CA-TMS Dokumentation

Haixin Cai, Jannik Vieten, Pascal Weisenburger
Wintersemester 2013 / 2014

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Theoretische Informatik
Kryptographie und Computeralgebra

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Zusammenfassung

Dieses Dokument stellt die Dokumentation zum *CA Trust Management System (CA-TMS)* dar und gibt Hinweise zur Benutzung sowie zur Architektur der entwickelten Komponenten. Dabei handelt es sich um eine Java-Anwendung die das Trust-Management übernimmt und eine Firefox-Extension, die die CA-TMS-Funktionalität im Browser benutzbar macht. Die hier vorgestellte Implementierung entstand im Rahmen eines Praktikums bei der CDC-Gruppe des Fachbereichs Informatik an der TU Darmstadt. Programmiert wurde das System von Haixin Cai, Jannik Vieten und Pascal Weisenburger, die Betreuung fand statt durch Johannes Braun und Moritz Horsch.

Inhaltsverzeichnis

1 Einleitung

Die heute im Internet eingesetzte *Public-Key-Infrastruktur (Web-PKI)*, die auf der Integrität von *Certificate Authorities (CAs)* basiert, gerät zunehmend in die Kritik, nicht ausreichend sicher und vertrauenswürdig zu sein. So kann jede CA ein Zertifikat für jede beliebige Domain ausstellen, was Angesichts der Einflussnahme von staatlichen Behörden auf einzelne CAs die Gefahr für Man-in-the-Middle-Angriffe real werden lässt.
In [1] stellen Braun et al. ein alternatives Trust-Modell vor, welches dieses Problem adressiert und der Vertrauensentscheidung einen benutzerspezifischen, individuellen *Trust View* zu Grunde legt, der sich den Bedürfnissen des Benutzers anpasst.
Dieses *CA Trust Management System (CA-TMS)* wurde nun implementiert und besteht im Wesentlichen aus zwei Komponenten. Die erste ist eine Java-Anwendung, die das Trust-Management übernimmt, die nötige Funktionalität für das Bewerten von Zertifikaten bereitstellt und sich vom Benutzer konfigurieren lässt. Die andere Komponente wird von einer Firefox-Extension gebildet, die auf die Trust-Berechnung der Java-Anwendung zurückgreifen kann, um während dem Surfen im Internet die Nutzung des CA-TMS möglich zu machen.

2 CA-TMS-Anwendung

Im Folgenden wird das CA-TMS-System beschrieben.

2.1 Entwicklungsumgebung

Die Anwendung baut auf Java 7 auf. Da zur Entwicklung Eclipse genutzt wurde, liegt ein Eclipse-Projekt vor. Die gesamte Anwendung lässt sich aber auch mit Hilfe einer Ant-Build-Datei kompilieren. Die dabei generierte `JAR`-Datei enthält alle nötigen Abhängigkeiten. Dabei handelt es sich um die folgenden Bibliotheken:

Bouncy Castle main package (bcprov-jdk15on-151.jar) wird benötigt, um CRLs und OCSP-Anfragen zu parsen

CertainTrust SDK (CertainTrustSDK-1.0.jar) ist das grundlegende Toolkit zur Modellierung/Berechnung von Computational Trust

Java JSON API (javax.json-1.0.4.jar) wird benötigt, um eingehende Validierungsanfragen (von der Firefox-Extension) zu lesen, die in JSON kodiert sind

Java JDBC connection pool manager (miniconnectionpoolmanager.jar) ist ein einfacher Manager, um JDBC-Datenbankverbindungen zu verwalten

SQLite connection pool (sqlite-connection-pool_v1.1.0.jar) stellt einen Pool für SQLite-Verbindungen zur Verfügung, der im Zusammenspiel mit dem *Java JDBC connection pool manager* genutzt wird, um einen Pool von wiederverwendbaren und potentiell nebenläufigen SQLite-Verbindungen zu verwalten

SQLite JDBC (sqlite-jdbc-3.7.2.jar) stellt den Zugriff auf SQLite-Datenbanken zur Verfügung, der zur persistenten Daten-Speicherung genutzt wird

