A Rusty Link in the AI Supply Chain: Detecting Evil Configurations in Model Repositories

Ziqi Ding[1], Qian Fu[2], Junchen Ding[1], Gelei Deng[3], Yi Liu[4], Yuekang Li[1]

[1]*UNSW Sydney, Australia, {ziqi.ding1, jamison.ding, yuekang.li}@unsw.edu.au*
[2]*Commonwealth Scientific and Industrial Research Organisation (CSIRO) Data61, Australia, qian.fu@data61.csiro.au*
[3]*Nanyang Technological University, Singapore, gelei.deng@ntu.edu.sg*
[4]*Quantstamp, yi009@e.ntu.edu.sg*

Abstract—**Recent advancements in large language models (LLMs) have spurred the development of diverse AI applications—from code generation and video editing to text generation. However, AI supply chains such as Hugging Face, which host pre-trained models and their associated configuration files contributed by the public, face significant security challenges. In particular, configuration files—originally intended to set up models by specifying parameters and initial settings—can be exploited to execute unauthorized code, yet research has largely overlooked their security compared to that of the models themselves. In this work, we present the first comprehensive study of malicious configurations on Hugging Face, identifying three attack scenarios (file, website, and repository operations) that expose inherent risks. To address these threats, we introduce CONFIGSCAN, an LLM-based tool that analyzes configuration files in the context of their associated runtime code and critical libraries, effectively detecting suspicious elements with low false positive rates and high accuracy. Our extensive evaluation uncovers thousands of suspicious repositories and configuration files, underscoring the urgent need for enhanced security validation in AI model hosting platforms.**

Index Terms—**AI Supply Chain, LLM, Configuration**

1. Introduction

Developing applications that leverage artificial intelligence is increasingly critical. The advent of large language models (LLMs) [1], [2] has spurred the emergence of applications in domains such as code generation [3], video editing [4], and text generation [5]. Moreover, AI supply chains (AISCs) such as Hugging Face [6] expedite development by hosting pre-trained models for reuse, thereby empowering researchers and developers. However, since these platforms provide out-of-the-box access to various AI models contributed by the public—similar to packages on PyPI [7]—the security of the hosted models is not assured. In particular, configuration files, which are originally intended to set up models by specifying parameters and initial settings, can be exploited by malicious actors to execute unauthorized code if not properly validated.

Unfortunately, research on the security of AI model hosting platforms within AISCs has been limited, particularly with regard to the configuration files of model repositories. In contrast, most pioneering studies have focused on the security of the AI models themselves, addressing issues such as backdoor attacks [8], adversarial attacks [9], and embedded code poisoning attacks [10]. However, AI model hosting platforms such as Hugging Face currently lack tools to alert users when potentially malicious configuration files are loaded. Therefore, it is imperative to validate the security of these configuration files by scrutinizing their constituent elements and assessing their associated risks.

There are two main challenges in identifying malicious configurations in AI models. First, the semantic complexity of diverse configuration files poses a significant challenge: different model frameworks employ configuration files with varied structures, contents, and dependencies, each relying on distinct imported packages. This heterogeneity complicates the development of a unified and extensible solution. Second, configuration files do not inherently present static risks; instead, defenders must analyze not only the file itself but also how its contents interact with the underlying code, potentially introducing vulnerabilities.

In this work, we have identified three attack scenarios—file operations, website operations, and repository operations—in which attackers exploit configuration files to launch attacks. Based on these observations, we have developed a tool, CONFIGSCAN, that leverages LLMs to analyze configuration files in the context of their associated runtime code, as described in the accompanying README, and the critical libraries they utilize. CONFIGSCAN is designed to detect suspicious configuration files by identifying potentially malicious elements, such as unusual keys related to file paths, websites, or repository IDs.

Our research makes the following key contributions

- **New Finding:** We present the first comprehensive study on malicious configurations in Hugging Face, highlighting the associated security risks in AISCs. Through our rule-based analysis, we identified 13,091, 1,324, and 35,761 suspicious repositories containing suspicious elements

related to file, website and repository operation risk, respectively.

