Entwicklung eines Browser-Plugins zur Routenüberwachung

Rheinische Friedrich-Wilhelms-Universität Bonn

Arbeitsgruppe IT-Sicherheit

Projektgruppe

Maximilian Häring
Matrikel-Nummer 2450260

Betreuer	Matthias Wübbeling
Prüfer	Prof. Dr. Meier

Inhaltsverzeichnis

1 Einführung

Das Internet ist für viele Menschen im Alltag selbstverständlich geworden. Nach einer Umfrage der Europäischen Kommission "besitzen fast sieben von zehn EU-Haushalten einen Internetanschluss"[1]. Wie man [1] des weiteren entnehmen kann, können 49 Prozent der EU-Bürger über ihr Mobiltelefon surfen. Dabei werden die angefragten Seiten von Webservern wie dem der Apache Foundation[2] beantwortet. Mittels des Kommunikationsprotokolls IP[1] und dem darauf aufbauenden Transportprotokolls TCP[2] werden die Daten der Webseiten in Form von HTTP-Paketen[3] übermittelt. Welchen Weg zum Ziel das Paket genommen hat, lässt sich aufgrund der dezentralen Struktur des Internets nicht zurückverfolgen. Trotzdem ist diese Information für den Nutzer durchaus von Bedeutung. So könnte er Schlüsse darauf ziehen, ob die Daten die er verschickt hat mitgelesen oder manipuliert wurden.

Mein Ziel ist es, den Nutzer auf Routenanomalien während des Surfens auf Webseiten aufmerksam zu machen. Mittels einer Browsererweiterung möchte ich den Nutzer über die von ihm genommenen Routen informieren. Wie man [13] detaillierter entnehmen kann, lässt sich die Ermittlung der Route mit dem Programm Traceroute [6] realisieren. Die Browsererweiterung soll Ketten aus den Nummern der Autonomen Systemen (kurz AS) bilden, die so als Pfad zu einer URL[4] abgespeichert werden. Immer wenn ein ermittelter Pfad von den Bekannten abweicht, soll der Nutzer informiert werden und die Möglichkeit haben den Pfad zu den Bekannten hinzuzufügen oder abzulehnen.

2 Grundlagen

In einer ersten Version und um die Machbarkeit der Idee zu belegen habe ich mich für folgende Rahmenbedingungen entschieden.
Als Betriebssystem unter dem ich entwickle nutze ich OSX 10.8.5, als Browser für den ich programmiere Mozilla Firefox in der Version 24. Für Firefox spricht, dass er der Browser mit dem größten Marktanteil ist [8] und auf unterschiedlichen Plattformen verwendet werden kann. Ein funktionierendes Addon auf diesem Browser hätte also direkt viele potentielle

1 Kurz für Internet Protocol, beschrieben in RFC 791[4]
2 Kurz für Transmission Control Protocol, beschrieben in RFC 793[3]
3 Kurz für Hypertext Transfer Protocol, beschrieben in RFC 2616[5]
4 Kurz für Uniform resource locator, spezifiziert in [7]

Nutzer. Zur Entwicklung eines Addons für den Browser der Mozilla Corporation steht ein SDK[5], mit dem Namen Jetpack[9], zur Verfügung, welches zurzeit in der Version 1.1.4 veröffentlicht ist. Die Dokumentation [10] erklärt die Bestandteile des SDKs, die auch Module genannt werden, und ermöglicht mir so einen schnellen Überblick über die Funktionsweise des SDKs und so die Realisierbarkeit meines Vorhabens.

Im Folgenden Kapitel lege ich den Aufbau des Addons dar und erkläre einzelne Bestandteile.

3 Software

3.1 Ablauf

Wie in Abb. 1 ersichtlich, basiert der Ablauf des Addons auf einem kurzen, sich wiederholenden Schema. Die einzelnen umrahmten Kästen stellen verschiedene Zustände des Addons dar. Nachdem der Browser gestartet ist und das Addon initialisiert wurde befindet es sich im Zustand 'Warte'.

