Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service

Bernhard Amann[*], Matthias Vallentin[§],
Seth Hall[*], and Robin Sommer[*ǂ]

TR-12-014

November 2012

Abstract

Much of the Internet's end-to-end security relies on the SSL protocol along with its underlying X.509 certificate infrastructure. However, the system remains quite brittle due to its liberal delegation of signing authority: a single compromised certification authority undermines trust globally. We present a novel *notary service* that helps clients to identify malicious certificates by providing a third-party perspective on what they should expect to receive from a server. While similar in spirit to existing efforts, such as Convergence and the EFF's SSL observatory, our notary collects certificates *passively from live upstream traffic* at seven independent Internet sites. Our data set currently includes 330k certificates extracted from 5.5B SSL sessions over a time interval of 6 months. The notary offers a DNS-based, near real-time query interface to the public that is compatible to existing systems. We will maintain the notary as an ongoing service to the community and plan to include further data providers in the future to extend its coverage. From a broader perspective, we also see our work as a case study on working successfully with network operators to provide researchers with real-world data.

*International Computer Science Institute, 1947 Center Street, Suite 600, Berkeley, California, 94704
§ University California, Berkeley, 2200 University Drive, Berkeley, California 94720
ǂ Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, California, 94720

This work was partially funded by the Deutscher Akademischer Austausch Dienst (DAAD) through a postdoctoral fellowship.

1 Introduction

The Secure Socket Layer (SSL/TLS) protocol serves as a centerpiece of the Internet's end-to-end security, providing secure encrypted channels and authentication through its underlying X.509 certificate infrastructure. In a nutshell, X.509 certificate authorities (CAs) sign SSL[1] server certificates, which clients then verify against a list of trusted root CA certificates that ship with their operating system or client software. In practice, most server certificates are not directly signed by a root CA; instead roots delegate signing authority to intermediate CAs. When validating a certificate, an SSL client attempts to build a valid *certificate chain* from the server certificate to one of the root certificates it knows, including intermediates as necessary.

This system is quite brittle due to its liberal delegation of trust. All root and intermediate CAs share the authority to sign *any* certificate Internet-wide. Hence, the compromise of a single intermediate undermines the entire X.509 certificate infrastructure, rendering CAs an attractive target for attackers. As demonstrated by several recent high-profile incidents [5, 20], an attacker typically acts as a man-in-the-middle (MITM) to assume the identity of a well-known secure site (e.g., a bank or email provider) by serving a malicious certificate. From the victim's perspective, the presented certificate validates correctly against their root store and hence the attack goes unnoticed [2]. While the weaknesses of the current state have been widely acknowledged, there is no real solution in sight. The security community has proposed a number of alternative PKI architectures (e.g., [8, 15]); however, widespread adoption remains unlikely for the medium-term future due to the fact that both clients and servers would have to support a new system.

SSL *notaries* represent an alternative approach to improve the existing state without architectural changes. A notary maintains a third-party database of known server certificates. When clients encounter a certificate, they can match it against the notary's version and flag mismatches as possible attacks. Notaries represent an attractive extension to the existing X.509 architecture as they introduce a non-obtrusive reputation scheme into the standard validation process. Perspectives [24] pioneered this method and Convergence [1] provides an improved implementation.

Our work follows this approach by offering a novel notary service for SSL clients. However, while existing notaries build their certificate databases actively—by either scanning the IP address space from different vantage points, or by asking users to submit what they see [7]—we *passively* extract X.509 certificates from live network traffic as observed on an ongoing basis at operational sites. As such, our notary complements existing services by providing a near real-time perspective on certificates *in actual use* by a large client population, along with further reputation information such when we first observed a specific certificate and the number of sites that have reported it.

The key challenge with this passive approach is to ensure broad certificate coverage. To feed our notary service, we are currently working with seven organizations that have instrumented their border gateways with our monitoring infrastructure, uploading all certificates they see to a central system at our institute in near real-time. As of August 2012, our data comprises approximately 4,000 hours of SSL activity from about 227k users, with the relevant information extracted from about 5.5 billion SSL sessions total.

[1] For the remainder of the paper, we will refer to either SSL or TLS as "SSL".

With these sessions come about 340k unique certificates that our notary makes available to the public through a DNS-based interface. As this passive data complements existing active services, clients could use them in conjunction to maximize coverage.

