**Zentrum für sichere Informationstechnologie – Austria**
**Secure Information Technology Center – Austria**

A-1030 Wien, Seidlgasse 22 / 9
Tel.: (+43 1) 503 19 63–0
Fax: (+43 1) 503 19 63–66

A-8010 Graz, Inffeldgasse 16a
Tel.: (+43 316) 873-5514
Fax: (+43 316) 873-5520

http://www.a-sit.at
E-Mail: office@a-sit.at

DVR: 1035461 ZVR: 948166612 UID: ATU60778947

# FIREFOX-PLUGIN ZUR ANZEIGE VON SICHERHEITSRELEVANTEN INFORMATIONEN

Version 1.0, 21. Juli 2015
Johannes Feichtner – johannes.feichtner@a-sit.at

**Zusammenfassung:** Browser beschränken sich bei der Anzeige sicherheitsbezogener Informationen über Webseiten zumeist auf ein Minimum. Obwohl deutlich erkennbar ist, ob die Verbindung zu einer Seite verschlüsselt ist oder nicht, werden detailliertere Informationen darüber ausgespart oder sind nur umständlich auslesbar. Als Konsequenz wurde in diesem Projekt ein modular erweiterbares Browser-Plugin entwickelt, das aufgerufene Domains auf sicherheitskritische Aspekte hin untersucht und die Ergebnisse für sicherheits-affine Anwender visuell ansprechend aufbereitet. Der Fokus wurde dabei auf das Auslesen und Auswerten von Daten gelegt, deren Werte essentiell für die vorliegende Verbindungssicherheit sind. Für jede aufgerufene Seite sind somit auf einen Blick Details zur eingesetzten Cipher Suite, verwendete Signaturalgorithmen, sowie die komplette Zertifikatskette ersichtlich. Ein Exzerpt dieser Daten wird auch für alle Domains angezeigt, die von einer aufgerufenen Domain eingebunden werden.

## Inhaltsverzeichnis

# 1. Einleitung

Gängige Web-Browser vermeiden es, sicherheitsrelevante technische Informationen über aufgerufene Webseiten in den Vordergrund zu rücken. Dies ist insofern nachvollziehbar, da ein Gros der Anwender mit einschlägigen Sicherheitsaspekten nicht vertraut ist. Eine Gemeinsamkeit aktueller Browser ist die Kennzeichnung von HTTPS-Verbindungen über ein „Schloss-Symbol“ in der Adresszeile. Je nachdem in welchem Zustand dieses Symbol dargestellt wird, ist für gewöhnliche Nutzer erkennbar, ob eine geschützte Verbindung vorliegt oder nicht. Technische Daten werden dabei vollständig abstrahiert und vom Anwender ferngehalten. Erst durch Klicks auf weiterführende Ebenen offenbart der Browser an verteilten Stellen Details zur eingesetzten Cipher suite und der Zertifikatskette einer Domain. Es obliegt schließlich sicherheits-affinen Anwenderinnen, relevante Daten an verteilten Stellen zusammen zu suchen. Abgesehen von diesen Parametern, die ein Browser auf weiterführenden Ebenen exponiert, werden viele sicherheitskritische Daten gar nicht angezeigt.

Unter den sicherheitsrelevanten Informationen, die über die Oberfläche des Browsers nicht einsehbar sind, finden sich beispielsweise Header-Parameter von aufgerufenen Domains. Einen direkten Einfluss auf die erreichbare Verbindungssicherheit haben dabei etwa die Verfahren „HTTP Strict Transport Security“[1] (HSTS) und „HTTP Public Key Pinning“[2] (HPKP). Bei diesen 2012 bzw. 2015 standardisierten Mechanismen wird in HTTPS-Antworten des Servers ein entsprechender Header-Tag mitgeschickt, der von allen aktuellen Browsern ausgewertet wird. HSTS schützt vor Man-in-the-middle-Angriffen und wirkt, indem der Browser für einen gewissen Zeitraum angewiesen wird, eine Domain ausschließlich über eine geschützte Verbindung aufzurufen. Ein „Downgrade“ von HTTPS auf das ungeschützte HTTP wird somit verhindert. Bei HPKP sendet der Server in der Antwort einen Fingerprint des von der Domain eingesetzten Zertifikats, der vom Browser für den angegebenen Zeitraum gemerkt wird und bei später erfolgenden Domainaufrufen übereinstimmen muss.

