Rijha Ehtsham

LAKE Solutions AG

7.6.2023

Ein Bild, das Design enthält.

Automatisch generierte Beschreibung mit geringer Zuverlässigkeit

Bildquelle: <https://www.indusface.com/blog/how-to-add-bot-management-solution-in-your-web-security-measures/>

Dokumentation Abschlussprojekt WebSecBot

**Inhaltsverzeichnis**

[**Änderungstabelle** 3](#_Toc138698339)

[**Motivation** 4](#_Toc138698340)

[**Was für ein Thema wähle ich?** 4](#_Toc138698341)

[**Warum will ich dieses Projekt durchführen?** 4](#_Toc138698342)

[**Welche Funktionen will ich anbieten?** 4](#_Toc138698343)

[**Was möchte ich mit meinem Projekt erreichen?** 5](#_Toc138698344)

[**Was für Umsetzungsmöglichkeiten habe ich?** 5](#_Toc138698345)

[**Planung/Entscheidung** 5](#_Toc138698346)

[**Was für Meilensteine und optionale Ziele setze ich mir?** 5](#_Toc138698347)

[**Was will ich bis wann erreichen?** 5](#_Toc138698348)

[**Welche Wege gehe ich und welche Möglichkeiten habe ich?** 6](#_Toc138698349)

[**Zeitplanung** 8](#_Toc138698350)

[**Soll-Zustand:** 8](#_Toc138698351)

[**Ist-Zustand:** 8](#_Toc138698352)

[**Umsetzung** 9](#_Toc138698353)

[**Umsetzung Meilenstein 1** 9](#_Toc138698354)

[**Was möchte ich beim ersten Meilenstein erreichen?** 9](#_Toc138698355)

[**Struktur und Funktionsweise von Webanwendungen** 9](#_Toc138698356)

[**Aufbau einer Webanwendung** 9](#_Toc138698357)

[**Komponenten** 9](#_Toc138698358)

[**Das Web-Frontend** 9](#_Toc138698359)

[**Das Web-Backend** 10](#_Toc138698360)

[**Die gängisten Sicherheitslücken und Angriffstechniken** 10](#_Toc138698361)

[**Cross-Site Scripting (XSS)** 10](#_Toc138698362)

[**Ein Beispiel für Cross-Site Scripting (XSS)** 11](#_Toc138698363)

[**SQL-Injection** 11](#_Toc138698364)

[**Ein Beispiel für SQL-Injection** 12](#_Toc138698365)

[**Umgehung von Queue-it (Warteschlangen-Management-System)** 13](#_Toc138698366)

[**Techniken und Ansätze, zur Umgehung von Queue-It** 13](#_Toc138698367)

[**Umsetzung Meilenstein 2** 13](#_Toc138698368)

[**Was möchte ich beim ersten Meilenstein erreichen?** 13](#_Toc138698369)

[**Was ist ein Bot und warum werden diese verwendet** 14](#_Toc138698370)

[**Mein eigenes Script entwickeln** 14](#_Toc138698371)

[**Umsetzung Meilenstein 3** 16](#_Toc138698372)

[**Was möchte ich beim zweiten Meilenstein erreichen?** 16](#_Toc138698373)

[**Funktionsweis Captcha-Systeme und warum sie eingesetzt werden** 16](#_Toc138698374)

[**Beispiele:** 17](#_Toc138698375)

[**2Captcha** 17](#_Toc138698376)

[**Mein eigens Script entwickeln** 18](#_Toc138698377)

[**Endergebnis** 20](#_Toc138698378)

[**Probleme** 26](#_Toc138698379)

[**Fazit** 27](#_Toc138698380)

# **Änderungstabelle**

|  |  |
| --- | --- |
| 07.06.2023 | Dokument erstellt |
| 07.06.2023 | Dokument an Angaben angepasst |
| 07.06.2023 | Motivation geschrieben |
| 08.06.2023 | Zeitplanung Soll-Zustand erstellt und hinzugefügt |
| 08.06.2023 | Kapitel zu Umsetzung Meilenstein 1 angefangen zu schreiben |
| 08.06.2023 | Abschnitt zu «Was möchte ich beim ersten Meilenstein erreichen?» geschrieben. |
| 08.06.2023 | Abschnitt zu «Struktur und Funktionsweise von Webanwendungen» geschrieben |
| 08.06.023 | Abschnitt zu «Die gängigsten Sicherheitslücken und Angriffstechniken» geschrieben |
| 08.06.2023 | Abschnitt zu «Umgehung von Queue-it (Warteschlangen-Management-System)» geschrieben |
| 08.06.2023 | Abschnitt zu «Techniken und Ansätze, zur Umgehung von Queue-It» geschrieben |
| 09.06.2023 | Kapitel zu Umsetzung Meilenstein 2 angefangen zu schreiben |
| 09.06.2023 | Abschnitt zu «Was möchte ich beim ersten Meilenstein erreichen?» geschrieben |
| 14.06.2023 | Mein eigenes Script hinzugefügt und mit Kommentaren erklärt |
| 14.06.2023 | Kapitel zu Umsetzung Meilenstein 3 angefangen zu schreiben |
| 14.06.2023 | Abschnitt zu «Was möchte ich beim ersten Meilenstein erreichen?» geschrieben. |
| 16.06.2023 | Abschnitt zu «Funktionsweis Captcha-Systeme und warum sie eingesetzt werden», geschrieben. |
| 16.06.2023 | Abschnitt zu «2Captcha» geschrieben |
| 21.06.2023 | Mein eigenes Script hinzugefügt und mit Kommentaren erklärt |
| 28.06.2023 | Abschnitt zu «Probleme» verfasst |
| 28.06.2023 | Abschnitt zu «Fazit» geschrieben |
| 28.06.2023 | Zeitplanung Ist-Zustand erstellt und hinzugefügt |

# **Motivation**

## **Was für ein Thema wähle ich?**

In meinem Projekt geht es darum, die Sicherheit von Webapplikationen zu verstehen und zu testen, indem ich ein Bot entwickle, welcher verschiedene Sicherheitsmechanismen umgehen kann.

Daraus kann ich lernen, wie (un)sicher das Web ist und wie man die Sicherheit verstärken kann.

Der Name des Projekts ist WebSecBot und ist zusammengestellt aus den Worten Web, Security und Bot.

## **Warum will ich dieses Projekt durchführen?**

Da ich mich seit längerem mit dem Thema befasst habe und ich dieses Projekt schon immer mal machen wollte, möchte ich die Chance nutzen und die Idee bei meinem Abschlussprojekt umsetzen.

Generell interessiert mich der IT-Security Bereich in meinem Beruf am meisten, weshalb ich mir mit meinem Projekt mein Wissen in diesem Bereich erweitern möchte.

Mein Ziel ist es, ein tieferes Verständnis für Sicherheitsmechanismen zu entwickeln und zu lernen, wie man diese überwinden kann. Durch die Umsetzung dieses Projekts werde ich meine Fähigkeiten im Bereich Webentwicklung, Automatisierung und Sicherheit stärken und neue Kenntnisse in Programmierung, Scripting und Sicherheitswerkzeugen erwerben.

## **Welche Funktionen will ich anbieten?**

Die folgende Funktionen sollte mein Bot anbieten können:

* Website-Einschränkungen wie zum Beispiel Queue-It also Warteschleifen bei Loadingpages mit der Codierung eines Bypass Scripts umgehen können
* Vorinstallierte Daten eingeben können (z. B. Name, Adresse, Kartendetails usw.)
* Der Bot sollte mit Captcha-Systemen wie Capmonster kompatibel sein, um das Captcha zu lösen

Optional wären folgende Funktionen:

* Verwendung verschiedener Proxys, um als „neue Benutzer“ zu erscheinen
* Überwachen des Backends der Website anhand des SKU-Codes

## **Was möchte ich mit meinem Projekt erreichen?**

Als Endergebnis möchte ich einen Bot entwickelt haben, welcher meine genannten Funktionen umsetzen kann. Durch dieses Projekt würde ich folgende Aspekte erlernen:

* Website Frameworks kennenlernen
* Gefährdete Bereiche analysieren können
* Verständnis der IT-Sicherheit im Web lernen
* Lernen, wie die Sicherheit auf Websites erhöht werden kann
* Lernen wie Dinge gefiltert werden

## **Was für Umsetzungsmöglichkeiten habe ich?**

Zur Umsetzung meines Projekts werde ich verschiedene Programmiersprachen und Frameworks nutzen, die sich auf Webentwicklung, Automatisierung und Sicherheit spezialisieren. Ich werde Programmierkenntnisse erwerben und praktische Erfahrungen sammeln, um meinen Bot zu entwickeln und die gewünschten Funktionen umzusetzen.