The most common deployment scenario for notaries concerns web browsers, and we support that directly by offering an interface compatible to existing systems. In addition, we also target two further applications: integration with a network security monitor and a validation service. First, our notary's per-query overhead is low and hence suitable for bulk-checking thousands of certificates quickly. This enables us to couple it to a network security monitor that validates certificates in real-time as they appear on the wire. Second, validation of certificates is not only CPU-intensive but also surprisingly difficult to implement correctly. Our notary precomputes validity for all its certificates and provides the information in its responses for clients to use.

In addition to the notary itself, we also see our work as a case-study on collaborating successfully with network operations. The challenge of getting access to real-world network data is a recurring theme in our research community, and our work provides evidence that operators are willing to share data if one addresses the constraints they face, even if performing automatic uploads of payload data to an external site.

We structure the remainder of this paper as follows: §2 presents our data collection process and discusses our experiences working with operations. §3 gives a short high-level overview of the data we have collected so far, and §4 presents the public notary service we set up. §5 discusses particular design questions. Finally, §6 summarizes related work and gives an overview of the current state of affair of X.509 certificates before we conclude in §7.

2 Data Collection

We begin the presentation of our system with an overview of our data collection setup now in operation at seven network sites (§2.1), and then report our experiences working with operations at these sites (§2.2).

2.1 Collection process

Generally, we assume the vantage point of a site's upstream border link where we passively monitor all external traffic (see Fig. 1). We use Bro [19, 4] to extract all X.509 certificates, which detects and parses SSL connections independent of any transport-layer ports [6]. To avoid revealing information about the collecting site, we configure the analysis to only inspect *outgoing* connections. In addition to recording certificates, we also record session-level SSL features such as timestamp, server address, cipher in use, and a hash of the client/server address tuple. While for privacy constraints we cannot make this additional data available to the public, it will allow us to further study properties of SSL communication internally. We note that none of the collected information directly identifies a client system.

We implemented this analysis as a Bro script that we give operators at participating sites. The script uploads both the certificates and the session-level information to a

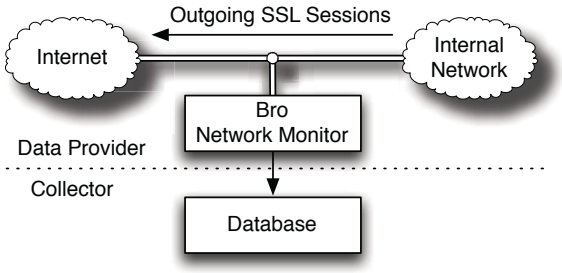

Fig. 1. Data collection setup

central server at our research institute in regular intervals. Our seven participating sites all operate ongoing Bro installations and added our script to their existing setups. [2]

2.2 Collaborating with Operations

Researchers in our community often find operators reluctant to provide real-world data for scientific studies. For this effort, we work with seven operational environments that all have agreed to instrumenting their networks for recording and exporting information in real-time, with full payload access for the analysis application. As such, we see our work also as a case study on overcoming the hurdles that researchers often face when interacting with operations, and we discuss our experiences in the following.

Generally, at the sites included in this study, we found operators with an *interest* in supporting our effort and contributing value to the larger community. We consider such support a crucial prerequisite as it provides sites with a motivation to invest time into the collaboration. With that perspective, participation becomes a question of realistically trading off benefits with the risk of unintentionally revealing sensitive information. None of the sites took that decision lightly, yet they all eventually approved going ahead.

The key to a successful collaboration lies in accepting and addressing the constraints that operators face. Most importantly, we design our data collection to minimize the risk of exposing site information. We *(i)* limit the collected features to a subset generally deemed low sensitive; *(ii)* separate between data for the public notary service and the further session-level features that we keep private; and *(iii)* accept that we generally do not control the collection setups and may hence experience artifacts such as non-continuous time intervals and packet loss. When operators were assessing the features we collect, their main concern was the risk of revealing specific sites their users access. For the notary, we account for that by leaving all sources anonymous and providing only coarse-grained, aggregated information.

[2] At three of the sites, members of our team performed the system integration after receiving administrative approval.

3 Data Sets

We now describe the data that our notary harnesses in more detail. Seven operational network sites of different sizes provide us with SSL information, captured at their upstream network links. Table 1 summarizes the data from each site. Our contributors requested to remain anonymous. Most of them are research environments, with six of the seven located in the US. As we added the sites incrementally to our effort, the individual sets span different time periods. For comparison, we list the total hours observed at each site (non-continuous due to occasional outages).

