HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

COMPASS SECURITY

IT-Security: Browser Uniqueness – Identifying Users

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2010

Autor(en): Florian Hengartner
Betreuer: Walter Sprenger
Projektpartner: Compass Security AG, Jona
Gegenleser: Andreas Rinkel

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Kurzfassung der Studienarbeit

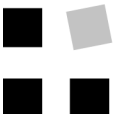
Abteilung	**Informatik**
Name[n] der Studierenden	**Florian Hengartner**
Studienjahr	**HS 2010**
Titel der Studienarbeit	**Browser Uniqueness – Identifying Users**
Examinatorin / Examinator	**Andreas Rinkel**
Themengebiet	**Internet-Technologien und -Anwendungen**
Projektpartner	**Compass Security AG**
Institut	**-**

Kurzfassung

Das Ziel ist Benutzer aufgrund Ihres Browser-Fingerprints zu Identifizieren. Der Fingerprint wird Aufgrund vom Browser gesendeten HTTP Header und Clientseitig (Javascript, Java, Flash) ermittelten Daten erstellt. Nicht alle verfügbaren Daten eignen sich für das Fingerprinting, weil sie z.b. zu oft ändern. Es ist erstaunlich welche Anzahl an unterschiedlichen Datenquellen ein moderner Web-Browser bietet.

Als Anwendungsmöglichkeiten bieten sich, dass Tracken von Usern ohne speichern einer Clientseitigen Session Id, wiederherstellen von Cookies oder das Erkennen von gestohlenen Sessions.

Das Resultat ist eine Web-Applikation welche Web-Browser über dessen Fingerprint identifizieren kann. Fingerprints von verschiedenen Browsern können verglichen werden. Eine Facebook Applikation demonstriert wie einfach persönliche Daten mit einem Fingerprint verknüpft werden können. Dadurch kann nicht nur der Browser, sondern auch die Person identifiziert werden.



HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Inhaltsverzeichnis (1)

Inhaltsverzeichnis (2)

Technischer Bericht

Studienarbeit:
"IT-Security: Browser Uniqueness - Identifying Users"
Florian Hengartner
2010

1. Inhaltsverzeichnis

2. Änderungsgeschichte

Version	Kommentar	Datum
1.0	Erstellt	Mittwoch, 15. Dezember 2010 (KW: 50)
1.1	Benutzeridentifikation	Donnerstag, 16. Dezember 2010 (KW: 50)
1.2	Erweitert	Freitag, 17. Dezember 2010 (KW: 50)
1.3	Ergebnisse	Montag, 20. Dezember 2010 (KW: 51)
1.4	Korrektur / Review	Dienstag, 21. Dezember 2010 (KW: 51)

3. Einführung

3.1. Problemstellung1

Viele Internet Benutzer denken, dass sie anonym im Internet surfen solange sie sich nicht an einer Web-Applikation angemeldet haben. Erste Tests (https://panopticlick.eff.org/) haben jedoch gezeigt, dass viele Browser ein einmaliges Fingerprinting zulassen und somit der Browser des Benutzers eindeutig identifiziert werden kann. Zusammen mit Social Network Diensten wie Facebook oder Xing könnte es möglich sein, einen Benutzer zu identifizieren bevor er sich an einer Web-Applikation anmeldet.

Was für das anonyme Surfen ein Nachteil sein kann ist für eine eBanking-Applikation ein Vorteil. Die Online Bank kann ermitteln, ob der Benutzer von seinem Computer ins eBanking einloggt oder ob der Benutzer unterwegs ist respektive ob die Session des Benutzer von einem Hacker geklaut wurde um auf dessen Konto zuzugreifen.

3.2. Aufgabenstellung2

In dieser Arbeit soll untersucht werden, inwiefern aufgrund von Browserinformationen auf die Identität des Benutzers geschlossen werden kann. Dabei soll abgeleitet werden, welche Informationen für ein eBanking nützlich sind um Missbrauch zu verhindern. Und was muss ein Benutzer unternehmen, der wirklich anonym surfen will?

Anhand einer ProofOfConcept Web-Applikation (Test-Portal) sollen die Erkenntnisse visualisiert werden. Allenfalls sollen dazu Social Network Applikationen (z.B. Facebook Apps) entwickelt werden um den Namen des Benutzers in Erfahrung zu bringen.

Die Aufgabe kann in folgende Tätigkeiten unterteilt werden:

- Erstellen einer Webseite mit Datenbankanbindung zur Visualisierung
- Erstellen z.B. einer Facebook Applikation

Die Arbeit hat unter anderem mit folgenden Themen zu tun:

- JavaScript und Java Programmierung
- Einfache Web-Applikation als ProofOfConcept mit Datenbankanbindung
- HTTP Protokolls
- IT-Security
- Social Networks (Facebook, Xing, MySpace, etc.)

[1] Zitat der originalen Einführung für die Studienarbeit. Autor: Walter Sprenger

[2] Zitat der originalen Aufgabenstellung für die Studienarbeit. Autor: Walter Sprenger

3.3. Rahmenbedingungen

3.3.1. Termine

20.09.10	Beginn der Studienarbeit, Ausgabe der Aufgabenstellung durch die Betreuer
20.12.10	Abstract / Kurzfassung und A0 Poster zur Prüfung an Betreuer senden
23.12.10	Abgabe Abstrakt / Kurzfassung Abgabe Bericht

3.3.2. Technologien

Name	Beschreibung	Link
Client-Seite		
HTML & CSS	Strukturierung & Darstellung von Inhalten für Webseiten	http://de.wikipedia.org/wiki/HTML http://de.wikipedia.org/wiki/Cascading_Style_Sheets
Javascript	Programmiersprache	
jQuery	Javascript Bibliothek	http://jquery.com
Server Seite		
Ruby	Programmiersprache	
RubyOnRails	Web Applikations Framework	http://rubyonrails.org/
Apache	Web Server	http://apache.org/
Phusion Passenger	Modul zur einbindung von Rails Applikationen in Apache	http://www.modrails.com/
Sqlite3	Datenbank (Serverlos)	http://www.sqlite.org/
Mysql	Datenbank	http://www.mysql.com

3.3.3. Entwicklungsumgebung

- Betriebssystem: Mac OS X 10.5
- Editor: Textmate (http://macromates.com/)

3.4. Vision und Ziel

Es soll ein Mechanismus implementiert werden um Fingerprints zu erfassen und zu speichern. Browser sollen über diese Fingerprints wiedererkannt werden können. Als Gimnik sollen dem Besucher einige der herausgefundenen Informationen schön Aufbereitet angezeigt werden (z.b. installierte Plugins).

Zur Demonstration der Technik wird eine Facebook Applikation erstellt welche einen Fingerprint des Besuchers erzeugt. Öffnet der Besucher nun die “Kontroll”-Seite - ausserhalb von Facebook - wird auch bei inzwischen gelöschten Cookies die Person wiedererkannt. Dies soll veranschaulichen wie Serverseitig erstellte Benutzerprofile innerhalb und ausserhalb von Facebook miteinander verknüpft werden können.

3.5. Fingerprint

In computer science, a fingerprinting algorithm is a procedure that maps an arbitrarily large data item (such as a computer file) to a much shorter bit string, its fingerprint, that uniquely identifies the original data for all practical purposes just as human fingerprints uniquely identify people for practical purposes.

Zitat von Wikipedia "Fingerprint (computering)" http://en.wikipedia.org/wiki/Fingerprint_%28computing%29, , zuletzt Abgerufen 21.12.2010

In unserem konkreten Fall, werden aufgrund der vom Browser gesendete und ausgelesenen Informationen ein Fingerprint erzeugt. Über diesen können Browser und im Endeffekt deren Benutzer wiedererkannt werden.

3.6. Stand der Technik

3.6.1. Forschungsprojekt Panopticlick[3]

Ein Anliegen der "Electronic Frontier Foundation" ist die Privatsphäre im Internet. Die Personen hinter Panopticlick haben sich die Frage gestellt wie eindeutig sich ein Browser Identifizieren lässt rein über die vom Browser übertragenen und zur Verfügung gestellten Informationen.

Das Ziel des Projektes ist es eine Datensammlung anzulegen die Rückschlüsse über die vom Browser gesendeten Informationen zulässt. Z.b. welche Informationen sind in User-Agent Strings enthalten? Wie ist die Verteilung eines bestimmten User-Agent Strings in der Gesamtmenge? Wie hoch ist die durchschnittlich übertragene Informationsmenge? Kann die Identität eines Besuchers auch nach einer Änderung des Fingerprints - z.b. Browserupgrade - etabliert werden?

Unterdessen wurden "1,302,316" Browser getestet. Es ist ebenfalls eine Auswertung[4] der bisher gesammelten Daten verfügbar.

Die Forscher sind zum Resultat gelangt, dass die Informationen des durchschnittlichen Browsers 18.1 Bits Entropie aufweisen.

3.6.2. Bericht "Browser Fingerprinting"[5]

In diesem Dokument der Compass Security AG werden sehr ausführlich die diversen Möglichkeiten aufgelistet und bewertet wie Information aus einem Webbrowser abgefragt werden können.

Folgendes können die Ziele bei der Verwendung von Fingerprinting sein:
- Fraud Detection
- Tracking ohne Cookies
- Wiederherstellen von Cookies

---

[3] http://panopticlick.eff.org/ How Unique & Trackable is your Browser?, zuletzt Abgerufen 21.12.2010

[4] http://panopticlick.eff.org/browser-uniqueness.pdf) How Unique Is Your Fingerprint?, Author: Peter Eckersley, zuletzt Abgerufen 21.12.2010

[5] "Browser Fingerprinting", 2010, Autor: Daniel Stirnimann, Compass Security AG

- Wiedererkennen von Socialnetwork Benutzern

Unvollständige Liste der betrachteten Möglichkeiten:
- HTTP Requests Header
- Javascript
  - Bildschirm Eigenschaften
  - Mime Typen
  - Browser Plugins
  - Storage Features
- Schriftarten (Java / Flash / CSS)
- Treiber
- MAC Adresse
- SSL Session
- TCP/IP
- ...

Die Reihenfolge in welcher Informationen (z.B. Schriftarten) aufgelistet werden kann ebenfalls als Informationsquelle in den Fingerprint miteinbezogen werden.

3.6.3. Kommerzielle Angebote

- 41st Parameter - Fraud Detection: http://www.the41st.com/products.asp
- Iovation - Fraud Detection: http://www.iovation.com/rm-360/
- ThreatMatrix - Fraud Dection: http://threatmetrix.com/our-solutions/show-me-how-it-works/
- United Virtualities - Cookie Replacement/Restoring: http://www.imediaconnection.com/news/5392.asp

3.7. Wie können Benutzer eindeutig identifiziert werden?

Nehmen wir an, wir erzeugen Fingerprints um Benutzer eindeutig zu Identifizieren und um Cookies wiederherzustellen

Nun erzeugen wir von unbekannten Besucheren einen Fingerprint. Ist der Fingerprint neu - nicht in unserer Datenbank vorhanden - so wissen wir nun, dass wir es mit einem neuen Benutzer zu tun haben.

Für den Fall, dass der Fingerprint bereits bekannt ist, können wir den Benutzer nun Identifizieren - oder nicht?
Es stellt sich nun die Frage ob der Fingerprint vom selben Browser gesendet wird, oder ob es einen zweiten Browser mit identischem Fingerprint ist.

Dieses Problem lässt sich nicht mit absoluter Gewissheit lösen. Ich habe nun den gleichen Ansatz wie Panopticlick verwendet. Mit folgender Annäherung kann man eine gute Aussage über die Eindeutigkeit eines Fingerprints treffen. Es wird für jeden Fingerprint dessen Entropie berechnet. Diese sagt aus, dass mit der Informationsmenge, welche vom Fingerprnt zur Verfügung gestellt wird ein Browser unter X Browsern eindeutig identifiziert werden kann.

Wenn nun die Zahl X grösser oder gleich gross der Anzahl Browsern auf dieser Welt ist, kann ich den Browser weltweit eindeutig Identifizieren.

Wie gross ist die gesamte Anzahl an Browsern auf dieser Welt? Da die Anzahl Browserinstallationen schwierig zu schätzen ist, wird stattdessen die Anzahl Personen auf dieser Welt verwendet. Die Weltbevölkerung beträgt im September 2010 6‘892

Millionen Personen. Hierbei lassen wir ausser Acht, dass nicht jeder Person zugang zu einem Browser besitzt, eine Person mehrere Browser verwendet (Heim, Büro, Laptop, Mobiltelefon, Spielkonsole, etc.) oder mehrere Personen denselben Browser verwenden (Heimcomputer, Internet Cafe), Crawlers, etc.. Die grösse der Weltbevölkerung ist meines Erachtens grösser als die Anzahl möglicher Browser - wir befinden uns also auf der sicheren Seite.

Eine Person aus 6‘892 Millionen entspricht einer Entropie (S) von 32.68 bits.

$$S = \log_2\left(\frac{1}{6'892\ \text{Mio.}}\right) = 32.68\ \text{bits}$$

Hiermit haben wir den Schwellwert für die Entropie festgelegt. Besitzt ein Fingerprint diese Entropie, oder mehr, können wir ihn als Eindeutig betrachten.

3.7.1. Implikationen in der Praxis

Es können durchaus zwei oder mehr Browserinstallation mit demselben Fingerprint existieren, die auch eine Entropie über dem Schwellwert besitzen. Dessen muss man sich immer bewusst sein. Beispielsweise könnten geklonte Installationen die ihre Anfragen über denselben Proxy senden von aussen sehr schwierig zu unterscheiden sein.

3.7.2. Berechnung der Entropie eines Fingerprints

Die Berechnung der Entropie ist schwierig, da die Anzahl möglicher Werte z.b. des User-Agents unbekannt ist. Ebenfalls ist die Verteilung unbekannt - wie oft taucht User-Agent XYZ auf. Eines der Ziele von Panopticlick ist es aufgrund der gesammelten Daten von tausenden Webbrowsern eine repräsentative Aussage über die Entropie machen zu können.

Ich verwende in diesem Projekt einerseits die durchschnittlichen Entropiewerte aus dem Panopticlick Projekt und andererseits, selbst berechnete Werte - denn ich sammle mehr Daten als Panopticlick.

Zu Beachten ist bei diesem vorgehen, dass die Entropie eines Fingerprints immer eine Mommentaufnahme ist. Zum Zeitpunkt A wurde der User-Agent XYZ bei 1 von 100 anderen Browsern gesehen. Zu einem späteren Zeitpunkt B könnte der User-Agent XYZ bei 1 von 50 anderen Browsern gesehen worden sein - die Entropie ist gesunken, der Schwellwert wird zum Zeitpunkt B möglicherweise nicht mehr erreicht!

3.7.3. Überwachung des Fingerprints einer Session

Verwenden wir das Fingerprinting in einem anderen Szenario, mit dem Ziel gehijackte Sessions zu erkennen, gestaltet sich die Sache einfacher. Die Problematik mit der eindeutigen User Identifikation wird abgenommen, dies kann über ein Cookie / Session-Identifier im URL Parameter / etc. erledigt werden.

In diesem Szenario wird der Fingerprint an eine Session gekoppelt. Die Session wird von einem separaten Mechanismus etabliert (Cookie, ...). Der Fingerprint wird nun ständig überwacht, ändert er sich (zu sehr) wird die Session terminiert und der Benutzer zu einem frischen Login gezwungen.

Nebst der eindeutigen Identifizierung, fällt in diesem Szenario auch die Problematik der Verfolgung von Änderungen im Fingerprint weg. Üblicherweise werden Login Sessions nach einem Browser neustart frisch aufgebaut, womit beim Login auch ein frischer Fingerprint zur Session hinzugefügt werden kann. Auch für den Fall, dass die Session z.b. 10 Tage gültig

ist, könnte falls in dieser Zeit ein Browserupdate den Fingerprint ändert, bereits vor ablauf der Session ein erneuter Login erzwungen werden.

3.7.4. Fazit

Fingerprints sind sehr hilfreich beim verifizieren von etablierten Sessions. Hier kann mit einfachen Mitteln viel erreicht werden.

