Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelorarbeit

Entwicklung eines Visualisierungswerkzeuges zur Demonstration datenschutzfreundlicher Dokumentspeicherdienste

vorgelegt von

David Kirchhausen Monteiro

geb. am 24. Januar 1994 in Hildesheim

Matrikelnummer 6530927

Studiengang Software-System-Entwicklung

eingereicht am 19. Juni 2018

Betreuer: Maximilian Blochberger, M. Sc.

Erstgutachter: Prof. Dr.-Ing. Hannes Federrath

Zweitgutachter: Tilmann Stehle, M. Sc.

Aufgabenstellung

Im Zuge dieser Bachelorarbeit soll ein einfacher Dokumentenspeicher entwickelt werden, welcher möglichst viele Nutzerdaten erfasst und speichert. Die erfassten Daten sollen anschaulich grafisch dargestellt werden können. Weiter sollen verschiedene Szenarien entwickelt werden, welche aufzeigen wie eine mögliche Benutzung des Services mit und ohne der Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten aussieht. Anhand der Szenarien soll eine grafische Auswertung Unterschiede zwischen anonymisierten Daten und nicht anonymisierten Daten visuell sichtbar machen und die Unterschiede somit leicht zugänglich sein.

Zusammenfassung

1. Dokumentenspeicherdienste Vorteile (Problemstellung erläutern)
2. Mögliche Datenschutz unfreundliche Aspekte von gängigen Anbietern (Problemstellung erläutern)
3. Entwicklung des Dokumentenspeichers und der Visualisierung zur deutlich Veranschaulichung von Potentiellen Unterschieden zwischen der Verwendung von Datenschutz freundlichen Methoden zum Anonymisieren oder nicht. (Bearbeitung der Problemstellung)
    a) Implementation des Dokumentenspeichers
    b) Implementation der API zur Datenübergabe
    c) Implementation des Visualisierungswerkzeug
    d) Darstellung der Szenarien zur Benutzung des Visualisierungswerkzeug

Inhaltsverzeichnis

1 Einleitung **5**

2 Grundlagen **6**
2.1 Terminologie 6
2.2 Set A / Set B 7

3 Hauptteil **8**
3.1 Implementation des Dokumentenspeichers 8
3.2 Darstellung: IP Tree Map und IP Google Map 9
3.3 Darstellung: Headerfingerprinting 10
3.4 Darstelung: Time Line 10

4 Schluss **11**
4.1 Zusammenfassung der Ergebnisse 11
4.2 kritische Bewertung des Ergebnisse 11
4.3 neue Problemstellungen, Möglichkeiten zur Weiterführung der Arbeit 11

1 Einleitung

Dokumentenspeicherdienste sind nützliche Alltagsgegenstände, welche für private sowie kommerzielle Nutzer meist unverzichtbar sind. Sie bieten nicht nur den Speicherplatz für wichtige Dateien der Nutzer sondern stellen auch die Sicherheit der Dateien sicher und machen sie global jeder zeit verfügbar. Durch die große Datensammlung dieser Dienstleister machen sie sich nicht nur selbst zu lukrativen Zielen von gezielten Angriffen (Yahoo, UBER etc.), jedoch auch die Dienstleister selber können die Daten auswerten und meitere Meta-Daten über die Nutzer sammeln und weiter verwenden. Vor allem private Nutzer sind meist gar nicht über die Risiken und das Missbrauchspotenziale aufgeklärt, welche die Verwendung solcher Dienstleistungen mit sich bringen. Methoden zur Verschlüsselung oder des Anonymisieren von Daten sind Benutzern meist nicht bekannt, werden von den Dienstanbietern nicht angeboten oder sind schwer umzusetzen da es einen meist erheblichen Aufwand für die Benutzer bedeutet und Kompetenzen erfordert welche diese Benutzer nicht besitzen. Um genau die Risiken und Missbrauchspotenziale aufzuzeigen wird im Zuge dieser Arbeit ein Dokumentenspeicherdienst entwickelt, welcher prinzipiell alle Meta-Daten der Nutzer sammelt und diese in einer visuellen Darstellung zusammen fasst. Zur Implementation des Dokumentenspeichers wird dabei das Microsoft ASP.NET Core Framework verwendet. Das Framework wird benutzt um die Webbenutzeroberfläche sowie der Web-APIs des Dokumentenspeichers zu realisieren. Dazu wird das Javascript Framework Data-Driven Documents, i.d.R. d3.js genannt, zur Visualisierung der Daten verwendet. Der Dokumentenspeicher soll vor allem den Unterschied zwischen der Verwendung von Methoden zur Verschlüsselung oder des Anonymisieren von Daten visualisieren und verwaltet dazu zwei verschiedene Datensätze, wobei eine Datenmenge ohne, und eine Datenmenge mit der Verwendung von Methoden zur Verschlüsselung oder des Anonymisieren von Daten erzeugt wird. Der entstehende Unterschied der gesammelten Meta-Daten durch die verschiedenen Methoden führt dann zu einer Veränderung in der Visualisierung was dan den Effekt und Nutzen der Methoden deutlich sichtbar macht.