As Table 1 shows, our data set includes a total of 18M unique X.509 certificates derived from several billion SSL connections. Of those, over 17 million originate from Grid traffic[3] and Tor servers[4]. Due to the highly dynamic and specialized nature of Grid and Tor certificates, we exclude them from both our notary and the further discussion in this paper. The *filtered* column in Table 1 shows the number of remaining certificates. We identify Tor certificates with a regular expression. Grid certificates theoretically could by validated by using the list of Grid roots accredited by the International Grid Trust Federation (IGTF). However, this approach proved problematic due to holes in certificate chains, and we hence opted to likewise use pattern matching for linking typical Grid-related "common names" to local certificates.

We note that our data sets exhibit artifacts of the collection process that are beyond our control. As we leverage operational setups that run our analysis on top of their normal duties, we must accept occasional outages, packets drops (e.g., due to CPU overload) and misconfigurations. As such, we deliberately design our data collection as a "best effort" process: we take what we get but generally cannot quantify what we miss. However, given the large total volume across the seven sites, we consider the aggregate as representative of many properties that real-world SSL activity exhibits.

Comparison with Holz et al. Holz et al. [12] present the most comprehensive academic study of X.509 so far. The authors collected several certificate sets by performing active scans from multiple locations and further derived two smaller sets from passive measurements. We briefly compare our more recent data with their findings. In aggregate, our data set has a comparable size—both sets comprise several 100k unique certificates.

The strength of a public key depends on the used encryption and hash algorithm as well as the key bit length. Nearly all X.509 certificates in use today are RSA certificates. Similar to Holz, we only see a handful of ECDSA and DSA keys in our data. For RSA, the cryptography community discourages key lengths less than 1024 bits [3]. However, Holz finds that their latest scan still includes 55% end-host certificates with smaller lengths, while earlier scans included 20% more than that. For the certificates we see this downward trend continuing: only 21% end-host certificates have such small key lengths. When looking only at currently validating certificates (see §4.3), the number decreases to 13%. Holz also found that the number of certificates using MD5 as an hash algorithm is steadily decreasing. By revisiting their latest active scan, we found that

[3] Grid services use certificates to identify both servers and users; most of the used certificates are dynamically generated and have lifetimes of only a few minutes or hours.

[4] The certificates of Tor servers we encountered do not seem to be used for authentication. Servers appear to change their certificates every few minutes.

Site			Certificates		Connections	Time	
Label	Type	Est. Users	Total	Filtered	Total	Hours	Start
US1	University	90,000	17,679,855	222,757	2,839,963,404	3,903	02/22
US2	Research site	250	154,840	22,377	40,073,665	4,603	02/17
US3	Research site	4,000	92,885	64,306	418,586,535	3,577	02/22
US4	University	50,000	327,783	185,231	2,423,110,152	4,080	02/22
X1	University	3,000	13,743	8,749	12,996,314	3,330	03/14
US6	University	30,000	18,920	17,617	13,447,775	56	08/27
US5	Gov. Network	50,000	92,361	90,053	251,646,788	3,639	03/30
All [1]		227,250	18,032,041	337,721	5,581,238,098	—	—

[1] The total reflects the number of *unique* items across all sites.

Table 1. Summary of data set properties from contributing sites.

6.6% of the distinct certificates therein used MD5 hashes. In our data set, this percentage has decreased to 3.3%, again suggesting an improvement in certificate quality.

However, the observed improvements may also come partially from our passive collection process. Our data will not contain certificates of very unpopular (or only privately used) hosts, which might more likely use weaker keys. For example, Holz reports seeing more than 60,000 certificates for Plesk and more than 38,000 for `localhost`. In comparison, we just have 388 and 2,303 of those certificates, respectively.

Another important security measure represents the cryptographic algorithms and key length in an SSL session. In line with Holz, we see changes to more secure algorithms. RC4 with 128-bit SHA prevails as most common cipher in our data set (20% of connections). In contrast, Holz observes RC4 with MD5 as most common cipher, which we observe only as the fourth popular one (15% connections). Another trend in our data shows a high number of connections using one of the different TLS ciphers with ECDHE (21% of connections), which we attribute to Google now supporting these ciphers in order to offer forward secrecy [14].

