Modeling Systemic Dependencies Through Attack Surface Analysis

Eric Osterweil

Danny McPherson

Lixia Zhang

More moving parts?

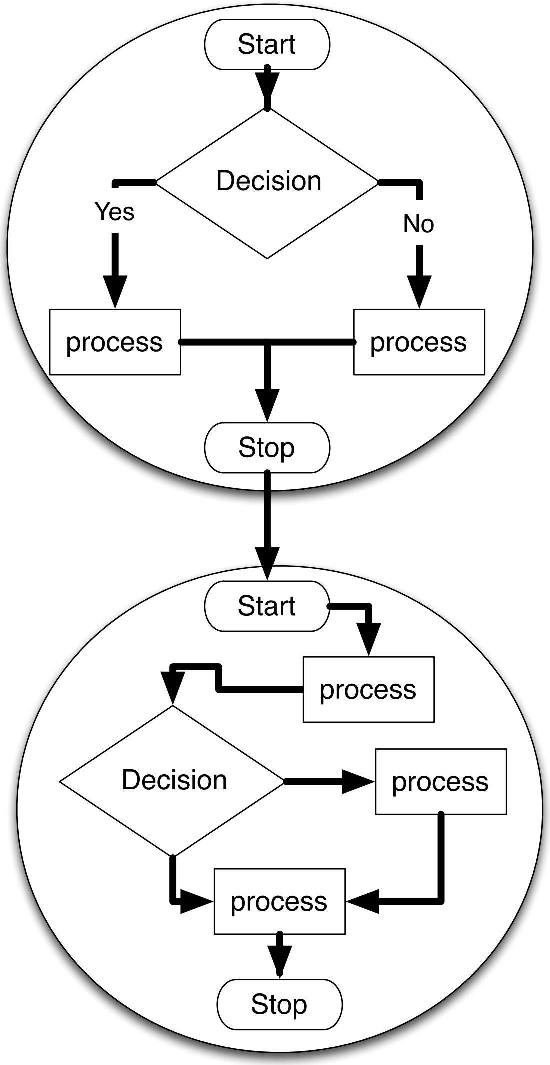
- Should we *measure* how many moving parts our systems need/use?
- We can see examples of systems that are designed to protect us that have varying degrees of systemic dependencies
  - Does not mean their bad, but we should be able to understand what they depend on
- Some systems offer strong assurances, but
  - What do they depend on
  - How large are their systemic dependencies?
  - Will our mouse traps fit in our houses?

Attack Surface Analysis

- We designed a methodology to quantify the systemic dependencies and used them to illustrate attack surface
- We measured DANE vs. CA verification
  - Existing standards
- We are also considering where else this work might apply

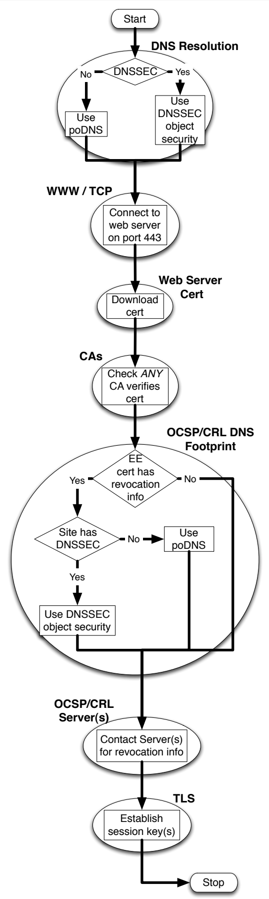
How our methodology works

- We have to quantify how systems depend on each other
- To do this, we create a Functional Process Digraph (FPD)
- Use that FPD to identify the elements used by the logical processes
- Measure the size of the set of these elements