Immer wenn der Nutzer eine neue Seite lädt, startet es mit dem ermitteln der Route zu der angesteuerten Seite[6] mittels Traceroute. Wenn der Pfad bekannt ist, gibt es dem Nutzer diese Information weiter und wartet auf einen neuen Seitenaufruf. Ist der Pfad neu, so informiert es den Nutzer ebenfalls und fragt darüber hinaus, ob der Pfad gespeichert werden soll. Ist dies der Fall, wird der Pfad gespeichert, aber ansonsten verworfen.

5 Kurz für Software Development Kit, eine Sammlung von Werkzeugen um eine Software zu erstellen.
6 Die Route wird anhand der URL ermittelt. URLs die auf verschiedene IP Adressen aufgelöst werden werden so einheitlich behandelt.

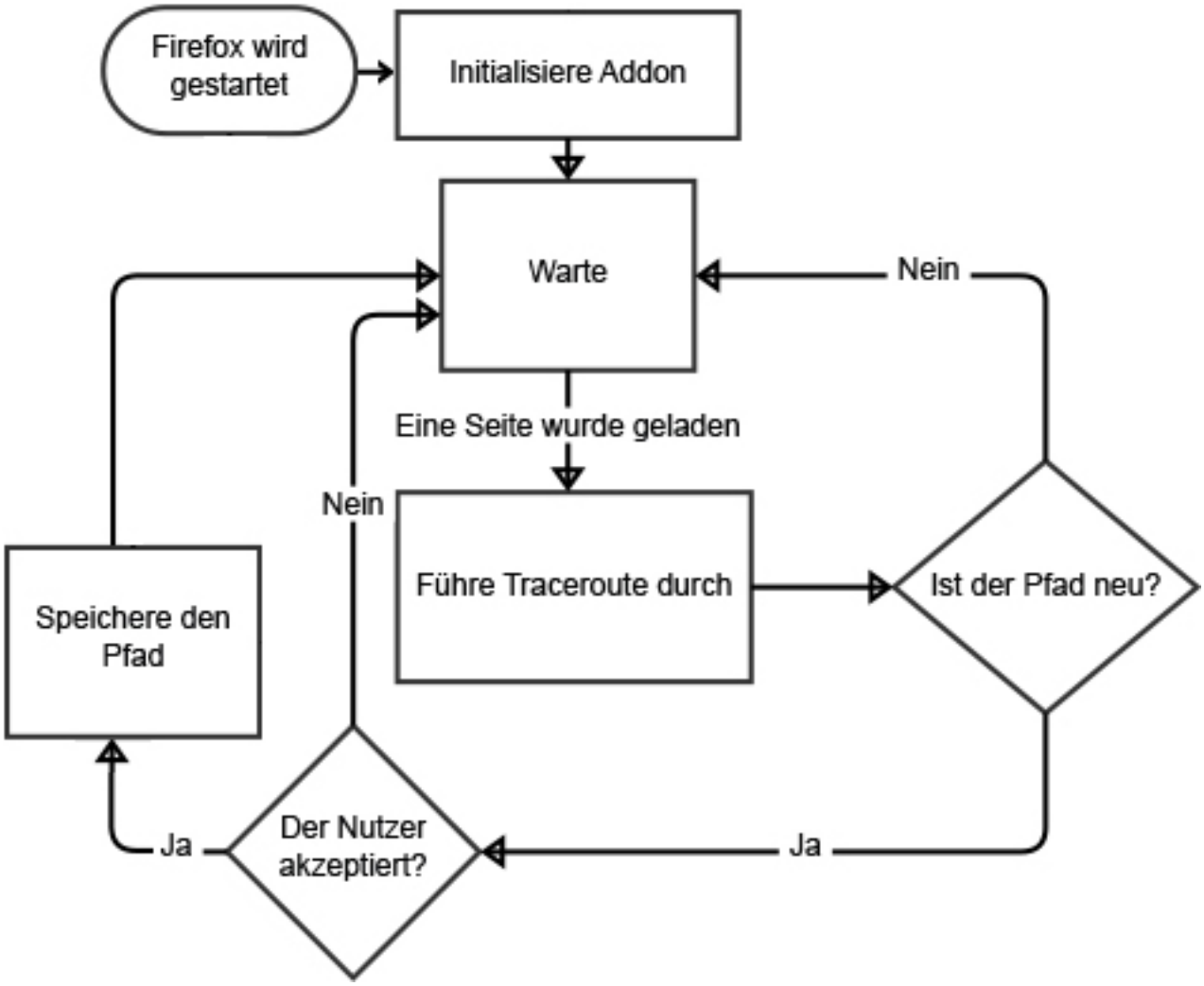

Abbildung 1: Ablaufdiagramm zum Addon

3.2 Verwendete Module und Skripte

Tabelle 1 ist eine Auflistung aller verwendeter SDK Module des Addons und deren Nutzung. Unter [10] findet sich für alle genannten Module die jeweilige Dokumentation.