Es ist auch wichtig, sich mit ethischen Aspekten und rechtlichen Rahmenbedingungen auseinanderzusetzen, um sicherzustellen, dass mein Projekt legal und ethisch verantwortungsvoll ist.

# **Planung/Entscheidung**

## **Was für Meilensteine und optionale Ziele setze ich mir?**

**Meine Meilensteine:**

* Website-Einschränkungen wie zum Beispiel Queue-It also Warteschleifen bei Loadingpages mit der Codierung eines Bypass Scripts umgehen können.
* Der Bot sollte vorinstallierte Daten eingeben können (z. B. Name, Adresse, Kartendetails usw.).
* Der Bot sollte mit Captcha-Systemen wie Capmonster kompatibel sein, um das Captcha zu lösen.

**Optionale Ziele:**

* Verwendung verschiedener Proxys, um als „neue Benutzer“ zu erscheinen.  
  Überwachen des Backends der Website anhand des SKU-Codes.

## **Was will ich bis wann erreichen?**

(Siehe Zeitplan)

## **Welche Wege gehe ich und welche Möglichkeiten habe ich?**

Meine Vorgehensweise habe ich durch meine spezifischen Meilensteine und optionalen Zielen wie folgt geplant:

**Vorgehensweise Meilenstein 1:**

1. Wissen in der Programmiersprache JavaScript vertiefen.
2. Grundwissen in HTML und CSS vertiefen/wiederholen.
3. Verständnis für die Struktur und Funktionsweise von Webanwendungen vertiefen.
4. Die Grundlagen der Webanwendungssicherheit, einschliesslich der gängigen Sicherheitslücken und Angriffstechniken wie Cross-Site Scripting (XSS), SQL-Injection, Cross-Site Request Forgery (CSRF) usw. erlernen
5. Verschiedene Techniken und Ansätze zur Umgehung von Website-Einschränkungen kennenlernen.
6. Verschiedene Web-Frameworks und deren Verwendung, um Webanwendungen zu entwickeln kennenlernen.
7. Mit dem erlernten Wissen Bypass Script entwickeln.
8. Das entwickelte Bypass Script auf seine Funktionalität überprüfen, indem ich es auf der betreffenden Website testen.

**Vorgehensweise Meilenstein 2:**

1. Verschiedene HTML-Tags und -Elemente erkunden, um das Formular für die Dateneingabe zu verstehen.
2. Die Struktur von HTML-Formularen studieren und wie sie Informationen erfassen.
3. Lernen, wie HTTP-Anfragen funktionieren und wie Daten zwischen einem Client (Bot) und einem Server ausgetauscht werden.
4. Die Unterschiede zwischen GET- und POST-Anfragen kennenlernen und wie Daten in Anfragen übertragen werden.
5. Automatisierungswerkzeuge und -frameworks erforschen, welche ermöglichen, Interaktionen mit Webseiten zu automatisieren.
6. Wissen über Web Scraping vertiefen, um Daten von Webseiten zu extrahieren.
7. Programmierkenntnisse anwenden, um eine Logik zu entwickeln, mit der der Bot vorinstallierte Daten in die entsprechenden Formularfelder eingeben kann.
8. Bot auf verschiedenen Webseiten testen, um sicherzustellen, dass er die vorinstallierten Daten korrekt eingibt.
9. Mögliche Fehler oder Probleme analysieren und diese beheben, um die Funktionalität des Bots zu verbessern.

**Vorgehensweise Meilenstein 3:**

1. Lernen, wie Captcha-Systeme funktionieren und warum sie eingesetzt werden, um automatisierte Bots zu erkennen.
2. Verschiedene Captcha-Techniken wie Textbasierte Captchas untersuchen.
3. Über die Capmonster-API und deren Funktionsweise informieren.
4. Lernen, wie ich HTTP-Anfragen an die Capmonster-API senden kann, um Captchas zu lösen.
5. Bot so zu erweitern, dass er das Captcha automatisch über die Capmonster-API lösen kann.
6. Eigenverantwortliches Arbeiten und Lernen
7. Bot auf verschiedenen Webseiten testen, die Captchas verwenden, um sicherzustellen, dass er das Captcha korrekt lösen kann.
8. Mögliche Fehler oder Probleme analysieren und diese beheben, um die Funktionalität des Bots zu verbessern.

**Vorgehensweise Optionales Ziel 1:**

1. Grundlagen von IP-Adressen, Ports, TCP/IP-Protokollen und Netzwerkverbindungen
2. vertiefen/wiederholen.
3. Proxy-Server erforschen und wie sie verwendet werden, um den Datenverkehr zwischen einem Client und einem Server umzuleiten.
4. Verschiedenen Arten von Proxys kennenlernen.
5. Vertrauenswürdigen Proxyservice-Anbietern suchen, die eine Liste von Proxys zur Verfügung stellen.
6. Über die unterschiedlichen Funktionen und Eigenschaften der Proxyservices informiere, wie z. B. die Möglichkeit, Proxys nach Standort, Geschwindigkeit und Protokoll auszuwählen.
7. Den Bot so zu erweitern, dass er Proxys verwenden kann, um als "neue Benutzer" zu erscheinen.
8. Lernen, wie ich HTTP-Anfragen über einen Proxy-Server senden und die Proxy-Adresse in Ihre Anfragen integrieren können.
9. Lernen, wie ich in meinem Bot die Fähigkeit implementieren kann, zwischen verschiedenen Proxys zu wechseln, um Ihre Identität zu ändern.

**Vorgehensweise Optionales Ziel 2:**

1. Lernen, was ein SKU-Code ist und wie er verwendet wird, um ein Produkt oder einen Artikel zu identifizieren.
2. Lernen, wie der SKU-Code in der Datenbank oder im Backend der Website verwendet wird.
3. Monitoring-Tools suchen, die helfen, das Backend einer Website zu überwachen.
4. Die Möglichkeiten erforschen, um Daten aus dem Backend einer Website abzurufen, wie APIs, Datenbankabfragen oder Protokolldateien.
5. Lernen, wie ich die gewünschten Informationen über den SKU-Code aus den Monitoring-Schnittstellen extrahieren kann.
6. Programmierkenntnisse verwenden, um einen Mechanismus zu erstellen, um die relevanten Daten aus dem Backend der Website anhand des SKU-Codes zu extrahieren.
7. Lernen, wie ich HTTP-Anfragen oder Datenbankabfragen verwenden kann, um die gewünschten Informationen abzurufen.
8. Lernen, wie ich die erhaltenen Daten aus dem Backend analysieren und relevante Informationen extrahieren können.
9. Implementierung testen.

# **Zeitplanung**

## **Soll-Zustand:**

Ein Bild, das Text, Screenshot, Wolke enthält.

Automatisch generierte Beschreibung

## **Ist-Zustand:**

Ein Bild, das Text, Screenshot, Wolke enthält.

Automatisch generierte Beschreibung

Quelle für Zeitplan: <https://trello.com/b/ptdlbEOJ/abschlussprojekt-websecbot>

# **Umsetzung**

## **Umsetzung Meilenstein 1**

### **Was möchte ich beim ersten Meilenstein erreichen?**

Website-Einschränkungen wie zum Beispiel Queue-It also Warteschleifen bei Loadingpages mit der Codierung eines Scripts umgehen können.

