MECAI - Mutually Endorsing CA Infrastructure

A proposal to improve PKI, OCSP, CAs.
Version 2.02 *
24 February 2012

Kai Engert
kaie@kuix.de – kaie@redhat.com

Table of Contents

Visit https://kuix.de/mecai/ for an architectural diagram and for updates to this document.

* new in version 2.02: added reference to architectural diagram

* new in version 2.01: typos fixed

MECAI Overview

This document assumes the reader has a basic understanding of the terms Public Key Infrastructure (PKI), Certificates (X.509), Certificate Authorities (CA), Online Certificate Status Protocol (OCSP), OCSP Stapling (RFC 4366), Certificate Revocation Lists (CRL), Secure Sockets Layer (SSL), Transport Layer Security (TLS), Man-In-The-Middle (MITM).

Goals:

- Solve the problem that an abuse of a single CA is sufficient to trick most SSL/TLS clients into accepting false certificates and perform a successful MITM attack, by introducing a second trust opinion.
- Design an architecture that is incremental to today's PKI, with a focus on incremental, short term deployment.
- Focus on MITM attack scenarios where the MITM is close to the victim's client, or at least where the attacker isn't immediately next to the attacked server.
- Enable SSL/TLS clients to make a guess about an existing MITM attack, if secondary trust statements are absent.
- Provide solutions that can avoid the need of communication channels besides the primary SSL/TLS channel, and thereby work in paid-for environments such as Wi-Fi Hotspots or captive portals in general.
- Function in combination with any existing protocol that uses TLS (wrapping another protocol) or STARTTLS (switch a channel using an arbitrary protocol to one that is wrapped inside TLS without a reconnect).
- Avoid the introduction of additional trusted entities, involve existing and future CAs.
- Require participation from the entities that benefit from PKI – the organizations that run a CA are the ones who experience a monetary benefit.
- Be data efficient, avoid high volumes of data to be kept publicly, avoid having to transfer high volumes of data to clients in order to be compatible with mobile devices, and compatible with environments where data transfer is expensive.
- Be decentralized and open.
- Function with old server software at the disadvantage of requiring a side channel initiated by clients to CA infrastructure.
- Allow servers that upgrade their software to increase the speed of the trust verification performed by clients and avoid the need of a side channel.
- Use the idea to have notaries in the web that confirm the use of server certificates from the notary's perspective. Require that CAs are notaries.
- Have a global infrastructure where a decision to distrust a (top level) CA certificate can quickly become effective, by minimizing the amount of parties that need to learn about the distrust decision.

The proposed infrastructure shall achieve the aforementioned goals by use of the following ideas:

- In addition to Server Certificates, introduce the concept of shorter lived vouchers issued by existing CAs
- Require that Certificate Authorities (CA) act as web notaries, introducing the term Voucher Authority (VA)
- A voucher confirms facts seen on the web from the perspective of a VA. A voucher contains a (hash of a) server's certificate, the (hashes of) intermediate CA certificate(s), a server's IP

address, (optionally) recent revocation status information (such as a recent OCSP response), a timestamp, a list of trust contexts, and a digital signature produced by the VA's certificate (a sub CA certificate issued by the CA's root certificate).

- A client connecting to a server will require the server's certificate as usual. In addition, it will require to receive a voucher that was issued by a CA. The CA/VA that issued the voucher must be different from the CA that issued the server certificate.
- The voucher might be transferred from a VA to a client using a protocol that shall be similar to the OCSP protocol. In addition, in order to avoid the need for a side channel and to improve the performance, a server shall obtain the voucher(s) in advance, cache them, and transfer them to a client as part of the TLS handshake, similar to OCSP stapling.
- When initiating a connection, a TLS client will randomly select a small set of candidate Authorities (e.g. three) that are allowed to vouch for the connection (in order to allow for offline VAs, and because the client doesn't know in advance which CA has issued the server's certificate).
- After the transition period that is required to deploy the VA servers and to upgrade the client software, client shall reject connections that cannot present a valid and fresh voucher.

If clients by default block connections that lack a valid voucher, then an attacker that abused the power of a single CA will no longer be successful.

