Matthias Tichy et al. (Hrsg.): Fachtagung Software Engineering,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 11

Automatisierte Identifikation von sicherheitsrelevanten Konfigurationseinstellungen mittels NLP

Patrick Stöckle[1] Theresa Wasserer[1] Bernd Grobauer[2] Alexander Pretschner[1]

Abstract: Dieser Vortrag wurde auf der 37. IEEE/ACM International Conference on Automated Software Engineering (ASE) präsentiert [St22]. Um Computerinfrastrukturen zu sichern, müssen die verantwortlichen Administratoren alle sicherheitsrelevanten Einstellungen konfigurieren und sichere Werte einsetzen. Hierbei stützen sie sich auf Sicherheitsexperten, die die sicherheitsrelevanten Einstellungen identifizieren und in Sicherheitskonfigurationsrichtlinien dokumentieren. Das Identifizieren der sicherheitsrelevanten Einstellungen ist allerdings zeitaufwändig und teuer, weshalb ihm oft keine Priorität beigemessen wird. Um dieses Problem zu lösen, nutzen wir aktuelle Verfahren der Computerlinguistik, um Einstellungen auf der Grundlage ihrer Beschreibung in natürlicher Sprache als sicherheitsrelevant zu klassifizieren. Allerdings zeigt unsere Evaluation, dass die trainierten Klassifikatoren nicht gut genug sind, um die menschlichen Sicherheitsexperten vollständig zu ersetzen sondern höchstens bei der Klassifizierung der Einstellungen helfen können. Durch die Veröffentlichung unserer gelabelten Datensätze und all unserer Modelle wollen wir Sicherheitsexperten bei der Analyse von Konfigurationseinstellungen unterstützen und weitere Forschung in diesem Bereich ermöglichen.

Keywords: Systemhärtung; Sicherheitskonfiguration; Computerlinguistik

Ein kritischer Teil der IT-Sicherheit in einer Organisation wie Siemens ist die sichere Konfiguration aller verwendeten Software. Hierfür müssen wir wissen, welche Konfigurationseinstellungen einer Software sicherheitsrelevant (SR) oder nicht-sicherheitsrelevant (NSR) sind. Wenn wir alle möglichen Einstellungen einer Software durchgehen, um alle SR Einstellungen zu finden, ist dies eine langwierige und zeitraubende Aufgabe. Daher lagern wir diesen Prozess an Organisationen wie das Center for Internet Security (CIS) aus. Das CIS bietet Sicherheitskonfigurationsrichtlinien für verschiedene Softwaresysteme, und wir können die CIS-Richtlinien verwenden, um unsere Software zu härten.

Es gibt jedoch Situationen, in denen dies nicht möglich ist: Erstens, wenn es keine CIS-Richtlinie für eine Software gibt, oder zweitens, wenn es ein neues Update der Software gibt und die CIS ihre Empfehlungen für das Update noch nicht veröffentlicht hat. Drittens, wenn wir in unserer Umgebung höhere Sicherheitsanforderungen haben und zusätzliche Regeln benötigen. Bei Siemens ist der dritte Anwendungsfall der wichtigste. In allen drei Fällen müssen die Sicherheitsexperten alle SR Einstellungen finden. Um das Auffinden der SR Einstellungen zu unterstützen und sicherzustellen, dass wir alle SR Einstellungen finden, wäre eine automatische Klassifizierung wünschenswert. Hierbei sind falsch-negative Ergebnisse (die Einstellung ist SR, aber wir klassifizieren sie als NSR) schwerwiegender

[1] Technische Universität München, Lehrstuhl für Software und Systems Engineering, Boltzmannstr. 3, 85748 Garching b. München, Deutschland vorname.nachname@tum.de
[2] Siemens AG, München, Deutschland bernd.grobauer@tum.de

als falsch-positive Ergebnisse. Ein Angreifer könnte eine falsch-negative, nicht gehärtete Einstellung ausnutzen, um das System anzugreifen. Ziel ist es also, dass Klassifikatoren falsch-negative Ergebnisse vermeiden, ohne jedoch jede Einstellung als SR zu kennzeichnen.

Unser laufendes Beispiel ist die automatische Härtung des Betriebssystems Windows 10 [SGP21] mit über 4500 Einstellungen. Außerdem gibt es eine CIS Windows 10-Richtlinie mit über 500 Regeln. Jede Regel bezieht sich dabei auf jeweils eine Einstellung. Im Mai 2021 veröffentlichte Microsoft das Update 21H1 für Windows 10 mit über 300 neuen Einstellungen. Daher mussten die Sicherheitsexperten bei Siemens nun die neuen SR Einstellungen identifizieren. Da das Durchsuchen nach SR Einstellungen langwierig und mühsam ist, forderten die Experten dafür automatisierte Unterstützung.

In diesem Artikel stellen wir unseren Vorschlag für diese Unterstützung vor. Wir verwenden verschiedene moderne Verfahren der Computerlinguistik (engl. natural language processing (NLP)), um automatisch zu klassifizieren, ob eine Einstellung SR ist. Um SR Begriffe zu identifizieren, verwenden wir bestehende Leitfäden. Anschließend nutzen wir die Beschreibungen der Einstellungen als Eingabe für die Klassifizierung.

Unser Forschungsbeitrag besteht aus drei Teilen: Erstens stellen wir den ersten Ansatz vor, der NLP-Techniken zur Identifizierung von SR Einstellungen verwendet. Zweitens ermöglichen wir durch die Veröffentlichung unserer gelabelten Datensätze, dass andere Forscher ihre Modelle ebenfalls auf ihnen trainieren können, um die beschriebene Problematik zu lösen. Drittens teilen wir den Code unserer Modelle, damit Sicherheitsexperten letztere bei der Richtlinienerstellung verwenden können.

Data Availability

Unser Datensatz ist öffentlich auf GitHub verfügbar.[3] Außerdem haben wir den Python-Code all unserer Modelle als Jupyter Notebooks auf Kaggle veröffentlicht.[4]

Literatur

[SGP21] Stöckle, P.; Grobauer, B.; Pretschner, A.: Automated Implementation of Windows-Related Security-Configuration Guides. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. ASE '20, Association for Computing Machinery, Virtual Event, Australia, 2021, URL: https://doi.org/10.1145/3324884.3416540.

[St22] Stöckle, P.; Wasserer, T.; Grobauer, B.; Pretschner, A.: Automated Identification of Security-Relevant Configuration Settings Using NLP. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. ASE '22, IEEE/ACM, Association for Computing Machinery, Rochester, MI, USA, 2022, URL: https://doi.org/10.1145/3551349.3559499.

[3] github/tum-i4/Automated-Identification-of-Security-Relevant-Configuration-Settings-Using-NLP

[4] Modell 1: kaggle/tumin4/sentiment-analysis, Modell 2: kaggle/tumin4/topic-modeling-and-latent-dirichlet-allocation, Modell 3: kaggle/tumin4/transformer-based-machine-learning