Notary module (sslcheck-1.0-jar-with-dependencies.jar) stellt das Modul zur Abfrage von Notaries zur Verfügung

2.2 Überblick

Das System besteht im Wesentlichen aus

- dem Datenmodell zur Verwaltung des *Trust Views* und der Benutzerkonfiguration
- der Trust-Validierung zur Berechnung der Vertrauenswerte im Trust-Modell
- Bindings zur Kommunikation mit Anwendungen, damit diese das Trust-Modell benutzen können
- einer graphischen Oberfläche, mit deren Hilfe der Benutzer den Trust View einsehen und verwalten kann

Abbildung 1 bietet einen Überblick über diese Komponenten und deren Interaktion.

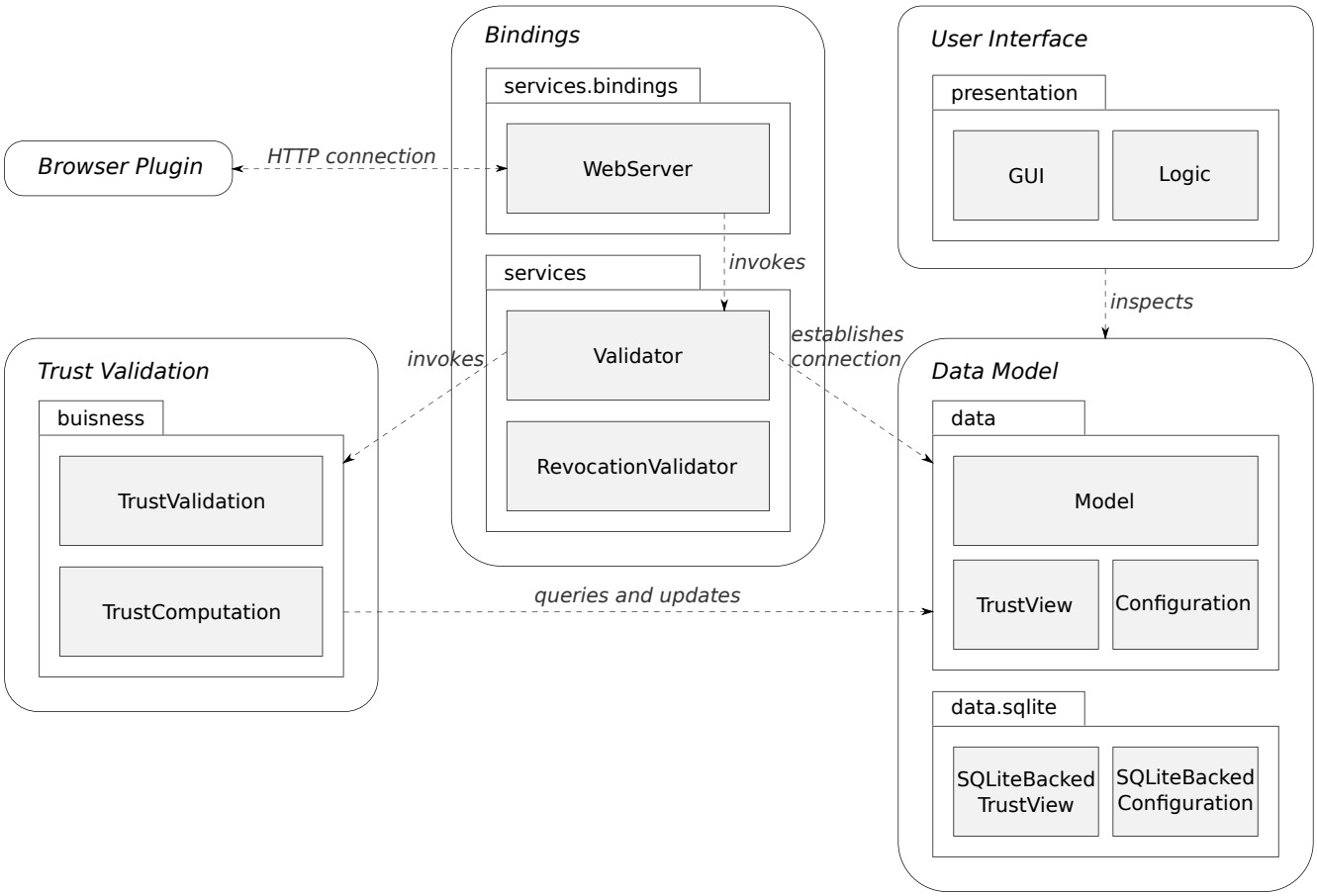

Abbildung 1: Überblick über die Komponenten des CA Trust Management Systems

2.3 Datenmodell

Das Datenmodell verwaltet den Trust View, in dem die *Public Key Trust Assessments* und die Menge vertrauenswürdiger und nicht vertrauenswürdiger Zertifikate gespeichert sind, wie in [1], Abschnitt 4.2 beschrieben.

Darüber hinaus stellt das Datenmodell den Zugriff auf die Benutzerkonfiguration zur Verfügung. Standardwerte sind in der Datei `configuration.properties` im Wurzelverzeichnis der Applikation/JAR-Datei festgelegt und können dort modifiziert werden. Vom Benutzer angepasste Werte werden von der zugrunde liegenden Implementierung des Modells an einem benutzerspezifischem Ort gespeichert, wie dies auch mit dem Trust View geschieht.

Die momentane Implementierung des Modells in den Klassen `data.sqlite.SQLiteBackedTrustView` und `data.sqlite.SQLiteBackedConfiguration` nutzt eine SQLite-Datenbank zum Speichern der Daten. Die Datenbank-Datei `catms.sqlite` wird abgelegt im Verzeichnis:

Windows %APPDATA%\catms

OS X ~/Library/Application Support/catms

andere (falls $XDG_DATA_HOME gesetzt) $XDG_DATA_HOME/catms