Wenngleich ein Browser die Cipher Suite und Zertifikatsdaten für die aufgerufene Domain in Fragmenten ausweist, so findet man diese Informationen nicht für fremd eingebundene Domains. Konkret spiegelt sich das darin wider, dass Webseiten oft Inhalte tertiärer Server einbinden. Analog dazu ist es üblich, Schriften, Bilder, Werbung und JavaScript-Code von externen Quellen zu inkludieren. Ebenso können Inhalte über Domaingrenzen hinweg z.B. per AJAX-Requests angefordert werden („Cross-Origin Requests“). Naturgemäß muss die Verbindungssicherheit dieser Dritt-Server dabei nicht mit jener der aufgerufenen Webseite korrespondieren. Bei TLS-Anfragen an fremde Server ist z.B. völlig plausibel, dass die Verwendung einer anderen Cipher Suite vereinbart wird oder das Zertifikat der eingebundenen Domain nicht die gleiche Qualität aufweist (Stichwort: „Extended Validation“), wie jene der eigentlich aufgerufenen Domain. Informationen darüber lassen sich in gängigen Browsern jedoch nicht ohne weitere Werkzeuge herausfinden.

Angesichts der geschilderten Situation war es das Ziel dieses Projekts, eine Erweiterung für einen Browser zu entwickeln, dass die zuvor erwähnten Aspekte für sicherheits-affine AnwenderInnen visuell ansprechend aufbereitet werden. Die Anforderungen an die Browser-Erweiterung bzw. die daraus resultierenden Ziele können wie folgt zusammengefasst werden:

- Das Plugin soll funktional zwei Lücken schließen:
    - Extraktion und Zusammenfassung kritischer Daten, die im Browser bisher nur auf weiterführenden Ebenen zugängig waren (z.B. Cipher Suite, Zertifikatskette).
    - Abbildung weiterer sicherheitsrelevanter Eigenschaften, die nicht ohne weiteres abrufbar sind, wie z.B. Server-Header (HSTS, HPKP) oder Informationen über die Verbindungssicherheit von tertiär eingebundenen Domains.

- Da sich die bereitgestellten Verbindungsinformationen in erster Linie auf die aufgerufene Webseite beziehen, sollte die Anzeige möglichst in der Nähe der Adresszeile abrufbar sein.

[1] https://tools.ietf.org/html/rfc6797
[2] https://tools.ietf.org/html/rfc7469

- Die dargestellten Informationen richten sich an sicherheits-affine Nutzer, die mit einschlägiger Terminologie vertraut sind.

# 2. Entwicklungsrahmen

Im ersten Schritt zur Realisierung einer Browser-Erweiterung wurden die für Entwickler zur Verfügung gestellten Schnittstellen mehrerer Browser (Mozilla Firefox, Google Chrome, Microsoft Internet Explorer, Opera und Safari) analysiert.

Dabei hat sich herauskristallisiert, dass ein Plugin üblicherweise nur dann ohne größeren Aufwand für mehrere Browser gleichzeitig entwickelt werden kann, wenn es ausschließlich mit dem *Document Object Model (DOM)* interagiert. Greift ein Browser also ausschließlich auf den Inhalt der aktuell angezeigten Seite zu (z.B. um etwas einzufügen), kann über JavaScript + CSS ein Plugin realisiert werden, das mit der gleichen Codebasis in Chrome, Firefox, Safari und Opera lauffähig ist.

Angesichts der eingangs angeführten Absicht, detaillierte Informationen über die Qualität einer verschlüsselten Verbindung anzuzeigen, reicht der Zugriff auf den DOM nicht aus. Es besteht daher die Notwendigkeit, die Erweiterung an die proprietären Schnittstellen eines Browsers anzupassen.

Eine weitere Untersuchung hat aufgezeigt, dass momentan lediglich Mozilla Firefox eine entsprechende Schnittstelle bietet, um Informationen über geschützte Verbindungen auszulesen. Google Chrome bietet Entwicklern nur Zugriff auf Inhalte des DOM[3]. In der offiziellen Dokumentation des Internet Explorer konnte kein Vorhandensein einer entsprechenden Funktionalität festgestellt werden[4]. Es wurde somit die Erarbeitung einer Firefox-Erweiterung lanciert.

# 3. Browser-Erweiterung

In diesem Kapital werden die Funktionalität und Darstellung des Plugins in Mozilla Firefox näher erläutert. Die Implementierung betreffende Aspekte finden sich im nachfolgenden Kapitel.