### **Struktur und Funktionsweise von Webanwendungen**

#### **Aufbau einer Webanwendung**

Webanwendungen bestehen grundsätzlich aus einem Frontend und einem Backend. Die Art und Weise, wie diese Elemente miteinander verknüpft werden, wird über die so genannte Geschäftslogik der Anwendung gesteuert. Vereinfacht ausgedrückt ist dabei das Frontend das, was der Nutzer der Webanwendung angezeigt bekommt. Demgegenüber steht das Backend, das der Verwaltung, der Administration sowie der Dateneingabe und -pflege dient. Die Daten selbst können in einer Datenbank, einem Dateisystem oder auf einem externen Server abgelegt sein.

#### **Komponenten**

Neben der Aufteilung in Front- und Backend kann die Architektur von Webanwendungen auch anhand ihrer Komponenten beschrieben werden:

* Die Benutzerschnittstelle (User Interface bzw. UI) umfasst alle grafischen Elemente, welche dem Nutzer Informationen oder Eingabemöglichkeiten zur Verfügung stellt. Beispielsweise zählen dazu Texte, Grafiken, Hyperlinks und Formulare.
* Die Prozesslogik steuert den Interaktionsablauf der Anwendung. Beispielsweise wird die Prozesslogik eines Online-Shops sicherstellen, dass der Anwender seine Zahlungsinformationen eingibt, bevor eine Bestellung abgeschlossen wird.
* Die Geschäftslogik verarbeitet alle Eingaben des Anwenders. Sie stellt somit das »Herz« der Anwendung dar und verwaltet alle Zugriffe auf die Datenbank. In einem Online-Shop wird die Geschäftslogik beispielsweise prüfen, welche Produkte überhaupt lieferbar sind.
* Die Datenhaltung beschreibt die Art der Speicherung aller nichtflüchtigen Daten der Anwendung. In einem Online-Shop werden hier beispielsweise die Bestellungen in einer Datenbank abgelegt.

#### **Das Web-Frontend**

Der Begriff Web-Frontend beschreibt typischerweise die Darstellung von Texten, Grafiken und Interaktionselementen im Webbrowser. In diesem Zusammenhang wird häufig vom GUI (graphical user interface) oder kurz UI (user interface) gesprochen. Die Implementierung der einzelnen GUI-Elemente, wie beispielsweise Buttons, Textboxen oder Auswahlfelder, erfolgt in den meisten Fällen mittels der Seitenbeschreibungssprache HTML, der Formatierungs- sprache CSS sowie der Scriptsprache JavaScript. Weitere, auf zusätzliche Browser Plugins angewiesene Techniken, wären z. B. Adobe Flash oder Microsoft Silverlight.

Das Frontend einer Webanwendung trägt oft entscheidend zur Akzeptanz und zur positiven Beurteilung der Qualität der Software durch den Nutzer bei. Eine angemessene Bedienung, ein klares übersichtliches Layout sowie eine stabile, fehlertolerante Steuerbarkeit ermöglichen eine flüssige Interaktion und ein einfaches Auffinden der gewünschten Informationen.

#### **Das Web-Backend**

Das Backend ist für die Nutzenden nicht sichtbar. Sie nehmen es jedoch wahr, indem sie durch Klicks auf Buttons, Symbole oder Ähnliches bestimmte Handlungen auf einer Webseite oder einem anderen digitalen Produkt auslösen. Diese Handlungen werden im Backend definiert. Auch die Datenverarbeitung, algorithmische Prozesse und die Leistung des Servers liegen im Backend einer Anwendung.

Darüber hinaus werden auch die sogenannten API-Schnittstellen im Backend konfiguriert. Sie ermöglichen, dass Drittanbieter in eine Webseite oder Anwendung integriert werden können. Backend-Entwickler arbeiten mit Programmiersprachen wie beispielsweise PHP, C, Python oder Ruby.

Neben der Programmiersprache wird für das Backend auch eine Datenbanksprache benötigt, um Abfragen abrufen zu können. Die wohl bekannteste Datenbanksprache ist MySQL.

Vereinfacht gesprochen ist das Frontend also der Browser und das Backend der Server. Der Programmierer des jeweiligen Bereiches sorgt dafür, dass die Inhalte für Client oder Server gerendert werden. So wird der Austausch zwischen Client und Server ermöglicht.

### **Die gängisten Sicherheitslücken und Angriffstechniken**

#### **Cross-Site Scripting (XSS)**

Die Abkürzung XSS steht für Cross Site Scripting und bezeichnet eine der häufigsten Angriffsmethoden im Internet. Durch eine Sicherheitslücke auf dem Client oder den Server gelingt es dem Angreifer, seinen Schadcode in eine vermeintlich vertrauenswürdige Umgebung einzubetten. Mit XSS lassen sich Internetseiten verändern, Browser übernehmen oder vertrauliche Informationen wie Passwörter entwenden.

Cross Site Scripting zählt zu den aktiven Angriffsmethoden und kann als Grundlage für weitere Angriffe verwendet werden. Die Angriffsmethode nutzt die im Internet weit verbreiteten Skriptsprachen wie JavaScript. XSS kann erfolgreich sein, wenn die Webanwendung die entgegengenommenen Daten nicht ausreichend prüft und sie anschliessend weiterverarbeitet oder weiterreicht. So lassen sich Skripte an einen Webserver oder an einen Browser senden, der diese ausführt.

Reflektiertes Cross Site Scripting läuft serverseitig ab. Ein Opfer klickt eine präparierte URL an, in der schädlicher Code eingefügt ist. Der Server übernimmt diesen Code aufgrund mangelnder Absicherung und generiert eine dynamisch veränderte Webseite. Der Anwender sieht eine vom Angreifer manipulierte Webseite und hält sie für vertrauenswürdig. Von ihm gemachte Eingaben lassen sich anschliessend beispielsweise an einen Server des Angreifers weiterleiten.

##### **Ein Beispiel für Cross-Site Scripting (XSS)**

Angenommen, eine Webanwendung enthält ein Gästebuch, in dem Benutzer Kommentare hinterlassen können. Die Kommentare werden auf der Seite angezeigt, ohne vorherige Überprüfung oder Filterung. Ein Angreifer könnte dann bösartigen JavaScript-Code in seinen Kommentar einschleusen.

Beispielhafte bösartige Eingabe des Angreifers:

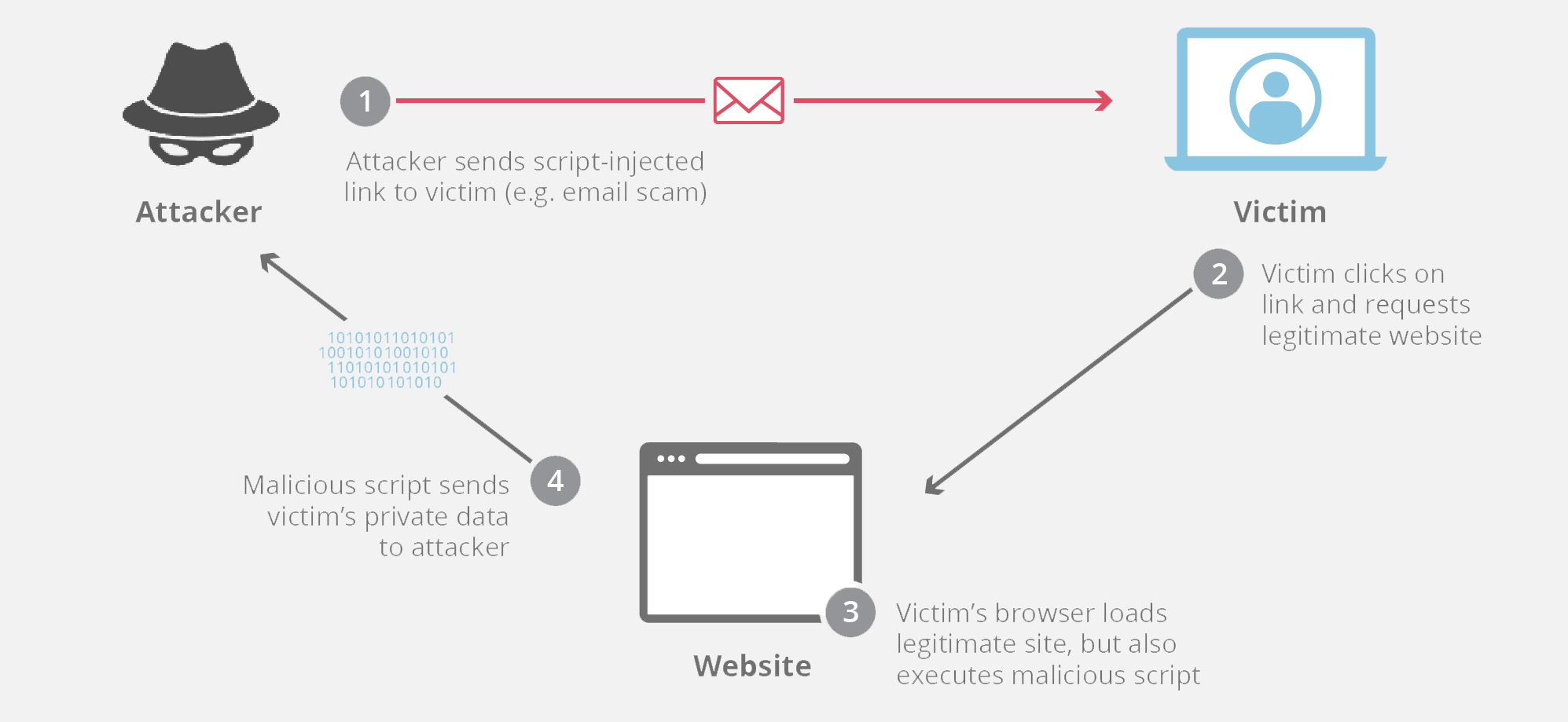
html

<script>

// Bösartiger Code, der sensible Daten stiehlt oder andere schädliche Aktionen ausführt

</script>

Wenn ein anderer Benutzer das Gästebuch aufruft und die Kommentare anzeigt, wird der bösartige JavaScript-Code im Browser des Benutzers ausgeführt. Dadurch erhält der Angreifer Zugriff auf sensible Informationen des Benutzers oder kann Aktionen in seinem Namen ausführen.



Veranschaulichung Cross-Site Scripting (XSS): <https://www.cloudflare.com/learning/security/threats/cross-site-scripting/>

#### **SQL-Injection**

SQL-Injection (zu Deutsch: SQL-Einschleusung) ist ein populärer Angriffsvektor. SQL ist eine beliebte Datenbanksprache, die zum Aufbau von Datenstrukturen in relationalen Datenbanken sowie zum Bearbeiten und Abfragen von darauf basierenden Datenbeständen zum Einsatz kommt. Aufgrund der weiten Verbreitung von SQL sind darauf basierende Programme und Web-Anwendungen auch sehr attraktiv für Angreifer.

Bei einer SQL-Injection (kurz SQLi) nutzen Cyberkriminelle gezielt Sicherheitslücken im Quelltext von Software aus, um etwa über Eingabemasken eigene Befehle oder Schadcode in Programme einzubinden. Auf diese Weise gelangen Angreifer an wertvolle Datensätze. Die meisten Attacken mittels SQLi sind auf fehlerhafte Skripte und Programme zurückzuführen.

Wie weit SQLi allgemein verbreitet ist, zeigt unter anderem das renommierte „Open Web Application Security Project“ – kurz OWASP. Der OWASP-Top-10-Bericht listet Injection-Attacken auf Platz eins der zehn kritischsten Sicherheitsrisiken für Webanwendungen.

##### **Ein Beispiel für SQL-Injection**

Angreifer nutzen SQL-Injections etwa um Datenbanken für User, Schüler, Produkte und dergleichen zu korrumpieren. Hier verwenden Software-Entwickler typischer Weise eine REST-API für den Abruf von Benutzern:

* https://myapi.com/users/123

Diese Eingabe würde wahrscheinlich zu einer (nicht gesäuberten) Anfrage wie dieser führen würde:

* SELECT \* from users where id = 123

Wenn die API über keine Eingabedaten-Validierung / -Hygiene verfügt, besteht die Möglichkeit, mittels SQLi jeden Benutzer im System zu löschen. Die hierzu erforderlichen Eingaben sehen etwa so aus:

* https://myapi.com/users/123;DELETE FROM users where id != 0

oder so:

* https://myapi.com/users/123;TRUNCATE users

Hat der Angreifer Kenntnisse über die Datenstruktur in der Datenbank, könnte er durch Hinzufügen oder Aktualisieren von Einträgen in der Datenbank an Admin-Rechte erlangen:

* https://myapi.com/users/123;UPDATE users set admin = 1 where id = 456

oder so:

* https://myapi.com/users/123;INSERT INTO user\_roles (userId, role) VALUES (456, 'ADMIN')

### **Umgehung von Queue-it (Warteschlangen-Management-System)**

Ein Queue-it-System ist ein virtuelles Warteschlangen-Management-System, das von der Firma Queue-it bereitgestellt wird. Es wird verwendet, um den Zugriff auf Websites oder Online-Veranstaltungen zu steuern und eine Überlastung der Server zu vermeiden.

Wenn eine Website oder eine Online-Veranstaltung eine hohe Nachfrage erwartet, kann das Queue-it-System eingesetzt werden, um Besucher in eine virtuelle Warteschlange zu stellen. Anstatt dass alle Besucher gleichzeitig auf die Website zugreifen, werden sie nacheinander in die Warteschlange gestellt und in regelmässigen Abständen schrittweise auf die Website gelassen. Dadurch wird die Serverbelastung gleichmässig verteilt und ein reibungsloser Zugriff für alle Besucher gewährleistet.

Das Queue-It-System zeigt den wartenden Besuchern eine anpassbare Warteseite an, auf der sie Informationen über ihre Position in der Warteschlange erhalten. Zusätzlich können Unternehmen oder Veranstalter benutzerdefinierte Nachrichten, Fortschrittsanzeigen oder geschätzte Wartezeiten anzeigen, um den Besuchern Transparenz zu bieten und ihre Geduld zu erleichtern.

Queue-it bietet verschiedene Funktionen und Konfigurationsmöglichkeiten an, um den spezifischen Anforderungen eines Unternehmens gerecht zu werden. Dazu gehören beispielsweise das Festlegen der Warteschlangenkapazität, das Hinzufügen von personalisierten Botschaften, das Einbinden von Branding-Elementen und die Integration mit anderen Systemen wie Ticketing-Plattformen oder E-Commerce-Websites.

#### **Techniken und Ansätze, zur Umgehung von Queue-It**

Das Umgehen von Warteschleifen bei Ladevorgängen ist eine spezifische Herausforderung und erfordert oft fortgeschrittenere Techniken. Hier sind einige Ansätze, die verwendet werden können, um solche Einschränkungen zu umgehen:

1. Verwenden von Browser-Erweiterungen: Es gibt bestimmte Browser-Erweiterungen, die entwickelt wurden, um Website-Einschränkungen zu umgehen. Einige dieser Erweiterungen können dazu beitragen, Warteschleifen zu umgehen. Ein Beispiel dafür ist das "Skip Redirect", "Redirect Bypasser" oder auch "Mod Header" Plugin.
2. Script schreiben: Um ein Queue-it zu übergehen hat man die Möglichkeit ein JavaScript Script zu schreiben, welcher genau diesen Queue-It übergehen kann.

## **Umsetzung Meilenstein 2**

### **Was möchte ich beim ersten Meilenstein erreichen?**

Bei meinem zweiten Meilenstein möchte ich erreichen, dass ich einen Bot programmiere, welcher vorinstallierte Daten wie zum Beispiel Name, Vorname, Adresse usw. eingeben kann.

### **Was ist ein Bot und warum werden diese verwendet**

Ein Bot ist eine Softwareanwendung, die automatisierte Aufgaben ausführt. Bots können programmiert werden, um wiederkehrende Aktionen oder Interaktionen auf Websites, in Anwendungen oder in anderen digitalen Umgebungen durchzuführen. Sie können Aufgaben wie das Ausfüllen von Formularen, das Durchsuchen von Webseiten, das Versenden von Nachrichten oder das Sammeln von Informationen automatisieren.