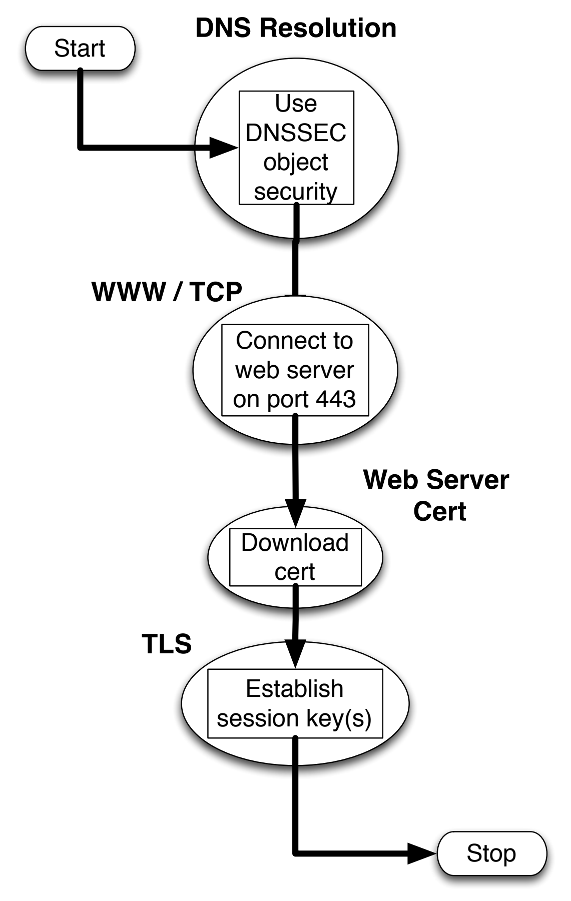


Functional Process Digraphs (FPDs)