An attacker that is able to control two CAs will only be successful in a small percentage of connections. Only if the second CA controlled by the attacker is the one randomly selected by the client, the client will accept a voucher from the second CA. In the vast majority of connections the attacker will be unable to satisfy the client's voucher expectations.

Only a very small number of security conscious users are required to understand the client's security errors (because of a missing voucher) correctly, and the rogue server certificate can be reported to authorities.

As soon as the (potentially) rogue certificate has been reported (to one of several public submission points), which should be combined with the IP address where the rogue certificate was seen, the information can be automatically distributed, at least to all publicly known CAs (to produce a public audit).

At least two CAs will be very much interested in analyzing such reports.

(a) The CA who issued the reported, potentially rogue certificate will probably be keen on analyzing their own infrastructure quickly, who can revoke the certificate quickly. The revocation has the result that no further vouchers will be issued for the rogue certificates.

(b) The CA who has issued a certificate for the involved hostname and has a customer relationship with the domain owner. The CA can contact the domain owner. Once the CA domain owner confirms that the certificate is rogue, the information about the abuse can be published. The public will demand an investigation. The public (and the CAs) can verify successful revocation of the rogue certificate. Depending on the extent of the security incident, the involved CA certificate can be revoked. All CAs could be informed to stop producing any vouchers for certificates issued by the affected CA. After a short delay (based on the validity period of vouchers) all client software that requires vouchers will distrust.

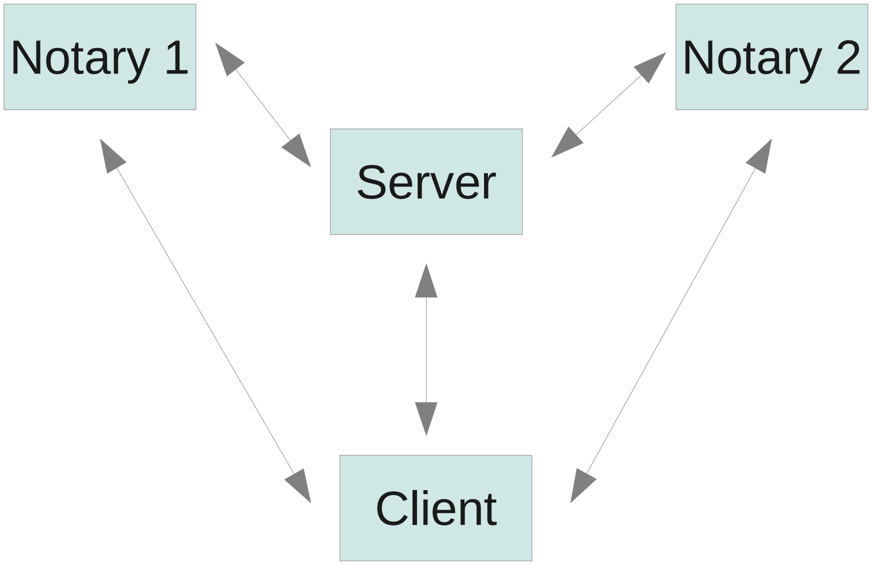
MECAI Details

Trusted Authorities

MECAI doesn't introduce a new set of Authorities, it rather expects additional contributions from the existing CAs.

CAs are expected to act as Web notaries, similar to what has been already proposed by other projects, such as the Perspectives Add-On for Mozilla Firefox, or as part of the Convergence project.

A web notary shall be one or multiple servers that are run by a CA. A web notary is expected to make statements about facts that can be discovered on the web.