Bots werden aus verschiedenen Gründen verwendet, und nicht alle Bots haben schlechte Absichten. Hier sind einige Beispiele für den Einsatz von Bots:

* Automatisierung: Bots können repetitive Aufgaben automatisieren und so Zeit und Aufwand sparen. Zum Beispiel können sie bei der Datenanalyse, im Kundenservice oder im E-Commerce eingesetzt werden.
* Informationsbeschaffung: Bots können verwendet werden, um Informationen von Websites zu sammeln, beispielsweise für Preisvergleiche, Marktforschung oder Überwachung von Datenänderungen.
* Benutzerunterstützung: Einige Bots sind darauf ausgelegt, Benutzern bei der Interaktion mit Websites oder Anwendungen zu helfen. Sie können Fragen beantworten, Anleitungen geben oder Informationen bereitstellen.

Allerdings können Bots auch für unerwünschte Aktivitäten eingesetzt werden, wie Spamming, das Erstellen von gefälschten Konten, das Verbreiten von Fehlinformationen oder das Durchführen von bösartigen Angriffen. In einigen Fällen können Bots verwendet werden, um vorinstallierte Daten einzugeben, um Systeme oder Dienste zu manipulieren oder zu missbrauchen.

### **Mein eigenes Script entwickeln**

// Importieren der erforderlichen Module

const puppeteer = require('puppeteer-extra');

const StealthPlugin = require('puppeteer-extra-plugin-stealth');

puppeteer.use(StealthPlugin());

// Definieren der Hauptfunktion als asynchrone Funktion

var finishline = async function () {

  // Starten des Browser-Instances

  const browser = await puppeteer.launch({ headless: false, slowMo: 10 });

  const page = await browser.newPage();

  // Setzen des User-Agents

  await page.setUserAgent('Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36');

  // Öffnen einer bestimmten Seite

  await page.goto('https://www.finishline.com/store/product/mens-nike-air-force-1-low-casual-shoes/prod795980?styleId=CW2288&colorId=111');

  // Warten für eine bestimmte Zeit

  await page.waitForTimeout(100000);

  // Klicken auf einen Button mit einem bestimmten Attributwert

  await page.click("button[data-size='8.0']");

  await page.waitForTimeout(5000);

  // Klicken auf ein Element mit einer bestimmten ID

  await page.click('#buttonAddToCart');

  // Warten für eine bestimmte Zeit

  await page.waitForTimeout(5000);

  // Warten auf das Erscheinen eines bestimmten Selektors und Klicken darauf

  await page.waitForSelector("a.button.expanded.mb-2.js-cart-proceed-btn");

  await page.click("a.button.expanded.mb-2.js-cart-proceed-btn");

  // Warten für eine bestimmte Zeit

  await page.waitForTimeout(5000);

  // Eingeben von Text in Textfelder mit bestimmten IDs

  await page.type("input[name='firstName']", 'Rijha');

  await page.type('#shippingLastName', 'Ehtsham');

  await page.type('#shippingAddress1', 'Teststrasse 1');

  await page.type('#shippingCity', 'Zuerich');

  await page.select('#shippingState', 'AK');

  await page.type('#shippingZip', '21224');

  await page.type('#shippingPhone', '553-321-2323');

  await page.type('#email', 'rijhaehtsham1@gmail.com');

  // Warten für eine bestimmte Zeit

  await page.waitForTimeout(5000);

  // Klicken auf einen Button mit einem bestimmten Attributwert

  await page.click("button[type='submit']", btn => btn.click());

  // Warten für eine bestimmte Zeit

  await page.waitForTimeout(5000);

  // Eingeben von Text in Textfelder mit bestimmten IDs

  await page.type('#billingCardNumber', '4539714410847600');

  await page.waitForTimeout(200);

  await page.select('#billingExpirationMonth', '05');

  await page.waitForTimeout(200);

  await page.select('#billingExpirationYear', '2027');

  await page.waitForTimeout(200);

  await page.select('#billingSecurityCode', '948');

  await page.waitForTimeout(200);

  // Versuch, auf ein Element mit einer bestimmten ID zu klicken und Fehlerbehandlung

  try {

    await page.click('#billingContinnueButton', btn => btn.click());

  } catch (ex) {

    await page.click("button[id='billingContinueButton']", btn => btn.click());

  }

  // Warten für eine bestimmte Zeit

  await page.waitFor(200);

  // Klicken auf ein Element mit einer bestimmten ID

  await page.click("button[id='submitOrder']", btn => btn.click());

}();

## **Umsetzung Meilenstein 3**

### **Was möchte ich beim zweiten Meilenstein erreichen?**

Der Bot sollte mit Captcha-Systemen wie Capmonster oder 2Captcha kompatibel sein, um das Captcha bei einer Sicherheitsprüfung zu lösen.

### **Funktionsweis Captcha-Systeme und warum sie eingesetzt werden**

Captcha steht für "Completely Automated Public Turing test to tell Computers and Humans Apart". Es handelt sich um ein Sicherheitssystem, das entwickelt wurde, um menschliche Benutzer von automatisierten Programmen, sogenannten Bots, zu unterscheiden. Captcha-Systeme werden verwendet, um sicherzustellen, dass bestimmte Aktionen im Internet von echten Menschen und nicht von automatisierten Programmen durchgeführt werden.

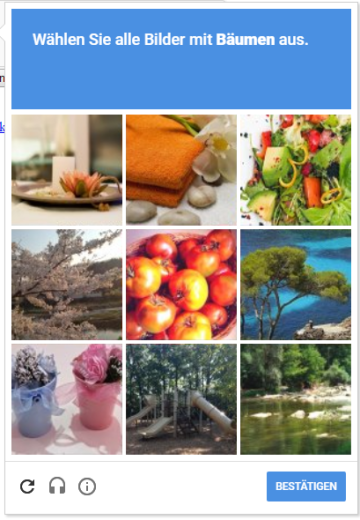
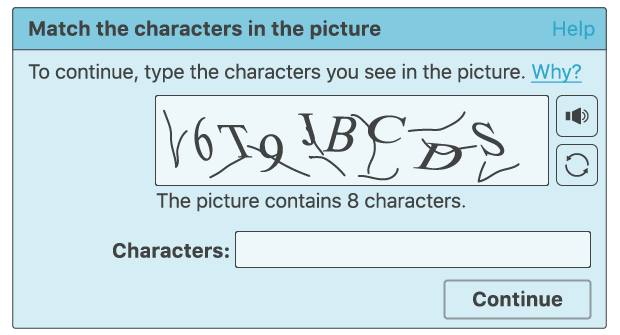
Die Funktionsweise von Captcha-Systemen beruht auf der Annahme, dass Menschen bestimmte visuelle oder kognitive Fähigkeiten besitzen, die für Maschinen schwer zu imitieren sind. Es gibt verschiedene Arten von Captchas, aber die gängigste Methode besteht darin, dem Benutzer eine Aufgabe oder eine Herausforderung zu präsentieren, die von einer menschlichen Intelligenz leicht gelöst werden kann, für Maschinen jedoch schwierig ist.

Ein bekanntes Beispiel für ein Captcha ist das "reCaptcha" von Google. Es kann beispielsweise ein verzerrtes oder verwischtes Bild von Buchstaben und Zahlen präsentieren und den Benutzer auffordern, die Zeichen in ein Eingabefeld einzugeben. Der Gedanke dahinter ist, dass Menschen in der Lage sind, die Buchstaben und Zahlen trotz der Verzerrung zu erkennen und einzugeben, während es für Maschinen schwierig ist, die Verzerrungen zu überwinden und die richtigen Zeichen zu identifizieren.