4 Notary Service

We now describe our X.509 notary in more detail. The setup is live and accessible to the public, and we intend to maintain it as an ongoing service to the community.

4.1 Client Interface

Clients access the notary via DNS requests for names of the form `<sha1>.notary.icsi.berkeley.edu`, where `<sha1>` represents the SHA1 hash of a certificate. The notary replies with a `TXT` record indicating *(i)* the day our data providers first saw the certificate (relative to 1/1/1970), *(ii)* the most recent day any of them saw it, *(iii)* the

number of days in between when at least one site reported it, and *(iv)* if it currently validates against the Mozilla root store. [5] A exemplary query looks like this:

```
# dig +short txt C195[..]3358.notary.icsi.berkeley.edu
"version=1 first_seen=15387 last_seen=15582 times_seen=196 \
 validated=1"
```

In addition to the primary interface described above, we also offer a DNS interface compatible to Google's (now defunct) SSL Catalog [17] at the domain `cc.notary.icsi.berkeley.edu`. Existing software like Convergence can leverage it directly.

To verify compatibility to existing services, we used our notary as a Convergence backend. This required patching Convergence to work with arbitrary DNS-based verifiers as opposed to supporting only Google's Certificate Catalog. We set up a local node that interfaced to our notary and tested the functionality using the Convergence Firefox plugin.

Furthermore, we wrote a Bro script which performs two DNS lookups per observed certificate to both of our notary domains, recording the responses along with other certificate details (such as subject and issuer) into a log file. The implementation proves concise, since Bro has the capability to perform asynchronous DNS lookups; the central lines are:

```
local seen = fmt("%s.%s", cert_hash, ".all.notary.icir.org");
when ( local result = lookup_hostname(seen) )
 # Work with 'result' when the asynchronous DNS lookup completes.
```

The script also maintains a cache with recent results to relieve the local DNS resolver. The full version is available in Bro's git repository[6] and will become part of a future Bro release. When running the script on a short network trace from one of our data providers, we did not notice any performance degradations due to the extra DNS lookups.

4.2 Architecture

Fig. 2 shows the internal architecture of our notary service. When a collection site uploads data to our institute, we first import it into a PostgreSQL database residing in a secured network. We create a text zone file in DNSBL format [21] at an hourly basis and transfer it over to the externally visible `notary.icir.org` system located in a DMZ where we serve the zone with *rbldnsd* [21], a stripped-down, high-performance DNS server commonly used for high-volume blacklist-style information.

On the central database server, we validate all certificate chains at import time and revalidate all of them once per day to account for changes in the root-store, expired certificates, etc. We spent significant time to optimize our import scripts for handling information from large numbers of SSL sessions. At peak times, our data providers upload more than a million new connections per hour. During bulk imports, we measured the script's maximum rate at about 100K sessions/minute for a single thread, running

[5] We also support `A` requests and return `127.0.0.1` if an entry exists for the certificate and 127.0.0.2 if it validates against the Mozilla root store.

[6] The code currently exist in the branch `topic/matthias/notary`.

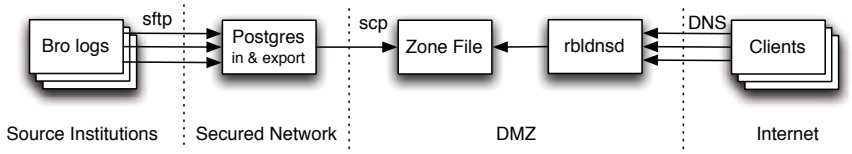

Fig. 2. Notary service architecture

on a Intel Xeon E5630 CPU. As certificate validation accounts for most the work, the processing parallelizes well across CPUs.

For statistical purposes, our custom version of rbldnsd logs the timestamp of each request, the SHA-1 hashed client IP, the geo-location of the client, and the query itself along with the notary's response.[7]

4.3 Certificate Validation

When validating certificates, we aim to match the results a typical browser would come to. It turns out that doing so is a surprisingly involved process due to the complexities of X.509 and the variety in certificate chains returned by the servers. In particular, many returned certificate chains remain incomplete, which tools like OpenSSL cannot handle. As we have not found the validation process documented elsewhere, we report the specifics of our approach in the following. We use the Mozilla root store as our trusted base.

When validating a certificate C from a server chain X_S, we incrementally assemble a temporary validation chain X_V that leads from C to one of the roots. Once we have such a sequence complete, we use OpenSSL to verify its correctness. If successful, we consider all certificates in X_V, including X_S, as *validated*.