Zusätzlich zu den bereits genannten Modulen wird noch ein weiteres verwendet. Dieses wurde laut [11] größtenteils von Patrick Brunschwig geschrieben. Das Subprocess genannte Paket von Dateien ermöglicht es in einem Addon Prozesse im Hintergrund zu starten

Name	Funktion
self	Das SDK selber
tabs	Zum benachrichten des Addon bei einem Seitenaufruf
url	Ermöglicht einfachen Umgang mit URLs
widget	Benötigt für die Platzierung eines Icons im Browserrahmen
simple-storage	Stellt einen Speicher für das Addon zur Verfügung
panel	Für die Dialoge mit dem Nutzer

Tabelle 1: Vom Addon verwendete Module des SDKs und deren Funktion

und deren Ein- und Ausgabe zu manipulieren und zu nutzen. Im Addon wird es benutzt, um das Traceroute Programm auf der Konsole zu starten und dessen Ausgabe zu lesen. Es wird hier zwischen der Standardausgabe und einer gesonderten Fehlerausgabe unterschieden. Dies ermöglicht eine komfortable Fehlerbehandlung seitens des Addons. Nach dem Aufbau des Modules zu schließen ist eine Verwendung unter Unix oder Windows vorgesehen. Getestet habe ich es nur unter OSX 10.8.5.

3.3 Kommunikation Addon - User

Der Ablauf des Addons, wie in Abb. 1 und Kapitel 3.1 erklärt, spielt sich hauptsächlich in einer immer wieder aufgerufenen Funktion ab. Diese wird bei jedem Seitenaufruf gestartet. Die Kommunikation mit dem Nutzer, die daraufhin anfällt, funktioniert über Pipes [12]. Dies ist die vom Modul 'panel' ermöglichte Form der Interaktion mit dem Nutzer.

3.4 Datenformat

Abb. 2 stellt exemplarisch dar wie die Daten des Addons gespeichert werden. Der Identifikationsschlüssel zu Pfaden ist das erste Autonome System vom Browser aus gesehen. Für jedes neue Autonome System als Ausgangspunkt wird ein Objekt[7] erzeugt und unter 'start' wird dessen Nummer eingetragen. So ist es möglich das Addon unter unterschiedlichen Lokationen und Anbindungen zu nutzen, aber nicht bei einem Wechsel von der IP bei seinem ISP[8] seine Pfadinformationen zu verlieren. Unter 'Data' wird zu jeder 'url' ein

7 JavaScript, die Entwicklungssprache mit dem SDK, unterstützt keine assoziativen Arrays
8 Kurzform für Internet Service Provider. In diesem Fall ist der Anbieter des Anschlusses zum Internet gemeint.

Array aus Pfaden gespeichert. Die Pfade sind, wie bereits wie in Kapitel 1 angemerkt, Ketten aus Nummern der Autonomen Systemen die per Traceroute ermittelt wurden. Zu beachten ist hierbei, dass die Pfade minimiert wurden. Das heißt, gleiche hintereinander folgende AS Nummern tauchen nur einmal auf.

```
{
    'start' : 'AS28513',
    'Data' : [
        {
            'url'  : 'reddit.com',
            'Path' : [
                ['AS28513','AS65534','AS24582','AS31578']
            ]
        },
        {
            'url'  : 'blog.fefe.de',
            'Path' : [
                ['AS28513','AS8422','AS48823','AS31333'],
                ['AS28513','AS8422','AS24582','AS31333']
            ]
        }
    ]
}
```