Sie können ebenfalls für die eindeutige Identifikation von Benutzern oder wiederherstellen von Cookies verwendet werden. Dies ist nicht immer 100% exakt, da es auf diversen Annahmen beruht. Trotzdem sind erstaunliche Resultate möglich - Panopticlick gelang es bei Fingerprints die sich über die Zeit verändert haben, 99.1% davon demselben Benutzer zuzuordnen.

4. Ergebnisse

4.1. Was wurde erreicht

Die fertige Demo-Applikation hat folgende Funktionalität.

4.1.1. Mit und ohne Javascript

Auf der Startseite wird der Link zum Fingerprinting angepasst, je nachdem ob der Browser Javascript fähig ist oder nicht.

Für Browser mit Javascriptunterstützung zeigt der Link auf “/pages/fingerprint”, ohne Javascript zeigt er auf “/pages/fingerprint_nojs”.

Ohne Javascript werden nur die Daten des HTTP Request Headers erfasst. Hierbei ist die Entropie grundsätzlich so klein, dass Sie nicht für eine eindeutige Identifikation ausreicht.

Ist Javascript aktiviert werden per Javascript diverse zusätzliche Tests ausgeführt und die per XmlHttpRequest an den Server gesandt. Hierbei können oft sehr viele Informationen gesammelt werden, die führ eine eindeutige Identifikation ausreichen.

4.1.2. Cookie zum Tracken des Benutzers

Für spätere Auswertungen ist es von Vorteil wenn wir ein Cookie setzen um Benutzer zu tracken. Es können mehrere unterschiedliche Fingerprints derselben Person zugeordnet werden. Somit können Änderungen am Fingerprint nachverfolgt und ausgewertet werden. Mögliche Anwendungen: mutwillige Änderungen (Javascript deaktiviert) aus statistischen Analysen ausschliessen, Veränderungen (Browserupdate, Plugininstallation) am Fingerprint über die Zeit verfolgen.

Natürlich können die Cookies vom Benutzer gelöscht werden und die Verfolgung der “History” unterbrochen werden.

4.1.3. Cookie wiederherstellen

Ist ein Besucher unbekannt (kein Cookie vorhanden) und die Entropy des Fingerprints genügend gross, wird versucht das Cookie wiederherzustellen. Es wird in der Datenbank nach dem Fingerprint gesucht und bei einem Treffer die zugehörige GUID wiederverwendet um das Cookie wiederherzustellen.

4.1.4. Konfigurierbarkeit

Der Entropie-Schwellwert kann im Administrations-Interface dynamisch angepasst werden.

Die Verwendung des Cookies kann aktiviert / deaktiviert werden. Ist das Cookie aktiviert, ist automatisch auch die Cookie-wiederherstellung aktiv.

Alle “Informations-Punkte” die gesammelt werden, sind als “Key” hinterlegt. Z.b. der Request Header “Accept-Language” oder ob der Browser Java aktiviert hat. Für jeden einzelnen Key, kann aktiviert werden ob:
1. der Key aktiv ist, d.h. die Daten gesammelt und gespeichert werden
2. dieser Key ein Teil des Fingerprints ist.

Die IP-Adresse zum Beispiel wird aus statistischen Gründen - und auch für mögliche spätere Auswertung - gesammelt (1), ist aber nicht Teil des Fingerprints (2) - denn die IP kann sich ändern.

Dies ermöglicht es sehr flexibel einen neuen Key hinzuzufügen, bestehende zu deaktivieren und den Inhalt des Fingerprints zu verändern. Ausserdem werden unbekannte HTTP Header automatisch als neue Keys erfasst. Mit wenigen Klicks können sie in den Fingerprint aufgenommen werden.

4.1.5. Administrations-Interface

Das Administrations-Interface biete CRUD Zugriff auf alle Datenbank Tabellen.

Zusätzlich können die vorhandenen Daten analysiert werden. Es gibt eine Auswertung der Useragents sowie der Entropie.

4.1.6. Updaten der Entropie

Es gibt ein Script, dass die durschnittliche Entropie jeden Keys aufgrund der aktuellen Daten berechnet und diesen durchschnitts Wert bei jedem Key Eintrag updatet und speichert.

4.1.7. Facebook Applikation

Die Facebook Applikation erstellt vom Benutzer einen Fingerprint und speichert zu diesem, den Benutzernamen und die Benutzer ID. Somit kann der Fingerprint nun einer Person zugeordnet werden. Da man nun die Facebook ID kennt, kann man über die öffentliche API von Facebook Informationen über die Person aufrufen. Es ist z.b. möglich das persönliche Bild des Benutzers anzuzeigen.

4.2. Verwendete Keys

Eine Key ist ein bestimmter Wert oder Test der im Browser abgefragt wird. Dies ist eine komplette Liste der für den Fingerprint verwendeten Keys.
Die “Quelle” ist entweder der Rückgabewert des aufgelisteten Javascript-Statements oder der Wert stammt aus den HTTP Header Daten.

Name	Quelle
[Opera] Build number of the browser	window.opera.buildNumber()
[Opera] Version number of the browser	window.opera.version()
ActiveX Enabled?	PluginDetect.ActiveXEnabled
Browser Vendor	window.navigator.vendor
Browser Name	window.navigator.appName
Class of CPU	window.navigator.cpuClass
Color Depth	screen.colorDepth
Cookie Enabled?	FingerprintAPI.browser.tests.cookie_enabled()
Engine of the browser	window.navigator.product
Fonts List (Java)	FingerprintAPI.browser.tests.fontslistjava()
HTML5 Database Storage (Webdatabase)	FingerprintAPI.browser.tests.databasestorage()
HTML5 Global Storage	FingerprintAPI.browser.tests.globalstorage()
HTML5 IndexedDB	FingerprintAPI.browser.tests.indexeddb()
HTML5 Local Storage	FingerprintAPI.browser.tests.localstorage()
HTML5 Session Storage	FingerprintAPI.browser.tests.sessionstorage()
HTTP Header Accept	HTTP Header
HTTP Header Accept-Charset	HTTP Header
HTTP Header Accept-Encoding	HTTP Header
HTTP Header Accept-Language	HTTP Header
HTTP Header Connection	HTTP Header
HTTP Header Content-Transfer-Encoding	HTTP Header
HTTP Header Header User Agent	HTTP Header
HTTP Header Header Wap Profile	HTTP Header
HTTP Header Keep-Alive	HTTP Header
HTTP Header Pragma	HTTP Header
HTTP Header TE	HTTP Header
Information about the OS and CPU (oscpu)	window.navigator.oscpu
Java enabled?	navigator.javaEnabled()
Language of the browser	window.navigator.language
Language of the installed operating system	window.navigator.systemLanguage
Language of the operating system's user interface	if (window.navigator.language === undefined) {window.navigator.browserLanguage}
MimeTypes Count	navigator.mimeTypes.length
MimeTypes List	FingerprintAPI.browser.tests.mime_types()
MS ActiveX XMLHttpRequest Object	FingerprintAPI.browser.tests.msxmlhttp()
Native XMLHttpRequest Object	FingerprintAPI.browser.tests.nativexmlhttp()
Operating System	window.navigator.platform
PluginDetect	FingerprintAPI.browser.tests.plugindetect()
Plugins Count	navigator.plugins.length
Plugins List	FingerprintAPI.browser.tests.plugins()
Regional and Language settings of the operating system	if (window.navigator.language === undefined) {window.navigator.userLanguage}
Time Zone Offset	(new Date()).getTimezoneOffset()
Timezone	FingerprintAPI.browser.tests.calculate_time_zone()
User Data (IE only, since Version 5.0)	FingerprintAPI.browser.tests.userdata()

4.3. Schwierigkeiten die es zu lösen galt

4.3.1. Ermöglicht eine Facebook-Applikation das erstellen von Fingerprints?

Die Antwort vorneweg: Ja es Funktioniert.

Zu Beginn des Projektes hatte keiner der Beteiligten eine Ahnung von Facebook Applikationen. Also wurde zuerst mit einem Prototypen, dass erfassen von Browserinformationen abgeklärt.

Früher wurden Facebook Applikationen von einem Server bei Facebook geparst und dargestellt. Diese Variante wird von Facebook in Zukunft deaktiviert. Heutzutage sind Facebook-Applikationen (sogenannte Canvas Apps) IFrames die Inhalte vom Server des Applikationserstellers laden.
Dies ermöglicht es unserem Applikationsserver direkt mit dem Browser des Benutzers zu kommunizieren. Dies hat zur folge, dass es möglich ist die HTTP Header Daten des Benutzters zu erfassen, sowie Javascripts, Applets, etc. auszuführen. Somit steht dem Erstellen von Fingerprints nichts im Wege.

4.3.2. Welche Keys sollen für den Fingerprint verwendet werden?

Durch kontinuierliches ausprobieren und testen wurden die Keys ausgewählt. Sie sind auf diese Anwendung zugeschnitten. Für andere Anwendungsgebiete muss die Auswahl überprüft werden.

Beispielsweise hätten mehr Keys verwendet werden können, wenn die Facebook-Applikation nicht gewesen wäre.

Für das Erkennen von gestohlenen Sessions könnte durchaus auch die IP Adresse spannend sein. Man könnte Festlegen, dass während der Session, zumindest der Netzblock nicht ändern soll. Etc.

Die grössten Schwierigkeiten bereitete die Facebook-Applikation. Ein Fingerprint im normalen Webinterface muss dasselbe resultat Liefern, wie ein Fingerprint im Facebook-Applikations-iFrame.

Tests auf die aus diesen gründen Verzichtet wurde:

- Schriftart Liste per Flash extrahieren:
  - Beschreibung: Bietet mehr Daten als die Java Version / Fallback falls kein Java vorhanden
  - Grund: konnte den Flash Movie in der Facebook-Applikation nicht zum laufen - nicht ausgeschlossen, dass dies Möglich ist, aus Zeitgründen nicht weiter verfolgt
- Cookie setzen und wieder auslesen
  - Beschreibung: Prüfen ob Cookie setzen/lesen auch wirklich möglich ist, nicht Blind auf ein Browser Attribut verlassen.
  - Grund: in einem Fall wurde beobachtet, dass dies mit Safari auf Windows fehl schlug
- Reihenfolge der MimeTypes & Plugin Liste wird verworfen
  - Die beiden Listen werden vor dem Erzeugen des Fingerprints sortiert. Damit gehen leider Informationen verloren.
  - Grund: Beobachtungen haben ergeben, dass derselbe Browser die Liste nicht immer in derselben Reihenfolge sendet

4.3.3. Gefahren

Alle Daten die für den Fingerprint verwendet werden sind Eingaben die vom Benutzer stammen. Werden diese direkt Weiterverwendet - z.b. eine Ausgabe im Webinterface - müssen Sonderzeichen escapt werden, damit kein Code eingeschleust werden kann.

4.4. Was wurde nicht erreicht

4.4.1. Cross Domain Problematik

Es wurde versucht das Fingerprint-Javascript als ein “Service” anzubieten. D.h. es wird von der Webseite X inkludiert und erhält als Antwort die Id des Fingerprints und womöglich noch Informationen zum Benutzer.

Als Schutz gegen Datendiebstahl gibt es in Browsern die sogenannte Same-Origin-Policy[6]. Das bedeutet, die Webseite welche vom Server X geladen wurde, kann Daten (in unserem Fal per XmlHttpRequest) nur wieder an die Webseite X, nicht aber an den Fingerprint Server senden.

Das Pattern JSONP[7] bietet einen Workaround. Kurz gesagt: es wird dynamisch ein Javascript-Include Statement auf der Webseite X eingefügt, dieses lädt dadurch Code vom Fingerprint Server nach und führt diesen aus. Voilà!
Nur leider steckt der Teufel im Detail. Das Include Statement verwendet einen GET Request. Bisher wurden die Daten mit der XmlHttpRequest Komponente als POST Request gesendet. Für den GET Request ist das Limit der Datenmenge viel kleiner. Siehe Artikel “WWW FAQs: What is the maximum length of a URL?“[8]. Dieses Limit wird bei Fingerprints die z.b. eine Liste der MimeTypes oder Plugins enthalten in gewissen Browser und Server Software locker überschritten. Somit kann diese Technik nicht universell verwendet werden.

Allenfalls könnte man durch klevere Optimierungen bzw. Verkleinerung der übertragenen Datenmenge oder Aufteilung der Daten in mehrere Requests eine Lösung mit JSONP entwickeln.

Als praktikabler Workaround für dieses Problem sehe ich den Einsatz eines “Cross-Domain Proxy”[9]. Hierbei gibt es ein Proxy Script, dass die Fingerprint Schnittstelle auf dem Webserver X zur Verfügung stellt - und im Hintergrund die Daten an den Webservice auf dem Fingerprint Server sendet. Dann wird die Antwort vom Fingerprint Server entgegengenommen und ebenfalls als Antwort retourniert.

Eine weitere Alternative ist das “Flash-enabled XHR”[10] Pattern. Damit begrenzt man sich allerdings auf Flash-Fähige Browser.

---

[6] http://en.wikipedia.org/wiki/Same_origin_policy Wikipedia “Same origin policy”, zuletzt Abgerufen 21.12.2010

[7] http://ajaxian.com/archives/jsonp-json-with-padding JSON With Padding, zuletzt Abgerufen 21.12.2010

[8] http://www.boutell.com/newfaq/misc/urllength.html What is the maximum length of an URL?, zuletzt Abgerufen 21.12.2010

[9] http://ajaxpatterns.org/Cross-Domain_Proxy Cross Domain Proxy Pattern, zuletzt Abgerufen 21.12.2010

[10] http://ajaxpatterns.org/Flash-enabled_XHR Flash Enabled XHR Pattern, zuletzt Abgerufen 21.12.2010

Das die Same Origin Restriktion für XmlHttpRequest auch valide Anwendungsfälle verhindert ist den Browserherstellern unterdessen bewusst. Viele haben begonnen oder werden eine separate XmlHttpRequest Komponente einführen, die Cross Domain Requests zulassen. Allerdings ist diese Komponente in jedem Browser unterschiedlich und für den praktischen Einsatz noch zu wenig verbreitet.

4.4.2. Keine Unterstützung von IE6 und älter

Der Javascript Code zum Fingerprint erfassen funktioniert nicht in IE Versionen kleiner gleich 6. Für die Behebung der Fehler blieb am Ende keine Zeit.

4.4.3. Webbrowser ohne Javascript

Die Identifikation von simplen Textbrowsern, Browsern mit deaktiviertem Javascript gestaltet sich schwierig. Der grösste Teil der Informationsbeschaffung läuft über Javascript. Die Entropy der HTTP Header ist üblicherweise zu klein für eine eindeutige Identifikation des Browsers.

Als Indikator ob eine Session gestohlen wurde könnten die Informationen im HTTP Header ausreichen. Der Browser muss in diesem Fall nicht eindeutig Identifiziert werden - es muss lediglich erkannt werden, dass er geändert hat.

4.4.4. Vergleich der Browser Features

Während dem Verlauf des Projektes, kam die Idee auf verschiedene Browsermerkmale zu vergleichen und dem Benutzer eine Auswertung zu Präsentieren. Es sollte Beispielsweise geprüft werden ob die Browserfeatures mit dem gesendeten User-Agent übereinstimmen, also beispielsweise sollte “userData” beim User-Agent “Internet Explorer”, aber nicht bei “Firefox” vorhanden sein. Dies wurde aus Zeitmangel gestrichen.

5. Ausblick in die Zukunft

Die Möglichkeiten zur Datensammlung wurden in dieser Semesterarbeit nicht ausgeschöpft. Angefangen von TCP/IP Fingerprinting über auslesen von Systeminformationen per Silverlight gibt es noch viele Möglichkeiten.

Wie die Recherche ergeben hat wird diese Technologie bereits eingesetzt. Das potenzial ist definitiv Vorhanden. Welcher Lieferant von Banner Werbung findet es schon Toll, dass Benutzer ihre Cookies löschen können. Deshalb werden Bereits jetzt Techniken wie Flash Cookies eingesetzt, die schwieriger zu löschen sind. Von den gefundenen Produkten sind viele im Security Bereich. Hier ist ein zusätzliches Instrument um sicherzustellen, dass man mit dem korrekten Gegenüber kommuniziert bestimmt willkommen.