andere (falls $XDG_DATA_HOME nicht gesetzt) ~/.local/share/catms

Die Implementierung des Datenmodells kann ggf. einfach gegen eine andere ausgetauscht werden. Dazu muss lediglich die Klasse `data.Model` angepasst werden, sodass eine andere Implementierung genutzt wird, welche Realisierungen der Interfaces `data.TrustView` und `data.Configuration` bereitstellt.

Zentraler Zugriff auf das Datenmodell erfolgt über die Klasse `data.Model`. Über die Methoden `openTrustView` und `openConfiguration` können Instanzen der Interfaces `TrustView` bzw. `Configuration` geöffnet werden. Nach dem Ende der Transaktion also dem Abfragen und Aktualisieren des Datenmodells, müssen die geöffneten Instanzen wieder geschlossen werden. Es ist dabei sicher gestellt, dass die Transaktion atomar ist, sie also komplett ausgeführt oder komplett nicht ausgeführt wird und die Daten somit konsistent bleiben.

2.4 Trust-Validierung

Die Algorithmen zur Berechnung der Trust-Werte und zur Trust-Validierung sind in der Klasse `buisness.TrustComputation` implementiert. Es handelt sich dabei um die in [1], Abschnitte 4.3, 4.4 und 4.5 beschriebenen Algorithmen der Initialisierung der Trust Assessments, der Trust-Validierung und der Trust-View-Aktualisierung. Eine Anfrage zur Trust-Validierung wird nach dem in Abb. 2 dargestellten Algorithmus verarbeitet.

Die Standardpfadvalidierung wird von Firefox ausgeführt und das Resultat wird der Java-Applikation übermittelt. Diese prüft die Gültigkeit des Zertifikatspfades nicht mehr selbst, sondern verlässt sich auf die Validierung, die von Firefox ausgeführt wurde. Dies bedeutet insbesondere auch, dass für die Gültigkeit der Root Certificate Store von Firefox maßgeblich ist und nicht der der Java Runtime.