Abbildung 2: Speicherformat der Pfade

Firefox stellt jedem Addon einen dedizierten Speicher über das Modul simple-storage [13] zur Verfügung. In dieses können die gerade beschriebenen Objekte direkt gespeichert werden. Laut Dokumentation stehen einem so 5,242,880 Bytes zur Verfügung. Dies reicht für einen Prototypen des Addons und auch im Verlaufe der möglichen Weiterentwicklung sollten diese Grenzen keine Problematik darstellen. Nach der Dokumentation zu schließen ist die einzige Möglichkeit die aktuelle Auslastung des Speichers herauszufinden das 'quotaUsage' Attribut des Storage Objektes. Dieses liefert einem eine Zahl die größer oder gleich null ist, wobei Eins als 100% und Null als 0% Speicherauslastung zu verstehen ist. Sollte die Funktion eine Zahl größer Eins ausgeben, so ist der Speicher überfüllt. Den Speicher kann man zwar kurzfristig überfüllen, beim Schließen des Browser werden diese Daten jedoch dann nicht gespeichert. Stattdessen wird der letzte Stand, bei dem der Speicher nicht voll ausgelastetet wurde, gespeichert. Bei zwei unterschiedlichen Einträgen für das 'start' und 'Data', gefüllt mit jeweils zwei URLs und wieder jeweils vier Pfaden, erhält man eine Auslastung von '0.00025634765625'. Also mit 0,025% deutlich weniger als einem Prozent. Fügt man noch eine weitere URL hinzu und dazu wieder vier Pfade, erhält man '0.00033016204833984374'. Hochgerechnet könnte man also noch $\sim$1250 URLs mit vier Pfaden mit einem gerundeten Wert von 0,00008 hinzufügen.

4 Evaluation

Im Folgenden werde ich einen Test beschreiben, der die Grundfunktionalität und somit die Funktionsfähigkeit der Software testet. Anschließend werde ich auf die daraus hervorgehenden Erkenntnisse eingehen.

4.1 Versuch

4.1.1 Aufbau

Um die Grundfunktionalität zu testen, bediene ich mich des Tools cfx [14]. Mit diesem Tool kann man den Code zu einem ausführbaren Addon umwandeln. Mit dem Parameter 'run' wird direkt eine Browserinstanz, mit einem leeren Profil, gestartet. Das bedeutet, dass sich der Firefox verhält, als hätte man ihn gerade installiert. Also ohne Plugins, Addons, Lesezeichen und besondere Einstellungen. So lässt sich die Grundfunktionalität eines Addons, ohne Einflüsse anderer Software, testen.

Als Grundfunktionalität des Addons verstehe ich das Starten des Addons, das Laden bereits bekannter Pfade und ermitteln von Pfaden bei Seitenaufrufen, das Einstufen in bekannte oder unbekannte Pfade, die Nutzerinteraktion und das anschließende Verarbeiten. Mit dieser Grundfunktionalität sind indirekt Routenanomalien zu erkennen. Wenn die Pfade von einem bestimmten Punkt aus zu der URL bekannt sind und diese sich bei gleichem Ort ändern, dann kann man von einer Anomalie ausgehen.
cfx stellt eine Konsole zur Verfügung über die man im Addon Ausgaben machen kann. Ich habe an einigen Stellen zum besseren Nachvollziehen des Verlaufes diese Möglichkeit genutzt.

4.1.2 Ablauf

Man startet Firefox mit dem Addon über cfx mit dem 'run' Parameter. Das Addon meldet sich mit einem Fenster, dass die Pfade geladen wurden. Das das Addon aktiv ist kann man auch daran erkennen, dass unten rechts in der Ecke ein 'e'-Icon ist. Dieses Icon ist am Anfang grau. Es wird grün, wenn man eine Seite besucht und der ermittelte Pfad bekannt ist oder akzeptiert wurde. Es wird orange, wenn der Pfad unbekannt ist und man ihn nicht

akzeptiert. Ein kurzer Blick auf das Icon teilt einem also den Status des aktuellen Pfades mit.
Gibt man 'blog.fefe.de' in das Adressfeld ein und drückt Enter, dann erscheint die Seite und einige Sekunden später erhält man eine Meldung, dass ein neuer Pfad ermittelt wurde. Wenn man nun die Frage, ob der Pfad akzeptiert werden soll, verneint, wird er nicht gespeichert. Wenn man die Seite neu lädt und einige Sekunden wartet, bekommt man erneut die Meldung dass ein neuer Pfad ermittelt wurde. Wenn man diesen nun hinzufügt, dann ist es wahrscheinlich dass beim nächsten Neu laden der Seite gemeldet wird, dass ein bereits bekannter Pfad ermittelt wurde. Nach mehrfachem neu laden der Seite können trotzdem noch ein bis zwei neue Pfade entdeckt werden.