Zur Zeit ist es sehr einfach Fingerprints von Browsern zu erstellen. Es können so viele Informationen über den Besucher ermittelt werden, dass es einem sehr einfach gemacht wird. Dies wird sich in Zukunft noch verstärken, da mit der Schrittweisen Implementation von diversen HTML5 Features, jedes neue Browser-Release bereits deshalb einen eigenen Fingerprint besitzt.

Die von Panopticlick angestossene Diskussion könnte eine Verbesserung erwirken, indem Browser dahingehend getrimmt werden, dass sie standardmässige Anonymer sind. Ein Beispiel sind anonymere Versionsnummern nur 6.1 statt 6.1.3.4567.

Projektplan

Studienarbeit:
"IT-Security: Browser Uniqueness - Identifying Users"
Florian Hengartner
2010

1. Inhaltsverzeichnis

2. Änderungsgeschichte

Version	Kommentar	Datum
1.0	Erstellt	Dienstag, 12. Oktober 2010
1.1	Ergänzt	Mittwoch, 13. Oktober 2010
1.2	Fertiggestellt	Freitag, 15. Oktober 2010
1.3	Anpassung der Aufgabenstellung, Risikoanalyse, Code Guidelines und Backuplan integriert	Dienstag, 19. Oktober 2010
1.4	Review	Sonntag, 19. Dezember 2010

3. Einführung

3.1. Zweck

Dieses Dokument beschreibt den Entwicklungsprozess der Studienarbeit “IT-Security: Browser Uniqueness - Identifying Users”.

3.2. Gültigkeitsbereich

Dieses Dokument gilt als Grundlage für die gesamte Studienarbeit und hat Gültigkeit über die komplette Projektdauer.

3.3. Definitionen und Abkürzungen

Beschrieben im Abschnitt Glossar.

3.4. Referenzen (References)

- Panopticlick: https://panopticlick.eff.org/
- Aufgabenstellung: doc/01 Aufgabenstellung/Aufgabenstellung_BrowserUniqueness_v1.0.pdf

3.5. Übersicht

Nachfolgend wird die Projektidee beschrieben und auf die Projektorganisation eingegangen. Im Abschnitt „Management Abläufe“ wird das Zeitmanagement und der Projektplan mit den geplanten Meilensteinen detailliert aufgeführt. Im Abschnitt „Risiko Management“ werden mögliche Risiken aufgezeigt und ihre Auswirkungen abgeschätzt. Anschliessend folgt eine Auflistung der geplanten Arbeitspakete und der verwendeten Infrastruktur. Zudem werden die Qualitätsmassnahmen beschrieben.

4. Projekt Übersicht

Die Studienarbeit hat zum Ziel die Erstellung einer Demonstrationsseite für Browseridentifikation in Verbindung mit einer Social-Networking Applikation.

Folgende Ziele stellt die Aufgabenstellung:
1. Erstellen einer Webseite mit Datenbankanbindung zur Visualisierung
2. Machbarkeitsstudie einer Socialnetworking Applikation (z.b. Facebook)
3. Ist die Facebook-Applikation machbar, eine einfache Facebook-Applikation erstellen

5. Projektorganisation

Die Studienarbeit wird von Florian Hengartner erstellt.
Zuständiger Betreuer ist Walter Sprenger (Compass Security AG).

Der Projektprozess basiert auf dem RUP[11]. Das Vorgehen wird an die Bedürfnisse eines Ein-Mann Projektes angepasst. Die Dauer einer Iteration ist auf eine Woche festgelegt. Typischerweise wird immer am Ende der aktuellen Iteration die nächste bzw. übernächste Iteration geplant.

5.1. Optionale Features

Schwergewicht wird auf die Entwicklung der Demonstrationswebseite zur Browseridentifikation gelegt.
Die Social Networking App (Facebook Applikation) soll mindestens als Machbarkeitsstudie und falls machbar als eine kleine Demo App erstellt werden.

5.2. Annahmen und Einschränkungen

Für den Fingerabdruck werden vorallem Daten die von einer Webapplikation relativ einfach ermittelt werden können verwendet. Methoden wie TCP Stack Fingerprinting liegen nicht im Rahmen dieser Arbeit.

[11] Rational Unified Process - siehe http://en.wikipedia.org/wiki/RUP und “Managers Introduction to RUP.pdf”

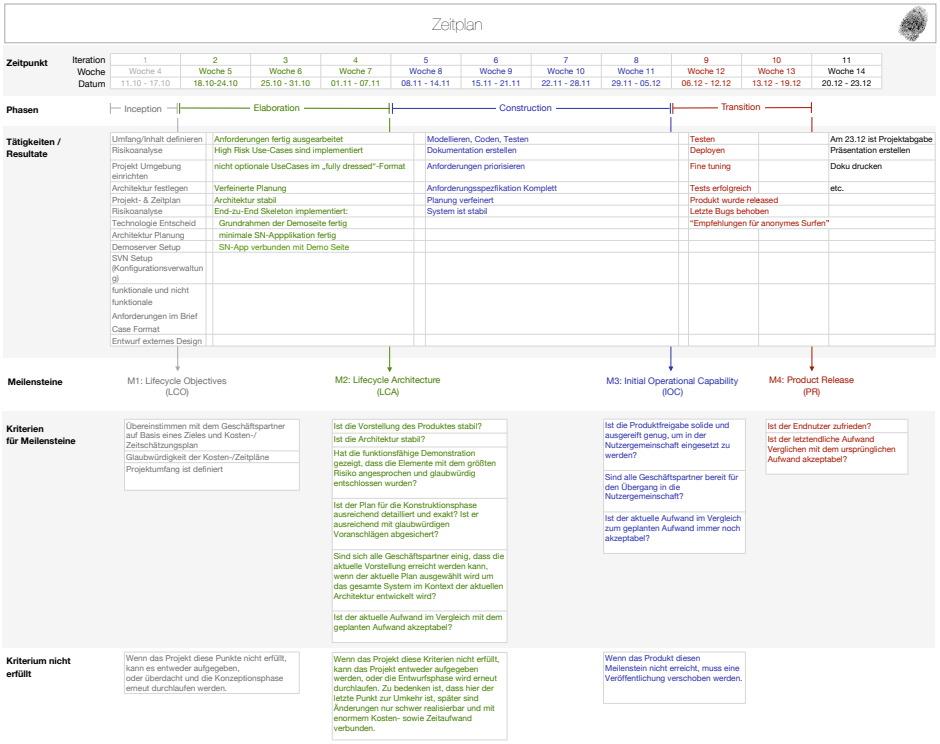
6. Management Abläufe

6.1. Projekt Kostenvoranschlag

Für die Studienarbeit werden 8 ECTS Punkte vergeben, was einer Leistung von 240 Stunden entspricht.

6.2. Projektplan

6.2.1. Zeitplan



6.2.2. Besprechungen

Grundsätzlich findet jede Woche eine Sitzung zwischen Student und Betreuer statt. Je nach Bedarf werden zusätzliche Sitzungen einberufen oder Sitzungen gestrichen.

7. Risiko Management

ID	Risiko	Auswirkung	Massnahmen	Kosten der Massnahmen [Std]	Max. Schaden [Std]	Wahrscheinlichkeit des Eintreffen	Gewichteter Schaden [Std]	Priorität
1	Projektdaten gehen verloren	Arbeit geht verloren	Einsatz von Subversion Repository und Backup	4	50	5%	2.5	klein
2	Teammitglied fällt aus infolge Unfall/ Krankheit	Verzögerung des Projektverlaufs	Zeitreserven einplanen	-	240	10%	24	sehr hoch
3	Projektanforderungen ändern	Projektplanung muss überarbeitet werden	Regelmässige Sitzungen, Zeitreserven einplanen	-	10	20%	2	klein
4	Aufwand für die Erstellung einer Facebook Applikation sehr hoch	Facebook Applikation wird nicht fertig	Mindestens die Machbarkeit abklären	2	48	15%	7.2	mittel
5	Zugriff in Facebook Applikation auf Browserfringerprinting-Informationen sind eingeschränkt/ nicht möglich	Die Facebook Applikation kann nicht im geplanten umfang erstellt werden.	Während Elaboration Phase abklären	10	15	40%	6	mittel

8. Arbeitspakete

Siehe die Issue (Tasks) Planung im Redmine: https://ssl.hengartner.biz/services/redmine_new/projects/sa

Zusätzlich ist hier ein Screenshot der Planung.

#	Tracker	Übergeordnete Aufgabe	Status	Thema	Beginn	Abgabedatum	Geschätzter Aufwand	% erledigt
138	Planung		Erledigt	1010 - Projekt und Zeitplan erstellen	11.10.2010	17.10.2010	14.0	
139	Planung		Erledigt	1020 - Codierungsrichtlinien	11.10.2010	18.10.2010	1.0	
140	Planung		Erledigt	1030 - Versionsverwaltung	11.10.2010	17.10.2010	2.0	
141	Planung		Erledigt	1040 - Dokumentvorlagen erstellen	15.10.2010	17.10.2010	1.0	
142	Planung		Erledigt	1050 - Q-Massnahmen	11.10.2010	17.10.2010	3.0	
143	Planung		Erledigt	1060 - Risiko Management	11.10.2010	17.10.2010	0.5	
144	Planung		In Bearbeitung	1070 - Sitzungen	12.10.2010	15.12.2010	20.0	
207	Planung		In Bearbeitung	1080 - Planung (Zeit/Tasks) aktualisieren am Ende der Iteration	11.10.2010	05.12.2010	15.0	
145	Planung		Erledigt	2010 - Use Case Brief	11.10.2010	17.10.2010	1.0	
146	Planung		Neu	2010 - Use Case Fully Dressed	18.10.2010	14.11.2010	4.0	
147	Planung		Neu	3010 - Domain Modell	11.10.2010	05.12.2010	5.0	
148	Planung		Neu	3020 - System Sequenzdiagramme	15.10.2010	05.12.2010	3.0	
149	Planung		Neu	3030 - Operation Contracts	15.10.2010	05.12.2010	2.0	
150	Planung		Neu	3040 - Activity Diagram	15.10.2010	05.12.2010	2.0	
151	Planung		Neu	3050 - Externes Design UI	11.10.2010	07.11.2010	6.0	
153	Planung		Erledigt	3060 - Analyse Browser/Fingerprinting	11.10.2010	24.10.2010	10.0	
206	Planung		Neu	3070 - Analyse Architektur	18.10.2010	07.11.2010	5.0	
154	Planung		Neu	4010 - Design Model	18.10.2010	23.12.2010	10.0	
155	Planung		Neu	4020 - Logische Architektur	11.10.2010	23.12.2010	3.5	
156	Planung		Neu	5010 - Prototyp Demoseite	11.10.2010	07.11.2010	29.0	
157	Planung		Neu	5020 - Prototyp Facebook App	18.10.2010	07.11.2010	15.0	
158	Planung		Neu	5030 - Projektautomation	15.10.2010	07.11.2010	5.0	
159	Planung		Neu	5040 - Implementation Demoseite	08.11.2010	05.12.2010	40.0	
160	Planung		Neu	5040 - Implementation Facebook App	08.11.2010	05.12.2010	32.0	
161	Planung		Neu	6010 - Unit Tests	15.10.2010	23.12.2010	20.0	
162	Planung		Neu	6020 - Bugfixing	29.11.2010	23.12.2010	5.0	
163	Planung		Neu	6040 - Systemtest	04.11.2010	19.12.2010	5.0	
164	Planung		Neu	7010 - Technische Dokumentation	15.10.2010	23.12.2010	6.0	
165	Planung		Neu	7020 - Webseite	13.12.2010	23.12.2010	5.0	
166	Planung		Neu	7040 - Schlusspräsentation	12.12.2010	23.12.2010	8.0	

9. Infrastruktur

An folgenden Orten wird entwickelt:

- Persönliches Laptop (Macbook Pro, OSX 10.5)
- Computer für Studienarbeit an der HSR

Applikationsserver für Demoseite und Facebook App: persönlicher Webserver von Forian Hengartner

Subversion Server: ebenfalls auf dem Webserver von Florian Hengartner.

9.1. Requirements/Issue Tracking

Für das Management wird die Software Redmine verwendet. Dies ist eine Webapplikation welche ein Wiki, integration mit Subversion, Requirement/Issue Tracking, Stundenerfassung und weiteres unterstützt.

9.2. Backup

Dokumente und Sourcecode müssen im Subversion Repository abgelegt werden.

Vom Backup erfasst werden:
- Subversion Repository
- Mysql Datenbank (Fingerprint Daten)

Diese Daten befinden sich auf dem Entwicklungsserver und werden dort 1x Täglich gebackupt. Das Backup wird auf dem vom Hoster zur verfügung gestellten Backupserver abgelegt.

Als zweites “Backup” dient der ausgecheckte Code und Dokumentationsstand auf dem Laptop von Florian Hengartner.

10. Qualitätsmassnahmen

10.1. Dokumentation

Die Dokumentation soll am Ende jeder Iteration aktualisiert worden sein.

Es werden im Code bei komplexen Fragmenten Kommentare angebracht. Alle Klassenschnittstellen werden mit den in der Sprache üblichen Methode (javadoc, tripple slash doc etc.) dokumentiert.

10.2. Sitzungsprotokolle

Sämtliche Sitzungen werden protokolliert. So werden spätere Missverständnisse vermieden und Entscheidungen festgehalten. Sitzungen werden im Wochenrhythmus gehalten, der Dienstag wird dabei bevorzugt, kann jedoch bei Terminkonflikt verschoben werden.

10.3. Projekt- und Zeitplan aktualisieren

Die Arbeitszeit wird im Redmine Web Interface eingetragen. Somit kann die Soll/Ist Planung jederzeit effektiv überprüft werden. Spätestens am Ende jeder Iteration sollen die Einträge dem aktuellen Stand entsprechen.

10.4. ToDo-Listen

Auf dem Redmine Web Interface werden alle ToDo’s inklusive Aufwandsschätzung als “Issue” eingetragen. Später wird dann die geleistete Zeit eingetragen.

10.5. Reviews

Da dieses Projekt von einer Person erarbeitet wird ist es nicht möglich Arbeiten im Team gegenseitig zu kontrollieren. Jedoch soll am Ende jeder Iteration die erstellten Dokumente kontrolliert werden sowie im Aufwand von ca. einer Stunde der Code gereviewt werden, dabei sollen fehlende Kommentare hinzugefügt und wo nötig Refactorings durchgeführt werden.

10.6. Code Guidelines für Ruby On Rails

Die offiziellen Guidelines bestehen aus wenigen Punkten (siehe Unten). Bei den Referenz Links gibt es einige zusätzliche Informationen.

10.6.1. Guidelines

- Two spaces, no tabs
- Blank lines should not have spaces
- Don't use and and or for boolean tests, instead always use && and ||
- MyClass.my_method(my_arg) -- not my_method(my_arg) or my_method my_arg
- 'a = b' and not 'a=b'
- Follow the conventions you see used in the source already

10.6.2. Referenzen

1. Blog Article - Style Guide: http://www.pathf.com/blogs/ruby-and-rails-style-guide/
2. RubyOnRails Style Guide: https://rails.lighthouseapp.com/projects/8994/source-style
3. Ruby Style Guide: http://www.caliban.org/ruby/rubyguide.shtml#style
4. 2009 Ruby Survey: http://robots.thoughtbot.com/post/308239139/2009-ruby-survey-results
   https://thoughtbot.wufoo.com/reports/style-part-1/
   https://thoughtbot.wufoo.com/reports/style-part-2/

10.7. Versionsverwaltungssystem

Die gesamte Dateistruktur (Dokumente und Quellcode) wird mit Subversion unter die Versionsverwaltung gestellt. Dies erlaubt das Nachverfolgen von Änderungen und das wiederherstellen von alten Datenständen.

10.8. Tests

10.8.1. Unit-Tests

Während der Entwicklung werden wo möglich und sinnvoll Unit-Tests durchgeführt. Diese Tests müssen erfolgreich durchlaufen, nach jeder Iteration.

