Automated Identification of Security-Relevant Configuration Settings Using NLP

1st Patrick Stöckle
Informatics 4
Technical University of Munich
Munich, Germany
patrick.stoeckle@tum.de

2nd Theresa Wasserer
Informatics 4
Technical University of Munich
Munich, Germany
theresa.wasserer@tum.de

3rd Bernd Grobauer
T CST
Siemens AG
Munich, Germany
bernd.grobauer@siemens.com

4th Alexander Pretschner
Informatics 4
Technical University of Munich
Munich, Germany
alexander.pretschner@tum.de

***Abstract*—To secure computer infrastructure, we need to configure all security-relevant settings. We need security experts to identify security-relevant settings, but this process is time-consuming and expensive. Our proposed solution uses state-of-the-art natural language processing to classify settings as security-relevant based on their description. Our evaluation shows that our trained classifiers do not perform well enough to replace the human security experts but can help them classify the settings. By publishing our labeled data sets and the code of our trained model, we want to help security experts analyze configuration settings and enable further research in this area.**

***Index Terms*—Hardening, Security Configuration, Natural Language Processing**

I. Introduction

A critical part of the IT security in an organization such as Siemens is the secure configuration of all used software [1]. Here, we need to know which configuration settings (from here on *settings*) of a software are security-relevant (SR) or not security-relevant (NSR) (see Fig. 1). We denote the classification predicate with p. Going through all possible settings Γ_θ of a software θ and classifying whether a setting $\gamma \in \Gamma_\theta$ is SR ($p(\gamma)$) to collect all SR settings $\Gamma_\theta^{SR} = \{\gamma | \gamma \in \Gamma_\theta : p(\gamma)\}$ is a tedious and time-consuming task. Thus, we outsource this process to organizations such as the Center for Internet Security (CIS). They provide a set of security-configuration guides $\mathbb{S}_{CIS}$ (from guides), and we use a CIS guide $\mathcal{S}_{CIS,\theta} \in \mathbb{S}_{CIS}$ to harden our software θ.

However, there are situations in which we cannot use a guide: First, if there is no CIS guide for software. Second, if there is a new update of the software and the CIS has not published its recommendations for the update yet. Third, we have higher security requirements in our environment and need additional rules. At Siemens, the third use case is the most important. In all cases, the security experts need to find all SR settings. To support finding the SR settings and assure that we find all SR settings, we use automated classification. False negatives, i.e., γ is SR, but $\neg p(\gamma)$, are more severe than false positives, because an attacker might use a non-hardened SR setting to attack the system. Classifiers should therefore avoid false negatives without labeling every setting as SR.

Our running example will be the hardening of the Windows 10 OS (in the following W10) with over 4500 settings ($|\Gamma_{W10}| > 4500$). Furthermore, there is a CIS W10 guide with over 500 rules, i.e., $|\mathcal{S}_{CIS,W10}| \approx 500$. Every rule r *targets* a setting γ, which we denote with $\varpi(r) = \gamma$ and this setting is unique, i.e., $\varpi_{CIS,W10}$ is injective and $|\Gamma^{SR}_{CIS,W10}| \approx 500$. In May 2021, Microsoft released the 21H1 update for W10 including over 300 new settings. The security experts at Siemens now needed the new SR settings, i.e., $\Gamma^{SR}_{W10'} \setminus \Gamma^{SR}_{W10}$.

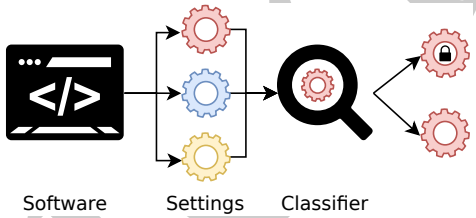

Fig. 1. Identification of security-relevant settings.

In this article, we present our solution to this problem. We use various state-of-the-art natural language processing (NLP) to model p and classify automatically whether a setting is SR. We use the settings' descriptions in natural language as input and existing guides to identify SR terms.

Our contribution is threefold. First, we present, to our knowledge, the first approach to use NLP techniques to tackle the identification of SR settings. Second, we publish our labeled data sets[1] so that other researchers can train their models on them to solve the described problem. Third, we share the code of our models on Kaggle so that security experts can use them when they create guides.