- **New Approach:** We introduce CONFIGSCAN, an LLM-powered tool designed to detect malicious configurations with high accuracy while minimizing false positives.
- **Comprehensive Evaluation:** We evaluated CONFIGSCAN on 1,000 samples, successfully identifying two malicious configuration files while resolving 998 false positives compared to rule-based analysis.

2. Related Work

Security of AI Supply Chains. As AISCs [6] have advanced, security concerns have become more prominent. Compared to traditional software supply chains, AISCs involve a larger number of components, creating more opportunities for attackers to inject malicious payloads [11]. Several studies [12], [10], [13], [14] have identified security vulnerabilities in AISCs. Zhou [12] explored the risks of insecure deserialization, specifically focusing on the dangerous use of `pickle.load` in AISCs. Similarly, Walker and Wood [13] examined the runtime risks posed by machine learning models using Keras files as an example. Zhao [10] analyzed over 705K models and 176K datasets, uncovering 91 malicious models and 9 dataset load scripts, highlighting significant security risks in publicly accessible machine learning resources. Zhu [14] conducted an in-depth analysis of TensorFlow APIs, demonstrating how 20 exploited APIs can pose various runtime risks. However, these studies primarily focus on code poisoning attacks and overlook the potential risks associated with configuration file abuses in AISCs.

Secure AI Supply Chains. To secure AISCs, prior work has focused on addressing security concerns across various domains. Trail of Bits developed Fickling [15], a tool that functions as a decompiler, static analyzer, and bytecode rewriter for pickle files. ModelScan [16] is another tool that assesses the threat of model serialization attacks using static analysis, supporting various machine learning libraries such as PyTorch, TensorFlow, Keras, and traditional machine learning libraries. Building on this, Zhao [10] introduced MalHug, a tool tailored for Hugging Face to analyze models and datasets. In contrast, Zhu [14] developed an analysis tool based on large language models (LLMs) and rule-based approaches. However, these tools primarily focus on securing the model itself. In this paper, we shift the focus to the security of configuration files.

3. Threat Model

To systematically analyze configuration-based attacks, we have developed a comprehensive threat model based on several key assumptions. Firstly, most users are unaware of the security risks associated with configuration files in AISCs, primarily because existing tools do not alert users to potential risks when executing code from a README associated with a configuration file. Additionally, the complexity of configuration files and the limited attention and

research in this area mean that many potential attack vectors remain unexplored. As a result, attackers can craft malicious configuration files with little concern for detection by AISC tools. Unlike code injection attacks, these attacks can employ sophisticated techniques to evade detection. For instance, an attacker might create a third-party Python library along with a custom configuration file to deceive users into executing it. Alternatively, they could exploit an official Python library (e.g., `retrieval_rag.py` in Transformers) on Hugging Face that includes `pickle.load`, combining it with a malicious configuration file to execute harmful actions.

4. Methodology

Algorithm 1 The Algorithm of ConfigScan

1: **Input:** README FILE R, CONFIG FILE C, Vulnerability LLM $VLLM$, Score LLM $SLLM$
2: **Output:** Vulnerabilities $\mathbb{V}$, POC $\mathbb{P}$, Confidence Score S, Process of Analysis $\mathbb{A}$, Reflection R
3: —-*Rule-based Analysis Starting*—-
4: $\mathbb{K}_\mho \leftarrow$ Json_Data_Parsing(C)
5: $\mathbb{K}_\mathbb{R} \leftarrow$ README_Summarization(R)
6: —-*Initial LLM Analysis Starting*—-
7: $\mathbb{V}, \mathbb{P}, \mathbb{A} \leftarrow VLLM(\mathbb{K}_\mho, \mathbb{K}_\mathbb{R}, C)$
8: $S, R \leftarrow SLLM(\mathbb{V}, \mathbb{P}, \mathbb{A}, C)$
9: —-*In-depth LLM Analysis Starting*—-
10: **if** $S < 0 \land S > 1$ **then**
11:     **return** None
12: **for** each V in $\mathbb{V}$ **do**
13:     $S_\mathrm{Pre} \leftarrow S$
14:     COUNT $\leftarrow 0$ and NUM $\leftarrow 0$
15:     **if** NUM $< N$ **then**
16:         $V, \mathbb{P}, \mathbb{A} \leftarrow VLLM(\mathbb{K}_\mho, \mathbb{K}_\mathbb{R}, C, V)$
17:         $S_L, R \leftarrow SLLM(\mathbb{V}, \mathbb{P}, \mathbb{A}, C, V)$
18:         **if** $S_L < S_\mathrm{Pre}$ **then**
19:             COUNT $\leftarrow$ COUNT $+ 1$
20:         **else**
21:             $S_\mathrm{Pre} \leftarrow S_L$
22:             COUNT $\leftarrow 0$
23:         **if** COUNT $> n$ **then**
24:             **return** $\mathbb{V}, \mathbb{P}, \mathbb{A}, C$
25: **return** $\mathbb{V}, \mathbb{P}, \mathbb{A}, C$

