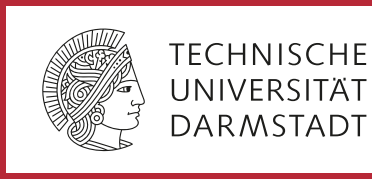
Implementierung in Forensik und Mediensicherheit: Darknet Crawler



Martin Pretz, Xian Chen
3. September 2023

Inhaltsverzeichnis

1 Einleitung

In diesem Prakiktum geht es um die Implementierung eines Darknet ”Crawlers”, das Informationen über das Forum ”Germania” sammeln soll. Dabei handelt es sich um eine sogenannte .onion Website, bei dem anonymisierte Nutzer unzensierte Meinungen äußern, ohne auf eine IP-Addresse zurückverfolgt werden zu können. In dem Forum können auch Handel betrieben werden, jedoch muss man registriert sein, um auf diese zugreifen zu können. Als dritte Partie ohne Account hat man nur beschränkten Zugriff auf den Inhalt des Forums. Ziel dieses Praktikums ist es so viele Informationen als dritte Partie automatisiert sammeln zu können mittels eines sogenannten ”Crawlers”.

Ein Crawler ist Bot, dass ein Netzwerk wie eine Spinne in einem Netz durchsucht und dabei gezielt Informationen sucht und speichert. Der Crawler der im Rahmen diese Praktikums implementiert wurde, durchforstet das Forum und besucht automatisiert weitere Webseiten, welche relevante Informationen beinhalten können. Dabei war der Fokus auf die relevantesten Informationen, wie die Forums Beiträge, die zugänglich waren ohne Authentifizierung und die Benutzer.

Das Forum liegt außerdem im ”Darknet”. Dies ist ein Netzwerk, dass nicht frei zugänglich ist vom allbekanntem Internet. Der Zugang benötigt eine Erweiterung TOR. TOR steht hierbei für ”The Onion Router”. Der Name entsteht aus der Verbildlichung des Netzwerk Zugangs, dass einer Zwiebel ähnelt. Um Webseiten im Darknet zu besuchen wird eine Route zur Zielseite gebaut, dass die eigene IP-Adresse hinter mehreren Schichten von Schutzmaßnahmen und anderen IP-Adressen verstecken soll, wie bei einer Zwiebel mit mehreren Schichten, wobei die eigene IP-Adresse das innere der Zwiebel symbolisiert.

Das heißt letztendlich, dass die Implementierung des Crawler Bots Zugang auf das Darknet verschaffen muss, damit dieser Zugriff auf die Webseite hat, um Informationen des Germania Forums zu sammeln und zu speichern.

2 Methodik & Implementierung

In diesem Bereich werden die Methodiken für die Implementierung des Crawlers und die Teile der Implementierung erläutert.

2.1 Methodik - Technische Details

In dieser Sektion wird die Umsetzung der Projektbeschreibung um die technischen Details erläutert. Die ausgewählte Sprache hierfür ist Python. Um eine Verbindung zum Darknet zu erstellen wurde die `requests` Bibliothek von Python verwendet. Im Hintergrund sollte bereits das TOR Netzwerk laufen, damit wir Zugang auf das Netzwerk haben setzen wir die Proxies.

Damit die Crawler die HTML-Seiten verstehen, benutzen wir weiterhin die `BeautifulSoup` und `lxml` Bibliotheken und filtern jeweils die Informationen raus, die sich auf den Webseiten in den HTML-Elementen befinden. Dabei findet man auch Links zu Seiten, die ebenfalls relevant sind und geben diese ebenfalls dem Crawler weiter.

Die Informationen werden jeweils strukturiert in Pythonn Dictionaries gespeichert und am Ende an eine SQL Datenbank weitergegeben. Die Datenbank selber wird beim Installationsprozesses direkt erstellt und die Tabellen nach erste Ausführung des Crawlers.

2.2 Implementierung

In der nächsten Sektion werden die Implementationen der einzelnen Crawlers erklärt und die Strukturierung der gesammelten Daten. Daraufhin, wird die Tabellen-Struktur der SQL Datenbank beschrieben und letztendlich die benutzten Tools für die Visualisierung.