2 Grundlagen

2.1 Terminologie

1. HTTP-Header
   a) Teil des Hypertext Transfer Protocol (HTTP)
   b) Headerblock und Headerfeld nach RFC 2616(https://tools.ietf.org/html/rfc2616)
2. Geo lookup
   a) eine Methodik zur Bestimmung des geografischen Standort einer IP-Adresse
   b) keine eigenen Implementation der Standortbestimmung, sondern Verwendung eines öffentlich zugänglichen Service, aus Zeit und komplexitätsgründen
   c) Patent US7752210B2
3. Header Fingerprint
   a) eine Methodik zur Identifikation eines Benutzers anhand der von ihm Verwendeten HTTP Header
   b) Aggregation über allen Http-Header oder einem ausgewählten Set an Headerfeldern
   c) erzeugt Fingerprint wird gehasht und gespeichert
   d) bei übereinstimmenden Fingerprints wird angenommen das diese vom gleichen Benutzer erzeugt wurden

2.2 Set A / Set B

Die angeführten DbSets FileEntryItemsA und FileEntryItemsB verfügen jeweils über Daten welche zu visualisieren sind. Dabei wird das DbSet FileEntryItemsA weiterhin als die ungeschützte Datenmenge beschrieben, da diese Datenbank nur mit Daten befüllt werden soll, bei welchen keine datenschutzfreundlichen Methoden zum anonymisieren verwendet wurden. Das DbSet FileEntryItemsB wird weiterhin als geschützte Datenmenge beschrieben, da ausschließlich Daten , welche mit Methoden zum anonymisieren hochgeladen wurden, verwendet werden sollen. Dabei sollte die geschützte und ungeschützte Datenmenge in den Zugrunde liegenden Datenmenge identisch sein und lediglich die Verwendung von Methoden zum anonymisieren die Datensätze unterscheiden, sodass die beiden Datenmenge miteinander in Hinblick auf den Effekt der Verwendung von Methoden zum anonymisieren untersucht werden können. So wird im jedem beschriebenen Szenario zur Benutzung des Dokumentenspeichers angenommen das ein Benutzer eine Datei einmal ohne die Verwendung von Methoden zum anonymisieren in die ungeschützte Datenmenge via HTTP hochlädt und einmal mit der Verwendung von Methoden zum anonymisieren in die geschützte Datenmenge via HTTP hochlädt.

Das führt dazu das bei der Auswertung der Datei durch die Verwendung oder nicht Verwendung von Methoden zum anonymisieren, die erfassten Meta-Daten sich nur durch die jeweilige Art der Methoden zum anonymisieren unterscheiden und die anschließenden Visualisierung nur diesen Unterschied darstellt und keine anderen Faktoren.

—

Das Visualisierungswerkzeug besitzt zwei verschiedene Datenbanken welche Set A und Set B genannt werden. Jedes Datenbank Set verfügt über eine eigene API und funktioniert identisch. Die angestrebte Benutzung sieht den Vergleich von Set A und Set B mit den gleichen Datensatz vor wobei die nicht Anwendung von Datenschutz freundlichen Methoden zum Anonymisieren bei Set A und die Anwendung von Datenschutz freundlichen Methoden zum Anonymisieren bei Set B. Set A ist somit die Basis und zeigt auf was ein Dokumentenspeicherdienst an Nutzerdaten sammeln kann und Set B kann im direkten Vergleich zeigen wie Datenschutz freundlichen Methoden zum Anonymisieren diese Daten verfälschen.

3 Hauptteil

3.1 Implementation des Dokumentenspeichers

Die Implementation des Dokumentenspeichers ist mit dem ASP.NET Core Framework von Microsoft umgesetzt worden.
„ASP.NET Core ist ein plattformübergreifendes, leistungsstarkes Open-Source-Framework zum Erstellen moderner, cloudbasierter mit dem Internet verbundener Anwendungen.“
Der Dokumentenspeicher ist mittels einer Model-View-Controller Architektur realisiert worden. Die Model-View-Controller Architektur definiert, Modelle als Klassen welche die zugrundeliegende Problemstruktur wiedergeben. Sie modellieren die Objekte welche keine Informationen über ihre Verwendung besitzen. Views sind die Klassen welche die Darstellung der Modelle implementieren und den Output einer Systems darstellen. Controller sind die Kontroll-Klassen welche Views steuern und Input in das System verarbeiten.