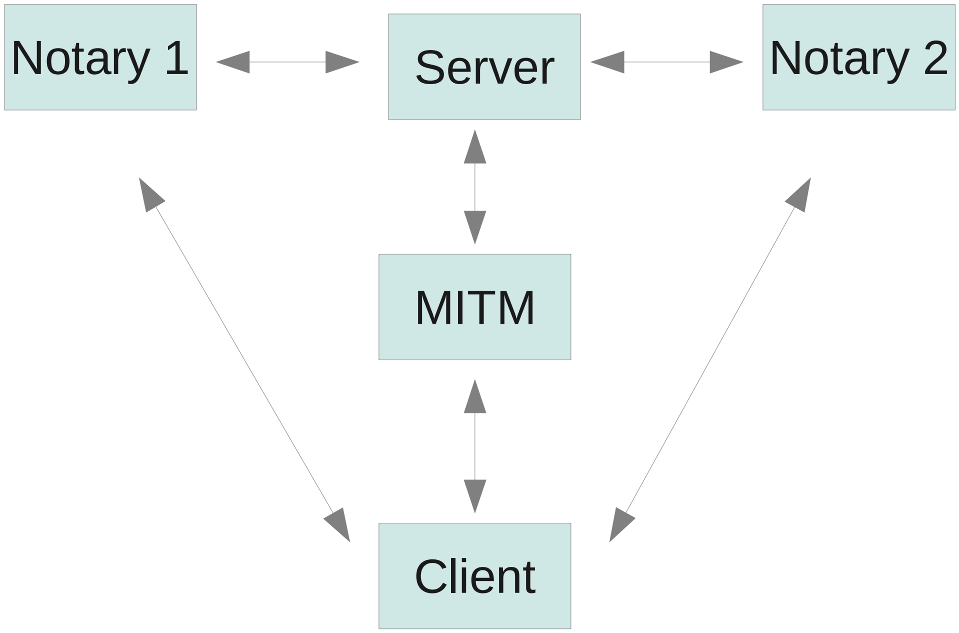


A client connecting to a server will receive the server's certificate as part of the TLS handshake.

A client could contact a notary, and ask the notary for the server certificate that can be obtained from the Notary's perspective, and send this information to the client. The client can compare the perspectives, they should be identical.

A MITM will need to use a different certificate that the MITM controls, in order to read and/or manipulate the data exchanged.

If the connection between client and server is influenced by a MITM, then the certificates seen will be different based on the perspectives. If the information returned by the notary is different than the information seen by the client, then the client probably shouldn't trust the information presented by the server.



MECAI uses the term voucher for the data that is sent from a Notary to a client.

How can a client trust the voucher sent by a Notary? A voucher should carry a digital signature.

Who is allowed to sign vouchers? MECAI proposes that each CA that is allowed to issue server certificates, shall be allowed to act as a web notary and digitally sign vouchers.

Each CA shall use its root certificate to issue a signing certificate for the purpose of signing vouchers.

In MECAI a notary is called a Vouching Authority (VA).

Any CA that is run in the public part of the web is allowed to participate in MECAI and operate a VA.

As usual it's still a decision of software vendors which CAs are trusted by the software.

If the web community agreed that MECAI were a reasonable path to improve today's PKI, then CAs should accept this preference and participate. As soon as a small set of CAs accepted the participation rules described in MECAI and agreed to participate by running a VA, it would be sufficient to start with the development and deployment of the MECAI system.

CAs should have an interest to participate. If a CA decided that it doesn't want to participate, then eventually software vendors (in particular those from the free software and open source world) might decide to exclude the non-contributing CAs from their list of trusted CAs.

Vouchers

The idea of the voucher is to have an independent opinion about the non-revoked state of a certificate, whether it seems to be used by a specific IP address from a different network perspective, and to bundle certificate status information from the original certificate issuer.

A voucher is the data package that is produced by a Vouching Authority and (indirectly or directly) delivered to a client. It shall contain the following information:

- a single IP address (or alternatively one IPv4 address plus one IPv6 address that point to the same server)
- the certificate (or a hash/fingerprint of the certificate) sent by the server in TLS handshakes
- optional: the intermediate certificates (or the hashes/fingerprints of the intermediate certificates) sent by the server in TLS handshakes
- optional: a recent OCSP response for the server's certificate
- optional: a recent OCSP response for each of the intermediate CA certificates sent by the server during the TLS handshakes
- a timestamp that states the time at which the Vouching Authority assembled this information
- a trust context (will be explained later)
- a digital signature, including the VA signing certificate
- (potentially, a proof that the VA signing certificate has not been revoked, such as a fresh OCSP response)

Including a hostname (such as www.my.site) in the voucher might not be helpful. Because of Content Delivery Networks (CDN) and varying DNS information based on network perspective, it might be impossible for a VA to confirm a relationship between a hostname and an IP address.

