liability already exists in the current CA deployment?

A. Liabilities: an Attack on One Defeats All

The converse of the CA model's easy-to-use design is that there is a correspondingly easy way to defeat it. Since any data source can be vetted by any CA, then if any CA is hacked, then all data sources are vulnerable. There have been several actual cases where this has happened [15], [20], [22], and one might reason that if this can happen to such high-profile zones (Google, Skype, Yahoo, etc.) then the same could happen for any zone, and may not even be widely reported. In addition to the outright issuance of fraudulent certificates, some CAs have been tricked into issuing certificates for names that simply *appear* to be valid for sites, but technically aren't. For example, in 2009, a Black Hat presentation [14] demonstrated the ability of a miscreant to use the "NULL byte hack" to use certificates that really vouch for one domain to trick browsers into applying them to a subdomain.[2]

The fact that any data source can choose whom they would like to be verified by means that any CA can authenticate a certificate for any data source they want. That is, the flexibility of whom a source trusts translates to the inability of a client to judge if a CA is *authorized* to vouch for a service. Consider the following (one of many possible attack vectors), suppose subzone.example.com runs an HTTPS server, and chooses to be certified by PiousCA (a fictional CA). They get a certificate from PiousCA, with a signature in it that can only be verified by PiousCA's CA certificate. The administrators of https://www.subzone.example.com/ then configure their webserver to use this (their new) certificate. However, an adversary can get the CA from (say) a compromised CA (such as one from a country whose political regime has recently been overthrown and whose climate may allow politics to interfere with Internet operations). This compromised CA (let's call it `.compromised`) can *also* issue a certificate for subzone.example.com. Thus, if an attacker launches a MitM attack with a certificate from `.compromised`, the client will not know it is any less valid than one from PiousCA, or any other CA.

In order to properly address the liabilities, we begin with the premise that the aggregate *attack surface* can be defined as the union of all attack vectors, and that the "size" of this surface roughly describes the vulnerability of the security model.

B. Attack Surface

The sum of all attack vectors, or the aggregate *attack surface* for the CA model:

- The CAs themselves are a vector. Each CA that a client has configured is an element that can be attacked (as seen in the example above). This means that if the average user has about 150 CAs configured, the corresponding attack surface is augmented once for

[2]The description of this hack is beyond the scope of this document, but is fully described by the citation.

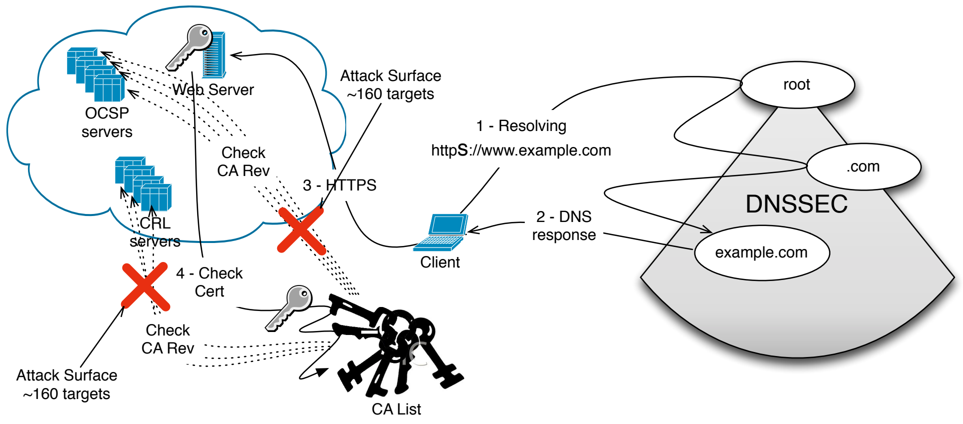

Fig. 2. Attack surface using CAs.

each. The most vulnerable is simply the weakest link, but any CA can allow for arbitrary certificate spoofing. Moreover, since the certificate delivery is point to point between a client and a server, it is not easy to detect when a MitM attack is underway. Bogus certificates can be delivered selectively to victims to avoid detection. This component of the attack surface is $O(n)$, where n is the number of CAs on a client. We note that many CAs employ subordinate CAs, which are entities who are allowed to sign on behalf of a given CA. While this clearly increases the attack surface, we omit this for now, and just estimate the predictable size of this component of the overall attack surface.