II. Data Set Creation

As we have only several thousand settings, we need data-efficient techniques and a labeled data set. For a given software θ, we first needed all settings Γ_θ. Second, we needed the descriptions $\mathcal{D}$ describing their function and purpose in natural language. Third, we needed to label each setting γ as SR or NSR. One can see the three steps depicted as arrows in Fig. 2.

As modern software can easily have thousands of settings [2], it is beneficial if we automate the three steps. Therefore, we choose W10 for our proof of concept. In W10, the Administrative Templates (ATs) define most settings. Microsoft

[1] github/tum-i4/ASE2022

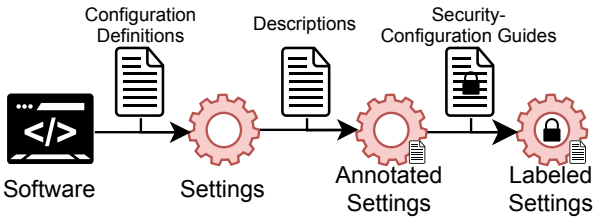

Fig. 2. Data set creation.

```
- setting: Control Panel \ Personalization \ Force a
↪  specific background and accent color
  description: "Forces Windows to use the specified colors
  ↪  for the background and accent. The color values ..."
  is_security_relevant: false
- setting: Control Panel \ Personalization \ Prevent
↪  enabling lock screen camera
  description: "Disables the lock screen camera toggle
  ↪  switch in PC Settings and prevents a camera from being
  ↪  invoked on the lock screen..."
  is_security_relevant: true
```

Listing 1. Labeled settings for Windows 10, version 1909.

stores these configuration definitions in so-called ADMX files, and we can automatically generate the set of settings Γ_{W10} out of them. Furthermore, the ATs include all the description texts in different languages in so-called ADML files. For our proof-of-concept implementation, we limited ourselves to English. However, one could also investigate whether another language, e.g., Hindi, is better suited to identify SR settings. We parse the set of descriptions $\mathcal{D}$ from the ADML files and join the definitions with the descriptions using a shared identifier to a set of settings together with their description, i.e., $\mathcal{L} = \{(\gamma, d) \,|\forall \gamma \in \Gamma_{W10} : \exists d \in \mathcal{D} : id(d) = id(\gamma)\}$.

To automate the third step, we need ground truth whether a setting is SR or NSR. Here, we used CIS guides and especially the W10 guide $\mathcal{S}_{CIS,W10}$. Our assumption is that a setting γ is SR if and only if there is a rule r in the guide $\mathcal{S}_{CIS,W10}$ that is targeting this setting γ, i.e., $p_{CIS,W10}(\gamma) \iff \exists r \in \mathcal{S}_{CIS,W10} : \varpi(r) = \gamma$. We also use Siemens guides to evaluate our classifiers on guides from different organizations. In both cases we can automatically retrieve the set of all rules $\mathcal{S}_{CIS,W10}$ and extract for each rule the targeted setting γ, i.e., we have a list $\mathcal{K} = \{(r, \gamma = \varpi(r)) \,|\forall r \in \mathcal{S}_{CIS,W10}\}$. In the end, we can construct our labeled data set by joining L and K, i.e., $\{(\gamma, d, (\exists\,(r, \gamma') \in \mathcal{K} : \gamma = \gamma')) \,|\, (\gamma, d,) \in L\}$ and mark for every rule r the setting γ as SR that r targets. The result are the labeled settings with their descriptions (see Lst. 1).

The input of our implementation is the ADMX/L files of the ATs and a guide in the XML-based XCCDF format. Microsoft regularly updates the ATs. Thus, there are different versions of the ADMX/L files, e.g., *1909* or *21H2*. We uploaded different variants into our repository.