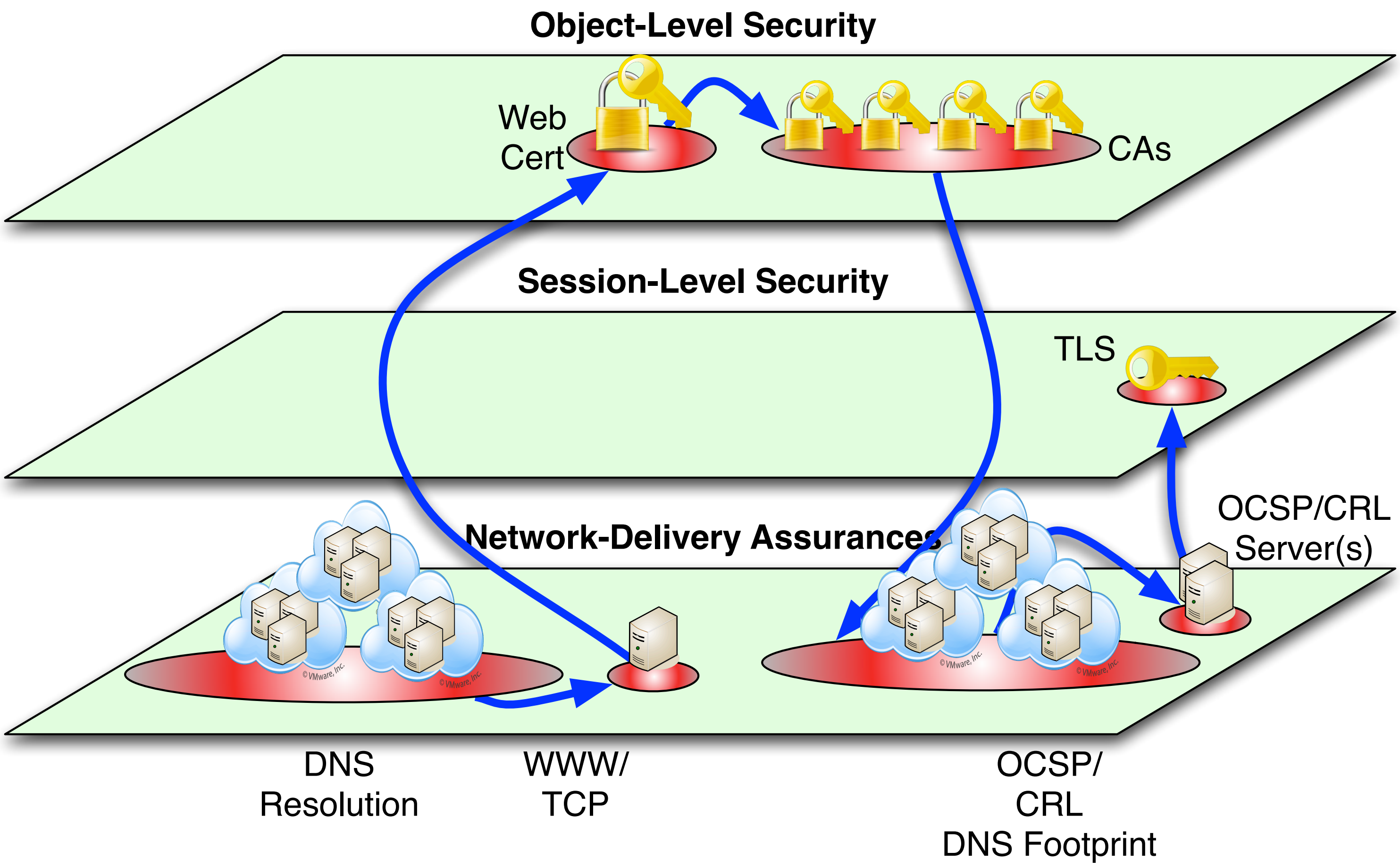
CA Verification



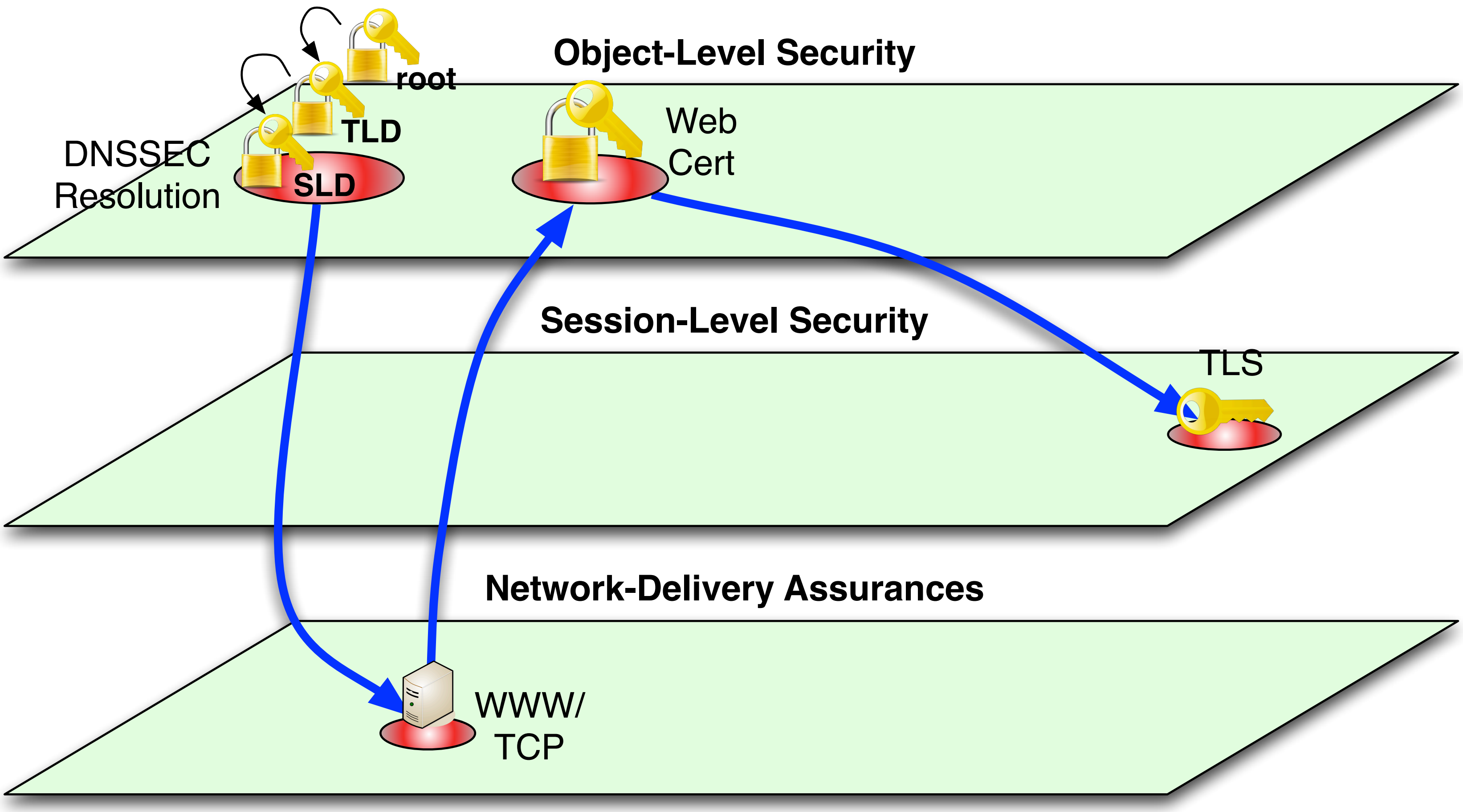
DANE



CA Verification's Attack Surface