- Any Online Certificate Status Protocol (OCSP) servers, or Certificate Revocation List (CRL) servers. These two services are designed to allow clients to check if a given certificate has been revoked. For example, if a certificate owner knows that their certificate has been compromised, they may want to issue a revocation so that clients won't use it any more. For clients to find this information they must use a service like OCSP, or a CRL. However, relying on these services means that they can be attacked in order to keep a client from learning of a revoked certificate (thus allowing a miscreant to continue using it). This component of the attack surface is $O(m+p)$, where m is the number of OCSP servers in *all* certificates a client encounters and p is the number of CRL servers a client encounters. This can effectively double the attack surface from just the CA list alone.
- The DNS zones that host OCSP services, the CRL servers, or even just the target domain's zone itself. This part of the attack surface may not be as obvious as others, but it is just as real. A zone's attack surface can be loosely derived from a notion that is often called *transitive trust* (first discussed in [18], and more recently elaborated on in greater detail in [17]), which relates to the opportunity cost of an attacker that aims to disrupt of hijack a DNS zone. In this case, such an attack would allow a miscreant to launch a DoS attack against service to OCSP or CRL servers by attacking DNS nameservers (instead of the data servers themselves). We generally quantify this as $O(|NS|)$, where $|NS|$ represents the number of name servers involved in serving the entire predecessor graph of a zone. An example of this is shown in Figure 4.

Figure 2 also shows a view of this attack surface.

III. DANE AS AN ALTERNATIVE

Now that DNSSEC has been deployed at the DNS root, several of the largest Top Level Domains (TLDs), such as `.com`, `.net`, `.se`, etc., and the global growth has begun to to gain visible traction (as seen in Figure 3, taken from [5]), zone owners are beginning to be faced with both economic incentives to deploy, and operational DNSSEC tools whose maturity is making the cost of deploying DNSSEC more realistic.

Under these operational conditions, operators, the IETF, and various other groups have begun considering the possibility of replacing the aging CA certification model with one based on the newly secured DNS. Most notably, the IETF has formed a working group called DNS-based Authenticated Named Entities (DANE) [1]. In this group, new Internet drafts have already begun the publication cycle towards RFC standards documents. In particular, the working groups first focus has been to specify *what* must be placed in DNS, in order to transfer trust to a TLS certificate. At the time of this writing, the working group has begun standardizing on a DNS resource record called the `TLSA` record and specifying its semantics. The basic role of this record is to exist in a DNSSEC signed zone, and to indicate the certificate information that corresponds to a specific service on a specific port of a name in that zone. Thus, when an administrator of (say) www.example.com wants to run TLS on port 443, he/she can place a `TLSA` record at _443._tcp.www.example.com so that browsers know that there is a service on that port and what the certificate should be. One of the subtle, but important, differences in this model is that it is attempting to verify the authenticity of certificates, but does not speak to the trustworthiness of the named entities themselves. Thus, one can know that the certificate being used is authentic for a domain name, but cannot necessarily use this knowledge to determine if that domain is (for example) a phishing site. That is, DANE properly decouples these two goals.

A. New Benefits

When clients (such as web browsers) lookup named entities (such as domain names like www.paypal.com), they currently query the site's DNS zone for the IP address of its server(s), then query the servers for the certificate, then use their CA list to verify the certificate (locally). By contrast, using the DANE protocol, a browser would query the DNS for the IP address and the `TLSA` record. Then connect to the webserver and see if the returned certificate matched what it had *already learned* and *verified* from the DNS. Any MitM attack would be thwarted before it even started. As a result, each named entity (i.e. a DNS domain name) can "staple" itself to either a key (placed in its zone), or to a specific CA (if the `TLSA` certificate explicitly allows for it). This protocol makes the verification process transparent to clients. When they lookup information about the zone, they can explicitly see the certificate/CA that a named entity.

In addition to the the `TLSA` record, the DANE group has also discussed the general ease with which similar work could be done to standardize on a verification mechanism for S/MIME (a protocol for "securing" email communications between arbitrary parties). Currently, some users use PGP keys to protect their emails. In order to verify the PGP keys being used, they turn to PGP's Web of Trust [12]. For many years, researchers have tried to turn the social multilateration of the Web of Trust into a trustworthy substrate, but now that DNSSEC has secured the same name space that email addresses rely on for service resolution, the DANE work can be applied to this need as well.

