24. Mai 2017 Studie · Web & Cloud · 24 Min. Lesezeit

Analyse von Browser-Erweiterungen

Archiviert. Dieser Inhalt wird nicht mehr aktualisiert. Er bleibt zur Information verfügbar.

Quelle https://technology2.a-sit.at/veroeffentlichungen/analyse-von-browser-erweiterungen/

Diese Studie beschäftigt sich deshalb mit Gefahren, die von Browser-Erweiterungen ausgehen können.

Mit Hilfe von Browser-Erweiterungen lässt sich die Funktionalität von modernen Webbrowsern nahezu beliebig erweitern. Durch die Möglichkeit, einfachen Zugriff auf sensible Daten wie Cookies zu erhalten, werden Browser-Erweiterungen allerdings häufig für bösartige Zwecke genutzt. Aber auch gutartige aber fehlerhafte Erweiterungen können durch Fehler in der Implementierung für gezielte Angriffe genutzt werden. Die Sicherheitsmechanismen von modernen Browsern liefern meist nur bedingten Schutz vor derartigen Angriffen.

Dazu wurde ein Analyse-Framework entwickelt, das diese automatisiert auf potentielle Schwachstellen untersucht. Es wurden dabei am Beispiel Google Chrome dessen Top 1.000 Erweiterungen analysiert. Die Ergebnisse zeigen, dass viele gutartige Erweiterungen Fehler in deren Implementierungen aufweisen, die es Angreifern unter gewissen Umständen ermöglichen, die Kontrolle über Browser-Erweiterungen zu erhalten. Die Analyse zeigt außerdem, dass viele Sicherheitsmechanismen meist nur auf den Schutz vor Malware ausgerichtet sind. Benutzer und Benutzerinnen haben häufig kaum Möglichkeiten, Übersicht über genutzte Funktionalität zu erhalten.

De Ergebnisse werden im Sinne responsive disclosure so statistisch präsentiert, um aus der Studie keine Rückschlüsse auf tatsächliche Schwachstellen konkreter Erweiterungen zu erlauben. Vielmehr soll die Arbeit auch Entwicklern von Browser-Erweiterungen dienen, häufige Fehler zu vermeiden.

1. Einleitung

Browser-Extensions oder auch Browser-Erweiterungen sind Programme, die die Standardfunktionalität oder das Verhalten von Webbrowsern nahezu beliebig erweitern oder modifizieren können. Sie können beispielsweise das User-Interface, den Inhalt von Webseiten oder direkt den Netzwerkverkehr verändern und somit beliebige Browserinhalte definieren und Aktionen ausführen. Dies ermöglicht es, die Benutzerfreundlichkeit von Browsern zu steigern oder Funktionen nur gewissen Nutzergruppen bereitzustellen. Bekannte Beispiele für Browser-Extensions sind Werbeblocker, Browser-Toolbars oder auch

Passwort-Manager. Alle wesentlichen Browser (z.B. Google Chrome [1], Mozilla Firefox [2], Apple Safari [3], Mircosoft Edge [4]) bieten mittlerweile Möglichkeiten an, die Funktionalität der Browser mittels Extensions oder Plugins zu erweitern. In den meisten Fällen greifen Browser-Extensions hierfür auf vom Browser definierte JavaScript-Schnittstellen zurück, um gewisse Funktionen zur Verfügung zu stellen. Diese Schnittstellen eröffnen allerdings auch Möglichkeiten, verschiedene gezielte Angriffe durchzuführen. Gelingt es einem Angreifer beispielsweise, eine bösartige Erweiterung am Zielcomputer zu installieren, können, je nach verfügbarer Schnittstellen, unter anderem private Daten von Nutzern und Nutzerinnen (z.B. Cookies, Passwörter) ausgelesen oder mitgelesen oder Weiterleitungen auf Phishing Seiten (z.B. bei der Nutzung von Online-Banking) durchgeführt werden. Die Gefahr, die von bösartigen Extensions ausgeht, wurde bereits in mehreren Studien untersucht [1] [2]. Um diese Gefahr zu minimieren, haben die meisten Webbrowser mehrere Sicherheitsmechanismen implementiert und Maßnahmen gesetzt. Viele Herstellen bieten zum Beispiel eigene Services an, von denen vertrauenswürdige Extensions geladen werden können. Dies stellt sicher, dass bei erfolgreicher Identifikation von bösartigen Erweiterungen diese schnell wieder entfernt werden können. Damit soll die Infektionsrate möglichst gering gehalten werden. Außerdem können Erweiterungen über einen eigenen Prozess zusätzlichen Analysen unterzogen werden. Die Praxis zeigt allerdings, dass dies nicht immer zuverlässig funktioniert und bösartige Erweiterungen den Weg in offizielle Extension-Stores finden bzw. dort über längere Zeit verfügbar sind [3] [4] [5]. Ein weiterer Schutzmechanismus ist der Einsatz von sogenannten Zugriffsberechtigungen. Bei der Installation von Erweiterungen ist es notwendig, dass der Nutzer oder die Nutzerin seine bzw. ihre Zustimmung erteilt, dass gewisse, vom Browser zur Verfügung gestellte Schnittstellen genutzt werden dürfen. Ist dies nicht der Fall, wird meist die Installation der Extension verweigert. In vielen Fällen ist dieser Vorgang allerdings kaum nachvollziehbar, da zum Zeitpunkt der Installation meist notwendiges Hintergrundwissen oder der aktuelle Kontext fehlen. Vor allem für technisch unbedarfte Nutzer und Nutzerinnen ist es deswegen meist schwierig, schon bei der Installation allgemeine Aussagen zu treffen, ob gewisse Schnittstellen benutzt werden dürfen. Viele Browserhersteller setzen deswegen auf die Kombination von mehreren Sicherheitsmechanismen, um die Verbreitung und Möglichkeiten schädlicher Software einzudämmen. Auf Fehler in der Implementierung oder Sicherheitslücken in Erweiterungen wird allerdings oft nicht explizit geprüft. In den meisten Fällen untersuchen Browser Hersteller lediglich, ob Erweiterungen gegen festgelegte Regeln oder Sicherheitsbestimmungen verstoßen. Aus diesem Grund beschäftigt sich die vorliegende Studie mit gutartigen aber fehlerhaften Erweiterungen. Sie gibt einen Überblick über mögliche Angriffsvektoren bei herkömmlichen Browser-Erweiterungen sowie über mögliche Gegenmaßnahmen. Begleitet wird die Studie von einer Analyse der Top 1000 Erweiterungen für Google Chrome. Dabei wird vor allem auf die richtige Verwendung von Zugriffsberechtigungen geachtet bzw. auf die Einhaltung von gegeben Sicherheitsempfehlungen. Außerdem wird auf die Verwendung von externen Programmbibliotheken sowie deren Aktualität und Häufigkeit Bezug genommen. Die vorliegende Studie gliedert sich in fünf Bereiche. In Kapitel 2 werden die Grundlagen von Browser-Erweiterungen behandelt sowie Hintergrundwissen, welches für diese Studie notwendig ist, gegeben. Außerdem wird die grundlegende Struktur von Browser-Erweiterungen sowie Sicherheitsmechanismen von Browsern im Detail diskutiert. In Kapitel 4 werden die Ergebnisse der Analysephase präsentiert. In Kapitel 5 werden Schlüsse aus der Analysephase gezogen sowie ein möglicher Ausblick bzw. allgemeingültige Empfehlungen gegeben.