## *1.1. Anzeige und Usability*

Um relevante Informationen bei Bedarf stets einsehen zu können, sollte sich die Erweiterung in den Bereich nahe der Adresszeile integrieren. Bei Auswahl des Plugin-Symbols öffnet sich ein Overlay-Fenster und fasst Informationen konzise zusammen.

Um möglichst viel Daten kompakt anzuzeigen, wurde eine Unterteilung in Tabs vorgenommen. Wie im Screenshot in Abb. 1 ersichtlich, gliedern sich die dargestellten Informationen in die Reiter „Connection“, „Certificates“ und „Domains“. Je nachdem welche Informationen für BenutzerInnen relevant sind, kann zwischen den Tabs hin- und hergewechselt werden. Wenn umgeschaltet wird, merkt sich die Erweiterung den zuletzt ausgewählten Tab und zeigt bei einer anderen aufgerufenen Domain den gleichen Tab wieder an. Dies ist insbesondere nützlich, wenn Domains per HTTP angezeigt werden, zu denen es naturgemäß keinen Informationen zur Verbindungssicherheit auszulesen gibt. Wenn hier also z.B. der Tab „Domains“ ausgewählt ist, muss somit bei einer Folgeseite nicht immer wieder neu auf diesen Tab gewechselt werden. Ebenso wird die Darstellung von Tabs mit leerem Inhalt vermieden, indem bei ungeschützten Domains der Reiter „Certificates“ nicht auswählbar ist.

Die Darstellung der Daten selbst erfolgt gemäß den aktuellen Richtlinien von Mozilla über eine HTML+CSS-Formatierung. Sofern sich an den darunter liegenden Schnittstellen durch Updates von Firefox keine Änderungen ergeben, ist damit größtmögliche Kompatibilität zu zukünftigen Versionen des Browsers gewährleistet.

[3] https://code.google.com/p/chromium/issues/detail?id=107793
[4] https://msdn.microsoft.com/en-us/library/hh772374%28v=vs.85%29.aspx