In any of these cases, one of the most compelling observations is that a named entity is able to *explicitly* indicate its verification information to its clients. Rather than trusting the two-party matching protocol of the CA model, the owners state their trust assertions to their clients in a clear and transparent protocol.

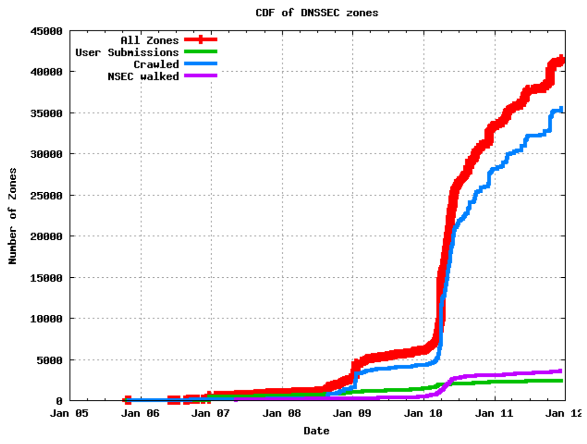

Fig. 3. The size of the global growth of DNSSEC, as seen by SecSpider.

B. Protecting DNS' Model

Many of the benefits described above are really inherent in DNS' core model. Thus, changes to the way DNS is operated, or its core approach to universal resolvability has the potential to undermine its protections. DNS was designed to be a globally unique name space served on top of a globally available database. This has allowed websites to brand themselves, users to trust named entities to deliver content from *specific* authorities, and more. This global name space and its universal resolvability have formed a substrate on which even the CAs (let alone DANE) have built a verifiable trust model. Clearly, without these properties, many of the security assurances that have been built and are being designed would suffer. Yet, some groups have proposed that resolution of DNS and resolvability of DNS' domain names should be subject to external legislation, or filtering techniques that are built into the DNS' control plane itself [21]. Recently, some Internet researchers approached the US legislative body that has been contemplating this form of legislation and issued a statement against such action [2].

In addition to the implications of prescribing resolution policy through legislative bodies, DNS' control-plane itself is an important consideration when adding additional layers of trust. In recent work [17] we outlined the need for zone owners to understand how much transitive-trust they may inadvertently be incurring through the deployment and sharing of DNS secondary servers. Figure 4 shows that the DNS resolution of the DNSSEC-enabled Top Level Domain (TLD), `.bg` can depend on querying a great many secondaries for information. By contrast, the more conscientious deployment of starbucks.com's zone (as seen in Figure 5) will generally require far fewer secondary

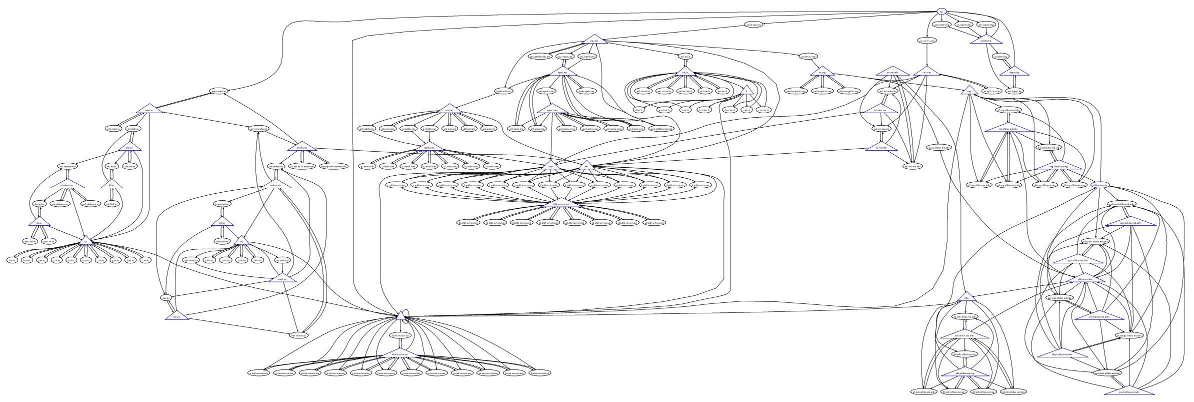

Fig. 4. The control plane dependency graph of `.bg`

servers to be trusted (and in this particular case, they all reside within a single administrative domain).