1 https://www.google.com/chrome/ 2 https://www.mozilla.org/firefox/ 3 https://www.apple.com/lae/safari/ 4 https://www.microsoft.com/windows/microsoft-edge

2. Grundlagen

Browser-Erweiterungen sind meist Programme, die in JavaScript bzw. HTML die Funktionalität von Browsern über eigens definierte JavaScript Schnittstellen erweitern. Als solche können sie natürlich ähnlich fehleranfällig wie herkömmliche Webseiten sein. Im Gegensatz zu gewöhnlichen Webseiten können Browser-Extensions allerdings auch zusätzliche Berechtigungen besitzen, um beispielsweise an private Daten wie Cookies oder der Browser-Historie zu gelangen. Außerdem können sie direkt Veränderungen am Browser (z.B. durch Einstellen eines Proxies) durchführen oder Inhalte von Webseiten verändern. Damit stellen Browser-Erweiterungen auch ein Angriffsziel dar, um einfach und schnell gezielte Angriffe durchzuführen. In diesem Kapitel wird auf die Grundlagen von Browser-Erweiterungen genauer eingegangen. Dazu werden beliebte Angriffsszenarien sowie mögliche Gegenmaßnahmen und Sicherheitsmechanismen behandelt. Repräsentativ wird außerdem die Struktur sowie der Aufbau von Google Chrome Erweiterungen erklärt. Dies ist einerseits durch die weite Verbreitung von Google Chrome begründet (rund 61% stand 2017 [6]) andererseits bietet Google Chrome seit 2009 weitere Sicherheitsmechanismen, die zusätzlichen Schutz für Browser-Erweiterungen bieten sollen [1] .

2.1. Browser-Extension Struktur

Im Grunde bestehen Google Chrome Erweiterungen aus mehreren JavaScript bzw. HTML und CSS Dateien. Für die Bereitstellung über den offiziellen Extension-Store werden diese Dateien in einer ZIP-Datei gebündelt und mit einem (privaten) Entwickler-Schlüssel signiert. Dieser Vorgang stellt sicher, dass Erweiterungen nicht von unberechtigten Personen aktualisiert werden. Die resultierenden Dateien tragen die Dateiendung *.crx. Erweiterungen für Google Chrome bestehen zumindest aus folgenden Dateien:

- **Manifest** : Das Manifest beinhaltet die wichtigsten Informationen über die Erweiterung wie zum Beispiel den Namen, die Version aber auch die benötigten Berechtigungen bzw. die wichtigsten JavaScript Ressourcen. Das Manifest ist der Einstiegspunkt bei der Installation von Erweiterungen und wird vom Browser automatisch geladen und interpretiert. Abbildung 1 zeigt ein Beispiel für eine einfache Manifest-Datei.
- **HTML-Dateien:** Erweiterungen für Google Chrome können zusätzliche HTML-Dateien beinhalten, die für visuelle Repräsentationen (z.B. Einstellungen) benötigt werden. HTML-Dateien können bei einem Theme [5] ausgelassen werden.
- **JavaScript Dateien** : In den meisten Fällen befinden sich eine oder mehrere JavaScript Datei(en) in einer Erweiterung. Diese beinhalten Programmcode, der für die Funktionalität der Extension notwendig ist.
- **Zusätzliche Dateien:** Erweiterungen können zusätzliche Ressourcen wie z.B. Bilder, die für die Funktion der Erweiterung von Bedeutung sind beinhalten.

Zusätzlich wird bei Erweiterungen in Google Chrome zwischen verschiedenen Komponenten und Inhalten unterschieden. Diese werden im Folgenden behandelt.

5 Persönliches Browser-Design, das über Erweiterungen verändert werden kann

```
{
  "manifest_version": 2,
  "name": "Sample Extension Example",
  "description": "Sample Extension",
  "version": "1.0",
  "background": {
     "scripts": [ "background.js"]
   },
   "content_scripts": [
    {
      "matches": ["<all_urls>"],
      "js": ["content.js"]
    }
  ],
  "browser_action": {
     "default_icon": "icon.png",
     "default_popup": "popup.html",
     "default_title": ""
  },
  "permissions": [
    "activeTab",
    "<all_urls>",
    "cookies",
    "tabs"
  ]
}
```

Abbildung 1: Beispiel einer Manifest Datei für Google Chrome Erweiterungen

2.1.1. Hintergrund Seite

Die meisten Erweiterungen besitzen eine Hintergrund-Seite bzw. eine so genannte „background page". Diese (HTML-) Datei erlaubt es, weiteren JavaScript Code einzubinden. In den meisten Fällen hat eine background page keinen (für den Benutzer und Benutzerin sichtbaren) Inhalt. Google Chrome unterscheidet zwischen permanenten Hintergrund-Seiten und eventbasierten Seiten. Während permanente background pages dauerhaft beim Start des Browsers geladen werden, werden eventbasierte Seiten, wie der Name suggeriert, nur bei Bedarf geladen bzw. geschlossen. Im Allgemeinen empfiehlt es sich, eventbasierte Seiten zu nutzen, um beispielsweise Ladezeiten zu verkürzen.

2.1.2. UI Seiten

Über UI Seiten haben Erweiterungen die Möglichkeit, zusätzlichen Inhalt Benutzern und Benutzerinnen anzuzeigen. Diese können beispielsweise als Konfigurationsbildschirm oder als zusätzlicher Informationsfaktor dienen.

2.1.3. Content-Scripts

In einigen Fällen ist es notwendig, dass Erweiterungen direkt mit dem Inhalt von Webseiten interagieren. Die Sicherheitsmechanismen in Google Chrome (siehe Kapitel 2.3.2) sehen jedoch eine strikte Trennung von Inhalten vor. Aus diesem Grund werden sogenannte Inhaltskripte benötigt. Im Gegensatz zu Core-Extension Code werden diese Skripte direkt in die Webseite eingebunden und können somit ohne weitere Umwege

den DOM der Webseite modifizieren oder darauf zugreifen. Aus diesem Grund sind Inhaltsskripte auch logisch von der restlichen Erweiterung getrennt. Um trotzdem auf Funktionen, die nur direkt von der Extension genutzt werden dürfen zuzugreifen (z.B.

Zugriff auf Cookies), können von Inhaltsskripten zuvor definierte Nachrichten an die zugehörige Erweiterung geschickt werden. Im Gegenzug kann natürlich auch die Erweiterung selbst Nachrichten an das Content-Script schicken um Interaktionen mit dem DOM durchzuführen. Ein Beispiel hierfür ist das Ändern der Hintergrundfarbe.