Other technical proposals:

- Use Cryptographic Message Syntax (CMS, RFC 5652) to represent a digitally signed voucher. The message shall include the VA's certificate that was used to sign the voucher.
- The signed data inside CMS shall be a single data container that allows for flexible contents and space efficiency, for example an ASN.1 encoding following a template defined in a MECAI RFC.

Voucher Authority

The entity producing vouchers, a department of a CA. Also, a certificate issued by a CA for the purpose of creating digital signatures. A VA certificate MUST specify an OCSP responder.

Voucher Authority Server (a Notary)

(VAS) A server hostname that is associated to the VA service run by a CA. A server hostname that has been communicated to software vendors and has been embedded into client and server software products that support the MECAI system. A VAS shall listen to requests, perform notary services for network perspective, create vouchers, sign vouchers, and send the vouchers back to the requesting party. A VAS implements the server side of the Voucher Request Protocol.

Voucher Request Protocol

In order to obtain a voucher, a connection must be established to a Voucher Authority Server (VAS). The requesting party will transmit a voucher request. The request will contain the certificate, the intermediate certificates, the server's IP address. The VAS is expected to keep the connection alive while performing a validation, until it is able to return the voucher.

As soon as the VAS has received a request, it will initiate an (additional) connection to the requested IP address, in order to perform probing. The VAS will perform a TLS handshake with the server at the destination address and verify that it uses the certificate that was mentioned in the voucher request. By being able to perform the handshake, it can be confirmed that this IP address is in possession of the certificate's private key. As an additional (optional?) check the VAS will verify that the list of intermediate certificates sent in the voucher request is identical to the list of intermediate certificates sent by the IP address in the TLS handshake. The VAS will terminate the connection after the TLS handshake has been completed, no application data will be exchanged on the probing connection.

In order to allow for multiple services at a single IP address, with potentially different certificates, the voucher request must contain the destination port number to which the VAS shall connect to perform the TLS handshake probing.

By including a port number in the voucher request, multiple irregular scenarios can be handled.

(a) A server might use separate certificates for distinct services, such as HTTPS, IMAP/SSL, etc.

(b) A server might require to transition from an older certificate to a newer certificate, and need to acquire a voucher for the newer certificate in advance, prior to deploying it.

(c) A server might use separate certificates based on the cryptographic abilities of a client.

(d) Some servers use a variation called STARTTLS, which begin with a protocol specific plaintext handshake, then switch to TLS on the existing network socket. By using a secondary port for the probing performed by the VA, a standard TLS service can be offered in addition to the STARTTLS variant, thereby avoiding the problem that a VA will not be able to speak all variations of STARTTLS protocols.

Voucher Stapling

Similar to RFC 4366, the proposal is to define a TLS handshake extension, where a client and a server can learn that they are both able to exchange vouchers inside the TLS handshake.

Some will argue this will increase the size of TLS handshakes excessively, to an extent where it is no longer possible to have a quick communication between client and server. However, some approaches can be mentioned that may compensate or limit this increase in size. (a) Instead of transmitting full certificates inside the voucher, it should be sufficient to include the fingerprints, because the certificates are contained in the TLS handshake anyway. (b) The client could cache the recently seen vouchers. If the client has the server's certificate cached, and the client also has a sufficiently fresh voucher cached, then the client could decide that no voucher needs to be sent at this time, thereby increasing the speed of connections that happen after the first one. This could work by sending a fingerprint of the server certificate the client has cached as part of the TLS handshake. If the server is using the certificate that matches the fingerprint, the server can skip sending the voucher. (c) See also ietf-tls-cached-info which is another approach to reduce the size of handshakes.

Which party requests the voucher?

The proposal is that web clients (such as browsers or email clients) usually will not request the voucher on their own. Instead each server offering a TLS service should obtain the required vouchers (e.g. twice a day), in advance to accepting client connections, cache the vouchers, and transmit the respective voucher to a client as part of the TLS handshake.