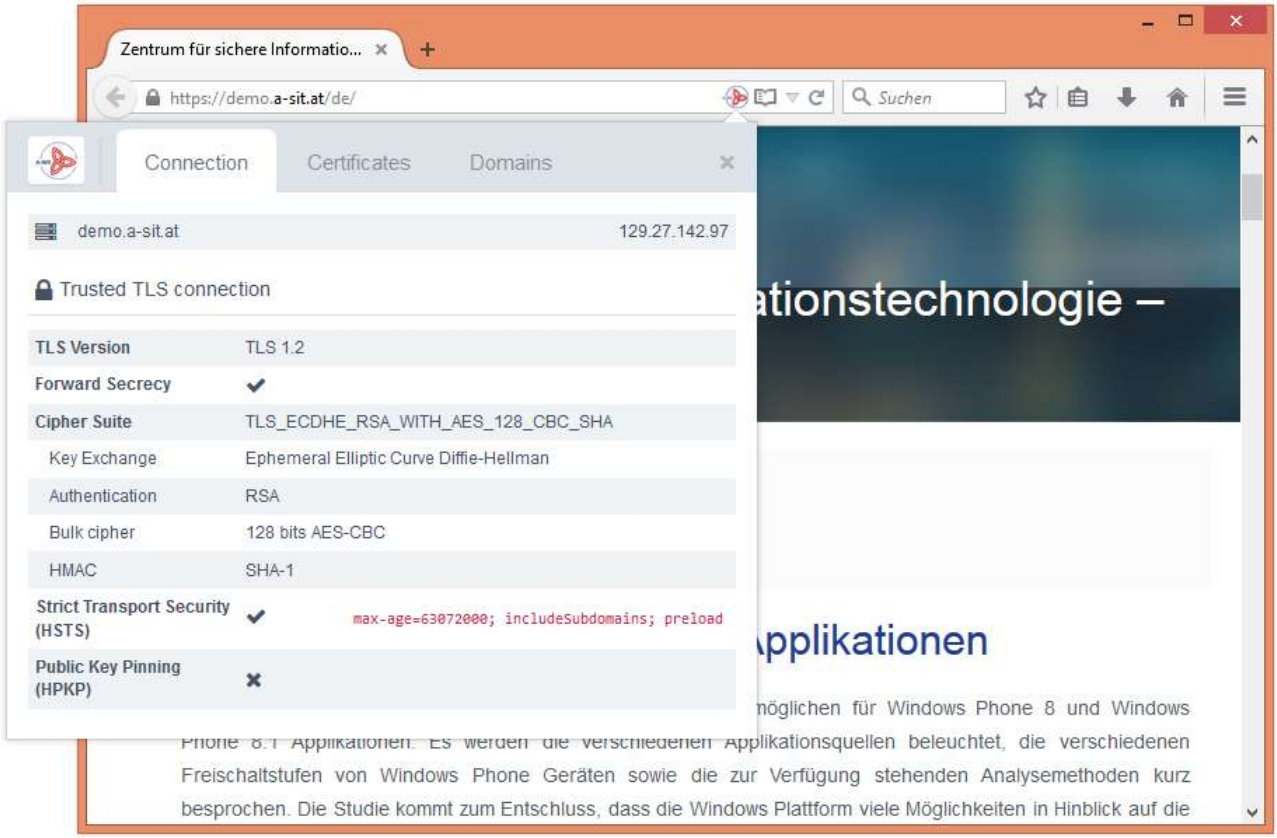


*Abbildung 1. Anzeige des Plugins im Browsers, Tab „Connection“*

## *1.2. Funktionsübersicht*

Nachfolgend wird erläutert, welche Daten in den einzelnen Tabs des Plugins angezeigt werden. Der Übersicht halber wurden die Informationen dabei nach Zugehörigkeit aufgeteilt.

### 1.2.1. Tab „Connection“

Im Reiter „Connection“ finden sich ausschließlich Parameter, die einen Einfluss auf die aktuell hergestellte Verbindung zu einem Server haben. Eingangs wird der *Hostname des Servers* ausgegeben, der sich prinzipiell mit dem Domainnamen decken sollte, der auch in der Adresszeile ersichtlich ist. Daneben wird die *IP-Adresse* des Servers angezeigt, sofern die Webseite nicht vollständig aus dem Browser-Cache geladen wird und somit keine IP-Adresse auslesbar ist.

Die nachfolgende Anzeige des „Trust Status“ bezieht sich sowohl auf den Verbindungszustand, als auch auf die Zertifikatskette. Konkret wird dabei angezeigt, ob eine Verbindung vertrauenswürdig ist oder nicht. HTTP-Verbindungen werden somit als ungeschützt angezeigt. Bei HTTPS-Verbindungen wird die Zertifikatskette ausgewertet, wobei die Anzeige des Plugins nicht zwangsläufig mit dem „Schloss-Symbol“ des Browsers übereinstimmt. Wenn in Firefox z.B. eine Domain mit selbst signiertem oder ungültigem Zertifikat aufgerufen wird, warnt der Browser die/den AnwenderIn und ermöglicht es, eine Ausnahme festzulegen. In weiterer Folge wird die Verbindung mit einem „Schloss“ gekennzeichnet, wodurch sie von einer tatsächlich vertrauenswürdigen Verbindung auf den ersten Blick nicht mehr zu unterscheiden ist. Die Browser-Erweiterung weist hingegen den tatsächlichen Status aus und warnt z.B. wenn der Hostname der Domain nicht mehr dem im Zertifikat hinterlegten übereinstimmt.

Bei HTTPS-geschützten Verbindungen werden in der unteren Hälfte dieses Tabs Informationen zur aktuellen TLS-Verbindung dargestellt. Neben der eingesetzten TLS-Version wird auch die Cipher Suite in IANA-Notation nochmal explizit ausgegeben und unterteilt in die Komponenten „Key Exchange“, „Authentication“, „Bulk Cipher“ und „HMAC“. Sofern für den Schlüsselaustausch DHE oder ECDHE verwendet wird, ist „Forward Secrecy“ gegeben.

Schließlich wird noch angezeigt, ob der Server HSTS- oder HPKP-Header mitsendet. Wenn dies der Fall ist, werden die zugehörigen Attribute ausgegeben. Bei der Anzeige der HPKP-Header werden aus Übersichtsgründen die Public-Key-PINs weggelassen.

### 1.2.2. Tab „Certificates“

Im Tab „Certificates“ wird die vom Browser aufgebaute Zertifikatskette für die aktuell aufgerufene Domain eingeblendet. An oberster Stelle wird stets das Zertifikat für die Domain angezeigt, darauf folgend alle „Intermediate Certificates“ und schließlich die „Root CA“.