Die Modelle welche im Dokumentenspeicher verwendet werden sind

1. FileEntryItem
2. FileEntryItemA
3. FileEntryItemB

Wobei FileEntryItemA und FileEntryItemB von FileEntryItem erben und alle relevanten properties in FileEntryItem definiert sind. Mit Hilfe des Entity Framework Core von Microsoft werden aus diesen Modell-Klassen ein Datenbankschema erzeugt welches dem Dokumentenspeicher zugrunde liegt. Das Datenbankschema ist somit an die Modell-Klassen gekoppelt und führt dazu das Änderungen an den Modell-Klassen auch direkt Änderungen des Datenbankschemas erfordern, welche vom Entity Framework Core einfach erzeugt werden können. Das erzeugte Datenbankschema für FileEntryItemA ist in Abbildung 2.1 zu erkennen, das Datenbankschema für FileEntryItemB ist analog.

```
1  [ID]                 INT            IDENTITY (1, 1) NOT NULL,
2      [City]               NVARCHAR (MAX) NULL,
3      [Country]            NVARCHAR (MAX) NULL,
4      [DateTime]           DATETIME2 (7)  NOT NULL,
5      [Filename]           NVARCHAR (MAX) NULL,
6      [HeaderFingerprint] NVARCHAR (MAX) NULL,
7      [Headers]            NVARCHAR (MAX) NULL,
8      [IPAddress]          NVARCHAR (MAX) NULL,
9      [Isp]                NVARCHAR (MAX) NULL,
10     [Lat]                REAL           NOT NULL,
11     [Lon]                REAL           NOT NULL,
12     [RegionName]         NVARCHAR (MAX) NULL,
13     [Size]               BIGINT         NOT NULL,
14     [Filepath]           NVARCHAR (MAX) NULL,
15     CONSTRAINT [PK_FileEntryA] PRIMARY KEY CLUSTERED ([ID] ASC)
```

Listing 3.1: Datenbank Schema für FileEntryItemA

Die einzelnen Properties werden für bestimme Informationen verwendet.

1. ID: Datenbank Index
2. City: Stadt aus welches die Datei hochgeladen wurde
3. Country: Land aus welches die Datei hochgeladen wurde
4. DateTime: Als Zeitstempel für das Hochladen der Datei
5. Filename: Der Dateiname
6. Headers: Ein String bestehend aus den Headern der Datei
7. HeaderFingerprint: Ein zusammenschluss aus ausgewähten Headern um eine möglichst eindeutige Signatur zu erzeugen
8. IPAddress: Die IP-Adresse von der die Datei hochgeladen wurde
9. Isp: Internetanbierter des Benutzers der die Datei hochgeladen hat
10. Lat: Breitengrad welcher mit den bekannten IP-Adresse assoziiert wird
11. Lon: Längengrad welcher mit den bekannten IP-Adresse assoziiert wird
12. RegionName: Region (Bundesland) aus welches die Datei hochgeladen wurde
13. Size: Die Dateigröße in Byte
14. Filepath: Der Pfad zur gespeicherten temporären Datei

Für eine hoch geladene Datei werden diese 14 Informationen gespeichert und verwaltet.

Der UploadController, welche die API's implementiert besitzt eine Reihe von

3.2 Darstellung: IP Tree Map und IP Google Map

Die Darstellung der gesammelten Daten über IP-Adressen Gruppierte Mengen.

Erzeugen des Gruppierungen aus der gegebenen Datenmenge. Nutzung des d3.js zur Darstellung der Tree Map. Dabei wird jeder IP-Gruppierung eine andere Farbe zugeordnet. Die Farbliche Visuelle Darstellung macht zusammengehörige Dateien nach IP-Adresse direkt sichtbar. Mit Hilfe der IP-Adressen und eines Geolookup kann ein ungefährer Standpunkt (Lat/Lon) der IP-Adresse gewonnen werden. Anhand dieses Standpunkts kann auf der Google Map der Standpunkt von wo eine Datei hochgeladen wurde aufgezeigt werden.

Bei Verwendung von Methoden zur Anonymisieren der IP-Adresse wie z.B. das verwenden eines Proxys oder der Verwendung des TOR-Netzwerks, werden die IP-Adressen-Gruppierungen verzerrt, durch verzerrte Gruppen können Dateien eines Benutzer nicht mehr zu 100% auf diesem Benutzer gemappt werden.

1. Proxy -> IP-Adresse wird maskiert
    a) Falls der Proxy wird nur durch einen Benutzer benutzt -> keine Gruppenverzerrung
    b) Proxy wird von mehreren Benutzern benutzt -> Gruppenverzerrung

c) Benutzer wechselt Proxy mehrfach -> Gruppierungen werden in der Gesamtheit verzerrt -> pro Benutzer mehre Gruppierungen

2. IP-Gruppierung über die Endknoten des Tor-Netzwerks

   a) Benutzer die den gleichen Endknoten benutzen werden gruppiert -> Gruppenverzerrung

   b) Durch automatischen wechsle der Endknoten -> Pro Benutzer zwangsläufig mehrere Gruppen -> Gruppenverzerrung

3.3 Darstellung: Headerfingerprinting

Der Headerfingerprint welcher beim hochladen der Datei erzeugt wird, wird nun verwendet um die Dateien jeweils

3.4 Darstelung: Time Line

4 Schluss

4.1 Zusammenfassung der Ergebnisse

4.2 kritische Bewertung des Ergebnisse

4.3 neue Problemstellungen, Möglichkeiten zur Weiterführung der Arbeit