Bei der Verarbeitung von Anfragen zur Trust-Validierung wird auf eine interne *Watchlist* für Zertifikate und auf Notare als externe Zertifikatsvalidierungsdienste zurückgegriffen. Darüber hinaus werden verschiedene Spezialfälle identifiziert, unter denen ein Zertifikat direkt als vertrauenswürdig eingestuft werden kann.

2.4.1 Watchlist für Zertifikate

Die Watchlist enthält Zertifikate, die vorläufig als vertrauenswürdig eingestuft sind (aufgrund einer Nutzerentscheidung oder weil einem anderen Zertifikat für den selben Host und Schlüssel bereits vertraut wird). Die Vertrauenswürdigkeit von Zertifikaten auf der Watchlist wird nach einer bestimmten Zeitspanne erneut bewertet.

2.4.2 Validierungsdienste für Zertifikate

Die Algorithmen zur Trust-Validierung und der Trust-View-Aktualisierung können zusätzliche Informationen über die Vertrauenswürdigkeit eines Zertifikats einholen. Abhängig vom festgestellten *Spezialfall* und davon, ob der *Bootstrapping-Modus* aktiviert ist, wird das Host-Zertifikat als *vertrauenswürdig* oder *unbekannt* eingestuft oder externe Validierungsdienste (Notare) werden angefragt.

Validierungsdienste werden ausschließlich für Host-Zertifikate angefragt, da die Notar-Dienste für CA-Zertifikate keine verwertbaren Informationen liefern.

2.4.3 Spezialfälle für Zertifikate

Für ein durch einen *Trust Validation Request* gegebenes Host-Zertifikat *H* wird versucht ein zugehöriges Zertifikat *C* für den selben Host wie *H* innerhalb der Menge vertrauenswürdiger Zertifikate im Trust View zu finden:

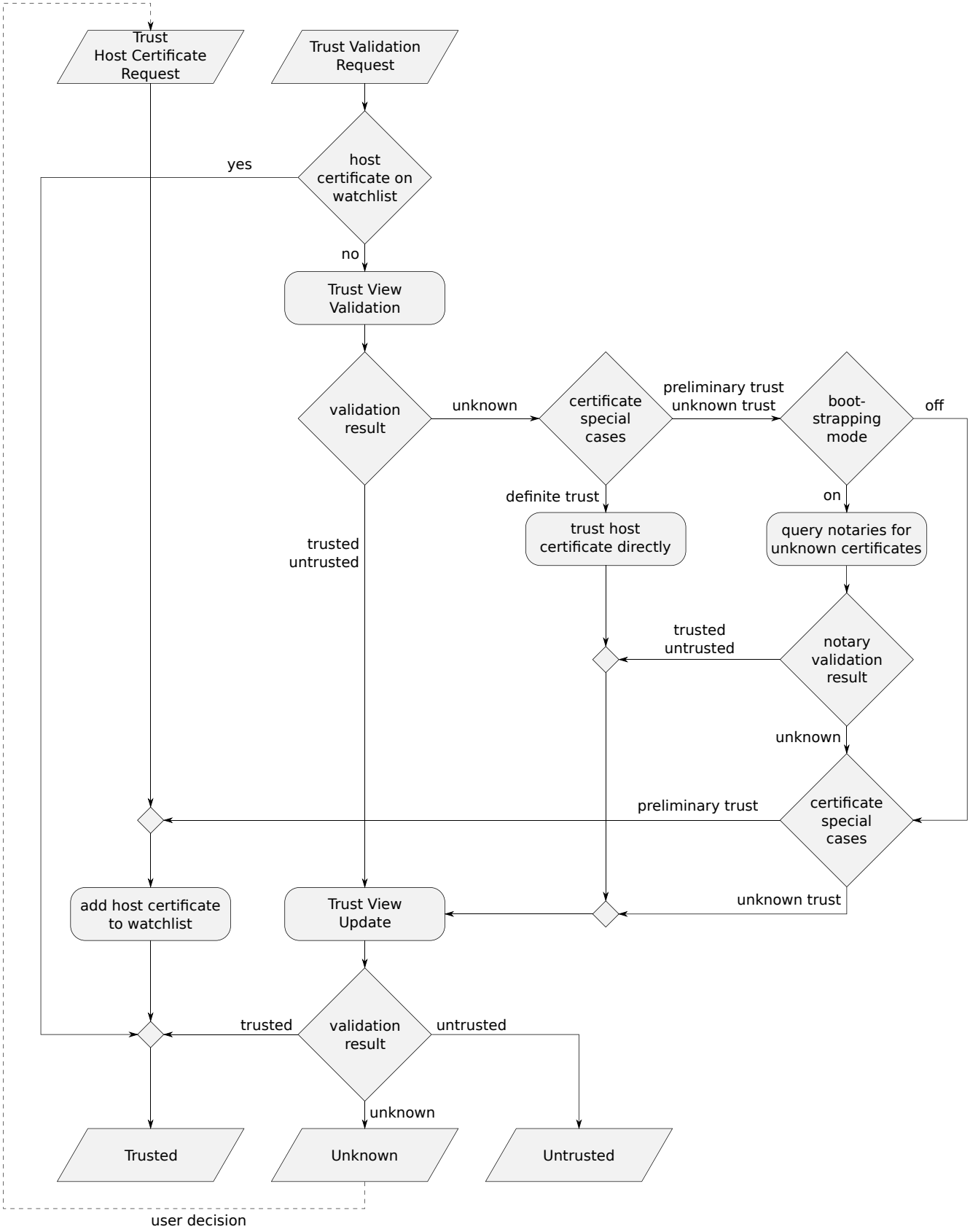

Abbildung 2: Verarbeitung von Anfragen zur Trust-Validierung

1. **existierendes Zertifikat für den Host mit demselben öffentlichen Schlüssel von derselben CA**

   Falls ein Zertifikat *C* existiert, das abgelaufen, aber nicht revoziert ist, mit demselben *Issuer* und demselben *Public Key* wie *H*, wird *H* als **definitiv vertrauenswürdig** eingestuft.

2. **existierendes gültiges Zertifikat für den Host mit demselben öffentlichen Schlüssel**

   Falls ein Zertifikat *C* existiert, das weder revoziert noch abgelaufen ist, mit einem anderen *Issuer* als *H*, aber demselben *Public Key* wie *H* (CA-Wechsel), wird *H* als **vorläufig vertrauenswürdig** eingestuft.

3. **existierendes abgelaufenes Zertifikat für den Host von derselben CA**

   Falls ein Zertifikat *C* existiert, das abgelaufen oder revoziert ist, mit demselben *Issuer* wie *H* (wahrscheinlich normaler Zertifikatswechsel), wird *H* als **vorläufig vertrauenswürdig** eingestuft.

4. **existierendes gültiges Zertifikat für den Host von derselben CA**

   Falls ein Zertifikat *C* existiert, das weder revoziert noch abgelaufen ist, mit demselben *Issuer* wie *H* (nebeneinander mehrere gültige Zertifikate von derselben CA), wird die Vertrauenswürdigkeit von *H* als **unbekannt** eingestuft.

5. **existierendes Zertifikat für den Host**

   Falls ein Zertifikat *C* existiert, mit einem anderen *Issuer* und einem anderen wie *Public Key* als *H*, wird die Vertrauenswürdigkeit von *H* als **unbekannt** eingestuft.

6. **erstes Zertifikat für den Host**

   In allen anderen Fällen (erstes Mal, dass ein Zertifikat für den Host verarbeitet wird), wird die Vertrauenswürdigkeit von *H* als **unbekannt** eingestuft.