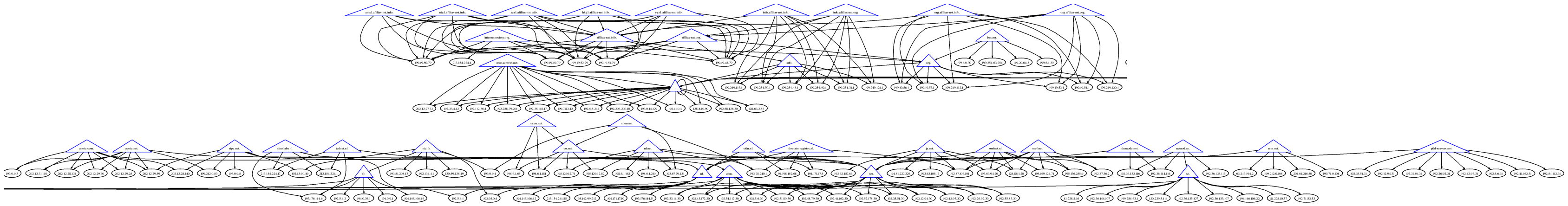


DANE’s Attack Surface

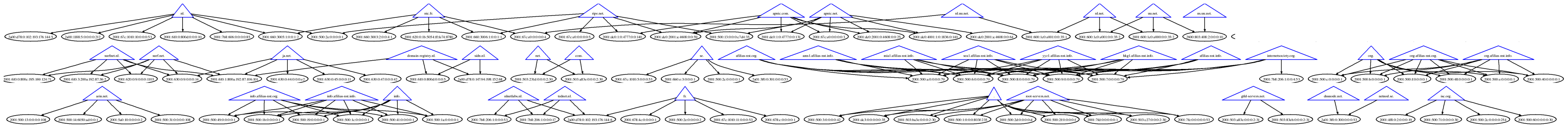


How big is this?
http://trans-trust.verisignlabs.com/