2.2. Angriffsvektoren

In Folge werden typische Angriffsvektoren auf Browser-Erweiterungen diskutiert. Dabei wird vor allem Bezug auf Angriffe auf gutartige Erweiterungen genommen. Die Liste stellt keine vollständige Sicherheitsanalyse dar, sondern soll einen Überblick über häufige Angriffe geben. Die Angriffsvektoren und mögliche Auswirkungen können sich demnach von Erweiterung zu Erweiterung unterscheiden und sind abhängig von der Funktionalität der Erweiterung.

2.2.1. XSS

Werden Nutzereingaben oder externer Inhalte vor dem Einfügen in einen neuen Kontext nicht bereinigt, können sie dazu führen, dass ungewollt Code ausgeführt wird. Diese Angriffe werden als Cross-Site-Scripting (XSS) Attacken bezeichnet. Ziel ist es an private Daten oder ähnliches zu gelangen. Vor allem Browser-Erweiterungen sind ein beliebtes Angriffsziel für XSS-Attacken, da bei erfolgreichem Angriff auf alle Funktionen der Erweiterung zugegriffen werden kann. Dies hat zur Folge, dass beliebiger Code im Kontext der Erweiterung ausgeführt werden kann. Viele erfolgreiche XSS-Attacken auf Browser-Extensions rühren auch von der unsachgemäßen Benutzung von JavaScript Funktionalität wie eval() oder document.write().

2.2.2. HTTP

Die Verwendung von unverschlüsseltem Verkehr kann dazu führen, dass Angreifer beliebigen Code in den Kontext von Browser-Erweiterungen einschleusen können. Beispielsweise können Erweiterungen externe Programmbibliotheken einbinden, oder einen Remoteserver kontaktieren, um z.B. Suchanfragen durchzuführen. Wird dabei eine unverschlüsselte Verbindung genutzt, kann ein Angreifer theoretisch Teile oder den kompletten Inhalt von Bibliotheken austauschen oder Code einschleusen. Wird diese Eingabe nicht korrekt verifiziert und überprüft, können Angreifer volle Kontrolle über die Erweiterung erhalten. Schutz vor solchen Angriffen kann unter anderem nur durch die korrekte Verwendung von Transport-Layer-Security erreicht werden.

2.2.3. Secrets in Erweiterungen

Die meisten großen Browser Hersteller setzen auf die Verwendung von JavaScript, um Browsertypische Schnittstellen für Extensions zur Verfügung zur stellen. Die Verwendung von JavaScript bedeutet aber gleichzeitig, dass Reverse Engineering von Erweiterungen ohne großen Aufwand möglich ist und diese

einfach analysiert werden können. Zwar können Code-Obfuscation-Techniken den Analyseaufwand erhöhen, dennoch können sie Analysen nicht vollständig verhindern. Damit bieten Erweiterungen keinen sicheren Weg, um Geheimnisse oder sensible Daten wie kryptographische Schlüssel sicher abzulegen.

2.2.4. Veraltete Libraries

Viele Erweiterungen binden weitere Programmbibliotheken ein, um auf erweiterte Funktionalität (z.B. DOM Manipulation oder AJAX) zuzugreifen. Die Verwendung von Bibliotheken von Drittanbietern ist dabei ein bequemer Weg, um schnell und einfach häufige genutzte Funktionen zu nutzen. Berühmte Beispiele hierfür sind jQuery [6] oder angular-js [7] . In der Vergangenheit wurde allerdings mehrmals gezeigt, dass verschiedene Angriffe auf externe Programmbibliotheken existieren [7] [8]. Zwar können diese relativ schnell behoben werden, vergisst ein Entwickler allerdings auf das Aktualisieren von externen Bibliotheken oder ist dies aus Kompatibilitätsgründen nicht sofort möglich, bedeutet 6 http://jquery.com 7 http://angularjs.org dies, dass alle betroffenen Erweiterungen ebenso fehleranfällig sind. Je nach Schwere der Attacke, kann dies dazu führen, dass Angreifer die Kontrolle über Erweiterungen erhalten können.

2.2.5. Drive-By-Downloads

Bei so genannten Drive-By-Downloads werden Webseiten präpariert, um bewusst Fehler in Implementierungen von populären Erweiterungen auszunutzen. Bei erfolgreichem Angriff können Angreifer unter anderem ungehindert Schadcode auf den Ziel-Computer zu laden bzw. ausführen.

2.3. Chrome Sicherheitsmechanismen

In diesem Kapitel werden Sicherheitsmechanismen, die Angriffe auf Erweiterungen erschweren oder verhindern sollen, diskutiert. Zwar unterscheiden sich die Sicherheitsmechanismen von Browser zu Browser, allerdings kommen in den populärsten Browsern ähnliche Sicherheitskonzepte zum Einsatz, weswegen an dieser Stelle nicht explizit zwischen den verschiedenen Browsern unterschieden wird. Stattdessen sei an dieser Stelle auf weiterführende Literatur verwiesen [9]. Die hier erwähnten Sicherheitsmechanismen konzentrieren sich primär auf gutartige aber fehlerhafte Erweiterungen. Sie sollen den Schaden minimieren, falls Erweiterungen erfolgreich angegriffen werden. Die in diesem Kapitel beschriebene Sicherheitsmechanismen schützen also nicht vor bösartigen Erweiterungen. Gelingt es einem Angreifer eine bösartige Erweiterung am Ziel Computer zu installieren, können im Prinzip beliebige Aktionen durchgeführt werden.

2.3.1. Permission-System

Ein Permission-System kommt meist zum Einsatz, um den Zugang zu zur Verfügung gestellten APIs oder den Zugriff auf den DOM zu limitieren. Zugangsberechtigungen sollen einerseits dabei helfen, bei möglicher Infektion von Extensions durch Malware den resultierenden Schaden zu minimieren, andererseits sollen sie Benutzern und Benutzerinnen einen besseren Überblick über verwendete Funktionalität geben. Erweiterungen, die keinen Zugriff auf gewisse Funktionalität benötigen (z.B. Zugriff auf Cookies) können selbst bei Kompromittierung durch Malware nicht auf andere als die vordefinierten APIs zugreifen. Im Allgemeinen ist es notwendig, dass Benutzer und Benutzerinnen bei der Installation von Erweiterungen Zugriff auf bestimmte APIs gewähren. Wird dies abgelehnt, kann eine Erweiterung meist nicht installiert werden. Diese Herangehensweise eignet sich jedoch nicht immer, um Malware oder bösartige

Erweiterungen zu verhindern. Fehlt bei der Installation der notwendige Kontext, ist es schwierig eine adäquate Aussage zu treffen, ob gewisse Funktionalität tatsächlich benötigt wird oder nicht. Beispielsweise setzt Google Chrome zum Zeitpunkt der Studie auf 64 unterschiedliche Permissions [10], welche Abbildung 2 illustriert. Vor allem für technisch wenig versierte Nutzer und Nutzerinnen kann diese große Anzahl an Berechtigungen überfordernd wirken. Der Einsatz eines Berechtigungs-Systems kann also nur bedingt Schutz bieten.