2.5 Bindings

Bindings stellen eine Schnittstelle für andere Anwendungen zur Verfügung, um das Trust-Modell nutzen zu können. Die allgemeine Funktionalität, die von allen Bindings genutzt werden kann, wie der Zugriff auf die Trust-Berechnung-Komponente und das Datenmodell, um die Trust-Validierung auf dessen Basis auszuführen, ist in der Klasse `services.Validator` implementiert. Momentan existiert ein Binding `services.bindings.WebServer`, das einen simplen WebServer zur Verfügung stellt, über den das ebenfalls im Rahmen dieses Projektes entwickelte Firefox-Plugin mit dem CA-TMS kommunizieren kann, um den Service der Trust-Validierung zu nutzen. Der Webserver kann über die graphische Oberfläche gestartet werden.

2.6 Graphische Oberfläche

Das CA-TMS kann über die grafische Benutzeroberfläche gesteuert werden, die sich in `presentation.ui` befindet. Wichtigste Funktion ist das Starten des Webservers, über den die Firefox-Extension mit der Java-Anwendung kommunizieren kann. Außerdem ist das Ansehen und Bearbeiten von vertrauenswürdigen und unvertrauenswürdigen Zertifikaten sowie den *Trust Assessments* möglich. In der Konfigurationsansicht können die Werte für die Security-Level und den TCP Port angepasst werden. Zudem ist es möglich, die Datenbank, in der der Trust View gespeichert ist, zu löschen oder ein Backup zu erstellen bzw. wieder einzuspielen. Die Nutzung des Notary-Moduls ist freiwillig und kann, insbesondere zu Testzwecken, mit einem gewünschten Ergebnis überschrieben werden. Dafür bietet die grafische Oberfläche eine Einstellungsmöglichkeit.

3 Firefox-Extension

Im Folgenden wird die zum CA-TMS-System gehörige Firefox-Extension beschrieben.

3.1 Überblick

Standardmäßig akzeptiert Firefox alle gültigen Zertifikate, die von beliebigen, im Browser registrierten CAs ausgestellt wurden. Das sind inzwischen sehr viele und diese Tatsache liegt dem Vertrauens-Problem der derzeitigen Web-PKI zu Grunde. Um das alternative Vertrauenskonzept der Trust Views zu benutzen, integriert sich die Firefox-Extension in den Browser und delegiert die Trust-Berechnung an die CA-TMS-Anwendung, um anschließend angemessen auf das Ergebnis zu reagieren und den Benutzer ggf. zu warnen.

3.2 Installation

Die Installation der Extension funktioniert wie von Firefox-Addons gewohnt. Liegt keine fertige .xpi-Datei vor, muss diese zunächst erzeugt werden. Dazu wird der Inhalt des Ordners, der die Extension enthält, in ein zip-Archiv gepackt. Die Dateiendung sollte anschließend in .xpi geändert werden. Es sollte nun also eine Datei `trustviewextension@cdc.informatik.tu-darmstadt.de.xpi` existieren, in der sich auf oberster Ebene (unter anderem) die Datei `install.rdf` befindet. Die .xpi Datei kann dann in den Firefox gezogen werden und wird folglich installiert.

3.3 Benutzung

Während der Installation der Extension wird ein zusätzlicher Button in die Toolbar des Browsers integriert. Über diesen Button wird ein Menü bereitgestellt, in dem Einstellungen vorgenommen werden können. Dort lässt sich das Sicherheitslevel auf eine der drei Stufen „hoch“, „mittel“ oder „niedrig“ einstellen, was das Sicherheitsbedürfnis des Benutzers für die Webseiten widerspiegeln soll, die als nächstes besucht werden. Für Online-Banking oder e-Commerce dürfte beispielsweise ein höheres Level erforderlich sein, als für das Lesen einer Nachrichtenseite. Das ausgewählte Sicherheitslevel geht dann als Parameter in die Vertrauensberechnung der CA-TMS-Anwendung ein. Außerdem lässt sich über das Menü das Konfigurationsfenster der Extension aufrufen, in dem die Verbindungsinformationen (Host und Port) zur CA-TMS-Anwendung geändert werden können, mit der die Extension über HTTP kommuniziert.
Wird eine mit TLS geschützte Seite aufgerufen, nimmt die Extension Kontakt mit der CA-TMS-Anwendung auf und erwartet von ihr das Urteil, ob die Seite bzw. das ausgelieferte Zertifikat vertrauenswürdig ist. Ist das nicht der Fall, wird der Benutzer auf eine Warnseite umgeleitet, die den Sachverhalt erklärt, dem Benutzer aber gleichzeitig die Möglichkeit gibt es mit geändertem Sicherheitslevel erneut zu versuchen, oder die Website in jedem Fall zu besuchen. Bei letzterer Möglichkeit wird die betroffene Website für die Dauer der Browsersitzung von weiteren Trust-Überprüfungen ausgenommen.
Ist die CA-TMS-Anwendung nicht erreichbar, wird ebenfalls eine Warnseite angezeigt, die den Benutzer dazu auffordert die CA-TMS-Anwendung zu starten oder die Extension zu deaktivieren.
In dem Fall, dass die Anwendung das Zertifikat nicht korrekt validieren kann (das Zertifikat also unknown ist), gelangt der Benutzer auf eine Seite, auf der er das Zertifikat manuell als vertrauenswürdig markieren kann.