To build the chain we first match C's *Issuer* against the *Subject* of all root certificates. If we find a match, this completes X_V. Otherwise, we examine X_S for a matching intermediate certificate. If found, we insert that into X_V and proceed recursively. If not found, we search for a matching intermediate across *all* already validated CA certificates in our data set that have not yet expired. This step matches the behavior of typical browsers, which *cache* intermediate CAs they have already encountered in past sessions in their local certificate store.[8] Certificates can include an *AuthorityInformationAccess* (AIA) extension to specify an URI for intermediate certificates not included in a chain. Web browsers parse this extension, download the intermediates, and add them to their local certificate store. If we encounter an *AuthorityInformationAccess* extension, we likewise attempt to download and use the certificates if it contains an HTTP URI (we ignore other schemes, such as LDAP). Finally, if we have come to a complete chain X_V, yet find that

[7] Our rbldnsd enhancements are publicly available at `https://github.com/mavam/rbldnsd`.

[8] This behavior makes validation dependent on a browser's state for incomplete chains. Indeed, we noticed that the web server of a major research institution failed to include a necessary intermediate certificate and thus remained inaccessible with a fresh browser installation.

OpenSSL does not accept it, we attempt to extend it further. This addresses cases where the names of intermediates match that of a root, but their keys differ.

The presented approach seems to generally match well with what browser return. While an exact comparison is technically difficult[9], we manually checked a random sample of 10% of the certificates our method leaves unvalidated and found no cases that a standard browser would accept.

5 Discussion

Using DNS as the notary's interface leverages its distributed nature for providing both scalability and client anonymity. A client's local DNS server will typically cache queries, which reduces the load on our systems for frequently requested certificate hashes. Furthermore, if the server recursively resolves requests, we will see *its* IP address on our end, not that of the client using the notary. If users want to avoid any evidence of their location, they can switch to public DNS services, such as OpenDNS or Google [9]. This is different from the REST-based interfaces that Perspective and Convergence offer. The latter implements a complex onion-routing approach to protect its clients' identity.

Our notary's view of the X.509 world depends inherently on the SSL activity seen by its data providers. As currently most of them are located in the US, coverage skews towards services accessed from there. While a highly diversified user population at these sites helps to mitigate the effect somewhat, we are seeking to add further international sites in the future. Doing so will also add coverage for localized services, such as CDNs.

Attackers could poison our notary's database if they had access to one of the networks contributing data; by establishing SSL connections to external systems under their control, they could inject malicious certificates. As a remedy, we could add a reputation score to the notary's response derived from the number of sites that have seen a certificate; alternatively we could skip certificates altogether that are not reported by a minimal quorum of sites. We may add either of these mechanisms in the future.

6 Related Work

The recent increase of security incidents involving certificate authorities turned the global X.509 infrastructure as an attractive subject of study. The Electronic Frontier Foundation (EFF) popularized the study of X.509 certificates infrastructure by publishing data sets derived from actively scanning the entire IPv4 address space on port 443 in mid 2010 and, again, in early 2012 [7]. Holz et al. [12] provide the most comprehensive academic treatment of the SSL infrastructure we are aware of, incorporating the 2010 EFF data set and comparing them with active and passive measurements of their own spanning 1.5 years. Vratonjic et al. [23] present a study of the X.509 certificates of the Alexa Top 1M list. Mishari et al. [18] study 30K X.509 certificates from randomly scanning different sets of domains. Heninger et al. [10] present a weak key study of TLS and SSH keys retrieved by an IPv4 address space scan. They offer a weak-key checking service based

[9] Specifics of the validation logic tend to be hidden deeply in a browser's code, with no easy way to factor it out, or access from external. Furthermore, different browsers disagree in some cases.

on their data. Our notary provides access to a novel, large-scale certificate data set to the community and can support future research studies.

Scenarios for man-in-the-middle (MITM) attacks are, e.g., discussed by Soghoian et al. [22]. Holz et al. [13] created a tool named Crossbear with the purpose of finding the origin of MITM in the Internet by deploying hunters that connect to affected servers from different origins. It is a research tool and not expected to be deployed by end-users.