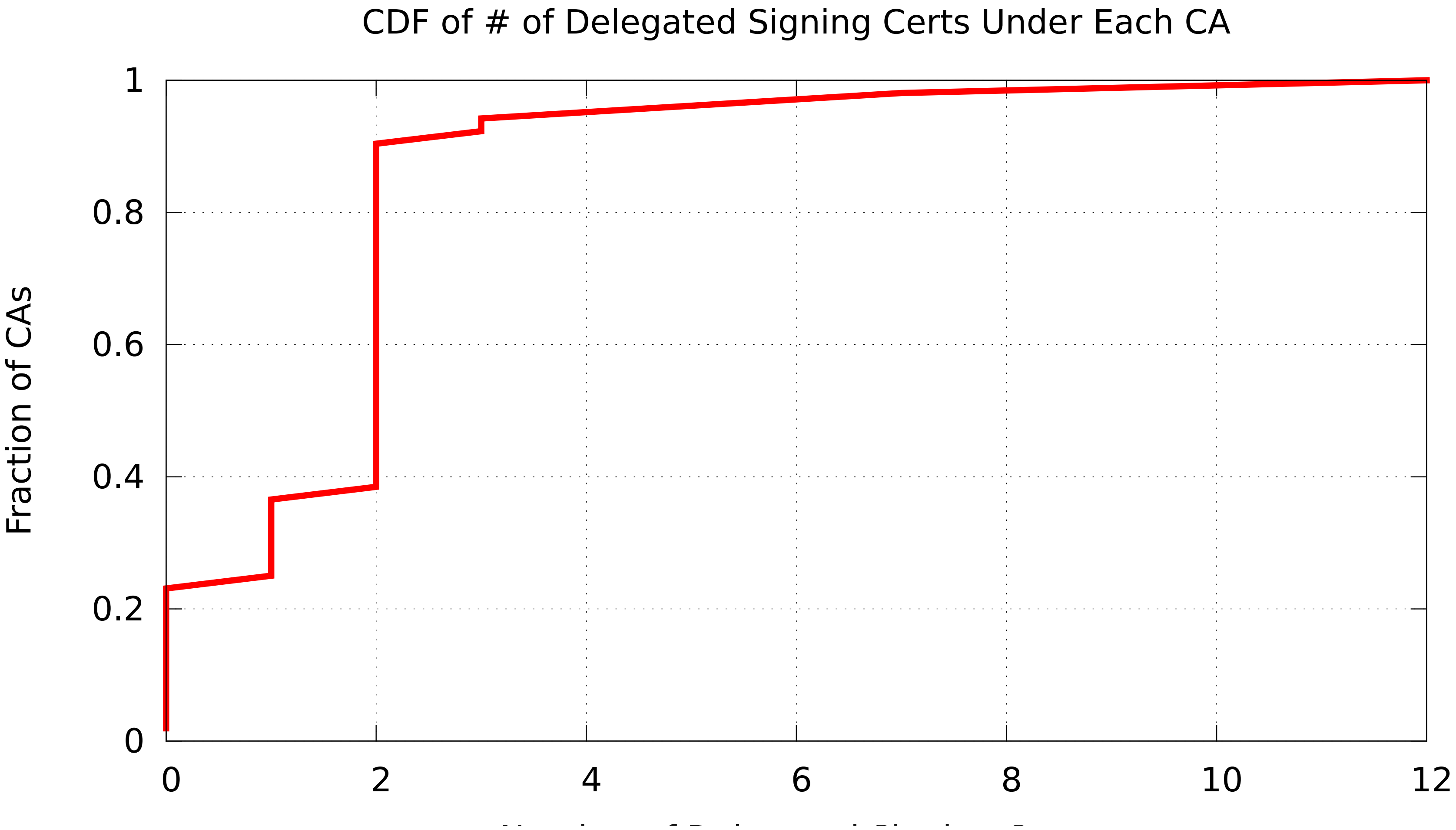
- Just consider the DNS portion



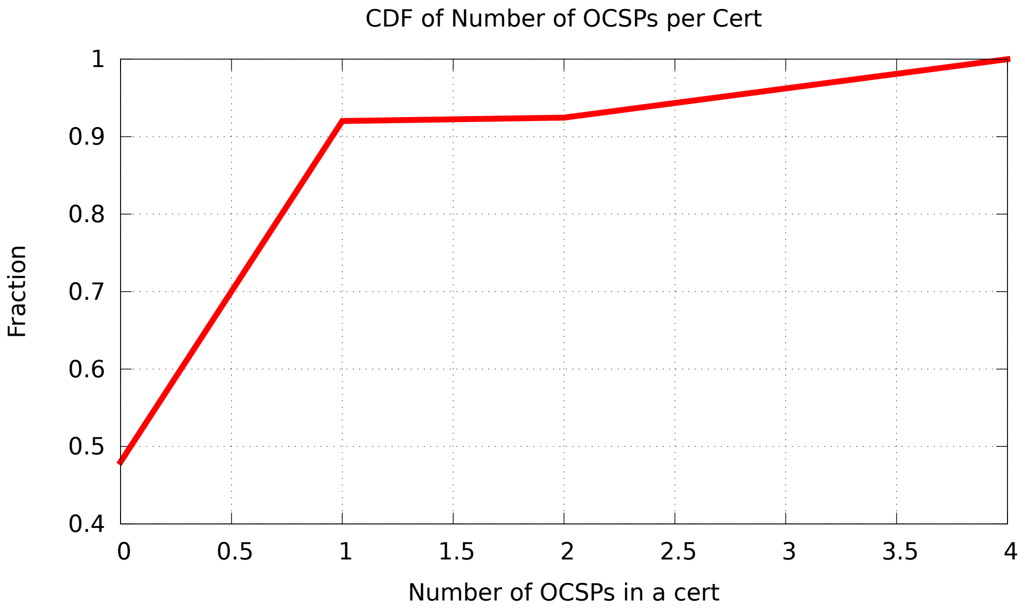
- Transitive trust graph of internetsociety.org