This approach has several advantages:

- no client-initiated side channel is necessary to transmit the voucher from a VAS to a client
- the number of requests that a VAS must be able to handle is greatly reduced
- the VRP could require that the requesting party uses TLS client authentication, where the server uses its own (server) certificate for (TLS client) authentication with the VAS. This could reduce the extent of denial of service attacks, as a VAS might choose to only accept voucher requests from a party that has been able to identify itself (using a certificate that has been issued by one of the publicly known Certificate Authorities).

The proposal is that each server obtains fresh vouchers from each of the publicly known Voucher Authorities, twice a day and cache them. The idea is to be prepared for all vouching requirements of a client. It will be explained later why this helps.

Feedback for an earlier version of the MECAI proposal contained the criticism that some servers might be unwilling to run a software on production servers that will initiate connections to the outside world, or might be unwilling to accept daily changing state on their servers.

One could argue there are multiple approaches that could be used to address this criticism. The program that sends voucher requests and initiates connections to the VAS could be completely isolated from the primary service run on a server. Or two separate machines could be used, where the production service never initiates connections on its own, and a separate machine talks to the VAS. The separate machine could copy the vouchers into a designated directory on the production server, where the production server will find and use them.

The idea of a service requesting certificate status information on its own isn't new. RFC 4366 defines the protocol described as OCSP stapling, which has the same requirements that a server must update its state frequently with recent status information, which will be used in the TLS handshake.

(In case the above arguments are not convincing, and some server operators insist on not supporting the Voucher-Stapling, it can be considered to allow that clients talk to the VAS directly, at the disadvantages that side channels will be required and that VAS have to handle a greater amount of requests.)

Random VA selection

The introduction and requirement of vouchers has the benefit that controlling a single CA will no longer be sufficient. If the presence of a valid voucher were mandatory, at least two CAs would have to be involved to create a working rogue identity, one CA signing the certificate, another CA using it's VA to produce a voucher.

In an implementation where clients connect directly to a VAS in order to obtain a voucher, the client is able to select a VA that is not related to the CA that has issued the server's certificate. But because of the disadvantages of this approach (high latency and side channel), this proposal recommends the use of Voucher Stapling.

However, when using Voucher Stapling, the following threat scenario should be considered.

Let's assume an attacker was able to abuse the power of CA-1. In addition, an attacker is physically close to the VAS of CA-2. The attacker might be able to gain a benefit of the network position and intercept the VAS' probing connection using a system controlled by the attacker. If the VAS used only a single network perspective, then the VAS wouldn't be able to detect an inconsistency, and would give a voucher to the attacker. (It might be recommended that CAs setup multiple VAS and run parallel probing from multiple parts of the web.)

Because of this threat scenario, it shouldn't be the server that makes the decision which VA shall be used for a TLS connection. The proposal is that the client will make this decision.

Client software ships with a list of trusted CAs, and with a list of hostnames that act as a VA on behalf of the CAs.

At the time a client (e.g. a browser) wishes to open a connection to a TLS server, the client can randomly select a small set of candidate VAs that will be accepted by the client for the connection.

The initial proposal is that the client selects a random list of three candidate VAs, and transmits this list of candidate voucher authorities to the server as part of the TLS handshake. The amount of three candidates has been chosen to allow for conflicts and offline VA scenarios.

At the time a client connects to a server, the client cannot know (or cannot be sure) which certificate will be used by the server during the TLS handshake. Therefore the client cannot be sure which CA has issued the certificate that is used by the server.

Because the client cannot be sure which CA has issued the server certificate, the server's certificate might have been issued by one of the CAs that controls one of the VA that have been selected as candidates. Because the idea of the voucher is to have two independent opinions, this candidate must not be used by the server. Another reason why a server might not be able to present a voucher from one of the selected candidates is a temporary CA infrastructure outage, which may have prevented the server from obtaining a recent voucher from a VAS.

By allowing the client to select the acceptable VAs, even a powerful attacker that controls multiple CAs and multiple network routes will succeed only in a limited amount of attack attempts, where the client's random preference happen to be identical with the systems that the attacker can control.

However, a major percentage of connection attempts from potential victims will not satisfy the client's voucher expectations, and their client software can reject the connections and report the confirmation to the user. Users can decide to report such incidents, or client software could report such incidents automatically, based on user's and environment's preferences.