10.8.2. System-Tests

Auf den Use-Cases basierend werden in der Transition Phase Systemtests durchgeführt und protokolliert. Diese überprüfen, ob die Software den funktionalen Anforderungen genügt. Diese Tests sind Black Box Tests und sollen 100% der Use Cases abdecken.

10.8.3. Usability-Tests

Potentielle externe Benutzer testen die Prototypen nach jedem Meilenstein und geben Feedback. Das Feedback wird ausgewertet und gibt wertvolle Inputs für die weiteren Projektiterationen.

10.9. Automatisierung

Soweit sinnvoll (Aufwand/Ertrag) soll die Entwicklungs-/Buildumgebung automatisiert werden.

Anforderungsspezifikation

Studienarbeit:
"IT-Security: Browser Uniqueness - Identifying Users"
Florian Hengartner
2010

1. Inhaltsverzeichnis

2. Änderungsgeschichte

Version	Kommentar	Datum
1.0	Erstellt	Montag, 18. Oktober 2010
1.1	UC1 Fully Dressed hinzugefügt	Montag, 25. Oktober 2010
1.2	UC2 Fully Dressed hinzugefügt	Dienstag, 26. Oktober 2010
1.3	Leere Stellen entfernt / ergänzt	Dienstag, 9. November 2010
1.4	UC2/UC3 vereinigt, UC4/UC5 vereinigt. Alle Usecases sind Fully Dressed.	Sonntag, 14. November 2010
1.5	Review	Sonntag, 19. Dezember 2010

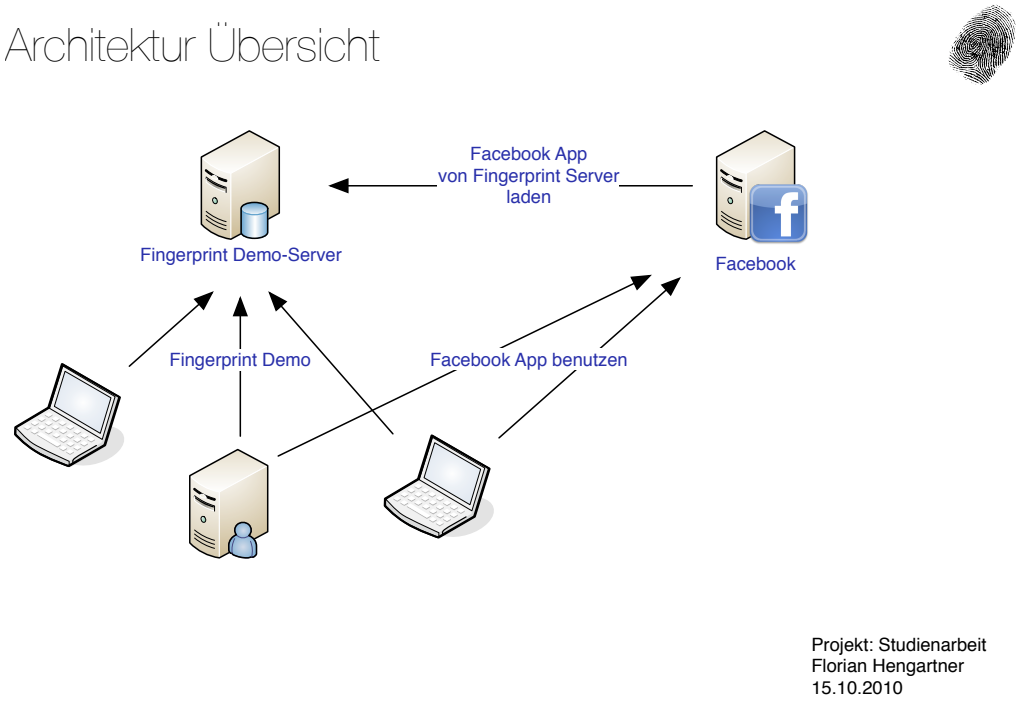
3. Übersicht

Dieses Dokument ist aufgeteilt in non-funktionale Anforderungen und funktionale Anforderungen in Form von Use Cases.

4. Allgemeine Beschreibung

4.1. Produkt Perspektive

Ziel ist es die Identifizierbarkeit eines Browsers mit Hilfe einer Webseite zu Demonstrieren. Zusätzlich soll mit einer Social Network Applikation der Browser einer bestimmten Person zugeordnet werden können.



4.2. Produkt Funktionen

- Fingerprint erfassen und in Datenbank speichern
- Browser aufgrund der gespeicherten Fingerprints wiedererkennen
- Social-Network Applikation sammelt Fingerprints & persönliche Benutzerinformation
- In Social-Network ausgeloggte User aufgrund des Fingerprints wiedererkennen

4.2.1. Optionale Funktionalität

- Social-Network Applikation bei Zeitmangel auf minimum reduziert
- Social-Network Applikation oder Demoseite als Spiel tarnen um eine grössere Nutzer und somit Datenbasis zu gewinnen
- Schnittstelle (z.b. Javascript Include) um auf externen Webseiten Fingerprints zu erfassen
- Erkennen wenn Browsersession gestohlen wurde

4.3. Benutzer Charakteristik

Wer die Demoapplikation Benutzt möchte erfahren wie Identifizierbar sein Browser ist.

Die zweite Benutzergruppe weiss möglicherweise nichts von ihrem “Glück” - indem nämlich Fingerprints für das Testen und statistische Auswerten auf bestimmten Webseiten gesammelt wird.

4.4. Einschränkungen

Verwendung von Daten für das Fingerprinting die gut aus einer Webapplikation erfassbar sind.

Zum Beispiel: Kein TCP Stack Fingerprinting.

4.5. Abhängigkeiten

Der Socialnetwork Applikation muss es möglich sein die benötigten Daten zu erfassen. Falls sie zum Beispiel kein Zugriff auf die vom Benutzer gesendeten HTTP_REQUEST Informationen oder kein Javascript ausgeführt werden kann, kann ein grosser Teil der Daten nicht ermittelt werden.

5. Spezifische Anforderungen

5.1. Funktionale Anforderungen

5.1.1. Sicherheit

Vertrauliche Benutzerdaten dürfen nie an Dritte gelangen.

5.1.2. Interoperabilität

Der Server soll Plattformunabhängig betrieben werden können.

5.1.3. Angemessenheit

Der Server sollte eine Last von 100 Usern die gleichzeitig die Webseite verwenden aushalten.

5.2. Zuverlässigkeit

5.2.1. Verfügbarkeit

Da es sich um eine Demo Applikation handelt, welche nicht im produktiven Einsatz ist: “Best Effort”.

5.3. Schnittstellen

5.3.1. Softwareschnittstelle

- Zwischen der Demoapplikation und dem Socialnetwork
- Interface zum erfassen von Fingerprints von externen Webseiten

5.4. Lizenzanforderungen

Es werden keine Lizenzen benötigt

6. Use Cases

6.1. Brief

6.1.1. UC1: Browserfingerprint erfassen und anzeigen

Wenn der User die Demo startet wird der Fingerprint erfasst und verfügbare Browserinformationen angezeigt.

6.1.2. UC2: Browserfingerprint vergleichen

Der User kann auf der Demoseite eine Session erzeugen. Es wird ihm dann eine eindeutige Session ID zugewiesen (URL).

Der User nimmt die Session ID und greift auf die Seite z.b. von einem anderen Browser zu. Die Demoapplikation zeigt die unterschiede des Fingerprints auf.

6.1.3. UC3: Socialnetwork Applikation - Benutzerdaten sammeln

Der Benutzer ist im Socialnetwork eingeloggt und verwendet die App. Dabei wird der Browserfingerprint zusammen mit der Benutzeridentität gespeichert.

Nach Verwendung der Socialnetwork Applikation und wenn der User dort wieder ausgeloggt ist, öffnet er eine Demo Seite die ihn immernoch erkannt und ihm z.b. seinen Namen anzeigt.

6.2. Fully Dressed

6.2.1. UC1: Browserfingerprint erfassen und anzeigen

User

User möchte seinen Fingerprint sehen.

Fingerprint wird berechnet, in Datenbank abgespeichert und dem User angezeigt.

1. Der User öffnet die Webseite und sieht die Willkommen-Seite.
2. Der User klickt auf den Link und die Fingerprint-Seite wird geöffnet
3. User sieht sich auf dieser Seite seinen Fingerprint an

2.a Der User klickt auf den Link und die Fingerprint-Seite für Browser ohne Javascript wird geöffnet
3.a User sieht auf dieser Seite einen minimalen Fingerprint

keine

10-500 Benutzer pro Tag.

Im Schnitt macht jeder Benutzer 2-3 Aufrufe.

Benutzungszeit: vorwiegend zwischen 0800 Uhr und 2000 Uhr MEZ.

6.2.2. UC2: Browserfingerprint vergleichen

User

User möchte den Fingerprint von z.b. zwei verschiedenen Browsern vergleichen.

Benutzer befindet sich auf der Fingerprint-Seite (Seite 2).

Es wurde eine eindeutige stabile URL zur Session-Identifikation erstellt, auf dieser kann der User die differenzen ansehen.

1. Der User klickt auf den Link zur erzeugung einer ID
2. Er wird auf die stabile Session URL umgeleitet
3. Der Benutzer kopiert die URL und öffnet Sie in einem anderen Browser

3.a. Der Benutzer deaktiviert Flash, Java, Javascript, bestimmte Plugins, etc. und lädt die URL neu.

keine

1 - 500 Aufrufe pro Tag.

6.2.3. UC3: Erkennen von Socialnetwork Usern

User

Facebookbenutzerdaten aufgrund des Fingerprints dem User anzeigen.

Benutzer hat einen Facebook Account.

Facebook Account Daten werden auf Demoseite angezeigt.

1. Benutzer loggt sich bei Facebook ein
2. Benutzer sucht die Demo Facebook App
2.1 Benutzer öffnet die Demo Facebook App
3. Benutzer geht auf die Demo Webseite und öffnet die Fingerprint Seite
3.1 Dem Benutzer werden nun die Facebook Account Daten angezeigt.

1.a Benutzer ist bereits eingeloggt
2.a Benutzer öffnet das bereits gespeicherte Bookmark der App

keine

1 - 500 Aufrufe pro Tag.

Domain Analyse

Studienarbeit:
"IT-Security: Browser Uniqueness - Identifying Users"
Florian Hengartner
2010

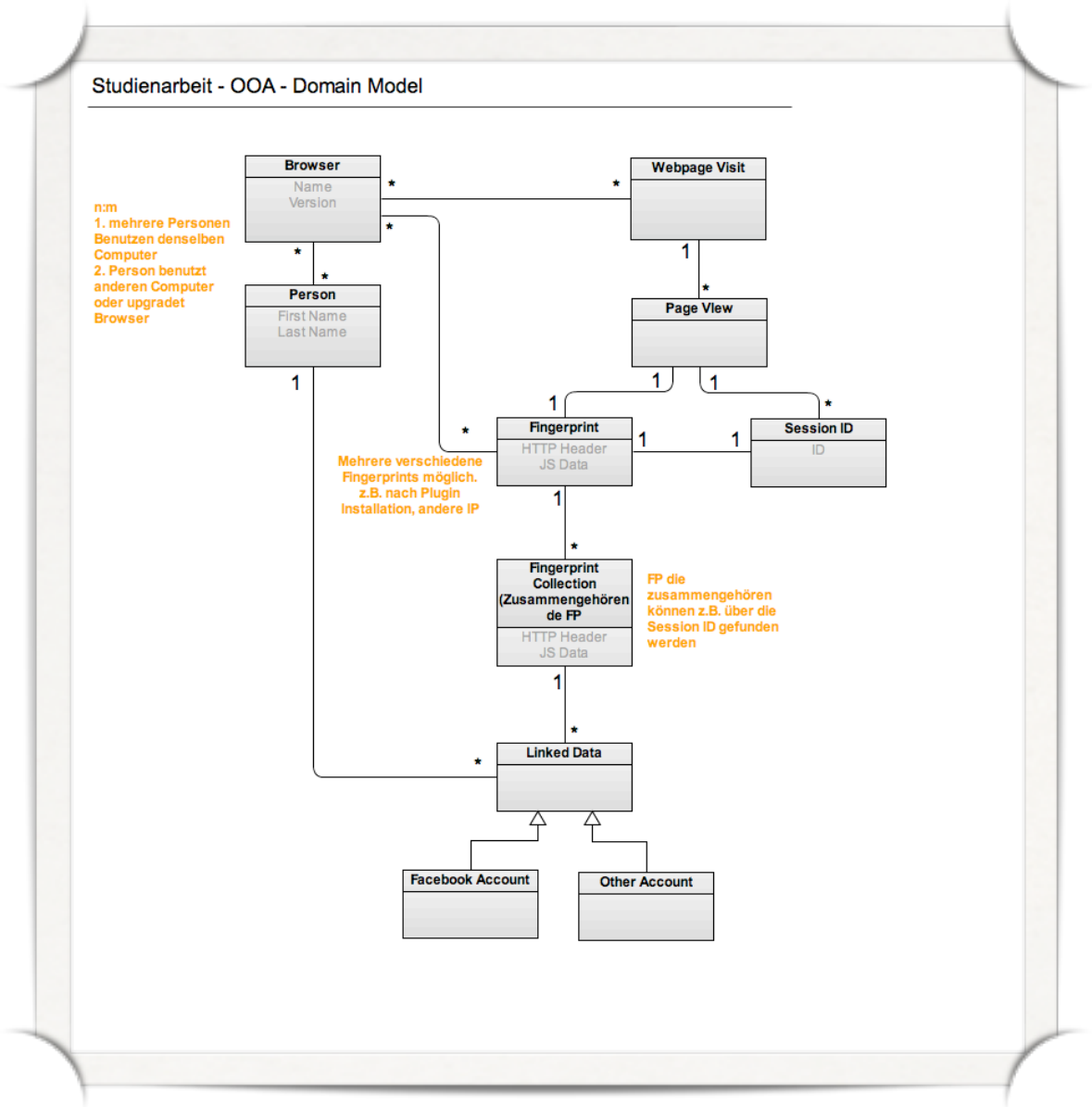
1. Inhaltsverzeichnis

2. Änderungsgeschichte

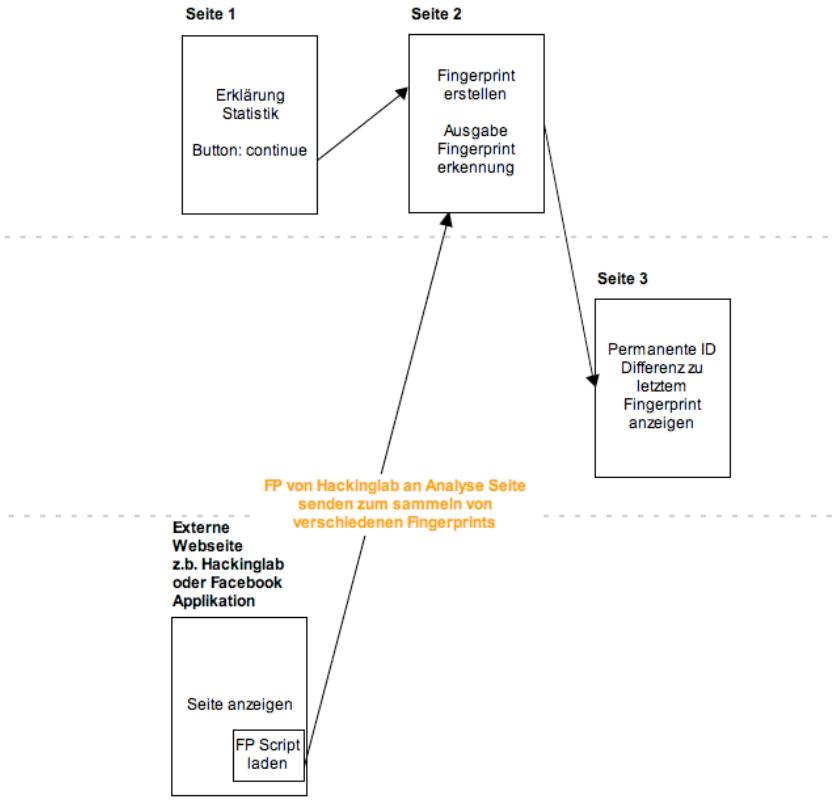
Version	Kommentar	Datum
1.0	Erstellt	Freitag, 22. Oktober 2010
1.1	Erweitert	Montag, 25. Oktober 2010
1.2	Sequenzdiagramme durch neue Version ersetzt	Samstag, 30. Oktober 2010
1.3	Abschnitt über Entropy erweitert	Montag, 1. November 2010
1.4	Informationen zu Facebook Applikation hinzugefügt	Dienstag, 2. November 2010
1.5	Review	Sonntag, 19. Dezember 2010