2.2.1 Funktionsweise der Crawler

Um gezielt Daten zu sammeln, muss die Zielseite analysiert werden, damit die relevanten HTML-Elemente identifizert werden können. Nach der Identifikation der relevanten Elemente, kann man den Crawler gezielt implementieren, um Daten zu sammeln. Bei der Kollektion von Informationen auf mehreren Webseiten, muss der Crawler beachten, dass entweder die Struktur der Webseite ähnlich ist, wie bei der analysierten Seite oder es müssen weitere Methoden implementiert werden, die anders strukturierte HTML-Seiten crawlen können.

Forum Crawler Dieser Crawler fokussiert sich hauptsächlich auf die existierenden Foren, Sub-Foren und einzelnen Beiträge. Dabei werden Sub-Foren von Foren entsprechend markiert und Beiträge entsprechend untergeordnet. Die extrahierte Information der Foren wird in der Python Dictionary wie folgt dargestellt:

```
forum =
    {
        "element_type": "forum",
        "crawling_date": crawling_date,
        "title": forum_description,
        "link": forum_link,
        "topics_count": topics_count,
        "posts_count": posts_count,
        "last_post_time": last_post_time,
        "last_post_author": last_post_author
    }
```

- **element_type**: Typ des Eintrags
- **crawling_date**: Datum und Uhrzeit des Zeitpunkts der Sammlung der Information
- **title**: Titel des Forums
- **link**: Link zum Forum
- **topics_count**: Anzahl der Titel in dem Forum
- **posts_count**: Anzahl der Beiträge in dem Forum
- **last_post_time**: Datum des letzten Beitrags
- **last_post_author**: Author des letzten Beitrags

Sub-Foren haben die gleiche Struktur mit dem Unterschied des Element Typs. Sub-Foren sind Foren untergeordnet und haben in der Datenbank einen entsprechenden Indikator.

```
subforum =
    {
    "element_type": "subforum",
    "crawling_date": crawling_date,
    "title": subforum_name,
    "link": subforum_link,
    "topics_count": topics_count,
    "posts_count": posts_count,
    "last_post_time": last_post_time,
    "last_post_author": last_post_author
    }
```

- **element_type**: Typ des Eintrags
- **crawling_date**: Datum und Uhrzeit des Zeitpunkts der Sammlung der Information
- **title**: Titel des Sub-Forums
- **link**: Link zum Sub-Forum
- **topics_count**: Anzahl der Titel in dem Sub-Forum
- **posts_count**: Anzahl der Beiträge in dem Sub-Forum
- **last_post_time**: Datum des letzten Beitrags
- **last_post_author**: Author des letzten Beitrags

Letztendlich gehört jeder Beitrag entweder zu einem Forum oder Sub-Forum. Dies wird in der Datenbank auch entsprechend unterschieden. Die Struktur ist wieder sehr ähnlich zu den Foren Strukturen, trägt jedoch den Unterschied, dass es einen direkten Autor zu dem Beitrag gibt und Antworten gezählt werden.

```
post =
    {
        "element_type": "post",
        "crawling_date": crawling_date,
        "title": post_title,
        "link": post_link,
        "author": author,
        "replies_count": replies_count.replace(",", ""),
        "views_count": views_count.replace(",", ""),
        "last_post_time": last_post_time,
        "last_post_author": last_post_author
    }
```

- **element_type**: Typ des Eintrags
- **crawling_date**: Datum und Uhrzeit des Zeitpunkts der Sammlung der Information
- **title**: Titel des Beitrags
- **link**: Link zum Beitrag
- **author**: Author des Beitrags
- **replies_count**: Anzahl der Antworten auf das Beitrag
- **views_count**: Anzahl der Leute, die den Beitrag gesehen haben
- **last_post_time**: Datum des letzten Beitrags
- **last_post_author**: Author des letzten Beitrags

User Crawler Der User Crawler sieht sich die Benutzerliste des Germania Forums an und iteriert durch jede Benutzerlisten Seite. Dabei werden lediglich die Informationen die bereits auf der Nutzerliste präsentiert werden gesammelt und gespeichert. Zu diesen Informationen gehören beispielsweise Name, Titel, Community Punkte und das Datum der Registrierung. Dargestellt in einer Python Dictionary sieht diese wie folgt aus:

```
{
    'name': name,
    'title': title,
    'link': profile_link,
    'number_of_posts': int(num_posts),
    'points': int(points),
    'registration_date': reg_date,
    'crawled_datetime': date
}
```

- **name**: Nutzername
- **title**: Titel des Benutzers
- **link**: Link zum Profil des Nutzers
- **number_of_posts**: Anzahl der geposteten Beiträge
- **points**: Community Punkte
- **registration_date**: Datum der Registrierung
- **crawled_datetime**: Datum und Uhrzeit des Zeitpunkts der Sammlung der Information

In der Datenbank gibt es für die generellen Benutzerdaten eine Tabelle.