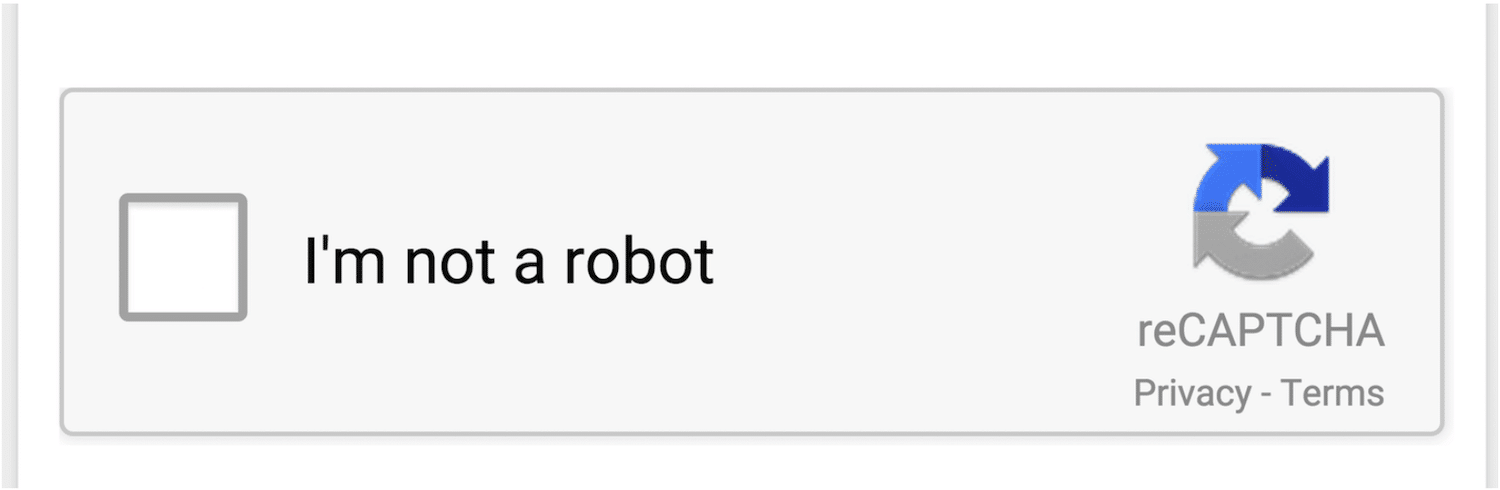
Captcha-Systeme werden aus verschiedenen Gründen eingesetzt. Hier sind einige Hauptgründe:

1. Bot-Abwehr: Captchas dienen dazu, automatisierte Bots daran zu hindern, automatisierte Aufgaben im Internet auszuführen, wie z.B. das Erstellen von Spam-Konten, das Senden von Massen-E-Mails, das Durchführen von DDoS-Angriffen usw. Durch die Einführung einer Captcha-Herausforderung wird die Hürde für Bots erhöht, um automatisierte Aktionen auszuführen.
2. Schutz vor Brute-Force-Angriffen: Bei einigen Diensten, wie z.B. beim Einloggen auf Websites oder bei der Registrierung von Konten, können Captchas verwendet werden, um Brute-Force-Angriffe zu verhindern. Ein Brute-Force-Angriff besteht darin, systematisch alle möglichen Kombinationen von Passwörtern auszuprobieren, um Zugriff auf ein Konto zu erlangen. Captchas erschweren diesen Prozess, da sie jeden Anmeldeversuch mit einer manuellen Eingabeaufforderung verlangsamen.
3. Datensammlung: Einige Captcha-Systeme, wie reCaptcha von Google, nutzen die menschliche Interaktion, um Daten zu sammeln und die maschinelle Bilderkennung oder andere Technologien zu verbessern. Zum Beispiel kann ein Benutzer gebeten werden, bestimmte Bilder zu identifizieren, um dazu beizutragen, maschinelle Lernalgorithmen zu trainieren.

#### **Beispiele:**

Beispiel Captcha System von Cloudfare



Beispiel Captcha System von Google reCAPTCHA.

Beispiel Captcha System von Google reCAPTCHA.

### **2Captcha**

2captcha ist ein Online-Dienst von Drittanbietern, der eine Lösung zur Umgehung von Captchas anbietet. Captchas sind Sicherheitsmaßnahmen, die von Websites verwendet werden, um zu überprüfen, ob ein Benutzer ein Mensch ist und kein Bot. Einige Benutzer finden Captchas jedoch schwierig zu lösen, insbesondere solche mit verzerrtem Text oder Bildern. 2captcha bietet ein automatisiertes System, das Captchas im Namen der Benutzer löst.

Um 2captcha zu nutzen, müssen Sie normalerweise ihre API in Ihre Anwendung oder Ihr Skript integrieren. Die API ermöglicht es Ihnen, Captcha-Herausforderungen an die Server von 2captcha zu senden, und sie senden Ihnen die gelösten Captchas zurück. Dies kann nützlich sein, um Aufgaben zu automatisieren, die das Lösen von Captchas erfordern, wie das Erstellen von Konten, das Ausfüllen von Formularen oder den Zugriff auf eingeschränkte Inhalte.

Es ist wichtig zu beachten, dass die Verwendung von automatisierten Captcha-Umgehungsdiensten gegen die Nutzungsbedingungen bestimmter Websites verstossen kann. Darüber hinaus haben einige Websites fortschrittlichere Captcha-Mechanismen implementiert, die für automatisierte Systeme schwer zu lösen sind.

Bitte beachten Sie, dass ich ein KI-Sprachmodell bin und meine Informationen möglicherweise nicht auf dem neuesten Stand sind. Daher ist es immer ratsam, die offizielle 2captcha-Website zu besuchen oder sich an den Support zu wenden, um genaue und aktuelle Informationen über ihre Dienste zu erhalten.

### **Mein eigens Script entwickeln**

const puppeteer = require('puppeteer');

const request = require('request-promise-native');

const poll = require('promise-poller').default;

// Dein API-Schlüssel für den Captcha-Lösungsdienst

const apiKey = 'adbc7d647ca8f20eebd39808f859aa81';

// Konfiguration für das Captcha und die Zielseite

const config = {

  sitekey: '6LdTUqEmAAAAAAe1-sUMJBC7a93qQfYPmlM7v-bs',

  pageurl: 'https://rijha.ictblj.ch/', // Ersetze es durch die URL deiner Zielseite

  apiSubmitUrl: 'https://2captcha.com/in.php',

  apiRetrieveUrl: 'https://2captcha.com/res.php'

};

// Funktion, um die Anfrage zur Captcha-Lösung zu initiieren

async function initiateCaptchaRequest(apiKey) {

  const formData = {

    method: 'userrecaptcha',

    googlekey: config.sitekey,

    key: apiKey,

    pageurl: config.pageurl,

    json: 1

  };

  const response = await request.post(config.apiSubmitUrl, { form: formData });

  return JSON.parse(response).request;

}

// Funktion, um den Captcha-Lösungsdienst nach der Antwort abzufragen

async function pollForRequestResults(apiKey, requestId, retries = 30, interval = 1500, delay = 15000) {

  await timeout(delay);

  try {

    return await poll({

      taskFn: requestCaptchaResults(apiKey, requestId),

      interval,

      retries

    });

  } catch (error) {

    if (Array.isArray(error) && error.includes('ERROR\_WRONG\_CAPTCHA\_ID')) {

      console.error('Die Captcha-ID ist ungültig oder abgelaufen.');

      return null;

    }

    throw error;

  }

}

// Funktion, um die Captcha-Lösungsantwort abzurufen

function requestCaptchaResults(apiKey, requestId) {

  const url = `${config.apiRetrieveUrl}?key=${apiKey}&action=get&id=${requestId}&json=1`;

  return async function () {

    const rawResponse = await request.get(url);

    const response = JSON.parse(rawResponse);

    if (response.status === 0) {

      throw new Error('Captcha-Lösung läuft noch.');

    }

    return response.request;

  };

}

// Timeout-Hilfsfunktion

function timeout(ms) {

  return new Promise(resolve => setTimeout(resolve, ms));

}

(async function main() {

  try {

    const browser = await puppeteer.launch({ headless: false });

    const page = await browser.newPage();

    console.log(`Gehe zu ${config.pageurl}`);

    await page.goto(config.pageurl);

    // Implementiere die Schritte, um die Benutzername- und Passwortfelder auszufüllen

    // mit page.type()

    const requestId = await initiateCaptchaRequest(apiKey);

    const response = await pollForRequestResults(apiKey, requestId);

    console.log('Captcha solved:', response);

    // Implementiere den Schritt, um die gelöste Captcha-Antwort in das

    // reCAPTCHA-Eingabefeld mit page.evaluate() einzugeben

    // Abschicken des Anmeldeformulars mit page.click()

    // Beispiel: await page.evaluate(() => {

    //   const captchaResponse = document.getElementById('g-recaptcha-response');

    //   captchaResponse.value = response; // Die gelöste Captcha-Antwort

    // });

    // Beispiel: await page.click('#submit-button');

    await browser.close();

  } catch (error) {

    console.error('Es ist ein Fehler aufgetreten:', error);

  }

})();

# **Endergebnis**

Zum Schluss meines Projekt habe ich einen Bot entwickelt, welcher alle Funktionen meiner einzelnen Meilensteine kombiniert. Mein Bot ist also in der Lage, vorinstallierte Daten einzugeben, das ausgewählte Produkt innerhalb von Sekunden zum Kauf weiterzuleiten und Captchas zu lösen.