Inspired by vulnhuntr [17], which leverages the power of LLMs to detect vulnerabilities on GitHub, we introduce CONFIGSCAN, a hybrid tool that combines rule-based and LLM-based approaches. CONFIGSCAN is the first scanning tool specifically designed to analyze configuration files on Hugging Face. As illustrated in Figure 1, the tool comprises three key components: Suspicious Configuration Analysis, Readme Summarization, and LLM-based Analysis.

4.1. Suspicious Configuration Analysis

Suspicious Configuration Analysis is the initial step of our CONFIGSCAN, which utilizes a rule-based approach to explore unknown configuration files by analyzing their keys
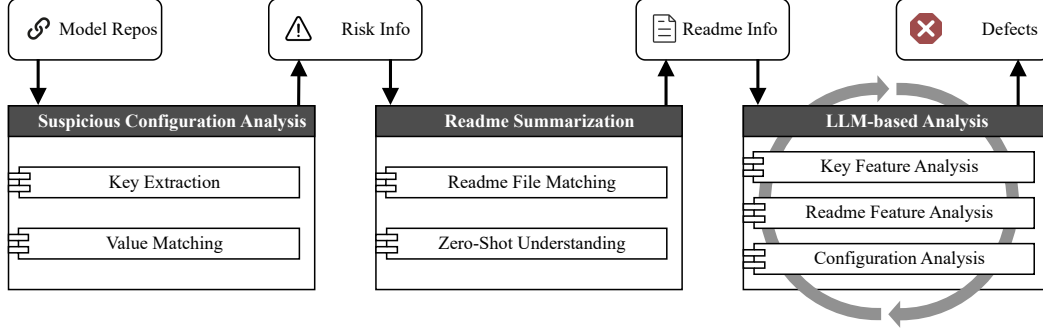
Figure 1. Workflow of ConfigScan

and values. In this phase, we focus on three potential risk scenarios that may arise, which we detailed below.

File Operation Risk. Many Hugging Face configuration files include file-related keys meant for secure file uploads and model execution. However, attackers can exploit these keys to load unauthorized or harmful files, creating security risks. Additionally, they may use these keys to read or write files, potentially accessing sensitive information from the user's system.

Website Operation Risk. Found in older Hugging Face configuration files, this feature was originally designed for model execution and weight extraction but introduces security risks by allowing access to unknown websites [18]. Visiting such websites can expose the user's IP address and lead to targeted attacks. Moreover, it may involve downloading third-party files or enabling remote code execution (RCE), which can harm the target system.

Repository Operation Risk. This vulnerability represents a common security flaw in Hugging Face, enabling attackers to launch customized attacks. For example, certain modules in the Transformers library require file inputs to download packages. Many configuration files contain the key (`_name_or_path`), which allows users to seamlessly fetch remote files. Attackers can exploit this by modifying the key to redirect users to malicious models hosted in unauthorized repositories. This can result in the distribution of poisoned models or backdoored scripts, ultimately compromising system integrity.