Wenn man den Browser schließt, werden die Pfade, weil man 'cfx run' benutzt hat, nicht dauerhaft gespeichert. Da ich aber die Pfade zusätzlich über die Konsole ausgebe, kann man diese direkt in den Code eintragen, um das Laden gespeicherter Pfade zu testen. Auf diese Weise belegt man den Pfadspeicher vor.

Wenn man Firefox nun wieder auf die selbe Art lädt, dann erscheint wieder die Meldung, dass die Pfade geladen wurden. Wenn 'blog.fefe.de' diesmal aufgerufen wird, erhält man die Meldung, dass der Pfad bereits bekannt ist. Dies kommt daher, da man diesen Pfad bereits per Hand eingetragen hat. Ändert man nun den Standpunkt indem man z.B. sein Mobiltelefon als Hotspot verwendet, dann wird bei erneutem laden des Addons ausgegeben, dass der ermittelte Pfad unbekannt ist.

Surft man auf 'www.reddit.com' erscheint auch nach $\sim$20 Sekunden keine Meldung vom Addon. Dieses Verhalten tritt bei sehr vielen Seiten auf.

4.1.3 Auswertung

Auf 'blog.fefe.de' verhält sich das Addon wie erwartet und gewünscht. Es startet, lädt vorbelegte Pfade, ermittelt Pfade und erkennt, wenn Pfade bereits bekannt sind. Dies kann man zusätzlich durch die Debugausgaben verifizieren.

Auf 'www.reddit.com' verhält sich das Addon offensichtlich nicht wie erwartet. Die Fehlerquelle hierbei ist der Aufruf von Traceroute. Ein Aufruf direkt in der Konsole gibt nur Zeilen mit '*' zurück. Die Ursachen hierfür werden in Kapitel 4.3 besprochen.

4.2 Schwachstellen in der Methodik

Grundsätzlich kann das Addon nur bewerten was es ermitteln kann. Nicht antwortende Gateways[9] oder Gateways die die TTL Value nicht berücksichtigen sind nicht sichtbar. Dies sollte beim benutzen des Addons immer im Bewusstsein sein. Anomalien, besonders jene die nicht absichtlich herbeigeführt sind, lassen sich aber dennoch erkennen. Das Addon verliert also nicht seine Bedeutung, es muss nur klar sein, dass die Aussagekraft begrenzt ist.

4.3 Traceroute in der Anwendung

4.3.1 HTTP-Requests simulieren

Zurzeit startet das Addon Traceroute mit den Parametern '-a -p 80 -q 1 -n'. Um einen HTTP-Request möglichst genau zu simulieren, wäre der Aufruf zusätzlich mit '-P TCP' nötig. So würde Traceroute TCP Pakete auf Port 80 verwenden. Dies ist der Port und das Protokoll auf dem Webserver hören. Auf dem Weg zu einem antwortenden Webserver dürfen solche Pakete nicht in einer Firewall hängen bleiben, sonst würde man die Seite auch mit dem Browser nicht sehen können.

Der Parameter '-P TCP' scheint jedoch unter OSX 10.8.5 und nach weiteren Tests auch nicht immer unter Ubuntu 12.4 wie erwartet zu funktionieren. Die Ursache dafür ist mir bisher nicht ersichtlich. Eine Ursache könnte sein, dass Traceroute unter jedem OS anders implementiert sein kann.

Der Aufruf mit den Parametern '-a -p 80 -q 1 -n', wie in der aktuellen Addonversion, landet scheinbar oft in einer Firewall. Für einen sinnvollen Einsatz des Addons muss geklärt werden wie man Traceroute mit '-P TCP', unter den verschiedenen Betriebssystemen, nutzen kann.