C. Liabilities in this Direction

The value proposition for the DANE work is relatively clear: since DNS zones are the starting point for TLS transactions in the Internet, and since DNSSEC provides origin authenticity, data integrity, and secure denial of existence, then verifying the TLS handshake from DNSSEC-protected data allows clients to transit the same trust they have in the origin zone to the webserver's authenticity. While the current CA model allows any CA that a client has configured to create a security hole, the new approach does not address some of the services that the current CA model does. Perhaps the most notable element that is missing from the DANE approach is the policy framework of the CA model. Currently, when many CAs make attestations, they take actual "responsibility" for them. For instance, some CAs will issue Extended Validation (or EV) certificates to zones. These certificates carry with them the implication that the issuing party is taking (in many cases, a legal) responsibility for the authenticity of the mapping from a certificate to the prescribed service. Thus, the implication is that if the CA is in err, they are liable. Some feel this adds more security to the CA model, and others claim that this new approach fundamentally decouples a conflated set of issues in the CA model and that this enables more utility in security policy expression.

As the DANE approach relies heavily on DNSSEC to verify data, it also ushers in another implicit need: DNSSEC validators. When an authoritative zone deploys DNSSEC, it creates signatures to verify its data. For example, when a user queries for www.verisigninc.com, the answer is returned with a digital signature. However, if the DNS resolver that is issuing the query does not check that signature, then there is no increase in security. ISPs and enterprises have begun rolling out these services, but our end-user devices are becoming increasingly *less* tied to specific DNS resolution infrastructure. For example, an iPad, or a cell phone may be acting on a wireless network, or a home cable network, or a coffee house's network. In each of these cases, the DNS resolver being queried may, or may not, have DNSSEC validation enabled.

Another liability in this direction is one in which people try to deliver DNSSEC verification over a non-DNS medium. In part, this discussion has arisen in the IETF [13] while the protocol is in its nascent stages. Thus, this treatise may not reflect a long-lived concern. Nonetheless, some have contended that a DNSSEC enabled zone should be able to serve certificate credentials for a service, or staple itself to a CA, but that this information should be encoded in the certificate itself (rather than taken from DNS). Though there are several operational justifications that proponents espouse, this approach opens an attack vector up against the otherwise immune DANE protocol. If verification of a certificate is taken from that certificate instead of validated off-axis in the DNS, then an adversary can replay a certificate that has been revoked in the DNS zone. That is, suppose a zone vouches for a certificate in its `TLSA` record at t_0. Then, at time t_1, the zone operator learns that the certificate has been compromised and is being used to forge data. The operator can then replace the `TLSA` record with one pointing to a new certificate at time t_2, and clients will automatically learn the new value from then on. By contrast, putting the DNSSEC verification information in a certificate would mean that an adversary can continue to replay the compromised certificate with impunity any time after t_0, regardless of when the zone is updated.

D. DANE's Attack Surface

To begin, let's consider a web server running HTTPs at the URL www.subzone.example.com. Further, we simplify our attack scenario by assuming that a zone operator has indicated an End Entity (EE) certificate as being authoritative for their web server. That is, initially this zone does not staple itself to a CA, but uniquely identifies the certificate it will use.

In order for an adversary (Eve) to attack this name, she can attack the name servers for the zone subzone.example.com, or any zone in the predecessor list (bar.com, .com, or

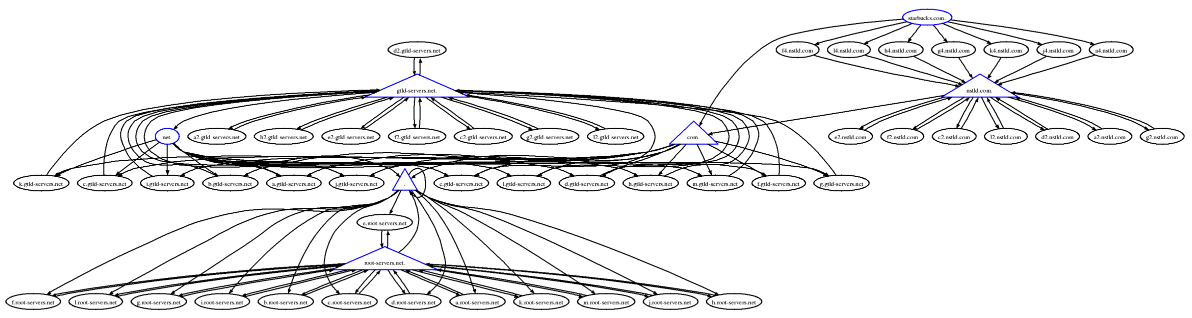

Fig. 5. The control plane dependency graph of `starbucks.com`.

the root zone). However, if each of these zones is DNSSEC enabled, she can (at best) launch a DoS attack against them. This would not allow her to spoof data, and would be globally detectable.