Such awareness and reports can lead to quick detection of MITM attacks and awareness of the CA involved that issued a rogue certificate.

As soon as the information about a rogue certificate has been confirmed (probably by involving the domain owner, who will make the final decision whether a certificate is rogue or not), the certificates in use can quickly be revoked, and no more vouchers will be issued by Voucher Authority Servers.

If required, if a major security incident has occurred where a CA's root key has been stolen, or has been used to issue certificates without revocation information hints, the CA's root key could be revoked at a global level, by informing all known VAS that it shall no longer be trusted, and VAS can stop issuing vouchers for the affected CA without further need for revocation checking of related certificates.

As soon as all valid vouchers have expired, clients can stop trusting certificates abused by a MITM.

Regional VA selection

Instead of a random selection, clients might optionally decide to select a set of candidate VAS based on their own geography and the destination server. For example, a person living in country C, wishing to visit a server in country D, might decide that it doesn't trust statements from country C for this connection.

VAS notifies domain owners

Let's consider the following scenario: The real website of of domain D has IP address 1.1.1.1 in country C1. The attacker's position is in country C2 and wants to attack the citizens living in country C2.

The attacker could abuse the power of any CA and get a certificate for domain D and install the certificate on IP address 2.2.2.2. In addition the attacker manipulates the DNS of the victims, so that D resolves to 2.2.2.2.

Because the attacker's server can be reached from anywhere in the world, valid vouchers from all VAS can be obtained. Victims will accept the server certificate and all vouchers.

In an earlier version of the MECAI proposal it had been proposed that a VAS performs a DNS plausibility test, to check that hostname and IP address are in relation. However, because of Content Delivery Networks (CDN), which can result in different IP address information for a single hostname, Voucher Authority Servers will often be unable to confirm that association.

In order to solve this problem, each Vouching Authority should keep a local log of all certificates that have ever been seen in at least one valid voucher request (and that have not yet expired). The very first time a certificate is seen, the VAS is supposed to reply to the requesting party with a „new certificate, come back later", and put the new certificate on hold, for a proposed period of 3 days. In addition, the VAS will send an automated, digitally signed email message, to each of the email addresses found in the public domain registry. The message shall contain a subject with the text „certificate for [hostname]: [fingerprint]", where [hostname] is the first hostname to be found in the certificate seen by the VAS in a voucher request, and where [fingerprint] is the certificate's fingerprint (e.g. SHA1).

The body of the email should contain the full certificate, the requesting IP address, and other interesting data found in the voucher request. The message shall be digitally signed in the S/MIME format, allowing recipients to reject all non-signed messages.

The domain owner can setup a mail filter that will filter or ignore all messages with the known good fingerprints.

If the certificate is a rogue certificate, the domain owner is expected to contact the CA that issued the rogue certificate. The domain owner should also report the rogue certificate to public trust watchers, such as the EFF SSL Observatory, or to report points operated by other CAs. The domain owner will need to present a proof of possession of the domain, to ensure the complaint will be accepted and an investigation started.

If a domain owner chooses to simply ignore all incoming reports that contain the expected fingerprints in the subject of the email messages, the VAS will automatically start issuing vouchers for the certificate after the initial hold period (suggested: 3 days).

The domain owner can trigger the start of the hold period by configuring servers to start requesting vouchers for the new certificate.

(If a domain owner wishes to accelerate the hold period, the domain owner could pay a higher amount of money for quicker verification. For example, the CAs could invent a peer system that is only used when a certificate applicant verification involved a phone call, and it would require a phone communication with two additional CAs, after which they agree to produce digitally signed statements, exchanged between CAs, that informs all CAs that the the hold period can be skipped.)

If a domain owner follows the advice to prepare for revocation events of major CAs and obtains a (set of) backup certificate(s) in advance, there shouldn't be a reason to obtain new certificates quicker than

3 days.

Exception: If the new certificate is a renewed certificate, in other words, if a certificate uses the same public key as the previously seen certificate, including the same (or a smaller) set of hostnames, not including any new hostnames, and the renewed certificate has been issued by the same CA, then the hold period is probably unnecessary.