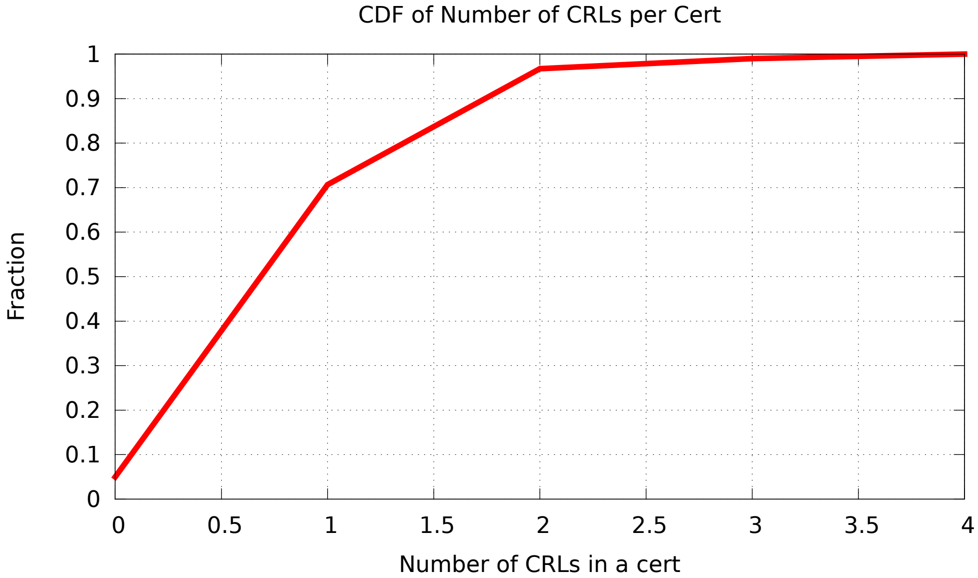


How deep the delegation chains are

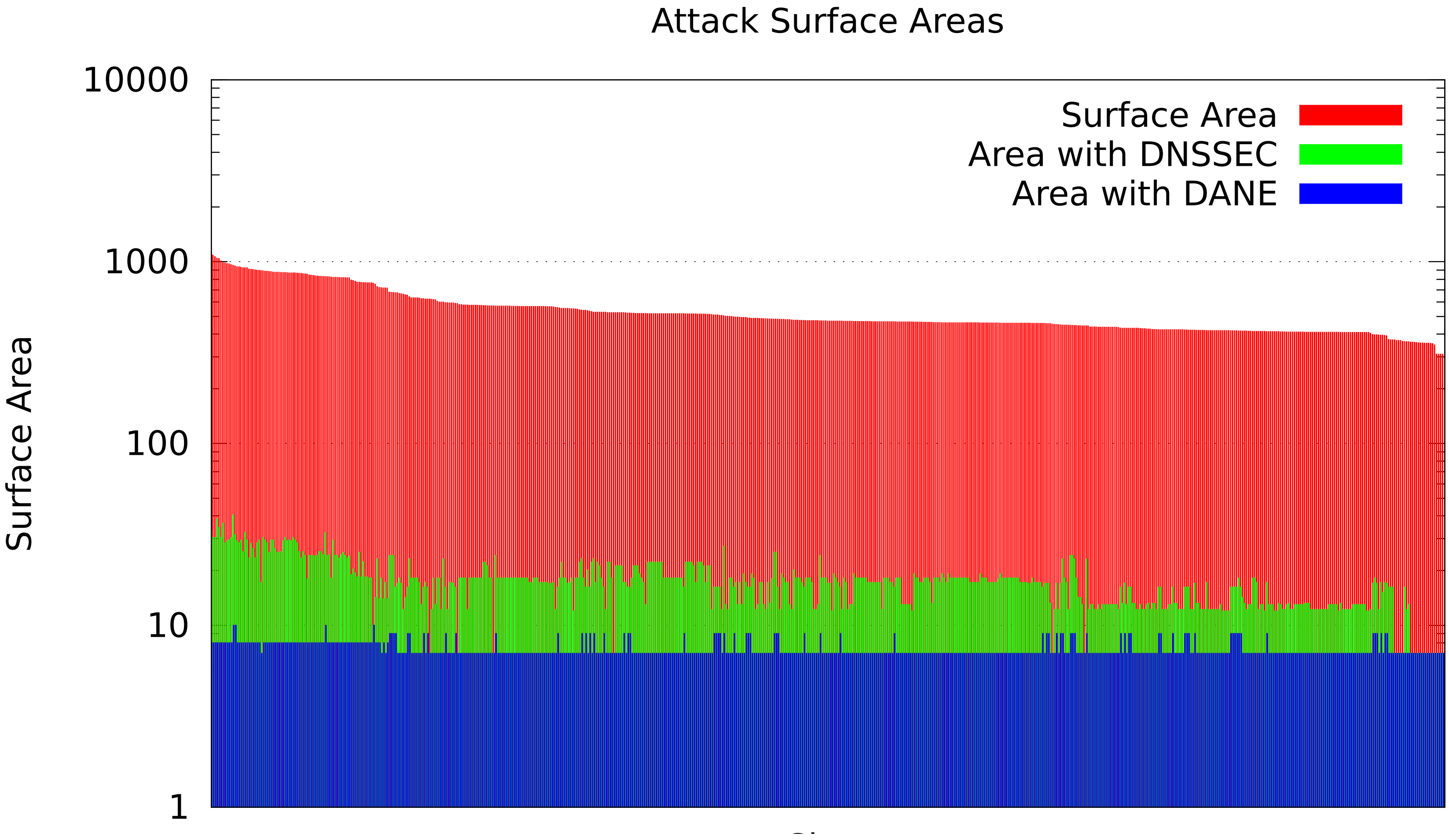


Revocation details (from CRL/OCSP URIs)