Die einzelnen Zertifikate werden anhand ihres „Common Name“ (CN) identifiziert. In Abb. 2 erscheint aus diesem Grund „www.a-sit.at“, wenngleich das Zertifikat für die Domain demo.a-sit.at ausgestellt ist. Im konkreten Beispiel wird der CN vom Browser ignoriert, da in diesem speziellen Zertifikat die optionale X.509-Erweiterung „Subject Alternative Name“[5] beigefügt ist, in der wiederum der Hostname des Servers vermerkt ist.

Zu jedem Zertifikat werden wesentliche Felder, wie etwa zur „Organization“, „Organization unit“ und der Gültigkeitszeitraum ausgelesen. Bei Domain-Zertifikaten wird außerdem erhoben, ob es sich um ein „Extended Validation“-Zertifikat handelt. Dies wird aber ohnehin in der Adresszeile des Browsers durch einen grünen Balken ebenso angezeigt. Außerdem werden bei jedem Zertifikat der Kette Daten zum öffentlichen Schlüssel und der angebrachten Signatur dargestellt sowie der Fingerprint des Zertifikats als SHA-1 Hash ausgegeben.

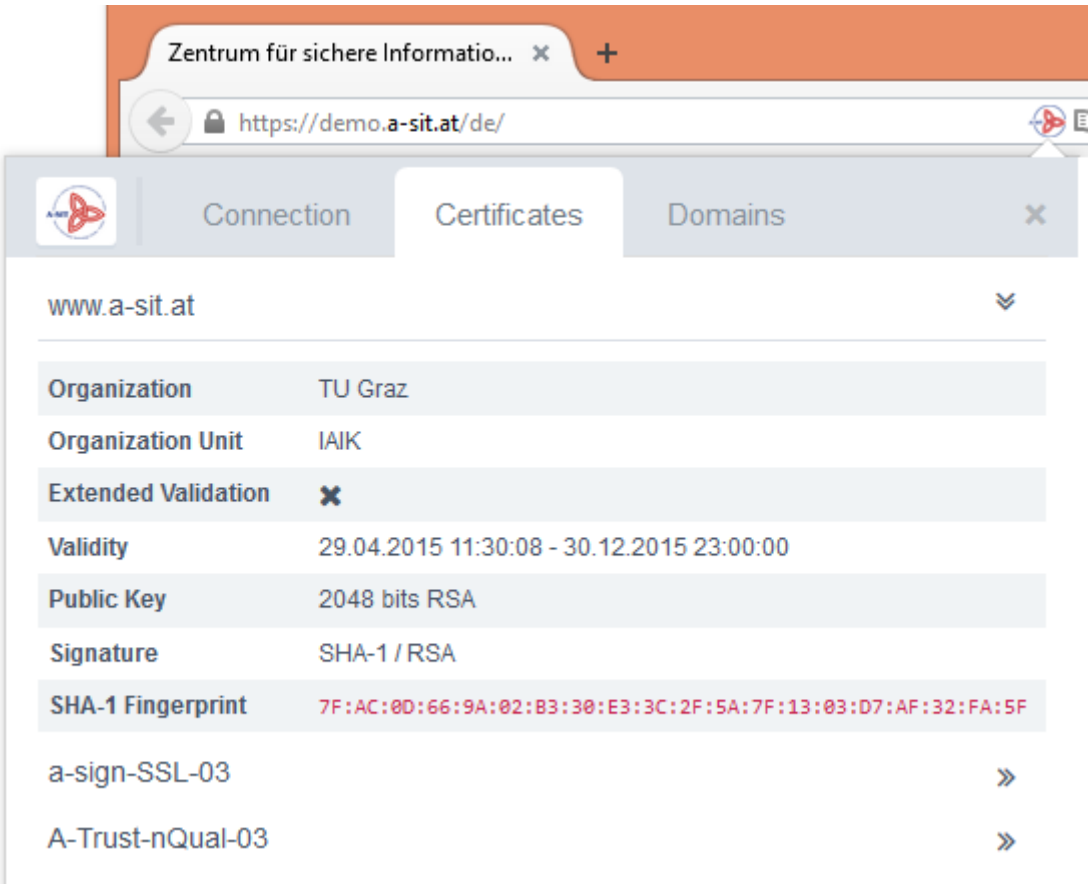


*Abbildung 2. Anzeige des Plugins im Browsers, Tab „Certificates“*

[5] https://tools.ietf.org/html/rfc5280#section-4.2.1.6

### 1.2.3. Tab „Domains“

Im Reiter „Domains“ wird eine Liste aller Domains angezeigt, von denen beim Laden einer Webseite im Hintergrund ebenso Informationen abgefragt werden. Typischerweise handelt es sich dabei um JavaScript-Code, Schriften oder Werbung.