Detailed User Crawler Der Detailed User Crawler sammelt mindestens die Informationen, die auch vom normalen User Crawler gesammelt werden, zuzüglich dem Link zum Profil des Nutzers. Dieser Link wird nämlich aufgerufen vom Crawler selbst, sodass zusätzliche Informationen auf dem Profil auch in die Datenbank gespeichert werden.

Dabei werden nur die Informationen gesammelt, die auch auf dem Profil existieren. Im Forum gibt es nämlich ein Handelsystem, wobei Kunden den Händlern Feedback geben können. Dies impliziert, dass nicht alle Benutzer Händler Feedback auf deren Profil präsentiert haben, wenn diese nichts anbieten. Somit muss der Crawler darauf achten, dass bei manchen Profilen bestimmte HTML-Elemente nicht existieren und manchen Profilen schon.

Die gesammelten Daten werden in einem Python Dictionary wie folgt definiert:

```
{
    'name': name,
    'title': title,
    'number_of_posts': num_posts,
    'points': points,
    'registration_date': reg_date,
    'badge': badge,
    'trade_activity': trade_activity,
    'feedback_statistic': feedback_stats,
    'feedback_reviews': feedback_reviews,
    'fingerprint': fingerprint,
    'public_key': pk,
    'crawled_datetime': datetime
}
```

- **name**: Nutzername
- **title**: Titel des Benutzers
- **number_of_posts**: Anzahl der geposteten Beiträge
- **points**: Community Punkte

- **registration_date**: Datum der Registrierung
- **badge**: Auszeichnung des Users
- **trade_activity**: Handelsaktivitäten/Statistiken über Kundenfeedback (Positive, Neutrale, Negative Bewertungen) als Dictionary
- **feedback_statistic**: Die Bewertung des Produktes, der Verpackung, Versand, Kundensupport im Detail als Dictionary
- **feedback_reviews**: Kunden-Bewertungen, eine Liste von Dictionaries
- **fingerprint**: Fingerabdruck des Benutzers
- **public_key**: Der öffentliche PGP Schlüssel des Benutzers
- **crawled_datetime**: Datum und Uhrzeit des Zeitpunkts der Sammlung der Information

In der Datenbank werden die Handelsaktivitäten und Statistiken, die hier als Python Dictionary gespeichert werden, in mehrere Zeilen unterteilt, und man findest diese in der gleichen Tabelle wie die Benutzer. Anders werden Kunden-Bewertungen, die eine Liste an Dictionaries sind, in einer separaten Tabelle abgespeichert.

Dieser Crawler benötigt am meisten Interaktion. Die vorherigen Crawler brauchen nur einmal einen valide Session Cookie, jedoch wird diese bei dem Detailed User Crawler immer wieder erneuert auf Grund von Schutzmaßnahmen der Webseite selbst. Erfahrungsgemäß wird der Cookie bei jedem ersten Profilaufruf auf einer Benutzerseite erneuert und zusätzlich bei jedem zehnten Aufruf eines Profils. Bei 25 Seiten und 51 Benutzer pro Seite kommt man somit aufgerundet auf 1275 Benutzer (beachte letzte Benutzerseite hat keine Auflistung von 51 Benutzern). Somit sind im Worst-Case (wenn neue Benutzer beitreten) mindestens 150 Interaktionen gefragt. Weshalb, es bei diesem Crawler die Option gibt, mittendrin beim crawlen bei jedem neuen detektiertem Captcha, den Fortschritt zu speichern und nicht weiter zu crawlen. Damit man beim nächsten Mal nicht wieder von vorne anfangen muss, wird bei der Ausführung des Programmes gefragt, auf welche Benutzerseite angefangen werden möchte. Somit ist man nicht beschränkt auf einen langen Crawl und man kann trotz Captcha weiterhin die Details jedes Benutzer in der Benutzerliste crawlen.