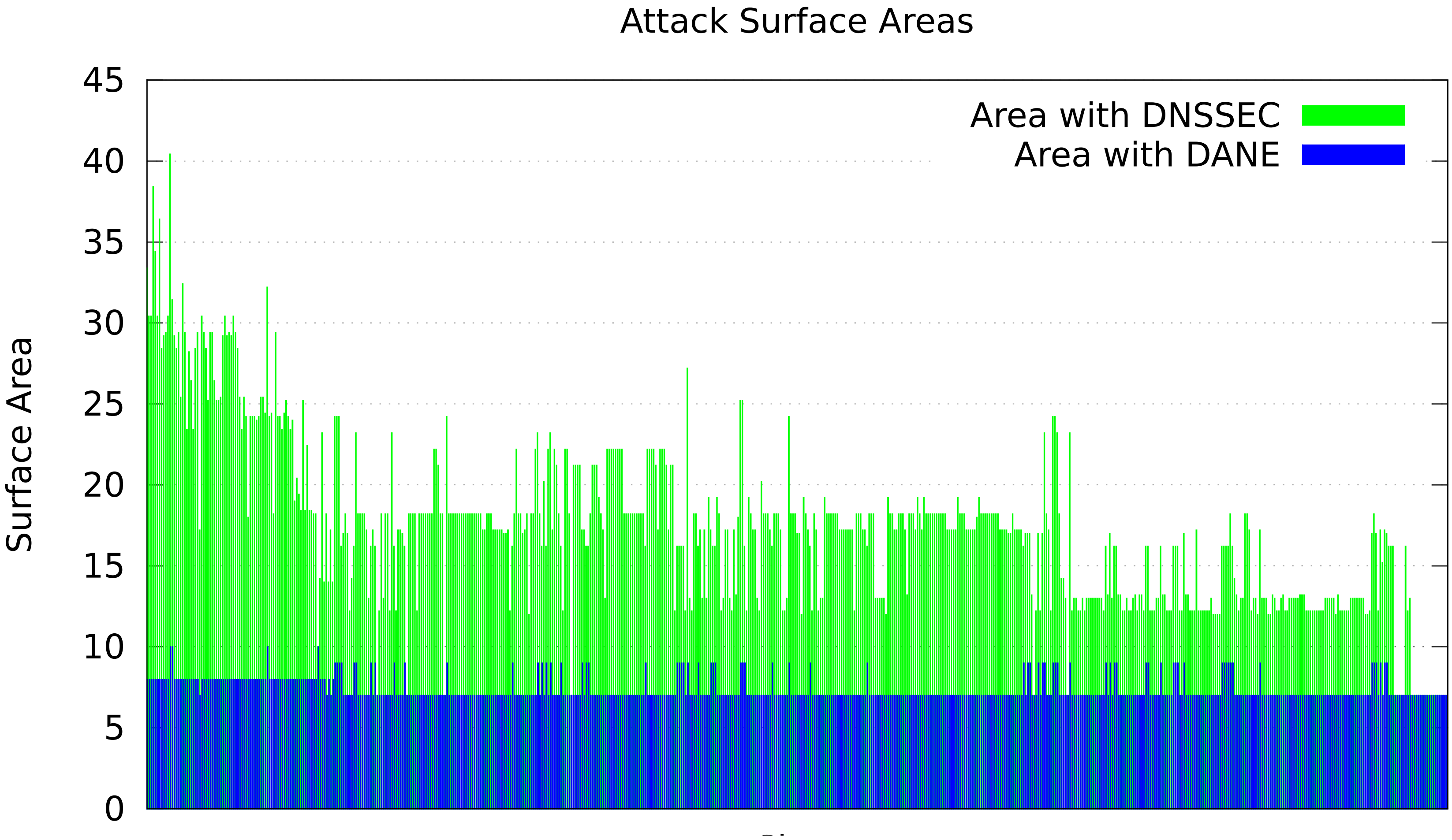




Quantitatively comparing these two systems



Non-log scale (just DNSSEC and DANE)