3. Domain Modell

3.1. Strukturdiagram

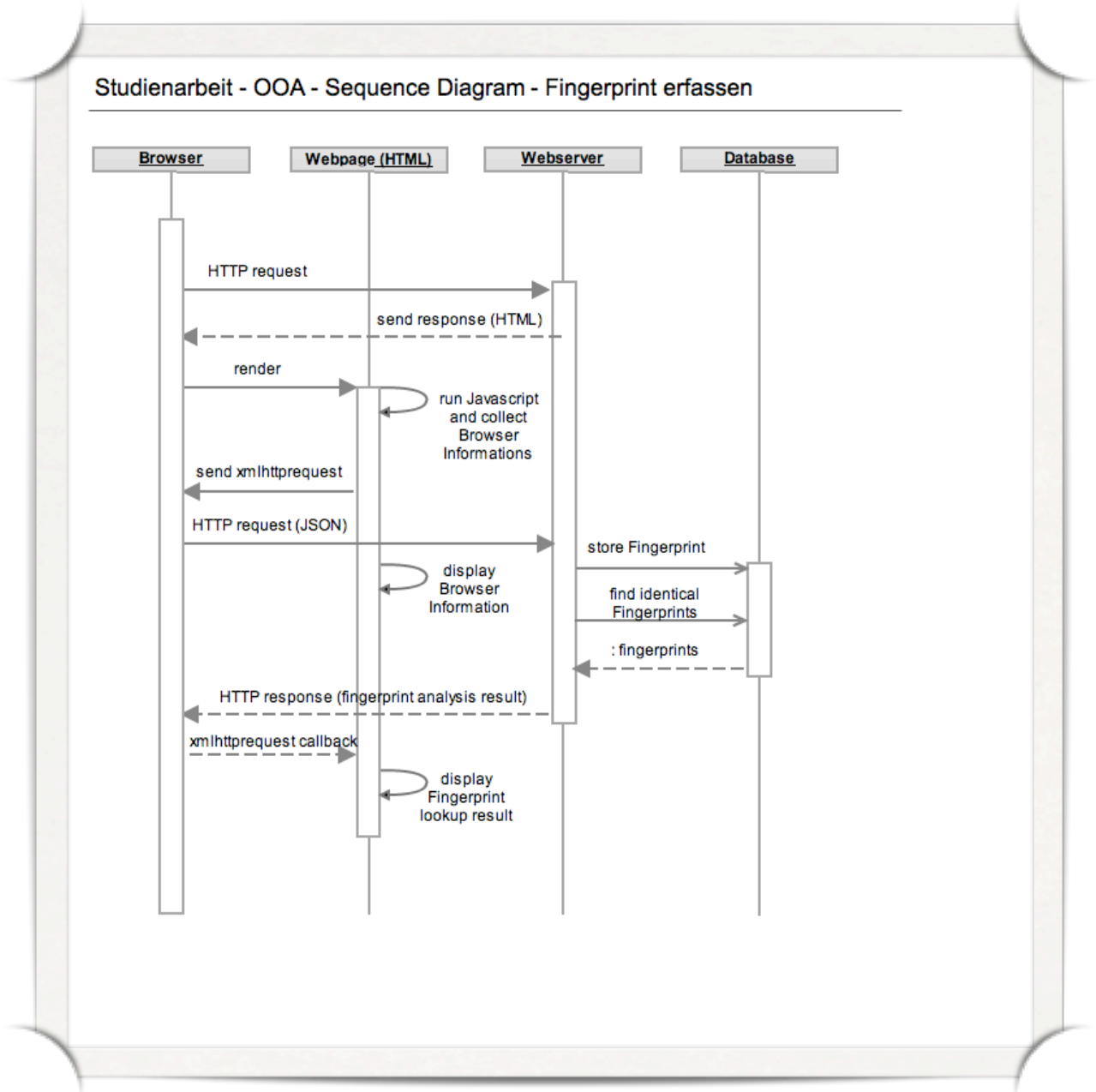


3.2. Beziehungen der Webseiten

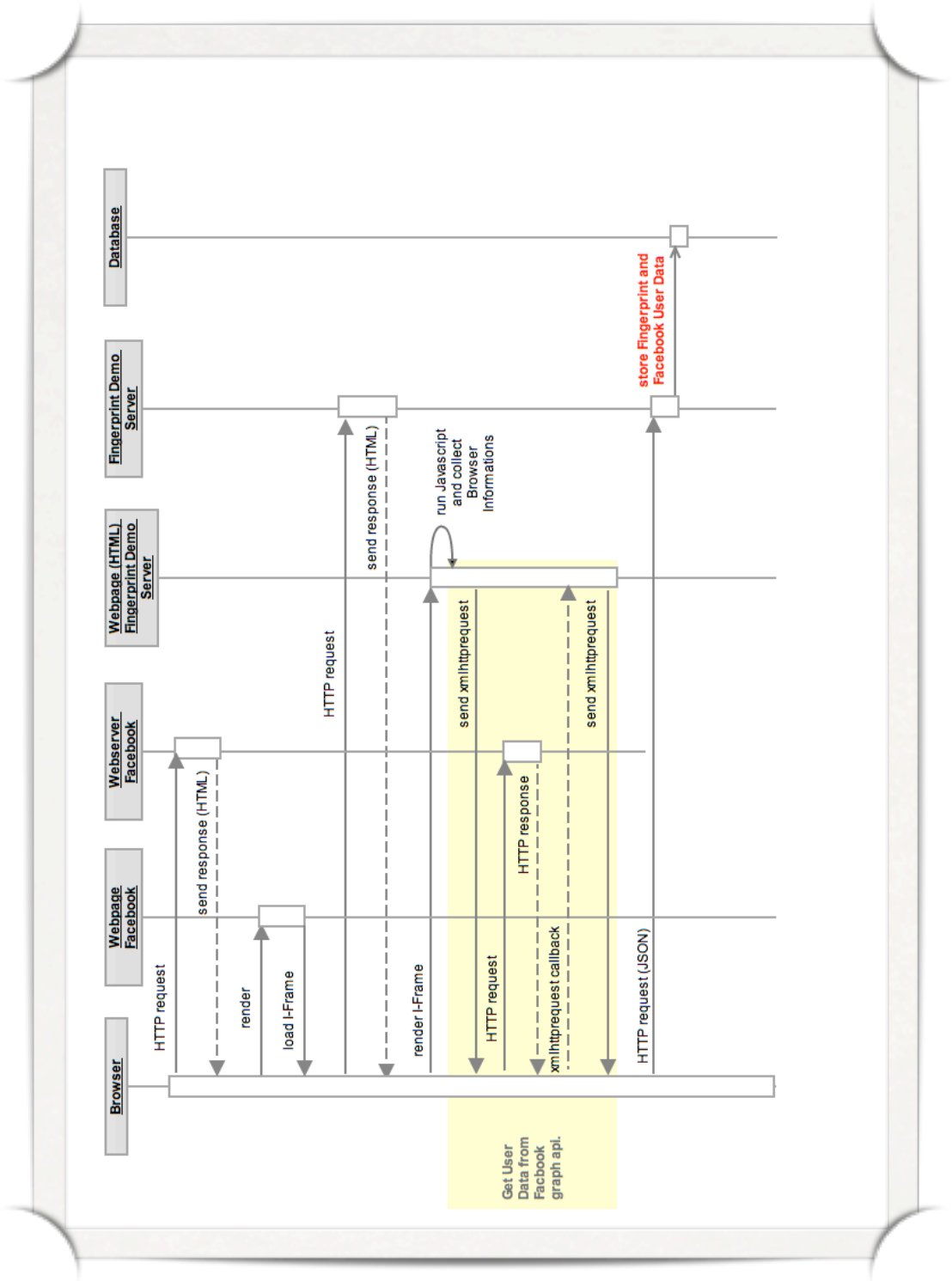


4. System Sequenzdiagramme

4.1. UC1: Browserfingerprint erfassen und anzeigen



4.2. UC3: Social-Networking Applikation - Benutzerdaten sammeln



5. Analyse

5.1. Einsatzszenarien

Der Browserfingerprint bietet verschiedene Einsatzszenarien:

1. Browser wiedererkennen
   - Verwendet bei: Panopticlick
   - Ermöglicht: Cookies wiederherstellen
2. Während Session darf sich Fingerprint nicht ändern oder nur um X%
   - Ermöglicht: Session Hijacking erkennen/verhindern
   - Kritische Attribute dürfen sich nicht ändern!
     - Z.b. während der Session plötzlich IP aus einem anderen Netzblock
3. Bei jedem Login prüfen ob der User von einem bekannten verifizierten Browser einloggt
   - Verwendet bei Facebook: Siehe http://blog.facebook.com/blog.php?post=389991097130
   - Falls der Browser unbekannt ist: Den User zusätzlich authentisieren

5.1.1. Schwierigkeiten:

1. Differenzierung von Usern die z.b. hinter Firmenproxies sind
   - IP könnte während Session hin und her wechseln
   - Sehr hohe ähnlichkeit des Fingerprints. 2 unterschiedliche User könnten als 1 einziger Identifiziert werden.
2. WAI / Legitime User werden ausgeschlossen: Facebook hat mit der zusätzlichen Authentisierung legitime User ausgeschlossen, die z.b von einem Internet-Café zugreiffen wollten
   - Probleme sind:
   - Erkennen von Bildern von fremden Personen nicht möglich / Bilder von Gegenstände / Blinde Personen
     1. Person muss vor der zusätzlichen Authentisierung eine vertrauenswürdige Kontaktmöglichkeit hinterlegt haben

5.2. Informationstheoretische Grundlagen

Die Theorie wird in diesen Artikeln sehr gut beschrieben:

- https://www.eff.org/deeplinks/2010/01/primer-information-theory-and-privacy
- https://www.eff.org/deeplinks/2010/01/tracking-by-user-agent
- http://www.eff.org/deeplinks/2009/09/what-information-personally-identifiable

Begriffe:

- Entropie:
  - http://en.wikipedia.org/wiki/Entropy_%28information_theory%29#Definition
  - http://de.wikipedia.org/wiki/Entropie_%28Informationstheorie%29
  - http://de.wikibooks.org/wiki/Entropie:_IT
- Self-Information: http://en.wikipedia.org/wiki/Self-information
- Conditional Entropy: http://en.wikipedia.org/wiki/Conditional_entropy

5.3. Bei wieviel Entropy ist eine Person eindeutig identifizierbar?

Um eine Person eindeutig zu Identifizieren benötigt man folgende Entropie (S):

$$S = \log_2(\frac{1}{\text{Weltbevölkerung}})$$

$$S = \log_2(\frac{1}{6'892\ \text{Mio.}}) = 32.68\ \text{bits}$$

Logarithmus Dualis

$$\log_2 = \frac{\ln(x)}{\ln(2)}$$

Ist die Entropie (S) der von einem Browser gesendeten Daten grösser als 32.68 bits, so kann man annehmen, dass diese Informationen nur von einer Person verwendet werden.

In der Praxis kann dieser Wert nicht als absolut, sondern als Hinweis angesehen werden, da unter Umständen geklonte Konfigurationen zu 100% Identisch sind.

5.4. Resultate bei Panopticlick und was wir daraus lernen können

- Fingerprints hat min. 18.1 bits Entropy
  - d.h. 1 von 286'777 Browsern hat denselben Fingerprint
- Sind Flash und Java aktiviert, führt dies zu einer Entropy von mindestens 18.8 bit
- 94.2% der Browser mit Flash oder Java sind Unique
- Fingerprints ändern sich über die Zeit (upgrades), die Änderungen können aber einfach verfolgt werden. (99.1% Erfolgsquote)
- Bei der Statistischen Auswertung muss beachtet werden, dass Personen oftmals die Seite besuchen und in einem zweiten Anlauf die Seite neu laden mit maximalen Privacy Settings (Kein JS, Flash, NoScript, ..) was zu einer Verzerrung führt.
  - Panopticlick schliesst deshalb suspekte Fingerprints von statistischen Analysen aus.
- Zur Regenerierung von Cookies reichen zusammen mit IP-Addresse/Subnetz/etc. 15-20 bits an Informationen

5.4.1. Learnings:

Daten unbedingt wie vom Browser gesendet 1:1 speichern. Panopticlick wurde auf eine Methode zur Identifizierung von Browsern aufgrund der Reihenfolge gewisser Werte aufmerksam gemacht, jedoch hatten sie diese Werte nur sortiert gespeichert.

5.4.2. Ideen für weitere Fingerprinting Mechanismen

- Differenz der Zeit des Browser Computers und des Servers
- CPU Clock Speed
- ActiveX und Silverlight erlauben Informationen wie CPU Typ usw. auszulesen

5.4.3. Ermittelte Daten[12]

The specific `fingerprint' information we collect is:

- The user agent string from each browser
- The HTTP ACCEPT headers sent by the browser
- Screen resolution and color depth
- The Timezone your system is set to
- The browser extensions/plugins, like Quicktime, Flash, Java or Acrobat, that are installed in the browser, and the versions of those plugins

[12] https://panopticlick.eff.org/privacy.php

- The fonts installed on the computer, as reported by Flash or Java.
- Whether your browser executes JavaScript scripts
- Yes/no information saying whether the browser accepts various kinds of cookies and "super cookies"
- In addition, we collect several kinds of `housekeeping' information to assist us in analyzing the fingerprint data. The housekeeping information is:
- Cookies
- Encrypted IP addresses
- Timestamps

5.5. Verwendung von Browserfingerprinting

- Panopticlick - Forschungsprojekt: https://panopticlick.eff.org/
- 41st Parameter - Fraud Detection: http://www.the41st.com/products.asp
- Iovation - Fraud Detection: http://www.iovation.com/rm-360/
- ThreatMatrix - Fraud Dection: http://threatmetrix.com/our-solutions/show-me-how-it-works/
- United Virtualities - Cookie Replacement/Restoring: http://www.imediaconnection.com/news/5392.asp
- Facebook - Zusätzliche Authentisierung: http://blog.facebook.com/blog.php?post=389991097130)
- Browserspy.dk - Demonstrationsseite zu Browser-Informationen: http://browserspy.dk/

5.6. Fingerprint möglichkeiten

Siehe Dokument “Browser Fingerprinting” von Compass Security.

5.7. Einschränkungen

Falls gewisse Mechanismen im Browser deaktiviert sind (z.b. Javascript, Java, ActiveX, Cookies, ...) ist die Berechnung eines Fingerprints erheblich schwieriger.

Es soll in diesem Fall ermittelt werden ob der Browser die jeweilige Technik nicht beherscht oder deaktiviert hat.

5.8. Facebook Applikation

5.9. Zugriff auf öffentliche Informationen ohne Autorisierung der Applikation

Ohne dass ein Benutzer eine Applikation hinzugefügt/authorisiert hat, kann man auf öffentlich freigegebenen Benutzerinformationen (Name, Profilfoto, etc.) zuzugreifen.