2.3.2. Isolation

Eine strikte Trennung von Webinhalten und Programmcode von Erweiterungen kann zusätzlichen Schutz vor Angriffen auf Erweiterungen bieten. Dadurch soll verhindert werden, dass unbeabsichtigt Programmcode über bösartige Webseiten in den Kontext von Erweiterungen eingeschleust wird. Zugriff auf den DOM der Webseite muss einerseits extra vom Benutzer oder der Benutzerin gewährt werden, andererseits ist dieser Zugriff nur über spezielle Nachrichten, die vom Entwickler zuvor definiert werden, möglich. Damit kann ein Angreifer nur Zugriff auf zuvor definierte Funktionalität erhalten, falls es gelingt über Schadcode in der Webseite, Code in den Kontext von Erweiterungen einzuschleusen. Zusätzlich bieten die meisten Browser Sandboxen, also abgesicherte Umgebungen mit limitierten Berechtigungen, an, in denen unsicher Code ausgeführt werden kann. Ein weiterer Isolationsmechanismus ist die Prozessisolation. Dafür werden Erweiterungen in einem eigenen Prozess gestartet, der sich vom Prozess in dem sich der Webseiteninhalt befindet, unterscheidet. Damit kann der Schaden, der von Low-Level Attacken wie beispielswiese Buffer-Overflows ausgehen kann, eingegrenzt werden.

2.3.3. Deaktivierung von XSS Angriffsvektoren

Viele Browser-Hersteller versuchen die Gefahr, die von XSS Angriffen ausgeht, durch zusätzliche Maßnahmen weitgehend zu unterbinden. So werden Mechanismen wie Same-Origin-Policy, also das Verhindern von Zugriffen auf Inhalt aus anderen Domains, standardmäßig eingesetzt. Das bedeutet gleichzeitig, dass Cross-Origin XMLHttpRequests im Allgemeinen verboten sind. Entwickler, die externe Inhalte einbinden und darauf zugreifen wollen, müssen dies extra im Manifest definieren. Weiters wird als gefährlich eingestufte JavaScript Funktionalität, wie zum Beispiel die Möglichkeit beliebigen JavaScript Code nachzuladen und auszuwerten, standardmäßig unterbunden. Auch hierfür sind weitere Einträge notwendig, um trotzdem Zugriff auf derartige Funktionalität zu erhalten.

activeTab alarms background bookmarks browsingData certificateProvider clipboardRead clipboardWrite contentSettings contextMenus cookies debugger declarativeContent declarativeWebRequest desktopCapture displaySource dns documentScan downloads enterprise.deviceAttributes enterprise.platformKeys experimental fileBrowserHandler fileSystemProvider fontSettings gcm geolocation history identity identity.email idle idltest management nativeMessaging networking.config notificationProvider notifications pageCapture platformKeys power printerProvider privacy processes proxy sessions signedInDevices storage system.cpu system.display system.memory system.storage tabCapture tabs topSites tts ttsEngine unlimitedStorage vpnProvider wallpaper webNavigation webRequest webRequestBlocking *Abbildung 2: Google Chrome verfügbare Verechtigungen*

2.3.4. Externe Installation

Um sicherzustellen, dass Erweiterungen vorgegebenen Sicherheitsmechanismen entsprechen und nicht gegen Policies verstoßen, bieten viele Browser-Hersteller eigene Dienste an, von denen überprüfte Erweiterungen geladen werden können. Um diesen Prozess zu forcieren, erlaubt Google Chrome beispielswiese keine Installation von Erweiterungen über externe Dateien. Erweiterungen können nur noch über den offiziellen Chrome Web Store [8] geladen werden. Dies stellt sicher, dass nur genehmigte Erweiterungen in den Umlauf gelangen bzw. bei verdächtigen Aktivitäten Erweiterungen schnell entfernt werden können. Eine Ausnahme stellt der Entwicklungsmodus dar, der allerdings manuell aktiviert werden muss. Dieser Modus ermöglicht es trotzdem Erweiterungen direkt zu installieren, beispielsweise für die Entwicklung neuer Extensions.

8 https://chrome.google.com/webstore/category/extensions

3. Analyse

Im Folgenden wird die Methodik für die automatisierte Analyse von Browser-Extensions im Detail erklärt. Die Tatsache, dass Erweiterungen typischerweise in JavaScript geschrieben sind, erlaubt einfaches Reverse-Engineering der Erweiterungen, da der Code meist unverändert in der Erweiterung erhalten bleibt. Zwar setzen einige Erweiterungen auf Code-Obfuscation-Techniken um den Analyseaufwand zu erhöhen, trotzdem sind kaum zusätzliche Schritte notwendig, um Code extrahieren können. Folglich können Erweiterungen effizient auf potentielle Schwachstellen untersucht werden. Ein vollständig automatisierter Prozess ist allerdings aufgrund der unterschiedlichen Komplexität und Funktionen der Erweiterungen nicht ohne weiteres möglich. Um schnell und effizient potentielle Schwachstellen zu finden, wurde im Rahmen dieser Studie ein Analyse-Framework entwickelt, welches automatisiert Browser-Extensions auf typische Angriffsvektoren untersucht. Dazu zählt unter anderem das automatische Auffinden von Geheimnissen, die Verwendung von unverschlüsselten Verbindungen oder die Verwendung von veralteten Bibliotheken. Der folgende Abschnitt beschreibt den Aufbau des Analyse-Frameworks bzw. die verwendete Methodik im Detail.

3.1. Analyse-Framework Im Rahmen der Studie wurde ein Framework erstellt, das es ermöglicht, automatisiert eine beliebige Anzahl an Erweiterungen für Google Chrome herunterzuladen, diese zu entpacken und den darin enthaltenen JavaScript Code sowie Metainformationen zu analysieren. Das auf Java basierte Programm unterstützt dabei die folgenden Befehle:

- collect: Um einen schnellen Überblick über verschiedene Erweiterungen zu erhalten, bietet dieser Befehl die Möglichkeit eine beliebige Anzahl an IDs von Erweiterungen für Google Chrome zu sammeln und diese in einer Liste abzuspeichern. Damit können, im Prinzip, beliebig große Datensätze für weitere Analysezwecke generiert werden.

- download: Bei diesem Kommando wird eine Liste an zuvor definierten Extension-IDs eingelesen und in einen definierten Ordner geladen. Zusätzlich werden Metainformationen wie der Name der Erweiterung sowie die heruntergeladene Version im Dateinamen hinterlegt. Durch das automatisierte Herunterladen von Erweiterungen können effektiv verschiedene Daten für eine genauere Analyse generiert werden.

- extract: Durch das spezielle ZIP Verfahren von Erweiterungen [9] (ZIP-Datei inklusive digitaler Signatur) können heruntergeladene Erweiterungen nicht mit herkömmlichen ZIP Verfahren entpackt werden. Aus diesem Grund bietet das Analyse-Framework die Möglichkeit, alle Erweiterungen in einem vordefinierten Ordner automatisch zu entpacken. Für eine automatisierte Analyse ist dieser Schritt unerlässlich.