(As an optional service, customers might request notification if certificates are seen on unexpected IP addresses.)

Trust Contexts

Whenever a major security incident occurs, there might be a dispute whether the evidence is sufficient to justify a global distrust of a Certificate Authority.

Each software vendor has their own list of trusted authorities, based on varying policies.

In the event of a security incident, software vendor A might decide to distrust a CA, but software vendor B might decide to keep the trust.

A CA-1 receiving a request to no longer issue vouchers for another attacked CA-2 might be unsure whether it shall comply with the request.

In order to solve this dilemma, the proposal is to introduce a trust context. A Trust Context Identifier (TCI) can be assigned for each large software vendor running their own list of trusted CAs. For example, identifier MOZ could be used to identify the trust list maintained as part of the Mozilla CA Policy.

At the time a VAS receives a request for a voucher, the VAS decides which TCIs can be included in the voucher. If Mozilla made the decision to distrust a CA-2, then CA-1 could still issue vouchers, however, they would no longer include the MOZ TCI.

A client requesting a voucher (such as a Mozilla browser) would only accept a received voucher if it contained its expected TCI (such as MOZ).

Verification performed by clients

Clients will verify that:

- the voucher presented is reasonable fresh (e.g. not older than 24 hours)
- the voucher signature can be successfully verified
- the voucher was indeed created by one of the candidates selected by client
- the certificate information inside the voucher matches the data being used during the TLS handshake with the server
- the voucher was created for the IP address being visited
- the voucher contains the expected Trust Context Identifier

If the voucher verification was successful, then all certificate status information (e.g. not revoked) contained in the voucher will be imported by the client and used for subsequent validations.

Clients will validate the server certificate and require that reasonably fresh revocation status information is available in the cache.

If all checks were successful, the connection can continue. If at least one problem was found, the behavior will depend on the client's configuration. In an early stage of MECAI deployment, only a warning information might be shown in the client's user information. In a later stage, the client should reject connections by default.

Not solved by MECAI

Following is a list of problems that aren't solved by MECAI itself, with comments.

Redundancy: After a CA gets revoked (and vouchers will no longer be issued), clients will no longer accept the related certificates from servers. How can web sites ensure that their customers will still be able to connect with security warnings? Either servers need to have an emergency plan and be able to quickly deploy a (set of) backup certificates (which the web site operator should have obtained in advance and kept handy), or the TLS protocol could be enhanced to allow clients to request an alternative secondary certificate, potentially allowing for zero downtime.

What happens if an attacker is very close to the server being attacked: The threat is that an attacker could abuse its geographic position, intercept all communication between a CA and the server under attack, and use it to obtain a minimum Domain Validation (DV) certificate. This problem could be solved at an organizational level. Proposal: In addition to the minimum DV certificates offered by CAs today, a new minimum Domain Validation Plus (DV+) could be established. The idea is to require an out-of-band communication channel between the CA and the domain owner, such as a phone call. Without requiring any organizational validation, without requiring any identity validation, it could be required that a CA must call each of the phone numbers visible in the domain registry, and inform each phone number about the requested certificate. One of the phone numbers must be controlled by the individual who requested the certificate. The CA will provide a transaction code via phone, which the individual must enter into a CA web form.

CAs inside Intranets / Corporate Networks: The principle of obtaining vouchers cannot be used for servers behind a Firewall, as external notary systems run by public CAs are unable to connect to such private servers. However, in order to run an attack against such servers, the attacker must be able to intrude into the private network and abuse the private CA. Inside a private network it should be possible to guarantee the uptime of OCSP infrastructure for the closed user group. Client software might offer a combined configuration that (a) relaxes the voucher requirement for the intranet domain and (b) pins the intranet domain to the private CA (c) strictly requires availability of fresh OCSP information. With this configuration, a hacker being able to abuse the powers of an external CA will not be able to trick local users with certificates issued by an external CA. The private environment can use appropriate measures to watch the Intranet systems and its CA, because it has full control over all involved systems. It can require out-of-band confirmation from employees for any issuance of certificates by the corporate CA.