III. Sentiment Analysis

We make a binary decision if a setting is SR based on its text. Therefore, our first idea was to use sentiment analysis and lexicon-based approaches in particular to solve our problem.[2] Due to the descriptions' formal language, spell correction was not necessary. First, we considered basing our classification on part-of-speech (POS) tags. However, we found SR words distributed over all groups. The same holds for high frequency, as most frequent words in the SR descriptions also occur frequently in NSR ones. We also extracted words that only occurred in SR descriptions. Several words, e.g., "attacker", showed a relation to a security aspect, but filtering for words with a frequency of greater than five left only 12 words identifying hardly all SR settings. As we could see subjects repeatedly mentioned in the descriptions, we used the term frequency–inverse document frequency (tf-idf) algorithm instead of the frequency. To reduce the words to the relevant ones, we set the threshold to 0.5 and ended up with 141 words. However, only a few came from the security domain, and so we combined the descriptions with the rationales (text explaining why one should configure a setting) of CIS rules. In the end, we could find the 80 SR words as depicted in Fig. 3. Nevertheless, these words also occur frequently in the NSR descriptions, and we constructed based on tf-idf a counterpart set of words that mark NSR settings, e.g., "color", but not enough to prevent a high number of false positives. The same problem occurred when we used n-grams or named entity recognition: The entity represents a particular case referring only to a few SR settings and, therefore, contributes little to the entire classification. Alternatively, the n-gram also appears within the NSR descriptions and therefore could lead to NSR descriptions being classified additionally as SR. With these findings, it becomes clear that the lexicon-based approaches lead to a large percentage of false positives, making them unsuitable in our case. SR words do not necessarily follow one after another. Therefore, increasing the size of n-grams is not suitable as well. Our insight here was that classifying the settings directly as SR performed not as good as expected.

IV. Topic Modeling

Next, we trained a Latent-Dirichlet-Allocation (LDA) topic model to determine topics within the SR descriptions.[3] The intuition behind the LDA is that a document typically not only treats one single topic but can be rather seen as a mixture of multiple latent topics. Once we trained the model on the SR descriptions, we can calculate the probability of each description referring to a security topic. If the probability exceeds a certain threshold, we classify the description as SR.

We tokenized the descriptions, removed stop words, selected only words between 2 and 16 characters, and built the lemma and the word stem. Of the 300 most frequent stems, we manually created a list of words that are irrelevant for the security aspect, e.g., "kilobyte", or not specific to one topic, e.g., "password". We trained the LDA model on the entire collection of SR descriptions as we cannot build the topics properly on a partial data set. For the evaluation, we tested the classifier on other data sets, e.g., labeled according to

[2]Code: kaggle/tumin4/sentiment-analysis

[3]Code: kaggle/tumin4/topic-modeling-and-latent-dirichlet-allocation

Fig. 3. Security-relevant words.

another guide for the same system variant or on data sets for different system versions and variants. We built a dictionary containing all words that remained after the preprocessing for the LDA model and assigned each word an id. The tf-idf-feature representation lists the ids with the tf-idf scores.

To optimize the performance of the model, we had to set the following parameters: amount of topics; passes, i.e., how often the algorithm iterates over the entire corpus to optimize the topic allocations; α, i.e., a priori assumption about the document-topic distributions; whether we would use per-word-topics; the probability threshold. We achieved the best results with nine topics and four passes on our training data. For α, we learned an asymmetric a priori probability distribution from the description corpus and used per-word topics, i.e., we calculated a list of most likely topics for each word. Finally, we set the probability threshold to 70%.

The LDA topics are not as descriptive as the topics based on the CIS categories. We would need to draw semantic relationships between the words of one CIS topic, e.g., typical key words relating to *Data Protection* are "send", "collect" etc., but the words "send" and "collect" do not necessarily share common context words. The knowledge about a semantic relationship between them is necessary to relate them to the same topic. However, LDA is not capable of doing this.

For the classification, we preprocess all descriptions, transform them into the tf-idf representation, and the model returns the probabilities that the description refers to a topic. The description in Lst. 2 has a strong relation of $\approx 80\%$ to Topic 3. If any topics are above our threshold, we classify the model as SR. Although the LDA obtained better results than the SA, we discovered several problems that we discuss in Sec. VI.

V. Transformer-based Machine Learning

The promising, but not convincing results of the topic model (see Sec. VI) and the recent success of deep learning models in standard NLP tasks motivated us to train an additional model using transformer-based machine learning, namely BERT [3].