- permissions: Erweiterungen für Google Chrome benötigen eine Manifest Datei, in der benötigte Zugriffsberechtigungen definiert werden. Während dem Benutzer oder Benutzerin bei der Installation von Erweiterung nur gewisse Berechtigungen angezeigt werden, müssen im Manifest alle benötigten Schnittstellen respektive Zugriffsberechtigungen definiert werden. Die Option permissions extrahiert nun automatisiert diese definierten Berechtigungen und speichert sie in eine CSV-Datei, die wiederum für weitere manuelle Analysezwecke genutzt werden kann.

- analyze: Die Struktur von Extensions für Google Chrome erlaubt eine semi-automatische Analyse von gewissen potentiellen Schwachstellen. Aus diesem Grund können mittels der Option analyze Erweiterungen auf mögliche Sicherheitslücken untersucht werden. Dazu zählt das Auffinden von Geheimnissen wie API-Keys, Benutzerdaten inklusive Passwörtern oder Tokens. Außerdem werden Erweiterungen auf eine falsche Verwendung von kryptographischen Primitiven, beispielsweise durch Definition statischer Schlüssel oder Initialisierungsvektoren, untersucht. Zusätzlich wird die Verwendung von unverschlüsselten

9 https://developer.chrome.com/extensions/packaging Verbindungen sowie die Verwendung von Funktionen, die potentiell für XSS Angriffe genutzt werden können, dokumentiert.

- library: Veraltete Bibliotheken können zu Sicherheitslücken in Browser-Erweiterungen führen. Aus diesem Grund bietet das entwickelte Analyse-Framework mittels der Option library die Möglichkeit, Erweiterungen nach externen JavaScript Bibliotheken zu untersuchen. Dazu werden zu Beginn Prüfsummen von über 300 beliebten JavaScript Bibliotheken gesammelt, die anschließend mit den Dateien in der Erweiterung abgeglichen werden. Die Ergebnisse werden in eine gesonderte CSV-Datei geschrieben. Somit ist es möglich schnell und einfach benutzte Bibliotheken bzw. deren Versionen in Erweiterungen herauszufiltern.

3.2. Methodik

Um einen allgemeinen Überblick über Browser-Erweiterungen zu erhalten, wurden im Rahmen dieser Studie, mit Hilfe der entwickelten Werkzeuge, 1000 Erweiterungen für Google Chrome geladen und analysiert. Dazu wurden zu Beginn der Studie die IDs der Top 1000 Erweiterungen im Google Web Store gesammelt. Die zugehörigen Erweiterungen wurden anschließend automatisiert geladen und extrahiert. Im Anschluss wurden die so generierten Datensätze mit den in Kapitel **Error! Reference source not found.** beschriebene Analysewerkzeugen untersucht. Unterstützt wurde diese Vorgangsweise durch eine manuelle Analyse der generierten Ergebnisse. Dies dient vor allem der Überprüfung der automatisiert generierten Resultate. In Kapitel 4 werden diese Ergebnisse im Detail beschrieben.

4. Ergebnisse

In diesem Kapitel werden die Ergebnisse der Analysephase präsentiert. Aufgrund der großen Datenmenge wurde jedoch keine vollständige Analyse durchgeführt. Vielmehr sollen die gewonnenen Analyseergebnisse einen Überblick über den aktuellen Stand der Technik geben und potentielle Schwachstellen bzw. häufige Fehler oder Vorgehensweisen identifizieren. Dazu wurden zuerst die zum Zeitpunkt der Studie Top 1000 Erweiterungen im Google Web Store auf die am häufigsten benötigten Berechtigungen untersucht. Im Anschluss wurden die Erweiterungen nach Geheimnissen wie Passwörtern, kryptographischen Schlüsseln sowie API-Token untersucht. Mit Hilfe der im Rahmen dieser Studie entwickelten Analyse Werkzeuge wurde außerdem nach unverschlüsselten Verbindungen sowie potentieller XSS-Schwachstellen gesucht. Im letzten Schritt wurden die Erweiterungen außerdem noch auf externe bzw. veraltete Bibliotheken untersucht.

4.1. Berechtigungen

Der Einsatz eines Berechtigungssystems ermöglicht es einerseits Benutzern und Benutzerinnen gezielt Informationen über genutzte Funktionalität von Browser-Erweiterungen anzuzeigen, andererseits kann ein Berechtigungssystem helfen, bei Infektion von Browser-Erweiterungen durch Malware den Schaden zu minimieren. Durch den Einsatz von Berechtigungen kann bei einer Infektion nur die vordefinierte Funktionalität genutzt werden. Um einen Überblick über die häufigsten Berechtigungen, sowie deren Auswirkungen zu bekommen, wurden die Top 1000 Erweiterungen im Google Chrome Web Store auf deren Zugriffsrechte untersucht. Abbildung 3 zeigt die Top 10 Berechtigungen dieser Erweiterungen. Die Bedeutung der jeweiligen Zugriffsrechte wird im Folgenden definiert. Berechtigungen in Fett lösen eine Warnung beim Installieren aus. Diese müssen dezidiert vom Benutzer oder Benutzerin erteilt werden:

- **tabs** : Die tabs Berechtigung erlaubt es Browser-Erweiterungen, die chrome.tabs bzw. chrome.windows API zu benutzen, mit deren Hilfe man neue Tabs bzw. neue Browser Fenster erstellen, bearbeiten oder verschieben kann.

- storage: Die storage Berechtigung wird benötigt, falls eine Erweiterung Zugriff auf die chrome.storage API benötigt. Mithilfe dieser API kann eine Erweiterung beliebige Daten in einer Datenbank speichern, verändern oder hervorholen. Außerdem können Änderungen in dieser Datenbank nachverfolgt werden.