Thus, the attack surface of this approach (which doesn't need OCSP, CRLs, or a list of CAs) is just $O(|NS|)$, where $|NS|$ is the number of name servers in the predecessor list (inclusive of the target zone). In fact, one could argue that if the DNSSEC zone's signing keys are kept offline, an adversary cannot compromise the integrity of the data on any single name server, and must compromise the organization's key. This could reduce $O(|NS|)$ to $O(|Inst|)$, where $Inst$ is the set of institutions running zones, and $|Inst| \leq |NS|$. However, we conservatively assume $O(|NS|)$ here. Because verification is based on the presence of a verifying `TLSA` record in a zone, revocation is implicit when it is removed. This obviates the zone from needing any revocation infrastructure. This effectively reduces the attack surface (from the CA model's) from a derivative of the roughly 150 CAs to just the predecessor graph (inclusive of the target zone). Figure 6 illustrates the reduction in attack surface.

E. A Usable S/MIME Infrastructure

As a globally unique and highly available name-space, DNS has allowed email addresses to become a form of unique identifiers. Users already rely on the fact that given an email address, any user can send messages to any other without knowing anything about them and that messages can be sent to a single unique place. However, this does not ensure that they will be delivered, be unaltered, or that they will not be read by third parties. To address this concern, S/MIME was created to sign and optionally encrypt email. However, as a cryptographic protocol, S/MIME suffers from the same problem that other Internet-scale security protocols suffer: how can arbitrary parties in the Internet securely learn and verify each others' cryptographic keys?

Just as with TLS, now DNSSEC can be leveraged to store S/MIME certificates for users. Just as the DNS named entity already directs mail to specific servers for the mailboxes that belong to it, it can now use the same authority to direct clients to the S/MIME certificate for each mailbox. This would overload the same 1-to-1 mapping that exists between the authority for a DNS domain (i.e. the address) and the verification mechanism for that address.

IV. Future Directions

The increased operational deployment and feasibility of DNSSEC will surely be considered a watershed event in the history of operational Internet security. Having designed Internet applications and protocols to use DNS names for service location, we are now able to use the same mechanism to resolve and securely authenticate system credentials. The deployment of DNSSEC has made this possible, and with the increased incidence of CA compromise, and the increasing need for security in our everyday online activities, it could not have happened soon enough.

In this paper we discuss a number of the motivations for using the DNS to bootstrap trust in other systems, and we outlined the need to treat DNS as a reliable service, rather than a policy enforcing framework. Researchers, protocol engineers, and operators alike have long awaited the arrival of an Internet-scale crypto key learning and verification system. It is important that now, as we are on the threshold of realizing this new service, we do not undermine its very foundations [2], and that we conscientiously attend to its own security properties [17] so that this promise becomes an operational reality.

One of the key outstanding issues is that some services may require both authenticity verification (as DANE proposes to enable), *and* the additional trustworthiness checks that were attempted by the CA model. While DANE offers a design that restores transparency and enables data owners to expose their policy preferences to RPs, a similar model is needed for those systems that want both. Currently, DANE allows data owners to serve certificates that chain back to *specific* (stapled) CAs. This is, perhaps, best viewed as a stepping stone on the way to a system that (like DANE) provides more transparency and embraces the RPs' needs directly. In the future, technologies may arise that allow name resolution to include more than just a domain name, but perhaps the entire URI and meta-data [11]. In such cases the DANE model would likely become even more influential as its applicability would reach more applications. That being said, DANE has become a relevant protocol *today*, and demonstrates that the promise of using DNSSEC