Wie in Abb. 3 erkennbar, wird jedem Eintrag der Liste ein „Schloss-Symbol“ vorangestellt, wenn eine TLS-Verbindung zur jeweiligen Domain aufgebaut wird, von der Daten eingebunden werden. Wenn der „Trust Status“ der Verbindung nicht vertrauenswürdig erscheint, wird dies durch entsprechende Symbolik gekennzeichnet (vgl. Kap. 1.2.1).

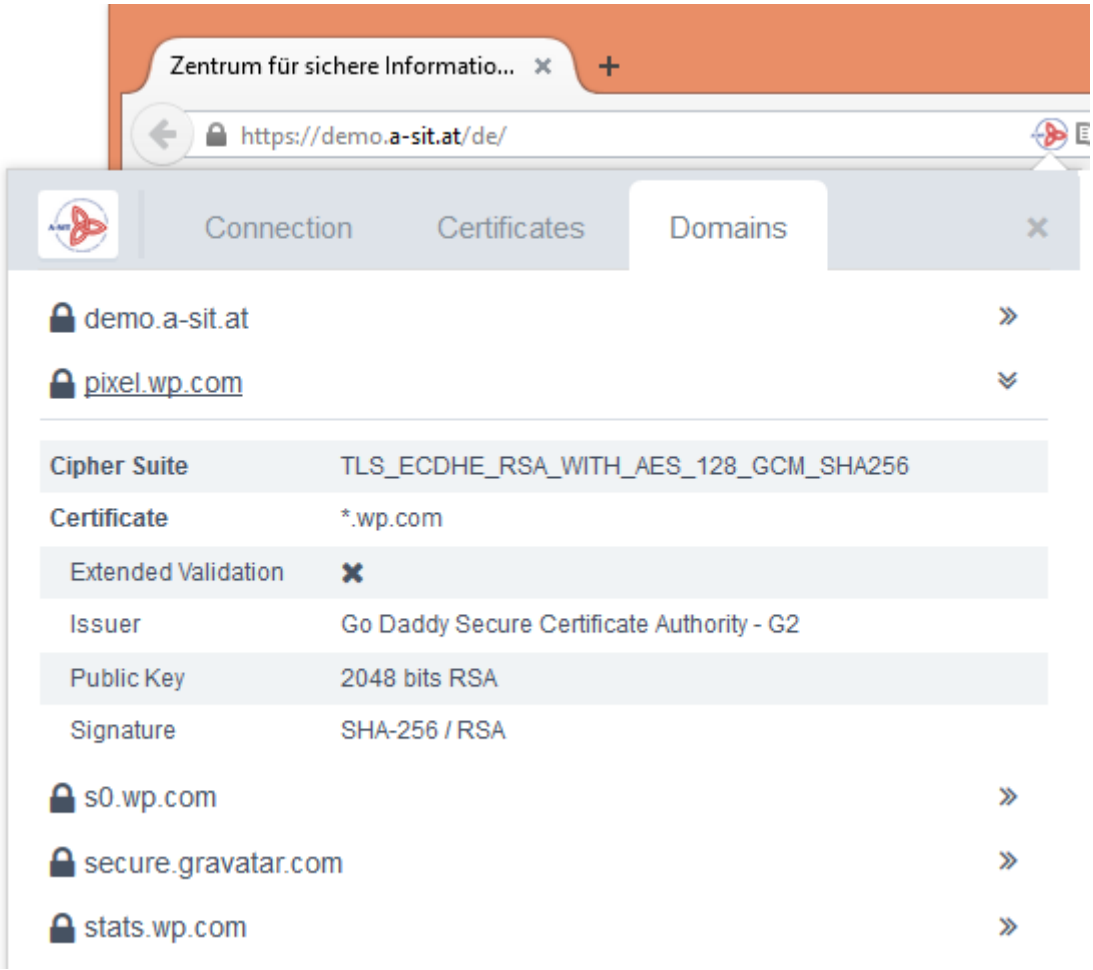


*Abbildung 3. Anzeige des Plugins im Browsers, Tab „Domains“*

Zu jeder Domain zu der eine TLS-Verbindung aufgebaut wurde, werden wesentliche Verbindungsinformationen extrahiert. Neben Angabe der „Cipher Suite“ wird auch der „Common Name“ des Domain-Zertifikats angezeigt und erhoben, ob es sich dabei um ein „Extended Validation“-Zertifikat handelt. Neben der Anführung des Zertifikat-Ausstellers werden weitere Zertifikatseigenschaften zusammengefasst (vgl. Kap. 1.2.2).