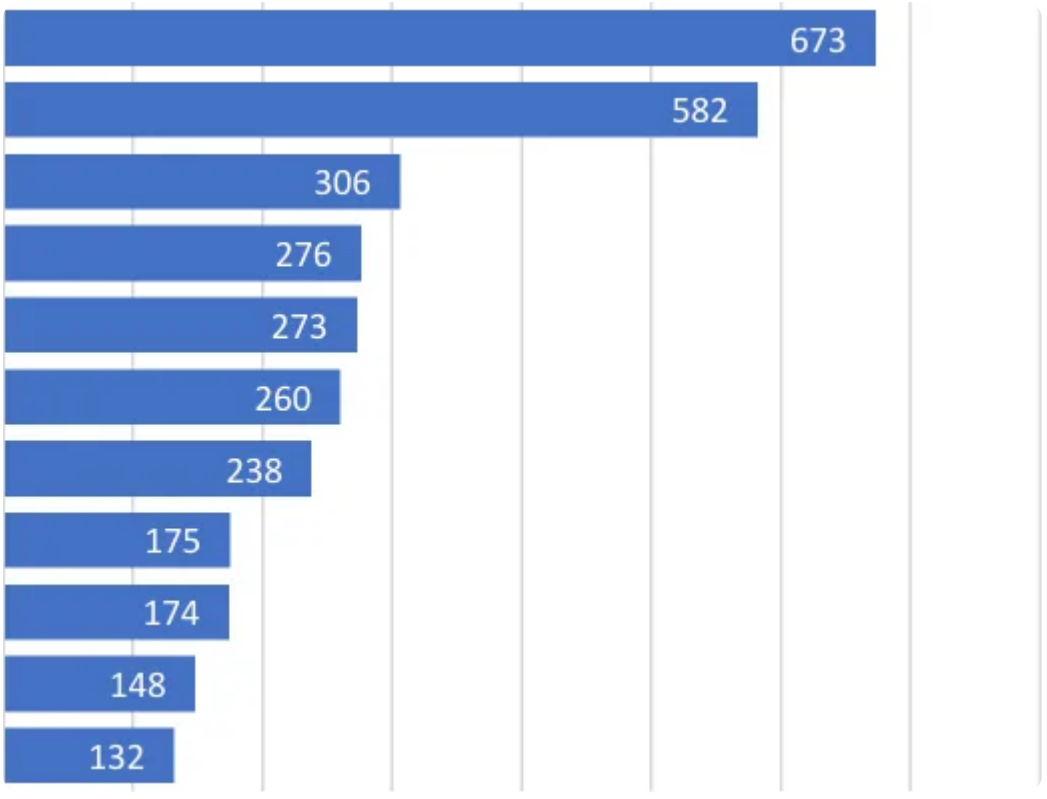
- contextMenus: Die contextMenus Berechtigung respektive der Zugriff auf die chrome.contextMenus API ermöglicht Erweiterungen Inhalte in das Kontext Menü von Google Chrome hinzuzufügen. Diese können sowohl Bilder, Links oder Seiten enthalten.

- webRequest: Mit Hilfe der webRequest Berechtigung können Erweiterungen Netzwerkverkehr im laufenden Betrieb überwachen bzw. analysieren oder sogar einzelne Netzwerkanforderungen bearbeiten und sperren. Allerdings müssen hierfür all jene URLs definiert werden von denen der Netzwerkverkehr überwacht werden soll.

- cookies: In Google Chrome können Erweiterungen, durch das Setzen der cookies Permission, Zugriff auf alle Cookies des Benutzers oder der Benutzerin erhalten. Zusätzlich ermöglicht diese Berechtigung das Verändern von existierenden Cookies.

- notifications: Erweiterungen, die Benachrichtigungen erstellen bzw. diese dem Nutzer oder der Nutzerin anzeigen wollen benötigen Zugriff auf die chrome.notifications API.
- webRequestBlocking: Die webRequestBlocking Berechtigung wird benötigt um mit Hilfe der chrome.webRequest API Zugriffe auf einzelne URLs zu verhindern.
- unlimitedStorage: Im Normalfall ist der maximal mögliche Speicher von Erweiterungen auf 5 MB limitiert. Ist dieser Speicherplatz zu gering, können Erweiterungen mit dieser Berechtigung im Prinzip unlimitierten Speicherplatz nutze.
- activeTab: Durch die activeTab Berechtigungen können Erweiterungen nach einer Benutzeraktion Zugriff auf den aktuellen Tab erhalten. Die Berechtigung erlischt, sobald der Benutzer oder die Benutzerin den Tab schließt oder eine Navigation stattfindet.
- webNavigation: Der Zugriff auf die chome.webNaviation API ermöglicht es Erweiterungen Zugriff auf den Status von Navigationsanfragen zu erhalten.
- **management** : Die Management Berechtigung wird sehr häufig benötigen, falls Erweiterungen Zugriff auf die installierten Erweiterungen oder Applikationen benötigten oder gewisse Funktionalität (z.B. Standardaktion bei Öffnen von neuen Tabs) verändern wollen.

tabs storage contextMenus webRequest cookies notifications webRequestBlocking unlimitedStorage activeTab webNavigation management



0 100 200 300 400 500 600 700 800 *Abbildung 3: Häufigste Permissions* Die vorliegende Studie konzentriert sich auf gutartige aber fehlerhafte Erweiterungen. Aus diesem Grund wird davon ausgegangen, dass die definierten Berechtigungen in guter Absicht definiert wurden. Die Analyse zeigt

allerdings, dass einige Erweiterungen zu viele Berechtigungen definieren, oder Zugriff auf Funktionalität erhalten die nicht zwingendermaßen für die Stabilität der Erweiterung notwendig sind. Dies vergrößert den Angriffsvektor von Browser-Erweiterungen, da der Schaden bei einer Infektion durch Malware größer ausfallen kann. Die Analyse zeigt außerdem, dass Nutzer oder Nutzerinnen nur selten bei der Installation über alle Berechtigungen informiert werden. Nur zwei der Top 10 Berechtigungen lösen tatsächlich eine Warnung beim Installieren der Erweiterung aus.

4.2. Geheimnisse

Aufgrund der Tatsache, dass Browser-Erweiterungen in JavaScript geschrieben sind, bieten sie kaum Möglichkeiten, Geheimnisse sicher abzulegen. Aus diesem Grund wurden die heruntergeladenen Erweiterungen auf Geheimnisse wie API-Keys oder Passwörter mithilfe des entwickelten Analyse-Frameworks untersucht. Die Untersuchungen zeigen, dass ein Teil der Erweiterungen Parameter für kryptographische Primitive statisch in der Erweiterung definieren und diese nicht neu generieren. Damit bietet diese kryptographischen Funktionen praktisch keinen Schutz, da die zugehörigen Geheimnisse von einem Angreifer leicht ausgelesen werden können. Außerdem wurde festgestellt, dass einige Erweiterungen private Zugangsdaten oder Zugrifftokens für APIs (z.B. Amazon AWS oder Google) in der Erweiterung gespeichert haben. Mithilfe dieser Daten ist es möglich, Zugriff auf nicht vorhergesehen Funktionalität zu erhalten oder Benutzerkonten zu übernehmen.

4.3. Unverschlüsselte Verbindungen

Die Verwendung von verschlüsselten Verbindungen im Internet ist unerlässlich, um die Integrität bzw. Geheimhaltung von sensible Daten zu wahren. Aus diesem Grund wurden die geladenen Erweiterungen auf die Verwendung von unverschlüsselten Verbindungen respektive HTTP-Verbindungen untersucht. Die Analyse zeigt, dass 77,3% der untersuchten Erweiterungen zumindest eine URL aufweisen, die auf die Verwendung einer unverschlüsselten Verbindung hindeutet. Insgesamt wurden 175.269 solcher URLs in den Erweiterungen gefunden. Zwar bedeutet die Verwendung von unverschlüsselten Verbindungen nicht zwingendermaßen, dass eine Sicherheitslücke vorliegt, allerdings können Angreifer theoretisch den Inhalt von unverschlüsselten Verbindungen beliebig ändern. Im richtigen Kontext kann es einem Angreifer unter Umständen somit gelingen, die Kontrolle über Erweiterungen zu übernehmen und beliebige Kommandos auszuführen. Die Untersuchungen haben außerdem gezeigt, dass dieser Umstand in mehreren Erweiterungen in Kombination mit anderen Sicherheitslücken ausgenützt werden kann um beliebigen Code im Kontext der Erweiterung auszuführen. Dadurch ist es beispielsweise möglich Cookies auszulesen oder den Nutzer und die Nutzerin auf andere Webseiten umzuleiten.