Z.b. über die URL http://graph.facebook.com/USER_ID

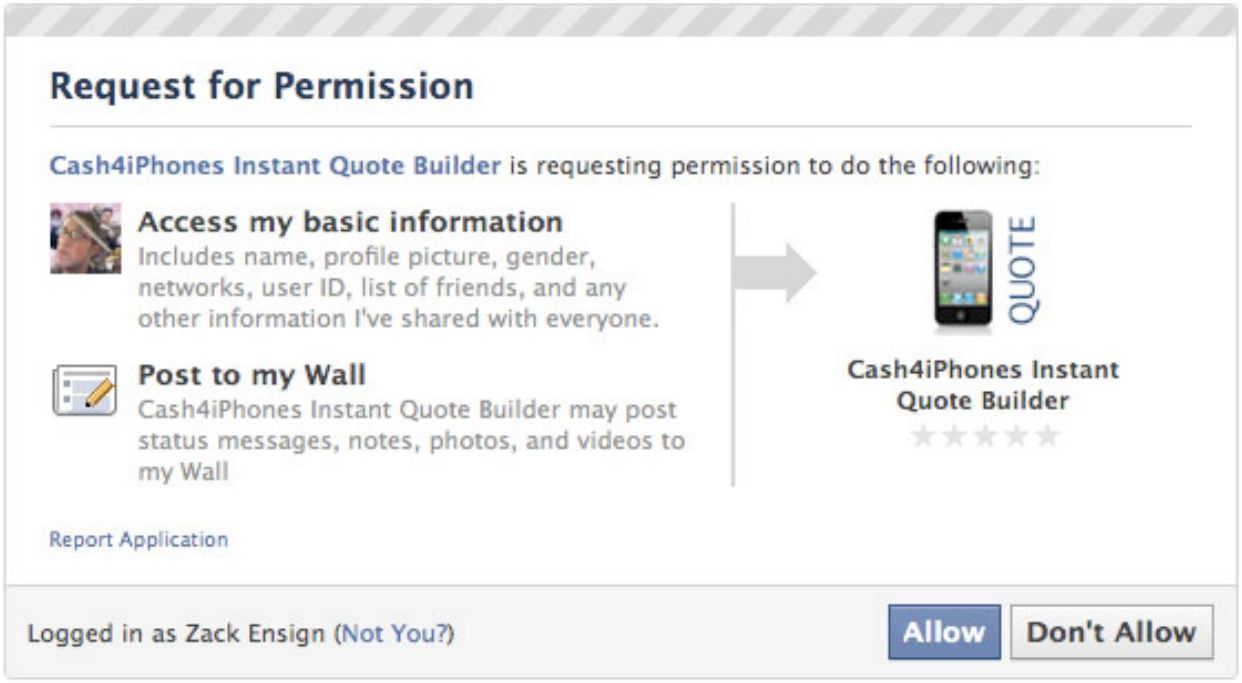
Der Haken dabei: die User-Id muss im vornherein bekannt sein. Es gibt keine möglichkeit die User-Id eines Users herauszufinden, bevor dieser eine Applikation authorisiert hat.

Somit kann man von einem Facebook Benutzer, dessen persönliche Informationen erst abrufen, wenn er die Applikation authorisiert hat!

Quellen:

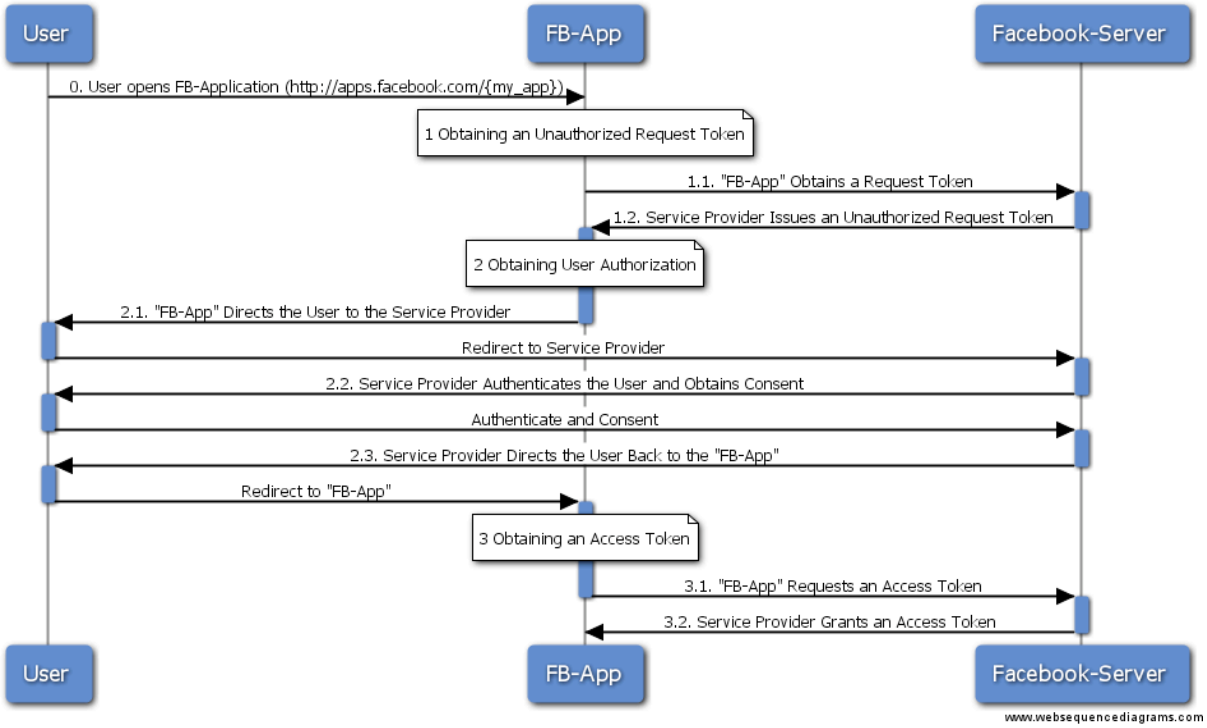
- http://developers.facebook.com/blog/post/135 (Etwas unklar beschrieben) Accessing User Information with Your Applications, zuletzt Abgerufen 21.12.2010
- http://forum.developers.facebook.net/viewtopic.php?id=72237 Getting public user data, zuletzt Abgerufen 21.12.2010
- http://forum.developers.facebook.net/viewtopic.php?id=74834 Fetchin User's Public Info, zuletzt Abgerufen 21.12.2010
- http://forum.developers.facebook.net/viewtopic.php?id=72798 Access to API without previous facebook user authentication, zuletzt Abgerufen 21.12.2010

Beispiel: Dialog zur Anforderung von Berechtigungen



Quelle: http://www.zen-sign.com/using-oauth-and-graph-apis-in-facebook-canvas-apps/

5.10. Authentisieren über OAuth



An Studienarbeit angepasste Vorlage des “OAuth Sequence Diagram Template”[13]

5.11. Wie funktionierte eine Facebook Applikation?

Es gibt zwei Applikationstypen. Die Canvas[14] und die FBML[15] Applikationen.

5.11.1. FBML Apps

FBML-Applikationen benutzten spezielle Markup Tags von Facebook (vermischt mit HTML). Sie interagieren nicht direkt mit dem Browser sondern werden von Facebook gerendet. Dieser Typ ist veraltet. Noch bis Ende 2010[16] können solche Applikationen erstellt werden.

---

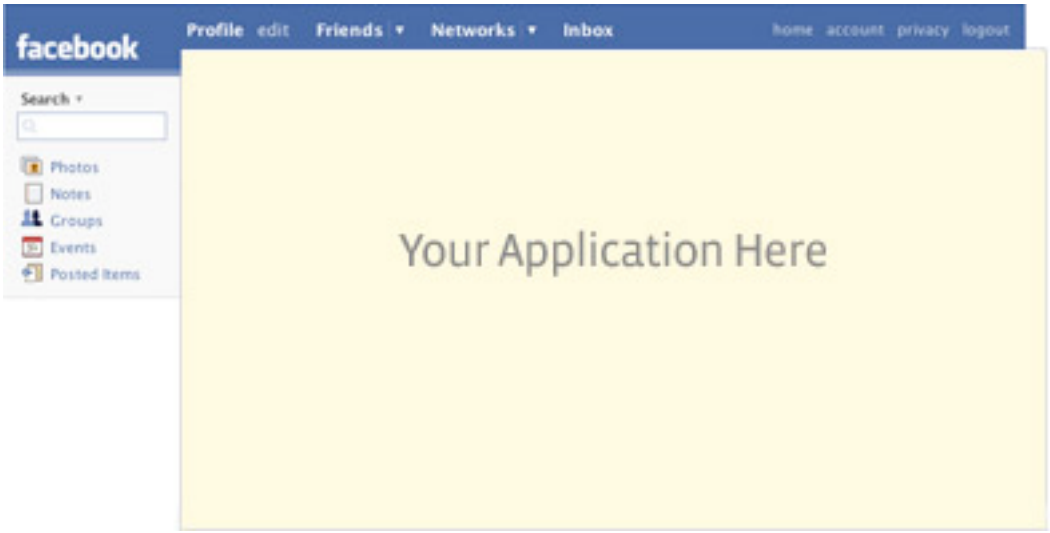
[13] http://d.hatena.ne.jp/ZIGOROu/20090811/1250006392 OAuth Sequence Diagram Template, zuletzt Abgerufen 21.12.2010

[14] http://developers.facebook.com/docs/guides/canvas/ Apps on Facebook.com, zuletzt Abgerufen 21.12.2010

[15] http://developers.facebook.com/docs/reference/fbml/ Facebook Markup Language (FBML), zuletzt Abgerufen 21.12.2010

[16] http://developers.facebook.com/blog/post/402 Facebook Platform Roadmap Update, zuletzt Abgerufen 21.12.2010

5.11.2. Canvas Apps



Quelle: http://www.keebler.net/blog/2007/06/02/facebook-application-basics/

Canvas Applikationen sind ein I-Frame das innerhalb der Facebook Webseite angezeigt wird.

Aus der Canvas Applikation kann man über die OAuth- und Graph-API mit Facebook kommunizieren, Benutzerdaten lesen und ändern. Sofern der User dies erlaubt hat, siehe Berechtigungen[17].

Weiterführende Links (alle zuletzt Abgerufen 21.12.2010):

- http://www.zen-sign.com/using-oauth-and-graph-apis-in-facebook-canvas-apps/
- http://www.keebler.net/blog/2007/06/02/facebook-application-basics/
- http://www.merchantos.com/makebeta/facebook/facebook-php-tutorial/

[17] http://developers.facebook.com/docs/authentication/permissions Extended Permissions, zuletzt Abgerufen 21.12.2010

5.12. Fingerprint ‘Engine’ von externem Webserver laden

Ganz praktisch wäre es wenn man die Fingerprinting Engine sehr einfach, z.b. nur Clientseitig auf beliebigen Seiten einbinden könnte. Damit könnte man einfach Daten sammeln für Testing und statistische Analyse der Browserpopulation.

5.12.1. Javascript von externem Webserver einbinden

Die einfachste Variante wäre das einbinden eines Javascript Tags im HTML, welches das Script von unserem Server holt.

5.12.2. Wird externes Javascript geladen falls Javascript deaktiviert ist?

Ein Test mit Firefox 3.6 und Safari 3.2 ergab, dass bei deaktiviertem Javascript, das Javascript-Include File nicht vom Webserver geladen wird. Die Fingerprint-”Engine” kriegt also gar nichts mit - nicht mal die normalen HTTP Header die beim Aufruf des Javascript Includes geloggt werden könnten.

5.12.3. Alternativen

1. Ein I-Frame inkludieren. Dieses wird immer geladen.
2. Zusätzlich zum Javascript ein Bild (1x1 Pixel, Transparent) auf dem Fingerprint Server abrufen. Somit erhält man zumindest einen “Hit” auf dem Webserver und kann die HTTP Header Daten analysieren.

Software Architektur

Studienarbeit:
"IT-Security: Browser Uniqueness - Identifying Users"
Florian Hengartner
2010

1. Inhaltsverzeichnis

2. Änderungsgeschichte

Version	Kommentar	Datum
1.0	Erstellt	Mittwoch, 1. Dezember 2010 (KW: 48)
1.1	Logische Architektur	Donnerstag, 2. Dezember 2010 (KW: 48)
1.2	Grafiken / Diagramme upgedatet	Montag, 20. Dezember 2010 (KW: 51)
1.3	Review	Dienstag, 21. Dezember 2010 (KW: 51)

3. Architektur Darstellung

Die Demo-Applikation ist als Ruby-On-Rails Applikation implementiert. Ruby-On-Rails ist ein Webframework das eine komplette Umgebung liefert.

Dazu gehören unter anderem:

1. ORM Mapper
2. MVC (Model-View-Controller)
3. Automatische CRUD generierung
4. Integriertes Build System (Rake)
5. Integriertes Test System
6. Unterstützt mehrere Umgebungen (Development/Testing/Production)
7. AJAX Funktionalität kann Serverseitig programmiert werden
8. u.v.m

RubyOnRails lebt nach den Grundsätzen von Dont-Repeat-Yourself (DRY) und Convention-over-Configuration (CoC) . Die Applikation ist im MVC Stil aufgebaut.

Die Models sind Abbildungen der Datenbanktabellen durch den OR-Mapper. Im Modelcode werden vorallem Relationen und Validierungen spezifiziert.

Nach CoC bildet sich eine URL direkt auf einen Controller und dessen Methoden und dem zugeordneten Model Objekt ab.

4. Architektonische Ziele & Einschränkungen

Ziele:

- Erstellen einer Web-Applikation welche Fingerprints von Browsern erstellen kann. Browser sollen über diesen Fingerprint Identifiziert werden können.
- Fingerprints sollen auch bei deaktiviertem Javascript erfasst werden können.

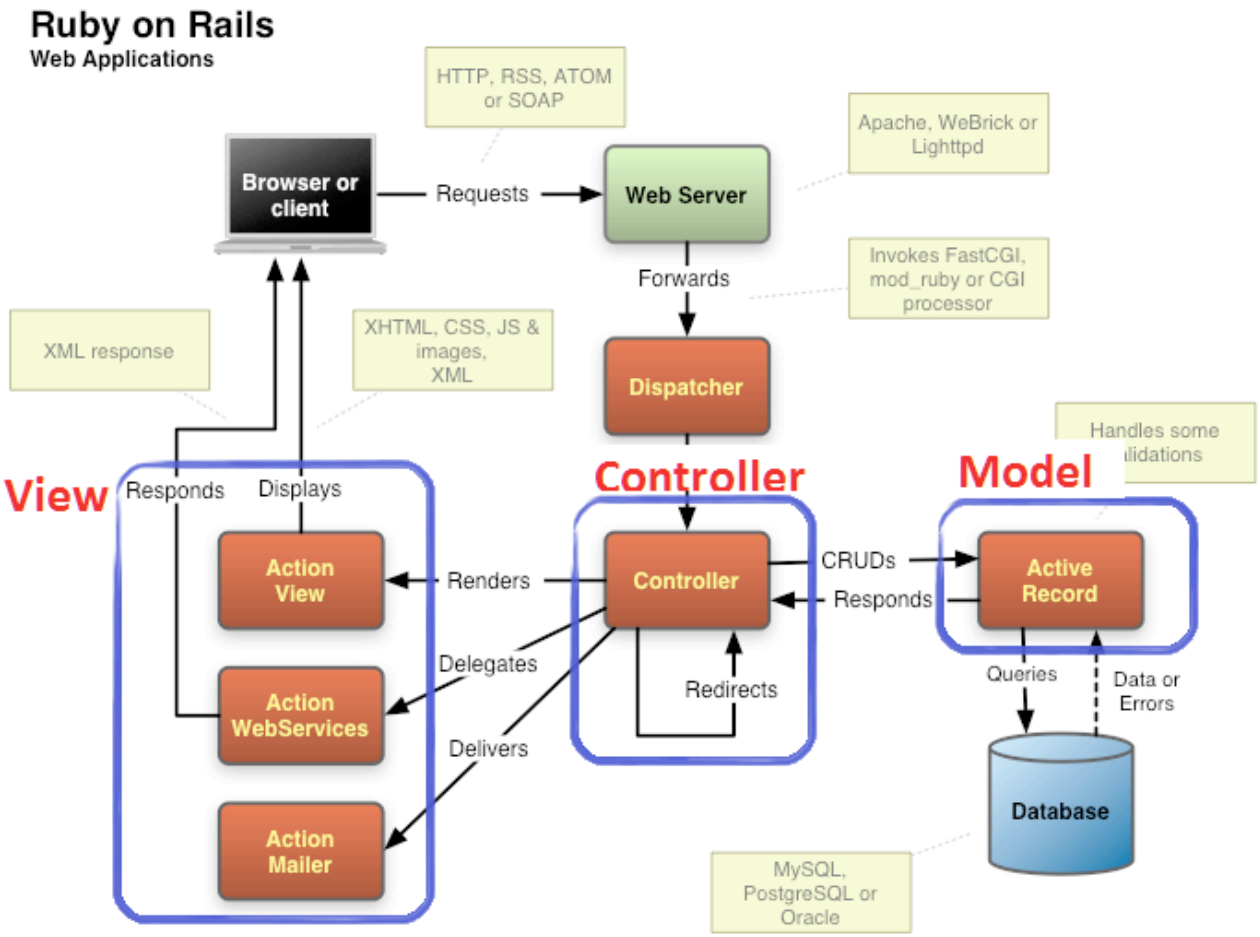
- Erstellen einer Facebook-Applikation welche mit dem Fingerprint Server interagiert und einen Fingerprint vom Facebook-Benutzer inklusive dessen Benutzerdaten anlegen kann.
- Die Zusammensetzung eines Fingerprints soll per Konfiguration geändert werden können.