4.3.2 Eigenheiten der Betriebssysteme

Die unterschiedlichen Benutzer und Rechteverwaltungen der Betriebssysteme können verhindern, dass Traceroute wie gewünscht gestartet werden kann. Unter Ubuntu muss man

9 Damit sind hier die einzelnen Wegpunkte auf dem Pfad eines IP Paketes zum Ziel gemeint.

sich standardmäßig mit dem Administratorkennwort ausweisen bevor man Traceroute mit den Parametern '-P TCP' aufrufen kann. So muss man dafür sorgen, dass der Browser und das Addon mit den entsprechenden Berechtigungen versehen ist.

5 Fazit

Mein Ziel war es einen Mechanismus zu schaffen mit dem der Nutzer beim Surfen auf Routenanomalien aufmerksam gemacht wird. Dieses Ziel habe ich teils erreicht.

Das Addon ist leider noch nicht so einsatzfähig wie ursprünglich geplant, dies wird durch die Komplikationen mit dem Parameter '-P TCP' verhindert.

Für die Zukunft des Addons sehe ich vielerlei Erweiterungsmöglichkeiten. Da man mittels der subprocess Bibliothek auf die Konsole zugreifen kann, ist es möglich, in hochsensibler Umgebung auch denkbar, bei nicht vertrauenswürdigen Pfaden, die Netzwerkkarte direkt zu deaktivieren.
Der Einstiegspunkt des Addons müsste noch anders gestaltet werden. So vermute ich, ist es möglich das Addon noch beim eigentlich http-Request, also vor dem Laden der Seite, mit dem Traceroute beginnen zu lassen. Auch wird momentan nur für den angesteuerten Host ein Pfad ermittelt. Viele Webseiten laden von unterschiedlichsten Quellen Skripte und Inhalte. So lädt www.spiegel.de, laut dem eingebauten Netzwerkanalyse Tool von Firefox, von 22 verschiedenen Hosts Dateien für Inhalt, Tracking und Werbung[10]. Zu diesen Servern könnte man auch Pfade ermitteln und müsste dies auch machen, um einen vollständigen Überblick zu erlangen.

10 Getestet am 15.10.2013 um 12:56 Uhr

Literatur

[1] Verena Gründel, "Zahl der Internetnutzer liegt in Deutschland höchstens im Durchschnitt", http://www.ibusiness.de/aktuell/db/465533veg.html, Beitrag vom 14.08.13

[2] Apache Software Foundation, "Apache Server Project Homepage", https://httpd.apache.org/

[3] IETF Trust, "RFC 793 - Transmission Control Protocol", https://tools.ietf.org/html/rfc793, 1981

[4] IETF Trust, "RFC 791 - Internet Protocol", http://tools.ietf.org/html/rfc791, 1981

[5] IETF Trust, "RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1", http://tools.ietf.org/html/rfc2616, 1999

[6] Aaron Hopkins, "traceroute(8) - Linux man page", http://linux.die.net/man/8/traceroute

[7] W3C, "URL", http://www.w3.org/TR/url/, 2012

[8] Team23 GmbH & Co. KG, "Webanalyse - Aktuelle Browser-Marktanteile", http://www.webmasterpro.de/portal/webanalyse-aktuell.html

[9] Mozilla Corporation, "Jetpack", https://developer.mozilla.org/en-US/docs/Jetpack

[10] Mozilla Corporation, "Add-on SDK Documentation", https://addons.mozilla.org/en-US/developers/docs/sdk/latest/dev-guide/index.html

[11] GitHub, Inc., "jetpack-subprocess", https://github.com/ochameau/jetpack-subprocess

[12] Wikimedia Foundation, Inc., "Pipe (Informatik)", https://de.wikipedia.org/wiki/Pipe_(Informatik)

[13] Mozilla Corporation , "simple-storage Add-on SDK Documentation", https://addons.mozilla.org/en-US/developers/docs/sdk/latest/modules/sdk/simple-storage.html

[13] Maximilian Häring, "Entwicklung eines Browser-Plugins zu Routenüberwachung", 2013

[14] Mozilla Corporation, "cfx - Add-on SDK Documentation", https://addons.mozilla.org/en-US/developers/docs/sdk/latest/dev-guide/cfx-tool.html

Alle Webseiten zuletzt aufgerufen am 14.10.2013