For our model, we extended the sequence size to 512 tokens as the average input length in our data set was higher than the standard sequence length. Before training the BERT model, we removed the hive duplications we recognized during our LDA experiments (see Sec. VI) from our data set. We used a combination of over- and undersampling to deal with our imbalanced data set and the low number of SR settings. We altered all hyperparameters, e.g., batch size or k-fold, in reasonable ranges of values.[4] We achieved the best results with

[4]Code: kaggle/tumin4/transformer-based-machine-learning

```
description: "Windows Components. AutoPlay Policies. Turn
↪  off Autoplay.[...]"
Topic probabilities: [(0, 0.025299275), (1, 0.02649991), (2,
↪  0.025674498), (3, 0.79593724), ..., (8, 0.026722105)]
```

Listing 2. Topic prediction.

a small BERT model with 2 hidden layers, a hidden size of 128, 2 attention heads, a batch size of 32, a dropout rate of 0.2, 20 epochs, and 5-fold cross-validation. Overfitting was a noticeable problem with BERT's storing capabilities and our limited data set. Thus, we added a dropout layer and used the AdamW optimizer with a decreasing learning rate.

VI. Evaluation & Discussion

We evaluated the performance of the LDA and the BERT model on different data sets. The correct classification of SR settings is critical, and, thus, we focus strongly on the recall. A *useful* classifier in our context should therefore score a recall close to 100% without producing too many false positives.

Our four main research questions were:

- RQ1 What is the highest recall a useful LDA-based classifier trained on the SR descriptions can achieve?
- RQ2 Can we use our LDA-based approach as a classifier p for Siemens and CIS guides? Does it make sense to train publisher classifiers, i.e., p_{SIE} and p_{CIS}?
- RQ3 Which recall/precision can our BERT-based classifier achieve on the unseen data? Could we achieve a sufficiently high recall of nearly 100% to replace the manual analysis by the security experts?
- RQ4 What are the main reasons for false negatives?
- RQ5 What are the main reasons for false positives? What would we need to avoid these problems in the future? Can we find settings that are not part of the CIS guides but are SR judged on the description?

Tab. I shows the classification results of the best performing LDA-based classifier on W10 and Windows Server 2016 (WS16) data sets. Our data sets are imbalanced as we have only a tiny percentage of SR settings. Therefore, we choose instead of the *normal* accuracy the balanced accuracy (BA) [4]. With LDA, we achieve a recall value of 91% on the W10 *1909* data set with a 92% BA on our training set answering **RQ1**. To make this clear, we used all SR descriptions to identify SR topics in the descriptions. The positive result here is that we have very few false positives, i.e., the LDA-based model can differentiate between the topics of the seen SR descriptions and the unseen NSR descriptions. One might now ask why we still missed 23 SR settings and could not meet our goal of $\approx 100\%$ recall. When we investigated the 23 false negatives, we could see that the LDA-based model could not take the context of a word into account and lacked the semantical understanding necessary to classify the settings correctly. Thus, we could train LDA-based classifiers with sufficient recall of $\approx 100\%$, but those resulted in large numbers of false positives. The context-sensitivity and semantical understanding motivated our BERT-based classifier as an improvement over LDA.

TABLE I: Classification results of the LDA-based classifier.

OS θ	Guide $\mathcal{S}$	Settings $\lvert\Gamma_\theta\rvert$	# of SR $\lvert\Gamma_\theta^{SR}\rvert$	Recall (%)	# classified as SR	BA (%)
W10 *1909*	CIS	2688	246	**91**	406	**92**
W10 *1803*	CIS	2576	238	89	382	91
WS16	CIS	2430	156	88	355	89
W10 *1909*	Siemens	2688	303	59	407	73
WS16	Siemens	2430	192	80	355	85

The LDA-based classifier has a high recall and BA values on all CIS guides ($\Delta_{recall} \leq 3pp$, $\Delta_{BA} \leq 1pp$), lower values on Siemens WS16, and performs bad on Siemens W10. The results are relatively stable between different W10 versions and W10/ WS16. Therefore, we assume that the CIS is consistent within its classification of settings based on their description. We know that the security experts used the CIS WS16 as a basis for the Siemens WS16 guide explaining the relatively good performance. After seeing the bad results on Siemens W10, we investigated the difference between the CIS and the Siemens guide. We found many settings targeted only in one guide but not in the other; even if such a setting was in the training data, the classifier could not predict it correctly. With this in mind, training a global p does not make sense, but a publisher classifier p_{CIS} is useful. With the limitation to two publishers and Windows-based OSs, we could answer **RQ2**.