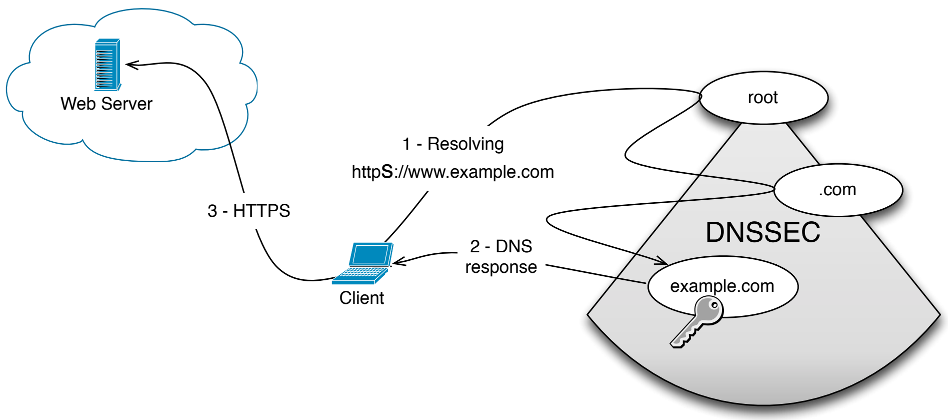

Fig. 6. Attack surface of DANE

to verify our online transactions is finally here. Moreover, the appeal that DANE likely holds to end users, and its reliance on DNSSEC could very likely lead to a deployment symbiosis whereby the benefits of each, stimulates operators to deploy them both.

ACKNOWLEDGMENTS

We would like to thank Joe St Sauver and David Dagon for their most helpful suggestions and feedback.

REFERENCES

[1] DANE. https://datatracker.ietf.org/wg/dane/charter/.

[2] Experts Urge Congress to Reject DNS Filtering from PROTECT IP Act, Serious Technical Concerns Raised. http://www.circleid.com/posts/20110525_experts_urge_congress_to_reject_proposed_dns_filtering_protect_ip/.

[3] Extended dnssec validator. https://os3sec.org/.

[4] RSA Data Security and Leading Vendors to Merge S/MIME Messaging Standard With U.S. Government's MSP Message Security Protocol. http://www.rsa.com/press_release.aspx?id=718.

[5] SecSpider. http://secspider.cs.ucla.edu/.

[6] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirement. RFC 4033, March 2005.

[7] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035, March 2005.

[8] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034, March 2005.

[9] P. Hoffman and J Schlyter. Using secure dns to associate certificates with domain names for s/mime - draft-hoffman-dane-smime. Technical report.

[10] DNSSEC Deployment Initiative. Dnssec certificates stable in chrome. https://www.dnssec-deployment.org/index.php/2011/09/dnssec-certificates-stable-in-chrome/.

[11] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *CoNEXT '09: Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12, New York, NY, USA, 2009. ACM.

[12] Rohit Khare and Adam Rifkin. Weaving a web of trust. *World Wide Web J.*, 2(3):77–112, 1997.

[13] DANE Mailing list. Draft for serializing DNSSEC chains. http://www.ietf.org/mail-archive/web/dane/current/msg02796.html.

[14] Moxie Marlinspike. Breaking SSL with null characters. *Black Hat*, 2009.

[15] Eddy Nigg. Untrusted Certificates, 2008. https://blog.startcom.org/?p=145.

[16] Eric Osterweil, Dan Massey, and Lixia Zhang. Managing trusted keys in internet-scale systems. In *The First Workshop on Trust and Security in the Future Internet (FIST'09)*, 2009.

[17] Eric Osterweil, Danny McPherson, and Lixia Zhang. Operational implications of the dns control plane. *IEEE Reliability Society Newsletter*, May 2011.

[18] Venugopalan Ramasubramanian and Emin Gün Sirer. Perils of transitive trust in the domain name system. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pages 35–35, Berkeley, CA, USA, 2005. USENIX Association.

[19] Joe St Sauve. S/mime simple public key server. http://pages.uoregon.edu/joe/simple-keyserver/keyserver-documentation.docx.

[20] Mike Wood Naked Security. Fraudulent certificates issued by Comodo, is it time to rethink who we trust?, 2011. http://nakedsecurity.sophos.com/2011/03/24/fraudulent-certificates-issued-by-comodo-is-it-time-to-rethink-who-we-trust/.

[21] Paul Vixie. Taking Back the DNS. http://www.isc.org/community/blog/201007/taking-back-dns-0.

[22] Warwick Ashford Computer Weekly. DigiNotar SSL certificate compromise widens to include security agencies, 2011. http://www.computerweekly.com/Articles/2011/09/05/247792/DigiNotar-SSL-certificate-compromise-widens-to-include-security.htm.