Bei einem detektiertem Captcha werden drei Optionen angeboten:

1. '`continue`': Diese Option gibt dem Crawler die Information, dass die Seite nochmal aufgerufen werden soll, mit den Cookies die bereits im Crawler existieren. Dies impliziert, dass man den gleichen Cookie wiederverwenden kann, wenn beispielsweise der Cookie wieder valide ist nach Lösen eines Captchas.
2. '`cookie:[BENUTZER_EINGABE]`': Hierbei handelt es sich um die Eingabe eines neuen Cookies. Will man also einen neuen Cookie verwenden, so gibt es diese Option. Bei der Eingabe sollen keine Whitespaces sein.
3. '`save`': Letztendlich kann man den Crawl auch abbrechen und den jetzigen Fortschritt speichern.

Voraussetzung für die Funktionstüchtigkeit der Optionen ist die korrekte Eingabe der Inputs.

2.2.2 Datenbank MySQL

Die Datenbank die wir nutzen ist eine SQL Datenbank und wird anfangs einmal installiert und nach erste Ausführung des Crawlers werden die Tabellen erstellt. Insgesamt werden sechs Tabellen erstellt, basierend auf die Python Dictionary Daten, die in der Sektion vorher vorgestellt wurden.

Die Foren haben dabei jeweils eine eigene `forum_id`, die mit jedem neuen Eintrag automatisch inkrementiert werden. Die Sub-Foren haben sowohl eine `subforum_id` als auch eine `forum_id`. Die `forum_id` soll dabei die Zugehörigkeit des Sub-Forums darstellen. Heißt also, dass ein Eintrag mehrmals die gleiche `forum_id` haben kann, jedoch hat jeder Eintrag eine einzigartige `subforum_id`. Zuletzt haben die Beiträge, sowohl `forum_id`, `subforum_id` als auch `post_id`. Gehört der Beitrag zu einem Sub-Forum, so findet man die ID des Sub-Forums in dem Eintrag wieder, andernfalls gehört der Beitrag zu einem Forum, dann ist die `forum_id` ausgefüllt. Heißt also, diese beiden Einträge sind mutually-exclusive ausgefüllt. Diese Struktur hilft der Zuordnung des Beitrag in einer der Foren/Sub-Foren. Die Auswahl dieser Struktur bildet somit eine klare Übersicht, über die Verhältnisse zwischen Foren, Sub-Foren und Beiträge.

Weiterhin, haben die Benutzer des Forums auch drei Tabellen. Wie vorher angedeutet, sind diese Tabellen einmal für die generellen Benutzer Daten, die auf den Benutzerlisten vorzufinden sind, die detaillierten Benutzerdaten und im Zusammenhang mit den detaillierten Benutzerdaten gibt es eine separate Tabelle für die Kunden-Bewertungen. Diese Struktur wurde entschieden auf Grund der Übersichtlichkeit der Daten. Der detaillierte Crawler enthält mehr Informationen, die sonst leer stehen würden, würde

man auch Informationen der generellen Crawls in die gleiche Tabelle speichern. Des Weiteren, ist nicht jeder Benutzer ein Händler, somit haben nicht alle Benutzer Kunden-Bewertungen auf ihrem Profil. Diese würden somit auch leer stehen würde man alle Werte in einer Tabelle speichern.
Hierbei haben beide Benutzer Tabellen eine eigene `user_id`, die auch jeweils einzigartig sind in den entsprechenden Tabellen. Die Tabelle für die Kunden-Bewertungen hat sowohl eine `feedback_id` als auch eine `user_id`, diese bezieht sich jedoch nur auf die ID in der detaillierten Benutzer Tabelle.

2.2.3 Visualisierung mit Hilfe von grafana

Grafana ist eine Open-Source Software, dass einen bei der Visualisierung von Daten unterstützt. Man kann Dashboards mit Hilfe von Anfrangen an beispielsweise einer SQL Datenbank die benötigten Daten für Visualisierungen bekommen und aufbauen. Es existieren Dashboards für die Foren und den generellen Benutzer Daten.

3 Installation

Die Step-By-Step Guide findet man in der `README.md` des Projektes.