Tab. II shows the result of our BERT-based classifier. As we present the first automated classification approach, we compare it with the best-performing dummy classifier, i.e., randomly classifying $x\%$ of the settings as SR, as a baseline. Although the dummy classifier has a better recall, its precision is only 11%, thus producing too many false positives. In precision and F1, the BERT classifier outperforms the baseline by 30pp respectively 24pp. However, our classifier misses more than half of the SR settings in the test data. Tab. III shows how our classifier performed on our other data sets. As the data sets share settings, we made sure that we used in the test data set no settings that we previously used in training. Nevertheless, although trained on CIS W10 *1803*, our classifier performs best on the W10 *1909* with a 60% recall and 46% precision. Our explanation for the good result on the newer version is that CIS marks some new settings as SR and changes some old settings from NSR to SR. However, CIS's updates to their guides make them more consistent, at least to what the classifier has learned from the descriptions. As we want to use the classifier in this use case of a new software version, we see this number as a basis for the future, but in the end, we are still far away from 100% recall. Therefore, we cannot replace the manual analysis of security experts, and we could not fulfill the second part of **RQ3**.

Going through the false negatives of our classifiers, we identified four main classification problems. Unique settings, short descriptions, descriptions with a vocabulary spread over multiple topics, and linked settings. An example of the first group is the setting *Enable Windows NTP Server*. The targeting rule's rationale state that it is SR for the validity of timestamps used, e.g., in authentication procedures. However, the setting's description neither includes "clock" nor "synchronization" and neither the LDA nor the BERT-based models label it as SR. An example of the second group is *Allow Cloud Search*. Here, the description only consists of one sentence, and we cannot assess the topic. The third group is settings whose description is SR according to two or more topics. However, no single probability is over the threshold. Our LDA classifier assigns the setting *Allow user control over installs* to 51% to Topic 3 and 35% to Topic 4. Thus, we classify it wrongly as not SR. The fourth group is settings that often occur in other settings' descriptions. Several NSR settings mention the SR setting *Prevent enabling lock screen slide show*. Thus, the classifier deducts that this setting is NSR. Linked settings also cause false positives if multiple SR settings mention a NSR setting. The four presented groups answer **RQ4**.

Next, we went through the classifiers' false positives. We could identify four groups of common problems: Overruled settings, hive duplication, correction candidates, and context-specific meanings. The first group is settings with SR descriptions. Nevertheless, they become ineffective if another setting is enabled or disabled. The setting *MS Support Diagnostic Tool \Configure execution level* states that it takes no effect if the "scenario execution policy" is configured. We would need a semantic model of the settings' relations to avoid such false positives. The second group is settings existing both in the Computer **and** the User hive. They usually have the same description, but the Computer setting has precedence over the User setting. Thus, the CIS marks the Computer setting as SR and the User as NSR. However, there are settings like *Always install with elevated privileges* stating that we should enable this policy in both hives. Thus, we needed to know which settings are essential on both hives to prevent these false positives. Since we trained the BERT-based model after the LDA evaluation, we removed this problem there. The third group is settings that indeed seem SR, e.g., because we found similar written SR descriptions. One example here is the *Prohibit non-administrators from applying vendor signed updates* setting. We do not know whether the CIS overlooked this setting or deliberately chose to omit this setting, e.g. because the impact is meager. The fourth group is settings that have words that are only in some contexts SR, e.g., *Prevent Application Sharing in true color*. "Application" and "Sharing" appear in many SR descriptions, but here, this color setting

TABLE II: Performance of the BERT and the dummy classifier on CIS Windows 10, version 1803.

Classifier	Recall	Precision	F1
BERT	0.44	**0.41**	**0.42**
Uniform	**0.54**	0.11	0.18

TABLE III: Classification results of the BERT-based classifier.

θ	$\mathcal{S}$	Recall	Precision	F1
W10 *1803*	CIS	0.44	0.41	0.42
W10 *1909*	CIS	**0.60**	**0.46**	**0.52**
WS16	CIS	0.49	0.28	0.35
W10 *1909*	Siemens	0.48	0.33	0.39
WS16	Siemens	0.48	0.43	0.45

is NSR. To filter out those rules, we would need to take the context of the words more into account. Only the third group provides candidates for the new rule. However, as we do not know whether the CIS forgot them or omitted them, we cannot answer the second part of **RQ5**.