# 4. Implementierung

Dieses Kapitel behandelt Implementierungsaspekte und fasst die wesentlichen Bestandteile der Browser-Erweiterung zusammen. Darüber hinaus werden Design-Entscheidungen erörtert.

## *1.3. Komponenten*

Für die Entwicklung der Erweiterung wurde auf das Addon-SDK von Firefox in aktueller Version 1.17 zurückgegriffen[6]. Um Entwicklungsfortschritte in einer stets neuen Demo-Instanz des Browsers zu testen und das finale Plugin-Paket zusammenzustellen, wurde das Build-System „gulp.js“[7] eingesetzt. Über Gulp wiederum können npm-Pakete[8] eingebunden werden, die den Build-Prozess beeinflussen. Ein Beispiel dafür ist „less.js“[9], welches als npm-Paket beim Building dafür sorgt, dass das Stylesheet vom LESS-Preprocessor ins CSS-Format umgewandelt wird.

## *1.4. Ausgewählte Implementierungsaspekte*

Nach dem Start fügt sich die Browser-Erweiterung als Listener für mehrere Events hinzu: „*http-on-examine-response*“, „*http-on-examine-cached-response*“, „*http-on-examine-merged-response*“. Events dieses Typs werden bei jeder HTTP-Response ausgelöst, die im Browser eintrifft. Die Browser-Erweiterung erhält dabei jeweils ein *nsIHttpChannel*-Objekt[10], aus dem entsprechend Informationen extrahiert werden müssen.

Bei der Entwicklung des Plugins gab es im Wesentlichen zwei große Hürden:

- Welche Responses gehören zu welchem Tab bzw. Browser-Fenster?
- Woher weiß man, dass die/der BenutzerIn aktiv eine neue Webseite aufgerufen hat?

Nachfolgend wird auf diese Fragestellungen Bezug genommen und die gewählte Strategie erläutert.

### 1.4.1. Zuordnung HTTP-Response <-> Browser-Tab

Jede HTTP-Response, die eingeht, enthält keine Informationen darüber, zu welchem Tab sie gehört. So ist es z.B. möglich, dass eine Seite A im Tab X gerade geladen wird und gleichzeitig die zuvor aufgerufene Seite B im Tab Y im Hintergrund noch weiter Daten lädt. Eine Zuweisung aller eingehenden Responses zum aktuellen Tab würde somit zu falschen Ergebnissen führen.

Die Zuordnung der Server-Antworten zu den jeweiligen Tabs konnte schließlich realisiert werden, indem geprüft wird, welcher Tab für die aufgerufene URL einen „Notification Callback“ hinterlegt hat. Im Normalfall registriert das aufrufende Browser-Tab also einen Callback, über welchen schließlich auf das Tab zurückgeschlossen werden kann.

### 1.4.2. Aktiver Aufruf einer neuen Webseite

Nachdem eine Relation zwischen *nsIHttpChannel*-Objekt und Browser-Tab herstellbar ist, erscheint es als logischer Schritt, den Aufruf einer neuen Seite dadurch zu erkennen, indem sich die Adresszeile bzw. die URL des Tabs geändert hat.

Bei Aufruf einer neuen Webseite lädt Firefox jedoch Inhalte vor und wechselt erst dann im Browser-Tab die angezeigte Adresse. Alle HTTP-Responses also organisatorisch jener Domain zuordnen, die im Tab gerade geladen ist, würde zu Informationsverlust führen, da die Aufzeichnung erst verspätet beginnen würde. Darüber hinaus ist dieser Ansatz ungeeignet wenn Anchor-Tags zum Einsatz kommen, die die Adresszeile ändern, jedoch nicht den Inhalt der Seite.