4.4. XSS

Ein wichtiger Sicherheitsmechanismus von Google Chrome ist es, standardmäßig als unsicher definierte JavaScript Funktionalität zu deaktivieren. Dazu zählt beispielsweise die Verwendung von document.write oder eval. Um trotzdem auf diese Funktionen zugreifen zu können, ist es notwendig, diese Ausnahme im Manifest zu definieren. Rund 20% der analysierten Erweiterungen weisen diesen Eintrag im zugehörigen Manifest auf. Zwar gibt es durchaus legitime Anwendungsfälle für das Reaktivieren von unsicheren Funktionen (z.B. die Verwendung von Bibliotheken wie jQuery), im Allgemeinen sollte diese Funktion allerdings deaktiviert bleiben. Das entwickelte Analyse-Tool untersucht definierte Browser-Erweiterungen auf genau diesen Eintrag im Manifest.

4.5. Veraltete Bibliotheken

Der Einsatz von externen Bibliotheken ist eine einfache und schnelle Möglichkeit um auf häufig genutzte Funktionalität zuzugreifen. Allerdings haben Angriffe in der Vergangenheit gezeigt, dass der Einsatz von veralteten Bibliotheken zu Sicherheitslücken in Anwendungen führen kann [7]. Aus diesem Grund wurden die Top 1000 Erweiterungen auf Erweiterungen von Drittanbietern untersucht. Zusätzlich wurde die verwendete Version dokumentiert. Abbildung 4 gibt einen Überblick über die Häufigkeit der Top 10 Bibliotheken von Drittanbietern. Mit 458 Treffern ist jQuery dabei die am häufigsten verwendete Bibliothek. Die Analyse zeigt allerdings, dass rund die Hälfte der verwendeten Versionen zumindest eine Schwachstelle aufweisen, die es einem Angreifer unter Umständen ermöglicht, auf Funktionalität der Erweiterungen zuzugreifen [8]. Abbildung 5 stellt diesen Sachverhalt dar.

jquery moment.js underscore.js twitter-bootstrap angular.js jquery-ui-timepicker-addon crypto-js history.js tinymce bootstrap-datepicker

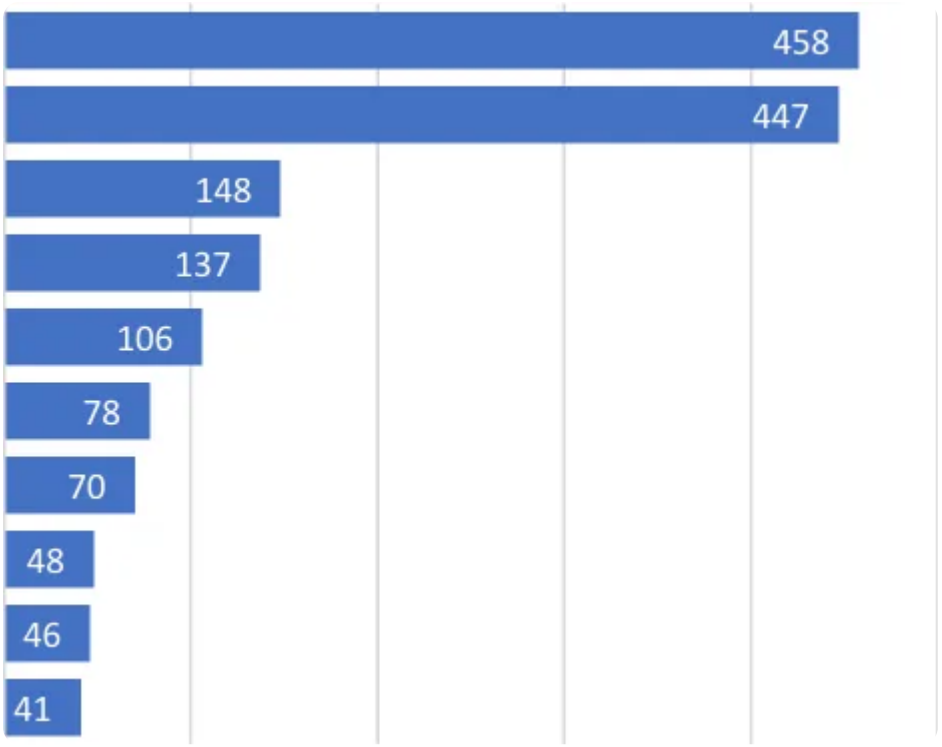


Abbildung 4: Häufigkeit der Top 10 externen Bibliotheken 1.7.1 1.6.2 2.1.4 1.7.2 1.11.1 2.2.4 3.1.0 3.1.1 2.2.0 2.0.0