Proposals to address the threat of malicious CA certificates include semi-centralized timeline servers and cryptographic append-only data structures [8, 15]. As adoption of such proposals is unlikely in the near future, other efforts aim to address shortcomings and problems of the current SSL and X.509 architecture [11, 16].

7 Conclusion

We present a novel SSL notary service that provides clients with a third-party perspective on certificates they should expect to receive from a server. While similar in spirit to existing services, we collect certificates passively from live Internet traffic at seven different sites on an ongoing basis, providing users with a near real-time view of certificates in actual use by a large client population. We intend to maintain the service as an ongoing service to the community and plan to include further sites as data providers. In particular, we recently received administrative approval from a US-wide backbone provider and are now working with their staff to set up the monitoring infrastructure. On the research side, our ongoing data collection will enable longitudinal studies of trends and developments of real-world SSL usage over time, and the notary's anonymized logs will allow us to estimate coverage based on what we see clients querying for.

References

1. Convergence. http://www.convergence.io
2. Adkins, H.: An update on attempted man-in-the-middle attacks (Aug 2011), http://googleonlinesecurity.blogspot.com/2011/08/update-on-attempted-man-in-middle.html
3. Bos, J.W., Kaihara, M.E., Kleinjung, T., Lenstra, A.K., Montgomery, P.L.: On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography. Cryptology ePrint Archive, Report 2009/389 (2009)
4. Bro IDS Web Site. http://www.bro-ids.org
5. Mozilla Security Blog: DigiNotar Removal Follow Up. http://blog.mozilla.org/security/2011/09/02/diginotar-removal-follow-up/ (Sep 2012)
6. Dreger, H., Feldmann, A., Mai, M., Paxson, V., Sommer, R.: Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection. In: USENIX Security Symposium (2006)
7. EFF: The EFF SSL Observatory. https://www.eff.org/observatory
8. EFF: The Sovereign Keys Project. https://www.eff.org/sovereign-keys
9. Google Developers – Public DNS. https://developers.google.com/speed/public-dns/ (Aug 2012)
10. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. In: USENIX Security Symposium (2012)

11. Hodges, J., Jackson, C., Barth, A.: HTTP Strict Transport Security (HSTS). RFC Draft (Aug 2012), http://tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-12
12. Holz, R., Braun, L., Kammenhuber, N., Carle, G.: The SSL Landscape: A Thorough Analysis of the X.509 PKI using Active and Passive Measurements. In: IMC (2011)
13. Holz, R., Riedermaier, T., Kammenhuber, N., Carle, G.: X.509 Forensics: Detecting and Localising the SSL/TLS Men-in-the-Middle. In: ESORICS (2012)
14. Langley, A.: Google Online Security Blog – Protecting data for the long term with forward secrecy (Nov 2011), http://googleonlinesecurity.blogspot.com/2011/11/protecting-data-for-long-term-with.html
15. Langley, A.: ImperialViolet – Certificate Transparency (Nov 2011), http://www.imperialviolet.org/2011/11/29/certtransparency.html
16. Langley, A.: ImperialViolet – Public key pinning (May 2011), http://www.imperialviolet.org/2011/05/04/pinning.html
17. Laurie, B.: Google Online Security Blog – Improving SSL certificate security (Apr 2011), http://googleonlinesecurity.blogspot.com/2011/04/improving-ssl-certificate-security.html
18. Mishari, M.A., Cristofaro, E.D., Defrawy, K.M.E., Tsudik, G.: Harvesting SSL Certificate Data to Mitigate Web-Fraud. CoRR abs/0909.3688 (2009)
19. Paxson, V.: Bro: A System for Detecting Network Intruders in Real-Time. Computer Networks 31(23-24), 2435–2463 (1999)
20. Percoco, N.J.: Clarifying The Trustwave CA Policy Update (Feb 2012), http://blog.spiderlabs.com/2012/02/clarifying-the-trustwave-ca-policy-update.html
21. rbldnsd: Small Daemon for DNSBLs, http://www.corpit.ru/mjt/rbldnsd.html
22. Soghoian, C., Stamm, S.: Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL. http://files.cloudprivacy.net/ssl-mitm.pdf
23. Vratonjic, N., Freudiger, J., Bindschaedler, V., Hubaux, J.P.: The Inconvenient Truth about Web Certificates. In: The Workshop on Economics of Information Security (WEIS) (2011)
24. Wendlandt, D., Andersen, D.G., Perrig, A.: Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In: USENIX Annual Technical Conference (2008)