Als Konsequenz wurde versucht, einen aktiven Seitenaufruf bereits dann zu erkennen, wenn noch kein „Preloading“ stattfand. Hierfür stellt das SDK sog. „Progress Listeners“[11] bereit, die ein *OnProgressChange-Event auslösen* sollen, sobald eine Seite geladen wird. Über ein entsprechendes Attribut sollten sich auch Anchor-Tags von normalen Aufrufen unterscheiden lassen. Praktisch hat sich diese Lösung als nicht brauchbar erwiesen, da die Notification erst dann

[6] https://developer.mozilla.org/en/Add-ons/SDK
[7] http://gulpjs.com/
[8] https://www.npmjs.com/
[9] http://lesscss.org/
[10] https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM/Reference/Interface/nsIHttpChannel
[11] https://developer.mozilla.org/en-US/Add-ons/Code_snippets/Progress_Listeners

eingeht, wenn sich die Adresszeile bereits geändert hat. Ein Informationsverlust würde also gleichermaßen eintreten, wie bei der vorher beschriebenen Lösung.

Eine akkurate Lösung für dieses Problem wurde schließlich vergleichsweise einfach gefunden, indem die *loadFlags* des *nsIHttpChannel*-Objekt ausgewertet werden. Wenn dabei das Bit *LOAD_INITIAL_DOCUMENT_URI* gesetzt und die Antwort dem hierarchisch obersten Browser-Fenster zuzuordnen ist, kann von einem aktiven Webseitenaufruf ausgegangen werden. Die Prüfung des Browser-Fensters ist deshalb wichtig, da ansonsten auch bei Frames bzw. Iframes ein neuer Seitenaufruf erkannt werden würde.

Konsequenterweise bedeutet das, dass alle HTTP-Responses, die einem Tab zuzuordnen sind, jedoch nicht das entsprechende Bit gesetzt haben, als subsidiär geladene URLs betrachtet werden. Die ursprüngliche Strategie, diese Responses anhand des „Referrer-„ oder „Ajax-Headers“ zu erkennen, ist somit obsolet.

## 5. Fazit

Das entwickelte Browser-Plugin bietet eine visuell gut aufbereitete Zusammenfassung kritischer Verbindungsdaten, die bisher nur umständlich zugänglich waren oder vom Browser nur intern ausgewertet wurden (z.B. HSTS- und HPKP-Einstellungen). Auf eine Bewertung von Verbindungen wurde zum momentanen Zeitpunkt ausdrücklich verzichtet, da das Aufkommen neuer Sicherheitsprobleme bei TLS oder eingesetzten Algorithmen sinnvollerweise auch unmittelbar im Plugin integriert werden müssten.

Die Fokussierung auf Mozilla Firefox erschien insofern unumgänglich, da vergleichbare Browser keine entsprechende Schnittstelle zum Auslesen bereitstellen. Die Darstellung der Daten ist thematisch in mehrere Reiter untergliedert, die alle wesentlichen Informationen für sicherheits-affine AnwenderInnen aufbereiten.

Durch ausschließliche Verwendung von als „stabil“ gekennzeichneten APIs des Firefox SDK wurde beim Implementierungsprozess auch Wert darauf gelegt, eine Erweiterung zu entwickeln, die auch mit künftigen Browser-Generationen noch lauffähig sein wird. Durch die Ausarbeitung entsprechender Strategien zur Sammlung und Auswertung von HTTP-Responses wurde der Fokus darauf gelegt, potentiell verfälschte Ergebnisse möglichst von vorneherein zu vermeiden.

Die Browser-Erweiterung ist modular aufgebaut und kann mit geringem Aufwand auch für weitere Analysen ausgebaut werden. Etwaigen Erweiterungen sind sowohl auf funktionaler als auch auf Darstellungsebene keine Grenzen gesetzt. Auf Verbindungsebene wäre beispielsweise denkbar, zusätzlich zu Sicherheitsinformationen auch DNS-Angaben zur jeweiligen Domain aus dem DNS-Cache des Browsers auszulesen oder weitere Angaben, wie z.B. zum ausgehandelten NPN anzuführen[12]. Auch von HTTP-Responses könnten abseits von HSTS und HPKP weitere Metadaten über Webseiten extrahiert und aufbereitet werden, wie etwa Angaben zu CORS[13] (zur Umgehung der „Same Origin Policy“ des Browsers) oder der „Content Security Policy“ (CSP)[14]. In der Praxis werden letztgenannte Sicherheitskonzepte jedoch gegenwärtig nur spärlich eingesetzt, weshalb sie im Browser-Plugin zurzeit auch unberücksichtigt bleiben.

[12] https://tools.ietf.org/html/rfc7301
[13] http://www.w3.org/TR/cors/
[14] http://www.w3.org/TR/CSP/