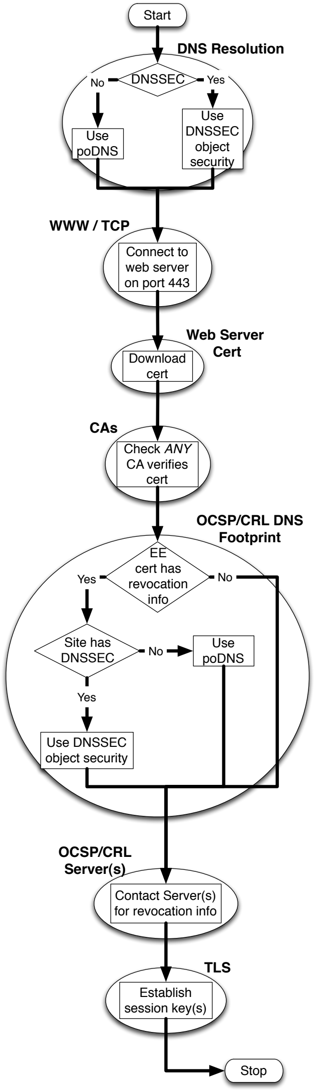


Kill chain analysis

- FPDs may lend nicely to kill chain-analysis

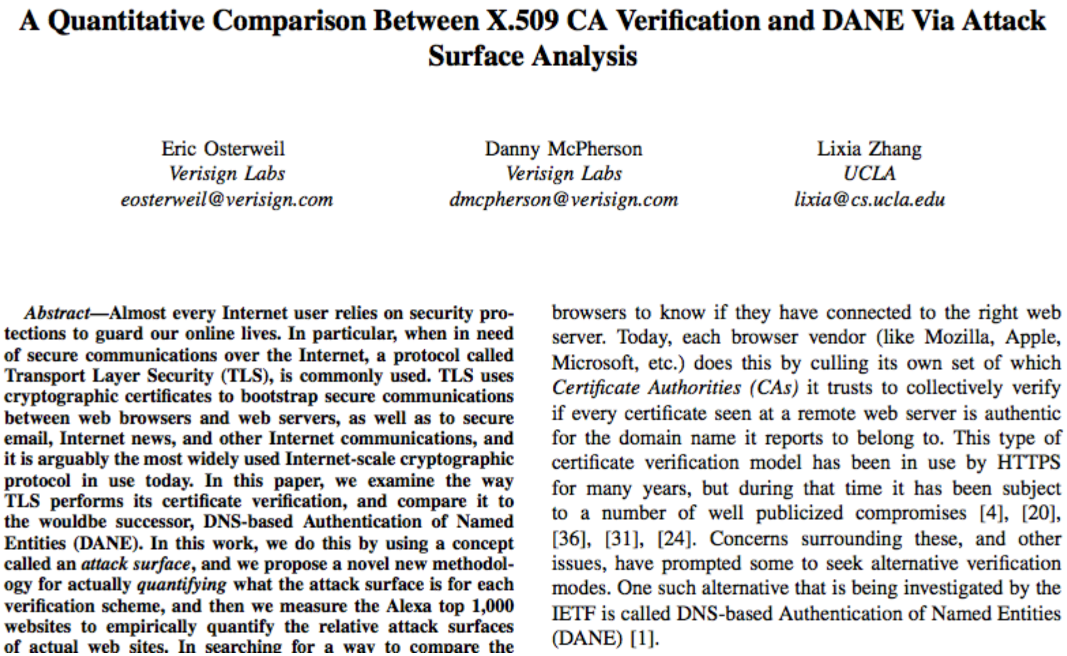
- Disrupting a step in an FPD can render the rest of the process moot

- Example: failing in the DNS stage renders following processing useless



Our Technical Report

http://techreports.verisignlabs.com/tr-lookup.cgi?trid=1120004



A Quantitative Comparison Between X.509 CA Verification and DANE Via Attack Surface Analysis

Eric Osterweil
Verisign Labs
eosterweil@verisign.com

Danny McPherson
Verisign Labs
dmcpherson@verisign.com

Lixia Zhang
UCLA
lixia@cs.ucla.edu

***Abstract*—Almost every Internet user relies on security protections to guard our online lives. In particular, when in need of secure communications over the Internet, a protocol called Transport Layer Security (TLS), is commonly used. TLS uses cryptographic certificates to bootstrap secure communications between web browsers and web servers, as well as to secure email, Internet news, and other Internet communications, and it is arguably the most widely used Internet-scale cryptographic protocol in use today. In this paper, we examine the way TLS performs its certificate verification, and compare it to the wouldbe successor, DNS-based Authentication of Named Entities (DANE). In this work, we do this by using a concept called an *attack surface*, and we propose a novel new methodology for actually *quantifying* what the attack surface is for each verification scheme, and then we measure the Alexa top 1,000 websites to empirically quantify the relative attack surfaces of actual web sites. In searching for a way to compare the**

browsers to know if they have connected to the right web server. Today, each browser vendor (like Mozilla, Apple, Microsoft, etc.) does this by culling its own set of which *Certificate Authorities (CAs)* it trusts to collectively verify if every certificate seen at a remote web server is authentic for the domain name it reports to belong to. This type of certificate verification model has been in use by HTTPS for many years, but during that time it has been subject to a number of well publicized compromises [4], [20], [36], [31], [24]. Concerns surrounding these, and other issues, have prompted some to seek alternative verification modes. One such alternative that is being investigated by the IETF is called DNS-based Authentication of Named Entities (DANE) [1].

What else could we look at?

- We started evaluating the systemic dependencies needed for candidate resource certification

- Very early thoughts and not fully evolved

- What might we see with RPKI?

How might RPKI look in this light?

Resources for a single repository

Size of each publication point's systemic dependencies

Adding up the surface

Thoughts going forward…

- There are lots of systems and protocols that have dependencies
  - Many times, these dependencies are non-obvious
- With this methodology we hope to offer a tool that lets people start evaluating what systems depend on

Thanks!

Questions?

Transitive Trust Checker

http://trans-trust.verisignlabs.com/

- Operational Implications of the DNS Control Plane, Eric Osterweil, Danny McPherson, Lixia Zhang, IEEE Reliability Society Newsletter, May 2011

  http://irl.cs.ucla.edu/~eoster/doc/trans-trust.pdf