Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangslage

Eine der häufigsten Schwachstellen im Internet sind so genannte Cross-Site-Scripting Angriffe. Dabei wird versucht auf eine fremde Website Schadcode zu hinterlegen, welcher danach von anderen Benutzern unbeabsichtigt ausgeführt wird. Die Websecurity Abteilung von SAP versucht solche Schwachstellen automatisiert aufzudecken. Aus diesem Grund wurde ein modifizierter FireFox entwickelt, welcher es ermöglicht herauszufinden, welche Eingaben des Benutzers an kritischen Stellen in der Website gelangt.

1.2 Aufgabenstellung

Oft ist es Sinnvoll nicht nur eine einzige Seite auf Schwachstellen zu testen, sondern automatisiert eine Vielzahl von Seiten zu besuchen. Um dies zu ermöglichen soll im Rahmen dieser Arbeit ein Crawler entwickelt werden, der eine Liste von Startseiten besucht und aus diesen Verweise auf neue Seiten extrahiert. Dafür muss eine Architektur gefunden werden, die sich in das aktuelle System einfügen lässt und sich gut skalieren lässt.

2 Grundlagen

2.1 Crawler

Ein Web Crawler (oft auch Spider gennant) ist ein Computerprogramm, welches es ermöglicht, eine große Anzahl von Webseiten automatisiert zu besuchen.

2.2 Einsatzgebiete

Einer der bekanntesten Einsatzgebiete von Crawlern ist in Suchmaschinen. Diese bauen eine große Datenbank mit indizierten Webseiten auf und erlauben es einem Benutzer diese Datenbanken mit einer Suchanfrage zu durchsuchen. Um diese Datenbank zu füllen und aktuell zu halten wird ein Crawler verwendet, wie zum Beispiel der Googlebot der Suchmaschine Google. [https://support.google.com/webmasters/answer/182072?hl=de]

2.3 Funktionsweise

Die Funktionsweise eines Webcrawlers lässt sich im wesentlichen auf einen einfachen Algorithmus reduzieren. Im ersten Schritt wird eine Seite identifiziert, welche heruntergeladen werden soll. Alle Seiten, die noch besucht werden sollen, sind in einer Warteschlange gespeichert. Zu Beginn des Programms wird diese Schlange mit einer Reihe von Startseiten initialisiert, welche als Ausgangspunkt des Crawlprozesses dienen.
Nachdem der Schlange eine zu besuchende Seite entnommen wurde muss geprüft werden, ob das Crawlen dieser Seite überhaupt erlaubt ist. Mit der Hilfe einer robots.txt Datei oder dem HTTP Header kann festgestellt werden, ob der Webseitenbetreiber das Crawlen nicht erwünscht. Falls die Seite

besucht werden darf, wird sie komplett heruntergeladen und zu einer Liste mit besuchten Seiten hinzugefügt. Aus dem so erhaltenen HTML-Code werden nun alle Links auf der Seite extrahiert. Diese neuen Links werden der Warteschlange aus dem ersten Schritt hinzugefügt, sofern sie nicht in der Liste mit besuchten Seiten enthalten sind. Somit wird verhindert, dass eine Seite mehrmals besucht wird.
Zum Schluss wird die heruntergeladene Seite analysiert und daraus gewonnene Informationen abgespeichert. Eine Suchmaschine könnte zum Beispiel alle Wörter die auf der Seite zu finden sind speichern. Danach beginnt der Prozess von die nächste zu besuchende Seite wird bestimmt. [http://www.ijcttjournal.org/Volume13/number-3/IJCTT-V13P128.pdf]

2.4 Herausforderungen

Dieser Algorithmus erscheint zunächst recht simpel, allerdings ist das World Wide Web sehr groß, wodurch sich einige Probleme ergeben. Um eine akzeptable Performance zu erreichen, muss der Crawlvorgang hoch parallel ablaufen. In modernen Anwendungen ist es sogar üblich diesen Vorgang auf viele verschiedene Computer zu verteilen. In einem solchen Fall spricht man dann von "Distributed web crawling".
Durch die enormen Datenmengen wird es auch schwierig die Datenstrukturen des Crawlers im Arbeitsspeicher zu halten. Zum Beispiel die Liste mit bereits besuchten Seiten kann sehr schnell groß werden. Diese Daten auf ein sekundäres Speichermedium zu verlagern und dabei annehmbare Performance zu behalten ist keine leichte Aufgabe.

2.5 Cross Site Scripting

3 Architektur

4 Implementierung

4.1 Worker Plugin

4.2 Master Plugin

5 Anwendung

6 Fazit

Listings

Abbildungsverzeichnis

Tabellenverzeichnis