Our evaluation shows that our classifiers could detect many settings correctly, but not enough for our use case. The main problem with the descriptions is that they should inform a user about the setting not a security expert about the setting's security implications. Our findings suggest that NLP techniques like the LDA topic model alone cannot replace the security experts and their domain knowledge in this task.

VII. Related Work

Research about configuration is an essential part of the software engineering [5], [6] as well as the security domain [1], [7]. Stöckle et al. demonstrated how one could use NLP to implement guides efficiently [8]. Most relevant for the problem of identifying SR settings is sentiment analysis, where we classify documents as being positive or negative, depending on the expressed sentiment [9]. We limited ourselves to SA approaches that do not need much data. Qiu et al. start from a seed lexicon containing a few meaningful features and expand it via the exploitation of a specific characteristic [10]. They use dependency rules to extract features from the data set and add words iteratively to the seed lexicon that occur in a particular dependency relation to a word from the seed lexicon. The lexicon-based approaches build on the assumption that specific words express either one of the opposing sentiments, i.e., *good* is characteristic for the positive and not for the negative sentiment. However, our evaluation shows that the assumption does not hold for the vocabulary of settings' descriptions.

VIII. Conclusion

We constructed labeled data sets for security-relevant configuration settings. We motivated our decision to train an LDA topic model and a BERT-based model to classify SR settings. Our evaluation could achieve good results on the different data sets. The required recall of close to 100 % due to the security implications could not be met. Therefore, our approach cannot replace security experts going through the settings. Nevertheless, it can provide good support for them. We published our labeled data sets so that other researchers can use them for training better models in the future.

Based on our results, we propose several improvements for the configuration hardening: First, we need data sets with settings, descriptions, and security relevancy for more systems, e.g., Linux-based systems or applications. Second, software vendors should improve the settings' descriptions and add security implications. Third, it would be better if the software vendors tag all SR settings directly in a machine-readable way, e.g., in the ADMX, so that we would not need NLP techniques to extract it from the natural language texts. Fourth, the software vendors could provide machine-readable security-configuration guides, e.g., in XCCDF or Scapolite, along with their software. With these guides, security-aware users could harden their systems directly during the installation and make them secure from day one.

References

[1] C. Dietrich, K. Krombholz, K. Borgolte, and T. Fiebig, "Investigating System Operators' Perspective on Security Misconfigurations," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 1272–1289. [Online]. Available: https://doi.org/10.1145/3243734.3243794

[2] J. A. Pereira, M. Acher, H. Martin, J.-M. Jézéquel, G. Botterweck, and A. Ventresque, "Learning software configuration spaces: A systematic literature review," *Journal of Systems and Software*, vol. 182, p. 111044, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221001412

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[4] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The Balanced Accuracy and Its Posterior Distribution," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 3121–3124.

[5] R. Bhagwan, S. Mehta, A. Radhakrishna, and S. Garg, "Learning Patterns in Configuration," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 817–828.

[6] K. Nguyen and T. Nguyen, "GenTree: Inferring Configuration Interactions using Decision Trees," in *36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 1232–6.

[7] P. Stöckle, I. Pruteanu, B. Grobauer, and A. Pretschner, "Hardening with Scapolite: A DevOps-Based Approach for Improved Authoring and Testing of Security-Configuration Guides in Large-Scale Organizations," in *Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 137–142. [Online]. Available: https://doi.org/10.1145/3508398.3511525

[8] P. Stöckle, B. Grobauer, and A. Pretschner, "Automated Implementation of Windows-Related Security-Configuration Guides," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 598–610. [Online]. Available: https://doi.org/10.1145/3324884.3416540

[9] F. Hemmatian and M. K. Sohrabi, "A survey on classification techniques for opinion mining and sentiment analysis," *Artificial Intelligence Review*, vol. 52, no. 3, pp. 1495–1545, Oct 2019. [Online]. Available: https://doi.org/10.1007/s10462-017-9599-6

[10] G. Qiu, B. Liu, J. Bu, and C. Chen, "Opinion Word Expansion and Target Extraction through Double Propagation," *Computational Linguistics*, vol. 37, no. 1, pp. 9–27, 03 2011. [Online]. Available: https://doi.org/10.1162/coli_a_00034