Einschränkungen:

- Benutzeroberfläche nur in Englisch

5. Logische Architektur

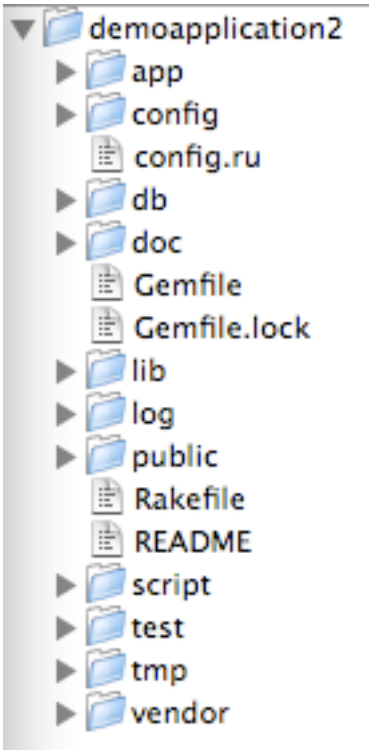
5.1. Logischer Aufbau der Applikation



Quelle: http://dedicatedwebserverhosting.co.uk/dedicated-server-hosting/tools-for-ruby-on-rails/
(Zuletzt Abgerufen am 20.12.2010)

Folgende Packages sind vorhanden: Model, View, Controller, Helper.
Sie sind im Diagram markiert – ausser das Packet „Helper“, es ist nicht eingezeichnet.

5.2. Ordnerstruktur



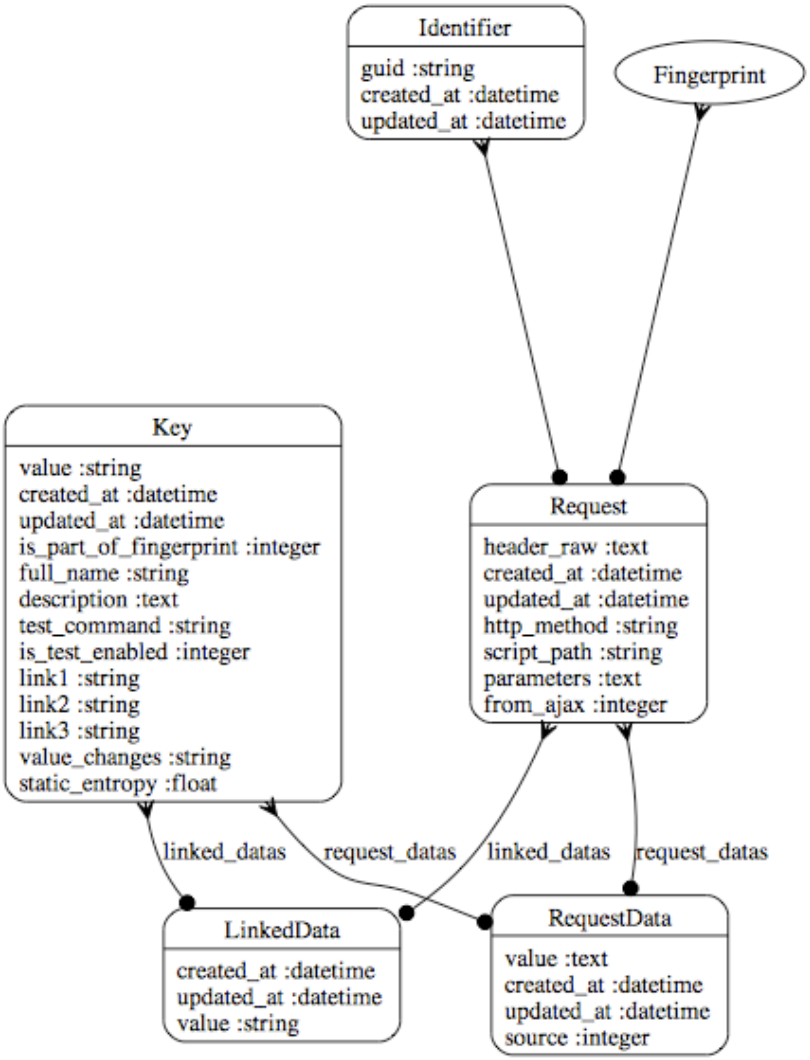
Pfad	Beschreibung
/app	MVC Klassen
/config	Datenbank, Environments, Plugins, Sprachen
/db	Datenbank Scripts, Dumps
/doc	Javadoc equivalent Rdoc-Dokumentation
/lib	U.a. selbst erstellte Scripts für die Projektautomation
/log	Logfiles
/public	Dateien die auf dem Webserver abrufbar sind.
/script	Webserver, Code Generatoren, etc.
/test	Fixtures, Functional Tests, Integration Tests, Unit tests
/tmp	Session Daten, Cookie Daten, Cache, etc.
/vendor	U.a. Plugins

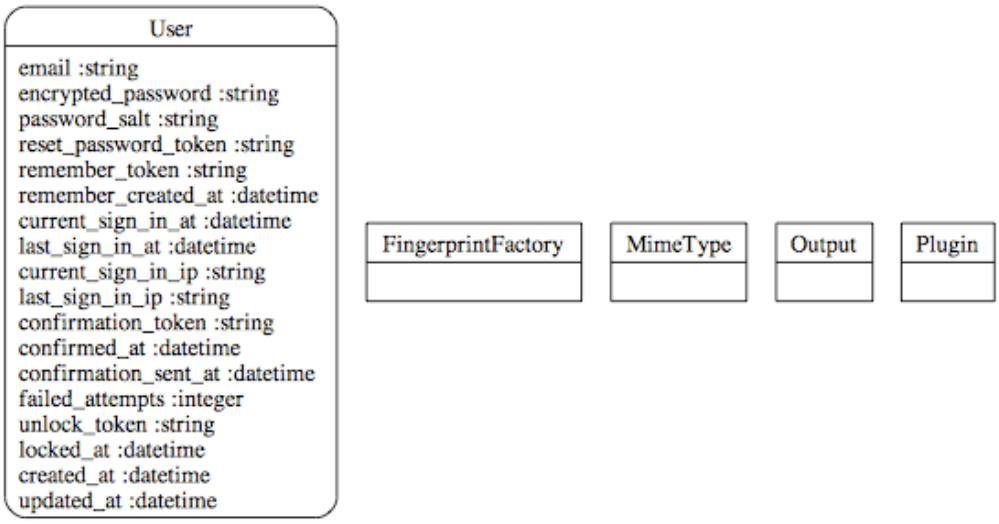
5.3. Design Packete

Per “Rdoc” (Javadoc equivalent für Ruby) generierte Dokumentation befindet sich im Projekt unter “demoapplication/doc/app/index.html”

5.3.1. Package Model

Beschreibung der Datenbankrelationen, Validierung, Model Spezifische Funktionalität.





5.3.2. Package VIEW

Besteht aus HTML/Javascript/XML/etc.-Templates die durch vom Controller bereitgestellte Daten gefüllt werden. Name von Controller-Methode korelliert üblicherweise mit Name der View Datei.

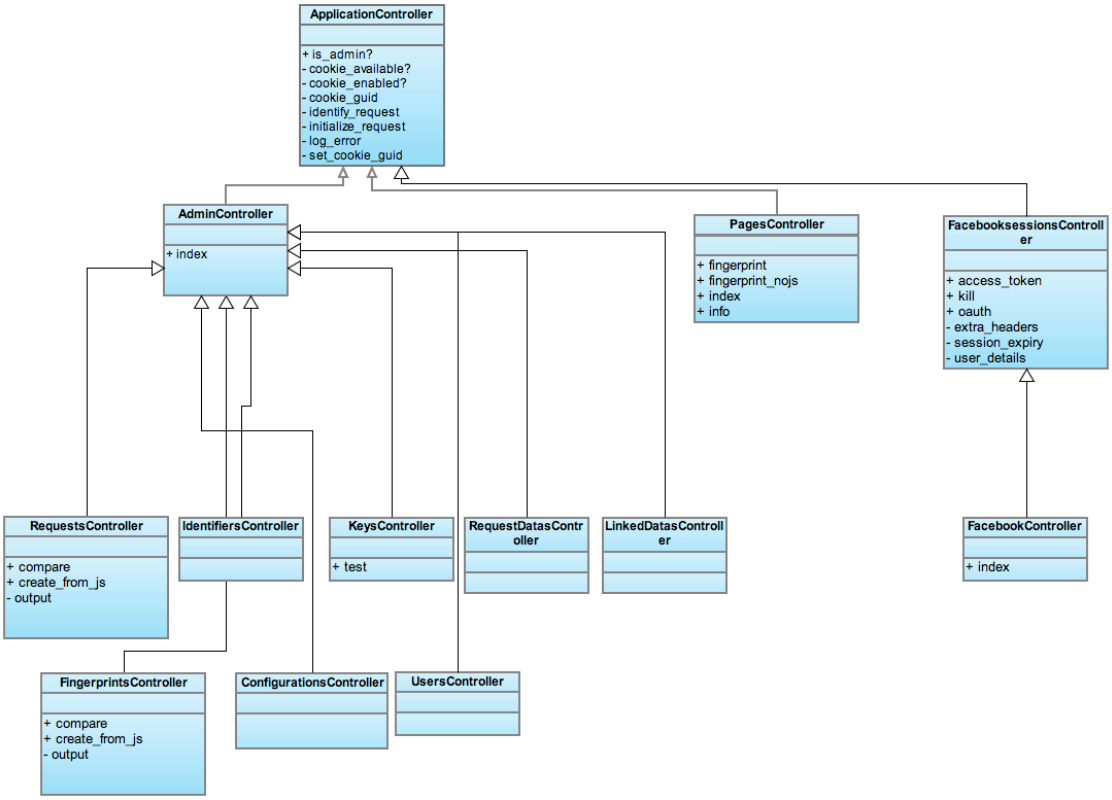
5.3.3. Package HELPER

Zum Auslagern von Code aus den Views. Methoden die im Helper definiert werden stehen automatisch in der entsprechenden View zu verfügung.

5.3.4. Package CONTROLLER

Die Logik wird hier abgelegt. Aufgrund der URL wird automatisch die entsprechende Controller Methode aufgerufen.

Beispiel: “http://localhost/people/show/1” ruft die Methode “show” des Controllers “people” mit dem Parameter “id=1” auf.



6. Datenspeicherung

Die Daten werden in einer relationalen Datenbank gespeichert. Der ORM ActiveRecord erzeugt aus den Tabellendefinitionen automatisch Objekte und lädt/speichert die Attribute sowie Relationen (1:1, 1:n, m:n).

Die Datenbanktabellen werden durch Metaprogrammierung (Skripte) erzeugt. Veränderungen an der Datenbank werden ebenfalls über Skripte durchgeführt. Sie können Schrittweise hinzugefügt und entfernt werden.

Das erlaubt das einfache, automatisierte und lückenlose Nachführen des Datenbankschemas auf verschiedenen Datenbankservern. Als Bonus wird die Entwicklung der Datenbank dokumentiert. Die Datenbank weiss selbst auf welchem Versions-Stand sie ist - dies wird in der Tabelle “schema_migrations” festgehalten.

7. Erstellen von Fingerprints

Hat der User Javascript aktiviert führt in der Fingerprint-Link auf die Seite “/pages/fingerprint” ansonsten nach “/pages/fingerprint_nojs”.

7.1. Ohne Javascript

Es werden aufgrund der Empfangenen HTTP-Header Daten ein Fingerprint erstellt und das Resultat direkt angezeigt.

7.2. Mit Javascript

In einem Zwischenschritt werden per Javascript diverse Tests ausgeführt. Diese Erkenntnisse werden per AJAX an den Server gesendet. Der Server sendet als Antwort das Resultat der Fingerprint erstellung als fertiges HTML Snippet, dass dann direkt im Browser angezeigt werden kann.

Test Dokumentation

Studienarbeit:
"IT-Security: Browser Uniqueness - Identifying Users"
Florian Hengartner
2010

1. Inhaltsverzeichnis

2. Änderungsgeschichte

Version	Kommentar	Datum
1.0	Erstellt	Mittwoch, 13. Oktober 2010
1.1	Durchführung Systemtest am Ender der Elaboration Phase	Dienstag, 9. November 2010
1.2	Durchführung Systemtest am Ender der Construction Phase	Freitag, 10. Dezember 2010
1.3	Rails Testing, Browser Testing	Sonntag, 19. Dezember 2010
1.4	Review	Dienstag, 21. Dezember 2010

3. Systemtest

3.1. Voraussetzung

- Demoserver muss gestartet sein
- Facebook Account ist vorhanden

3.2. Vorbereitungen

-

3.3. Ausführung des Systemtests

Getreu dem DRY Motto verwenden wir für die Systemtest's unsere UseCases als Testvorlage.

Zur durchführung des Systemtests werden alle UseCases mit allen Alternative Flows (soweit die möglich ist) durchlaufen.

Die UseCases befinden sich im Dokument: “06 Analyse/Anforderungsspezifikation-DATUM.pdf”

3.4. 09.11.2010 - Abschluss Elaboration

Use Case	Implementiert	Fehler / Unschönheiten	Status
UC1: Browserfingerprint erfassen und anzeigen	Ja	Erfassen funktionert. Es werden noch nicht alle angestrebten Informationen vom Browser ausgelesen. Rückmeldung der Informationen an den User ist unzureichend.	FAILED
UC2: Session erzeugen	Nein		FAILED
UC3: Session Id mit anderer Browser(-Konfiguration) öffnen und unterschiede anschauen	Nein		FAILED
UC4: Socialnetwork Applikation - Benutzerdaten sammeln	Ja	Ausgabe an Benutzer ist noch in “Rohform”.	FAILED
UC5: Erkennen von Socialnetwork Usern	Nein		FAILED

3.5. 13.12.2010 - Abschluss Construction

Use Case	Implementiert	Fehler / Unschönheiten	Status
UC1: Browserfingerprint erfassen und anzeigen	Ja	-	SUCCESS
UC2: Session erzeugen	Ja	-	SUCCESS

Use Case	Implementiert	Fehler / Unschönheiten	Status
UC3: Session Id mit anderer Browser(-Konfiguration) öffnen und unterschiede anschauen	Ja	-	SUCCESS
UC4: Socialnetwork Applikation - Benutzerdaten sammeln	Ja	-	SUCCESS
UC5: Erkennen von Socialnetwork Usern	Ja	-	SUCCESS

4. Functional & Unit-Tests

Rails bringt ein Testframework mit. Die Tests befinden sich im Verzeichniss “demoapplication/test”.