Hier ist mein Skript:

// Importieren der erforderlichen Module

const puppeteer = require('puppeteer-extra');

const StealthPlugin = require('puppeteer-extra-plugin-stealth');

const request = require('request-promise-native');

const poll = require('promise-poller').default;

puppeteer.use(StealthPlugin());

// Dein API-Schlüssel für den Captcha-Lösungsdienst

const apiKey = 'adbc7d647ca8f20eebd39808f859aa81';

// Konfiguration für das Captcha und die Zielseite

const config = {

  sitekey: '6LdTUqEmAAAAAAe1-sUMJBC7a93qQfYPmlM7v-bs',

  pageurl: 'https://www.offspring.co.uk/view/product/offspring\_catalog/2,20/x8qVNsFtXd',

  apiSubmitUrl: 'https://2captcha.com/in.php',

  apiRetrieveUrl: 'https://2captcha.com/res.php'

};

// Funktion, um die Anfrage zur Captcha-Lösung zu initiieren

async function initiateCaptchaRequest(apiKey) {

  const formData = {

    method: 'userrecaptcha',

    googlekey: config.sitekey,

    key: apiKey,

    pageurl: config.pageurl,

    json: 1

  };

  const response = await request.post(config.apiSubmitUrl, { form: formData });

  return JSON.parse(response).request;

}

// Funktion, um den Captcha-Lösungsdienst nach der Antwort abzufragen

async function pollForRequestResults(apiKey, requestId, retries = 30, interval = 1500, delay = 15000) {

  await timeout(delay);

  try {

    return await poll({

      taskFn: requestCaptchaResults(apiKey, requestId),

      interval,

      retries

    });

  } catch (error) {

    if (Array.isArray(error) && error.includes('ERROR\_WRONG\_CAPTCHA\_ID')) {

      console.error('Die Captcha-ID ist ungültig oder abgelaufen.');

      return null;

    }

    throw error;

  }

}

// Funktion, um die Captcha-Lösungsantwort abzurufen

function requestCaptchaResults(apiKey, requestId) {

  const url = `${config.apiRetrieveUrl}?key=${apiKey}&action=get&id=${requestId}&json=1`;

  return async function () {

    const rawResponse = await request.get(url);

    const response = JSON.parse(rawResponse);

    if (response.status === 0) {

      throw new Error('Captcha-Lösung läuft noch.');

    }

    return response.request;

  };

}

// Timeout-Hilfsfunktion

function timeout(ms) {

  return new Promise(resolve => setTimeout(resolve, ms));

}

(async function offspring() {

  try {

    const browser = await puppeteer.launch({ headless: false });

    const page = await browser.newPage();

    console.log(`Gehe zu ${config.pageurl}`);

    await page.goto(config.pageurl);

    // Prüfen, ob das Captcha existiert

    const captchaExists = await page.$('#g-recaptcha-response') !== null;

    if (captchaExists) {

      // Implementiere die Schritte, um das Captcha zu lösen und die Captcha-Antwort zu erhalten

      const requestId = await initiateCaptchaRequest(apiKey);

      const response = await pollForRequestResults(apiKey, requestId);

      console.log('Captcha solved:', response);

      // Implementiere den Schritt, um die gelöste Captcha-Antwort in das reCAPTCHA-Eingabefeld einzugeben

      await page.evaluate(response => {

        document.querySelector('#g-recaptcha-response').value = response;

      }, response);

      // Warten für eine bestimmte Zeit

      await page.waitForTimeout(2000);

    }

    // Klicken auf das Element mit der ID "onetrust-accept-btn-handler"

    await page.waitForSelector('#onetrust-accept-btn-handler');

    await page.click('#onetrust-accept-btn-handler');

    // Warten für eine bestimmte Zeit

    await page.waitForTimeout(2000);

    await page.waitForSelector('button.close');

    await page.click('button.close');

    // Klicken auf das Kombinationsfeld für die Schuhgröße

    await page.waitForSelector('button#selectSize');

    await page.click('button#selectSize');

    // Warten auf das Erscheinen der Schuhgrößenoptionen

      await page.waitForSelector('#productSizes');

      // Klicken auf die gewünschte Schuhgröße

      await page.evaluate(() => {

        const sizeOption = document.querySelector('li[data-value="045"]');

        sizeOption.click();

      });

      // Warten für eine bestimmte Zeit (z.B. um den Warenkorb zu überprüfen)

await page.waitForTimeout(2000);

// Klick auf den Button

await page.waitForSelector('#ajaxAdd');

await page.click('#ajaxAdd');

// Warten, um sicherzustellen, dass die Aktion abgeschlossen ist

await page.waitForSelector('.spc-mode-tile.js-deliveryMode[data-name\*="Standard UK Delivery"]');

// Lieferung auswählen

const deliveryOptions = await page.$$('.spc-mode-tile.js-deliveryMode[data-name\*="Standard UK Delivery"]');

if (deliveryOptions.length > 0) {

  // Das erste Element auswählen

  const deliveryOption = deliveryOptions[0];

  await deliveryOption.click();

  console.log('Das Element wurde angeklickt.');

  // Hier kannst du weitere Aktionen nach dem Klick durchführen

} else {

  console.log('Das Element wurde nicht gefunden.');

}

// E-Mail eingeben

await page.waitForSelector('#deliveryEmail');

await page.type('#deliveryEmail', 'rijhaehtsham1@gmail.com');

// Titel wählen

await page.evaluate(() => {

  const radioButton = document.querySelector('input[name="titleCode"][value="miss"]');

  if (radioButton) {

    radioButton.click();

    console.log('Das Radio-Button-Element wurde ausgewählt.');

  } else {

    console.log('Das Radio-Button-Element wurde nicht gefunden.');

  }

});

// Vornamen eingeben

await page.evaluate(() => {

  const firstNameInput = document.getElementById('deliveryFirstName');

  if (firstNameInput) {

    firstNameInput.value = 'Rijha';

    console.log('Das Eingabefeld für den Vornamen wurde bearbeitet.');

  } else {

    console.log('Das Eingabefeld für den Vornamen wurde nicht gefunden.');

  }

});

// Nachname eingeben

await page.evaluate(() => {

  const lastNameInput = document.getElementById('deliveryLastName');

  if (lastNameInput) {

    lastNameInput.value = 'Ehtsham';

    console.log('Das Eingabefeld für den Nachnamen wurde bearbeitet.');

  } else {

    console.log('Das Eingabefeld für den Nachnamen wurde nicht gefunden.');

  }

});

// Telefonnummer eingeben

await page.evaluate(() => {

  const mobileNumberInput = document.getElementById('deliveryMobileNumber');

  if (mobileNumberInput) {

    mobileNumberInput.focus();

    mobileNumberInput.value = '+447496821242';

    console.log('Das Eingabefeld für die Telefonnummer wurde bearbeitet.');

  } else {

    console.log('Das Eingabefeld für die Telefonnummer wurde nicht gefunden.');

  }

});

// Postcode eingeben

await page.evaluate(() => {

  const postcodeInput = document.getElementById('deliveryPostcodeLookup');

  if (postcodeInput) {

    postcodeInput.value = 'E11 1PD5';

    console.log('Das Eingabefeld für die Postleitzahl wurde bearbeitet.');