3.4 Architektur

Im Folgenden wird auf die einzelnen Komponenten der Extension eingegangen und die ihr zugrunde liegende Architektur beschrieben.

3.4.1 Namespace

Um Namenskonflikte bei der Wahl von Bezeichnern wie Funktionsnamen zu vermeiden, befinden sich alle zur CA-TMS-Extension gehörigen JavaScript-Objekte innerhalb des Namespace `TVE` (kurz für Trust-Views-Extension). Beispiele hierfür sind die Objekte `TVE.SSLListener` und `TVE.UI`. In den folgenden Beschreibungen wird das Präfix TVE meist weggelassen.

Der Namespace wird innerhalb der Datei `setupNamespace.jsm` angelegt, der als JavaScript-Modul [4] geschrieben wurde, s. d. es keine Probleme mit dem Scoping über mehrere XUL-Seiten hinweg gibt. Außerdem wird beim Setup auch der entsprechende Preferences-Branch and den Namespace gebunden.

3.4.2 Initiierung

Nachdem der Namespace erzeugt und alle nötigen Skripte geladen wurden, wird die Datei `init.js` ausgeführt. Dort wird das Objekt `SSLListener` als WebProgressListener registriert um über HTTPS-Seitenaufrufe informiert zu werden. Diese Registrierung muss jedoch durch die Methode `addTabsProgressListener()` [2] erfolgen, um auch mit mehreren Tabs zu funktionieren.
Wird `init.js` das erste Mal nach der Installation ausgeführt, werden noch zusätzliche Schritte unternommen um die Extension für die erste Benutzung vorzubereiten. Dazu wird der Button der Extension zur Toolbar von Firefox hinzugefügt und das Security-Level wird auf „mittel“ eingestellt. Die wesentliche XUL-Datei bis zu diesem Punkt ist `browserOverlay.xul`.

3.4.3 SSLListener

In der Datei `sslListener.js` befindet sich das Objekt `SSLListener`, welches bereits von `init.js` als WebProgressListener [3] registriert wurde. `SSLListener` wird aktiv, wenn eine neue HTTPS-Verbindung aufgebaut wird und regelt dann das weitere Vorgehen der Extension. Das CA-TMS wird nur kontaktiert, wenn Firefox/NSS die Zertifikatskette erfolgreich validieren konnte. Für die weitere Validierung durch das CA-TMS werden neben der URL und dem Validierungsergebnis von Firefox/NSS das vom Benutzer gesetzte Security-Level und die Zertifikatskette benötigt. Letztere wird vom `CertHandler` ausgelesen. Die Kommunikation mit dem CA-TMS wird dann vom `CATMSCommunicator` durchgeführt, der entsprechende Callback-Funktionen im `SSLListener` aufruft, um das Ergebnis der Validierung verfügbar zu machen. Dieses Ergebnis ist entweder „TRUSTED“, „UNKNOWN“ oder „UNTRUSTED“. In letzteren beiden Fällen wird das `State` Objekt angewiesen entsprechende Warnseiten anzuzeigen. Schlägt die Kommunikation mit dem CA-TMS fehl, etwa weil der Webservice nicht erreichbar ist, wird stattdessen eine entsprechende Fehlerseite angezeigt.