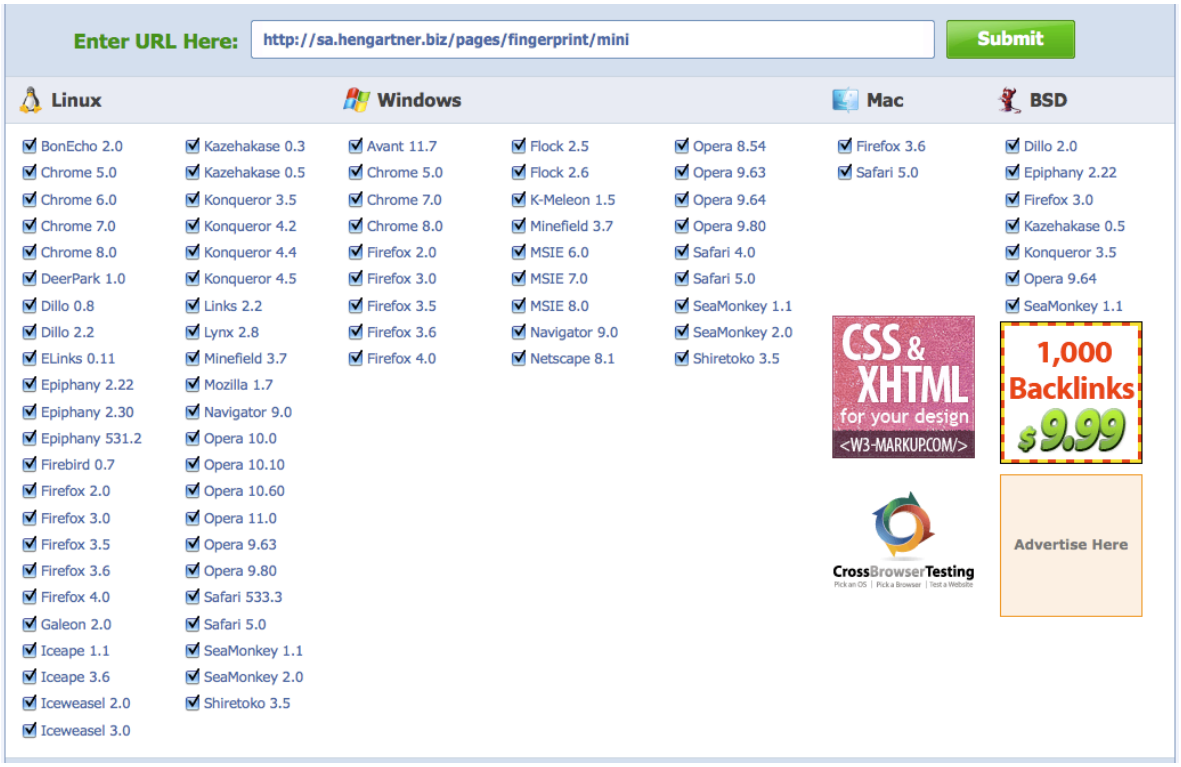
Die Tests können z.b. auf der Konsole ausgeführt werden. Dazu in das Verzeichniss “demoapplication” wechseln und folgendes Kommando ausführen:

```
rake test
```

5. Browser Testing

Das Testing des Fingerprinting Mechanismums und der Webseite hat mit diversen Browsern stattgefunden.

1. Browsershots



Die Webseite “http://browsershots.org” ermöglicht es einem Screenshots von seiner eigenen Webseite von unterschiedlichen Browsern auf verschiedenen Plattformen zu erstellen.

Dieses Tool war sehr praktisch um das Fingerprinting mit diversen Browsern zu testen.

Screenshots von einigen Testläufen sind unter “doc/08 Test/browsershots.org” abgelegt.

2. Manuelles Browser Testing

Ich habe die Webseite inkl. Facebook Applikation mit diversen Browsern getestet.

- OSX 10.5
  - Chrome 8.0.552.231
  - Opera 10.63
  - Safari 3.2.1 (5525.27.1)
  - Safari 4.0.4 (5531.21.10)
  - Firefox 3.6.12
- Windows 7
  - Chrome 8.0.552.215
  - Opera 10.63
  - Safari 534.10
  - Firefox 3.6.12
  - MSIE 8.0
  - MSIE 7.0
  - MSIE 6.0
  - MSIE 5.5
- LG Arena Mobile Phone

Installationsanleitung

Studienarbeit:
"IT-Security: Browser Uniqueness - Identifying Users"
Florian Hengartner
2010

1. Inhaltsverzeichnis

2. Änderungsgeschichte

Version	Kommentar	Datum
1.0	Erstellt	Dienstag, 21. Dezember 2010 (KW: 51)

3. Installation von Ruby on Rails

3.1. Version

Benötigte Software Versionen

- Ruby: 1.9.2
- Rails: 3.0.1

3.2. Installation auf Ubuntu

Folgende Kommandos ausführen:

```
apt-get install curl git-core build-essential zlib1g-dev libssl-dev libreadline6-dev subversion
bash < <( curl http://rvm.beginrescueend.com/releases/rvm-install-head )
```

Diese Zeile zur ".bashrc" hinzufügen

```
if [[ -s "$HOME/.rvm/scripts/rvm" ]]  ; then source "$HOME/.rvm/scripts/rvm" ; fi
```

Neues Terminal öffnen und RVM Installieren. Inklusive Workaround für kaputtes Iconv Gem[18]:

```
rvm notes
rvm package install readline
rvm package install json
rvm install --trace 1.9.2 -C --with-iconv-dir=$HOME/.rvm/usr
rvm --default ruby-1.9.2
```

Rails installieren

```
sudo gem install rails
```

SQLite installieren*

```
sudo apt-get install sqlite3 libsqlite3-dev
sudo gem install sqlite3-ruby
```

MySql installieren*

```
sudo apt-get install mysql-server libmysqlclient-dev libmysql-ruby
sudo apt-get install libsqlite3-dev build-essential
```

* eines Auswählen

[18] http://exceptionz.wordpress.com/2010/02/03/how-to-fix-the-iconv-require-error-in-ruby-1-9/ How to fix the iconv require error in Ruby 1.9, letzter Zugriff 21.12.2010

Quellen:

- http://www.web2linux.com/05/installing-rails-3-on-ubuntu-10-04-lucid-lynx/ Installing Rails3 on Ubuntu 10.04, letzter Zugriff 21.10.2010
- http://rohitarondekar.com/articles/installing-rails3-beta3-on-ubuntu-using-rvm Installing Rails 3 Beta on Ubuntu using RVM, letzter Zugriff 21.10.2010

3.3. Windows

Eine (ungetestete) Anleitung für Windows gibt es hier: http://allaboutruby.wordpress.com/2009/07/20/installing-rails-on-windows-3-years-later/

4. Kommandos

Um Kommandos auszuführen wechselt man erst in das Hauptverzeichniss der Applikation. Beispiel:

```
cd demoapplication
```

Web Server starten:

```
rails s
```

4.1. Kommandozeilen-Tools

- rails - Server starten, Code generieren, etc.
- rdoc - Javadoc equivalent
- rake - Make equivalent
- gem - Pendant zu CPAN/PEAR/APT
- bundle - Verknüpft “gem” und “rails”

5. Demoapplication einrichten

5.1. Gem’s installieren

```
cd demoapplication
bundle install
```

5.2. Datenbank Konfiguration

Die Datenbankkonfiguration ist unter “demoapplicaton/config/database.yml” abgelegt.
Aus diesem File können Benutzername / Passwort entnommen und auch angepasst werden.

Es gibt 3 Umgebungen “development”, “test” und “production”.

- development ist vorkonfiguriert auf eine Sqlite DB
- test - wird nur vom Test Framework verwendet
- production ist vorkonfiguriert auf eine Mysql Datenbank

Bitte passen Sie die Konfiguration der Datenbank für Ihre Umgebung an.
Anleitung: http://edgeguides.rubyonrails.org/getting_started.html#configuring-a-database, letzter Zugriff 21.12.2010

5.3. Datenbank einrichten

Für sqlite kann die mitgelieferte Datenbank “development.sqlite3” verwendet werden.

Für mysql müssen folgende Schritte durchführen:

```
rake db:create
rake db:migrate
```

Die Datei “initialize_mysql.sql” in die Mysql DB importieren.

5.4. Webserver einrichten

Die einfachste Variante ist es den von Rails mitgelieferten Webserver zu verwenden.
Webserver starten:

```
cd demoapplication
rails s
```

Für die Verwendung unter Apache gibt es den “Phusion Passenger”[19].

Auf http://sa.hengartner.biz verwende ich folgende Apache Konfiguration[20]:

```
NameVirtualHost sa.hengartner.biz:80
<VirtualHost sa.hengartner.biz:80>
    ServerName sa.hengartner.biz

    DocumentRoot    /path/to/demoapplication/public
    PassengerEnabled    off
    ProxyPass    / http://127.0.0.1:3000/
    ProxyPassReverse    / http://127.0.0.1:3000/

    # ... Log and stuff ...
</VirtualHost>
```

Den Passenger starte ich mit:

```
cd /path/to/demoapplication/public
sudo su www-data
```

[19] http://www.modrails.com/install.html, Installation Phusion Passenger, zuletzt Abgerufen 21.12.2010

[20] http://blog.phusion.nl/2010/09/21/phusion-passenger-running-multiple-ruby-versions/ Phusion Passenger & running multiple Ruby versions, zuletzt Abgerufen 21.12.2010

```
passenger start -a 127.0.0.1 -p 3000 -d -e production
```

6. Auswahl der Umgebung

Kommandos (rails, rake) werden Standardmässig in der Umgebung “development” ausgeführt. Hier wird beschrieben wie Sie das ändern können.

6.1. Als Parameter

Server in der Umgebung “production” starten:

```
rails s -e production
```

6.2. Als Variable

Server in der Umgebung “production” starten:

```
RAILS_ENV=production rails s
```

Rake Task “Datenbank erstellen” in production Umgebung ausführen

```
RAILS_ENV=production rake db:create
```

6.3. Apache konfiguration

```
<VirtualHost *:80>
  ..
  RackEnv production
  ..
</VirtualHost>
```

7. Exception Notifikation

Konfigurieren Sie den Empfänger für E-Mails bei Exceptions.

Anpassen im File “demoapplication/config/application.rb”:

```
# application.rb, inside the config block
config.middleware.use ::ExceptionNotifier,
    :email_prefix => "Fingerprint-Server-Errors: ",
    :sender_address => %w{mail@hengartner.biz},
    :exception_recipients => %w{mail@hengartner.biz}
```

Persönlicher Bericht

Studienarbeit:
"IT-Security: Browser Uniqueness - Identifying Users"
Florian Hengartner
2010

Die Studienarbeit war für 2 Personen geplant. Leider konnte mein Partner nicht teilnehmen. Ich hätte die Arbeit gerne im Team erarbeitet. Damit hätte ich immer einen Partner gehabt um mich über die aktuellen Probleme auszutauschen. Es wäre möglich gewesen mehr aus der Studienarbeit herauszuholen, da doppelt so viel Zeit hätte investiert werden können. Andererseits habe ich die Vorteile genossen. Es war kein Overhead für die Besprechung und Abgleich im Team nötig. Ich war immer auf dem aktuellsten Stand der Dinge. Es waren keine Einschränkungen bei der Wahl der Technologie nötig.

Die Arbeit mit dem Framework RubyOnRails war sehr produktiv. Der Bereits vorgegebene Rahmen erleichtert einem, seinen eigenen Code ebenfalls sauber zu strukturieren. Trotzdem hat die Arbeit am Admin-Backend (Benutzer-Verwaltung, CRUD) viel Zeit in Anspruch genommen. Im Nachhinein muss ich sagen, diese Zeit hätte ich auch besser Nutzen können. Als Backend würde PhpMyAdmin oder ähnliches reichen. Nichtsdestotrotz bringt das Backend einen Mehrwert indem die Darstellung Informativer ist als ein generisches Tool und die Applikation als ganzes abrundet.

Die Wahl von RubyOnRails hat sich definitiv gelohnt. Ruby, Rails, Ajax und Javascript sind Technologien die Zusammen gut funktionieren. Ich musste mich nicht mit den tücken der Technologie herumschlagen, auch keine “üble Hacks” einsetzen, sondern konnte mich voll auf die Lösung der gestellten Probleme konzentrieren.

Sozusagen als Bonus für mich konnte ich mich in diesem Projekt mit der Entwicklung einer Facebook-Applikation beschäftigen. Facebook ist heute nicht mehr wegzudenken und wegzukriegen. Als Fazit kann ich sagen, das erstellen einer Facebook-Applikation gestaltet sich einfacher als ich am Anfang dachte. Sie ist dank breiter Unterstützung in diversen Technologien möglich und es gibt im Internet genügend Anleitungen.

Es wahr immer konstruktiv und angenehm mit Herr Sprenger zusammenzuarbeiten.

Zum Abschluss kann ich sagen, dass mir die Arbeit am Projekt sehr gefallen hat.

Literaturverzeichniss

Studienarbeit:
"IT-Security: Browser Uniqueness - Identifying Users"
Florian Hengartner
2010

8. Dokumente

1. Browser Fingerprintingto detect Online Fraud, Author: Daniel Stirnimann, Compass Security AG, 12.08.2010

9. Internet / Webseiten

1. SwtWiki - Rational Unified Process , http://www.imn.htwk-leipzig.de/~weicker/pmwiki/pmwiki.php/Main/RationalUnifiedProcess, letzter Zugriff 13.10.2010

2. How Unique Is Your Web Browser?, https://panopticlick.eff.org/browser-uniqueness.pdf, letzter Zugriff 13.10.2010

3. A Primer on Information Theory and Privacy.pdf, https://www.eff.org/deeplinks/2010/01/primer-information-theory-and-privacy, letzter Zugriff 13.10.2010

4. What Information is "Personally Identifiable", http://www.eff.org/deeplinks/2009/09/what-information-personally-identifiable, letzter Zugriff 13.10.2010

5. Managers Introduction to RUP.pdf, http://www.ambysoft.com/downloads/managersIntroToRUP.pdf, letzter Zugriff 15.10.2010

6. Panopticlick - Privacy Policy.pdf, https://panopticlick.eff.org/privacy.php, letzter Zugriff 25.10.2010

7. User-Agent - Bits of Identifying Information, https://www.eff.org/deeplinks/2010/01/tracking-by-user-agent, letzter Zugriff 01.11.2010

8. Suprisal Example, http://www.umsl.edu/~fraundorfp/egsurpri.html, letzter Zugriff 2010.11.09

9. Defcon Panopticlick Presentation, http://www.defcon.org/images/defcon-18/dc-18-presentations/Eckersley/DEFCON-18-Eckersley-Panopticlick.pdf, letzter Zugriff 2010.11.09

10. Getting a list of fonts with flash and javascript, http://hasseg.org/blog/post/526/getting-a-list-of-installed-fonts-with-flash-and-javascript/, letzter Zugriff 2010.11.20

11. Facebook Platform Policies - Facebook Developers.pdf, http://developers.facebook.com/policy/, letzter Zugriff 2010.12.13

12. Terms of service - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Terms_of_service, letzter Zugriff 2010.12.13

13. Yahoo! Terms of Service, http://info.yahoo.com/legal/us/yahoo/utos/utos-173.html, letzter Zugriff 2010.12.13

Glossar

Studienarbeit:
"IT-Security: Browser Uniqueness - Identifying Users"
Florian Hengartner
2010

Begriff	Erklärung
XmlHttpRequest (XHR)	Eine API in Webbrowsern die in Scripting Sprachen wie Javascript vorhanden ist. Damit können HTTP/S Requests an Webserver gesendet und empfangen werden.
GET	HTTP Request Typ - GET Anfragen sollen keine Nebenwirkungen haben. (keine Daten modifizieren)
POST	HTTP Request Typ - Übermittelt Daten zur verarbeitung, darf Nebenwirkungen haben.
MimeType	Identifiziert Dateiformate im Internet
Plugin	Erweitert eine Software um Funktionalität.
Ruby	Programmiersprache
Ruby on Rails (Rails)	Framework zur Erstellung von Web-Applikationen
AJAX	Asynchronous Javascript And Xml Eine Technik zum asynchronen Austausch von Daten zwischen Webseite und Webserver.