  } else {

    console.log('Das Eingabefeld für die Postleitzahl wurde nicht gefunden.');

  }

});

// Adresse finden

await page.waitForSelector('button.btn.btn-secondary.btn-fluid.h-space-20.js-postcodeLookupButton');

await page.evaluate(() => {

  const findAddressButton = document.querySelector('button.btn.btn-secondary.btn-fluid.h-space-20.js-postcodeLookupButton');

  if (findAddressButton) {

    findAddressButton.click();

    console.log('Der Knopf "Find address" wurde gedrückt.');

  } else {

    console.log('Der Knopf "Find address" wurde nicht gefunden.');

  }

});

// Adresse wählen

await page.waitForSelector('button[data-id="deliverypostcodeLookupAddresses"]');

await page.evaluate(() => {

  const selectAddressButton = document.querySelector('button[data-id="deliverypostcodeLookupAddresses"]');

  if (selectAddressButton) {

    selectAddressButton.click();

    console.log('Der Knopf "Select your address" wurde gedrückt.');

  } else {

    console.log('Der Knopf "Select your address" wurde nicht gefunden.');

  }

});

await page.waitForXPath('//span[contains(text(), "Flat 5, Eagle Court, Hermon Hill, London, E11 1PD")]');

const element = await page.$x('//span[contains(text(), "Flat 5, Eagle Court, Hermon Hill, London, E11 1PD")]');

if (element.length > 0) {

  await element[0].click();

  console.log('Das Element wurde ausgewählt.');

} else {

  console.log('Das Element wurde nicht gefunden.');

}

// Mit der Bestellung fortfahren

await page.waitForSelector('button.btn.btn-primary.btn-fluid.h-space-30.js-submitBtn.js-continueToPaymentBtn');

await page.waitForTimeout(1000); // Warte 1 Sekunde

await page.click('button.btn.btn-primary.btn-fluid.h-space-30.js-submitBtn.js-continueToPaymentBtn');

console.log('Der Knopf "Continue to payment" wurde geklickt.');

// Zahlart "Gift Card" wählen

const giftCardElement = await page.waitForXPath("//div[contains(@class, 'spc-mode-tile\_\_name') and contains(text(), 'Gift Card')]", { visible: true, timeout: 5000 });

// Klicke auf das gefundene Element

await giftCardElement.click();

// Benachrichtigungen ablehnen

const labelElement = await page.waitForXPath('//span[contains(text(), "I do not want to sign up to OFFSPRING Mobile Text Alerts")]/ancestor::label');

await labelElement.click();

// Bestellung fortfahren

await page.waitForSelector('button.btn.btn-primary.btn-fluid.hover-opacity.btn--payment.js-formPaymentBtn');

// Klicke auf den Button

await page.click('button.btn.btn-primary.btn-fluid.hover-opacity.btn--payment.js-formPaymentBtn');

// Warte auf das Eingabefeld für die Geschenkkartennummer

await page.waitForSelector("input[name='giftCardNumber']");

// Gib die Nummer ein

await page.type("input[name='giftCardNumber']", '821471241723998');

// Warte auf das Eingabefeld für die Geschenkkarten-PIN-Nummer

await page.waitForSelector("input[name='giftCardPinNumber']");

// Gib die Nummer ein

await page.type("input[name='giftCardPinNumber']", '1234');

await page.evaluate(() => {

  const button = document.querySelector("input[name='checkBalance']");

  button.click();

});

  } catch (error) {

    console.error('Fehler beim Ausführen des Skripts:', error);

  }

  })();

# **Probleme**

In meinem Abschlussprojekt gab es einige Herausforderungen, mit denen ich konfrontiert wurde. Diese Probleme umfassten Folgendes:

Zu Beginn stellte ich fest, dass das Projekt sowohl aufwendiger als auch komplexer war als erwartet. Dies erforderte eine umfangreiche Recherche, um die notwendigen Informationen zu sammeln. Die Recherchearbeit nahm viel Zeit in Anspruch und verzögerte den Fortschritt des Projekts. Ich hatte die Komplexität meines Projekts etwas unterschätzt.

Aufgrund der Komplexität war ich auch oft auf Hilfe von meinem Berufsbildner Jörg angewiesen. Obwohl ich finde, dass die Unterstützung, die ich benötigte, gerechtfertigt war, da ich die meisten Probleme selbstständig lösen konnte.

Ein weiteres Problem, dem ich beim Schreiben von meinen Skripts begegnete, waren die Fehler im Code. Wie es in der Programmierung üblich ist, traten immer wieder Fehler im Code auf, die es zu beheben galten. Glücklicherweise konnte ich diese Fehler jedoch meistens selbstständig lösen, indem ich mein Wissen und meine Problemlösungsfähigkeiten bei der Programmierung einsetzte.

Insgesamt waren die Hauptprobleme in meinem Abschlussprojekt die umfangreiche Recherche zu Beginn, die Komplexität des Projekts, die gelegentliche Abhängigkeit von Jörgs Hilfe und das Lösen von Fehlern im Code. Trotz dieser Herausforderungen konnte ich jedoch erfolgreich mit ihnen umgehen und das Projekt letztendlich abschliessen.

# **Fazit**

Das Projekt war für mich eine wertvolle Lernerfahrung, die mein Wissen erheblich erweitert hat. Insbesondere konnte ich ein tieferes Verständnis für Sicherheitsmechanismen entwickeln und erfahren, wie man diese überwinden kann. Durch die praktische Umsetzung des Projekts konnte ich meine Fähigkeiten in den Bereichen Webentwicklung, Automatisierung und Sicherheit stärken. Zudem habe ich neue Kenntnisse in den Bereichen Programmierung und Scripting erworben.

Beim Schreiben des Scripts konnte ich meine Kenntnisse in der Programmiersprache JavaScript, sowie in HTML und CSS erheblich erweitern. Ich habe gelernt, verschiedene Arten von Fehlern im Code zu erkennen und zu beheben. Neben der eigentlichen Umsetzung des Projekts konnte ich meine Fähigkeiten beim Dokumentieren umsetzen und neue Kenntnisse bei der Gestaltung eines Handouts erwerben.

Insgesamt hat mir das Projekt sehr viel Spass gemacht und ich konnte eine Menge lernen. Es war eine wertvolle Erfahrung, bei der ich meine Fähigkeiten verbessern konnte. Das Projekt hat mir gezeigt, wie spannend und herausfordernd die Arbeit an komplexen Projekten sein kann, und ich bin stolz darauf, was ich erreicht habe. Ich bin zuversichtlich, dass die erworbenen Fähigkeiten und Kenntnisse mir in meiner zukünftigen beruflichen Laufbahn von Nutzen sein werden.

**Quellenabspann**

[Bildquelle Titelblatt](https://www.indusface.com/blog/how-to-add-bot-management-solution-in-your-web-security-measures/)

**Struktur und Funktionsweise von Webanwendungen:**

[Frontend](https://www.bitkom.org/sites/default/files/file/import/LF-Webanwendungen-150910-1.pdf)

[Backend](https://blog.hubspot.de/website/frontend-backend)

**Die gängisten Sicherheitslücken und Angriffstechniken:**

[Cross-Site Scripting (XSS)](https://www.heise.de/hintergrund/Sicherheit-von-Webanwendungen-270870.html)

[SQL-Injektion](https://www.myrasecurity.com/de/sql-injection/)

**Funktionsweis Captcha-Systeme und warum sie eingesetzt werden:**

[Captcha](https://www.pandasecurity.com/de/security-info/captcha/)

**Beispiele Captcha-Systeme**

[Cloudfare](https://www.cloudflare.com/de-de/learning/bots/how-captchas-work/)

[Google reCAPTCHA Bilder](https://www.ionos.de/digitalguide/online-marketing/verkaufen-im-internet/captcha-codes-und-bilder-als-spamschutz/)

[Google reCAPTCHA Ankreuzung](https://blog.xeit.ch/2017/02/captcha-der-schutz-gegen-roboter/)

s