3.4.4 Certificate Handler

Der `CertHandler` aus der Datei `certHandler.js` stellt eine Funktion bereit, um die Zertifikatskette der aktuellen Verbindung auszulesen. Zurückgegeben wird dabei ein Array, welches alle Zertifikate der Kette enthält. Jeder Eintrag ist wiederum ein Bytearray welches ein Zertifikat im DER-Format [6] repräsentiert. Das erste Zertifikat im Array ist das der Root-CA, das letzte Zertifikat das des Servers.

3.4.5 Kommunikation mit dem CA-TMS

In `catmsCommunicator.js` ist das Objekt `CATMSCommunicator` definiert, welches Validierungsanfragen an die CA-TMS-Anwendung stellen kann. Kommuniziert wird dabei über HTTP. Der `CATMSCommunicator` packt die erforderlichen Parameter (URL, Zertifikatskette, Standardvalidierungsergebnis, Security-Level, Flag um unbekannte Zertifikate zu akzeptieren) in ein JSON-codiertes Objekt und verschickt dieses per HTTP-POST mittels eines `XMLHttpRequest` [5]. Außerdem werden die `onload` und `onerror` Callback-Funktionen gesetzt, über die der aufrufende `SSLListener` vom Ergebnis erfährt.

3.4.6 State

Das in `state.js` definierte Objekt `State` ist dafür zuständig Warnseiten anzuzeigen, sowie eine Liste temporär erlaubter Seiten zu managen, die während der aktiven Sitzung von der Trust-Überprüfung ausgenommen werden.
Schlägt der Verbindungsaufbau mit dem CA-TMS fehl, wird eine Warnseite angezeigt, auf der dem Benutzer erklärt wird, dass der Webservice aktiviert werden sollte.

Wird hingegen ein Zertifikat für unvertrauenswürdig erachtet, wird der Benutzer auf der Warnseite gewarnt und erhält die Möglichkeit, es mit geändertem Sicherheitslevel erneut zu versuchen, oder die Warnung zu ignorieren und die Website dennoch zu besuchen. In letzterem Fall wird die entsprechende Seite für die aktive Sitzung von Trust-Überprüfungen ausgenommen.
Ist das Ergebnis unknown, gibt es eine Fehlerseite, auf der der Benutzer das Zertifikat manuell als vertrauenswürdig kennzeichnen kann. Dazu wird ein Flag gesetzt, welches bei der nächsten Trust-Validierung mitgesendet wird.

3.4.7 Optionen

In `options.xul` ist ein Einstellungsfenster definiert, in dem sich die Verbindungsinformationen zum CA-TMS verändern lassen. Es ist möglich, die Einstellungen auf die Standardwerte zurückzusetzen. Diese Funktionalität stellt das `UI` Objekt aus `ui.js` bereit.

Literatur

[1] Johannes Braun, Florian Volk, Johannes Buchmann, and Max Mühlhäuser. Trust views for the web pki. 2013.

[2] Mozilla Developer Network. Listening to events on all tabs. https://developer.mozilla.org/en-US/docs/Listening_to_events_on_all_tabs. [online, accessed 14-March-2014].

[3] Mozilla Developer Network. nsIWebProgressListener. https://developer.mozilla.org/en-US/docs/XPCOM_Interface_Reference/nsIWebProgressListener. [online, accessed 14-March-2014].

[4] Mozilla Developer Network. Using JavaScript code modules. https://developer.mozilla.org/en-US/docs/Mozilla/JavaScript_code_modules/Using. [online, accessed 14-March-2014].

[5] Mozilla Developer Network. Using XMLHttpRequest. https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest. [online, accessed 15-March-2014].

[6] Wikipedia. Abstract Syntax Notation One. https://de.wikipedia.org/wiki/Abstract_Syntax_Notation_One. [online, accessed 15-March-2014].