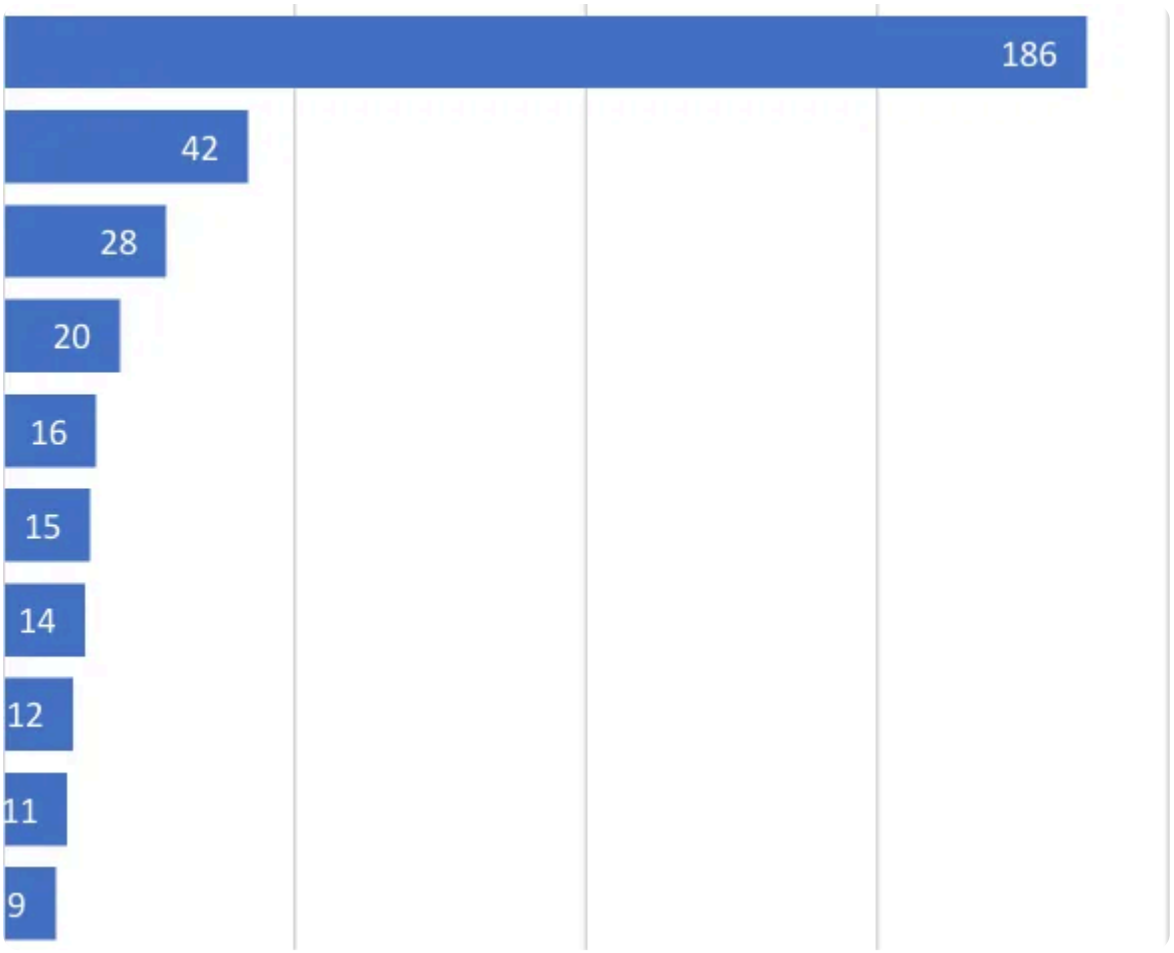


Abbildung 5: Top 10 jQuery Versionen

5. Schlussfolgerung

Browser-Erweiterungen können die Funktionalität von Web-Browsern nahezu uneingeschränkt erweitern oder verändern. Dies kann die Benutzerfreundlichkeit bzw. Verbreitung steigern. Allerdings öffnet diese Möglichkeit neue Angriffsmöglichkeiten bzw. ist ein beliebtes Ziel für Malware. Durch Fehler in Implementierungen oder durch Verwendung von externen Komponenten kann allerdings auch Gefahr von gutartigen aber fehlerhaften Erweiterungen ausgehen, wenn Angreifer gezielt diese Fehler ausnutzen. Die vorliegene Studie hat gezeigt, dass nicht alle Angriffszenarien von Browser Sicherheitsfeatures wie Zugriffsberechtigungen oder Deaktivierung von XSS Angriffsvektoren automatisiert abgedeckt werden können. Aus diesem Grund wurde eine Analyse Platform für Google Chrome Erweiterungen entwickelt, die es erlaubt, Erweiterungen automatisch nach potentiellen und bekannten Sicherheitslücken zu untersuchen. Dazu zählt unter anderem die Verwendung von unverschlüsselten Verbindungen oder das Ablegen von Geheimnissen in Browser-Erweiterungen. Zusätzlich können Erweiterungen auf veraltete Bilitohtken von Drittanbietern untersucht werden. Die Analyse zeigt, dass viele Erweiterungen Zugriff auf nicht benötigte Funktionalität erhalten. Außerdem setzten noch immer rund 80% der Erweiterungen, zumindest in gewissen Fällen auf unverschlüsselte Verbindungen. Die Ergebenisse zeigen weiters, dass auch das bewusste Dekativieren von Sicherheitsfeatures in Browser-Erweiterungen dazu führen kann, dass Erweiterungen anfällig auf bekannte Schwachstellen werden. Außerdem wurde gezeigt, dass die Verwendung von veralteten Bibliotheken zur Kompromitierung von Extensions führen kann. Zusammenfassend bieten moderne Web Browser viele Sicherheitsmechanismen um Browser-Erweiterungen vor Angreifern zu

schützen. Allerdings werden in der Praxis viele Mechanismen durch Fehler in der Implementierung oder der falschen Verwendung von kryptographischen Primitiven außer Kraft gesetzt. Berechtigungssysteme, wie sie unter anderem in Google Chrome zum Einsatz kommen, bieten meist nur Schutz für installierte Erweiterungen. In der Regel verhindern sie jedoch nicht einen unsachgemäßge Zugriff auf sensible Daten. Auch beim Einsatz von externen Bibliotheken sollte auf mögliche Sicherheitslücken geachtet werden bzw. Updateprozesse für diese Abhängigkeiten definiert werden.

6. Literaturverzeichnis

[1] A. Barth, A. P. Felt, S. Boodman und A. Prateek, „Protecting Browsers from Extension Vulnerabilities," *Ndss,* Bd. 147, pp. 1315-1329, 2010.

[2] A. Guha, M. Fredrikson, B. Livshits und N. Swamy, „Verified Security for Browser Extensions Arjun," *Proceedings - IEEE Symposium on Security and Privacy,* pp. 115-130, 2011.

[3] D. Goodin, „Google's Chrome Web store used to spread malware," arstechnia, 27 3 2012.

[Online]. Available: https://arstechnica.com/business/2012/03/googles-chome-web-storeused-to-spread-malware/. [Zugriff am 11 07 2017].

[4] D. Goodin, „Google kills 200 ad-injecting Chrome extensions, says many are malware,"

arstechnica, 1 4 2015. [Online]. Available: https://arstechnica.com/security/2015/04/googlekills-200-ad-injecting-chrome-extensions-says-many-are-malware/. [Zugriff am 11 5 2017].

[5] D. Goodin, „Chrome extension collects browsing data, uses it for marketing," arstechnia, 4 8 2015. [Online]. Available: https://arstechnica.com/security/2015/04/chrome-extension-collectsbrowsing-data-uses-it-for-marketing/. [Zugriff am 11 5 2017].

[6] „Browser & Platform Market Share," 30 4 2017. [Online]. Available: https://www.w3counter.com/globalstats.php. [Zugriff am 5 11 2017].

[7] D. Goodin, „Actively exploited WordPress bug puts millions of sites at risk," arstechnica, 5 6 2015. [Online]. Available: https://arstechnica.com/security/2015/05/actively-exploitedwordpress-bug-puts-millions-of-sites-at-risk/. [Zugriff am 17 04 2017].

[8] C. Details, „Jquery : Security Vulnerabilities," CVE Details, [Online]. Available: https://www.cvedetails.com/vulnerability-list/vendor_id-6538/Jquery.html. [Zugriff am 18 03 2017].

[9] N. Virvilisa, A. Mylonasa, N. Tsalisa und D. Gritzalisa, „Security Busters: Web browser security vs. rogue sites," *Computers & Security,* Bd. 52, pp. 90-105, 6 2015.

[10] Google Inc., „Declare Permissions," 2016. [Online]. Available: https://developer.chrome.com/extensions/declare_permissions. [Zugriff am 02 01 2017].