Therefore, to address these issues, we implemented a rule-based system to extract key-value pairs containing sensitive terms related to files and URLs (e.g., "url," "file"). Using regular expressions, we match keywords like "url" or "file" in the keys, and patterns such as "http," "root," or paths starting with "/" in the values, which may indicate potential risks. This approach helps identify and flag critical data that could be a security threat. Additionally, we check the value of `_name_or_path` to see if it matches the repository name, which may indicate the configuration file links to a suspicious or malicious repository. This also serves as a rule-based tool for testing configuration files.

4.2. README Summarization

README provides critical information for users to run code and execute models in AISC. In this section,

we first identify the file type of the README and then extract key information from it. In Hugging Face, many README files contain not only code but also extensive explanations. Therefore, we instruct the LLM to focus exclusively on content relevant to running code. Additionally, not all Hugging Face repositories include a README file. In such cases, we link the repository name to assist the LLM in inferring the library used by the repository. Our analysis relies on zero-shot learning, and we have crafted a specialized prompt for the LLM to address the following questions: (1) What is the aim of this project? (2) Does the running code in the README involve any unknown network connections? (3) Does the running code in the README involve local file access or other file operations? This approach enables us to extract crucial information regarding the project's objectives and its overall workflow.

4.3. LLM-based Analysis

As shown in Algorithm 1, this part consists of two distinct components: Initial LLM Analysis and In-depth LLM Analysis.

Initial LLM Analysis. In this section, CONFIGSCAN begins with a methodical analysis, starting with an initial examination of key features within the configuration file, followed by generating a detailed summary of the README. This phase is essential for establishing a foundational understanding of the project's structure and functionality. Based on these preliminary analyses, CONFIGSCAN then advances to a thorough evaluation of the configuration file. During this phase, CONFIGSCAN applies advanced heuristics to identify potential vulnerabilities, assess the integrity of the configuration settings, and flag any anomalies that could pose a security risk. The results of this evaluation are presented in a clear, actionable format, including a Proof of Concept (POC), a detailed list of discovered vulnerabilities, and a step-by-step breakdown of the analysis process. These processes are conducted by an LLM, which analyzes the vulnerabilities. Additionally, another LLM is used to assist in scoring and reflecting on the previous analysis. This helps increase confidence in the identified vulnerabilities and generates reflective statements to guide further analysis.

In-depth LLM Analysis. After completing the Initial Analysis, CONFIGSCAN progresses to the In-depth Analysis phase. Unlike the broader Initial Analysis, the In-depth

Analysis focuses specifically on a vulnerability identified during the initial phase. This targeted approach allows CONFIGSCAN to investigate the specific issue in greater detail. The decision to conduct multiple levels of analysis is intentional, aiming to reduce hallucinations or inaccuracies that might arise from a single-pass evaluation. By breaking the process into stages, CONFIGSCAN ensures that each vulnerability is verified, cross-checked, and assessed with greater precision, minimizing the likelihood of false positives or misinterpretation. In the In-depth Analysis, the identified vulnerability is examined independently, emphasizing its configuration context, potential impact, and root cause. By leveraging LLM-based Score Analysis, the tool performs a comprehensive evaluation of vulnerabilities and iteratively refines its assessments. Each score is generated based on criteria such as severity and exploitation likelihood and is re-evaluated iteratively. If the confidence score does not increase within the predefined iteration limit, the highest recorded confidence score is selected as the final result. Combining this multi-step analysis approach with advanced scoring mechanisms, CONFIGSCAN enhances accuracy and ensures that critical vulnerabilities are identified and addressed with confidence.

5. Experiments

In this section, we briefly describe the experimental setup and results. To demonstrate the risks associated with configuration files on Hugging Face, we have structured the experiments around two distinct research questions:

- **RQ1: Identifying Suspicious Risks in Configuration Files on Hugging Face.** Do Hugging Face repositories contain configuration files with elements that pose potential security or operational risks?
- **RQ2: Evaluating the Effectiveness of CONFIGSCAN.** How effective is CONFIGSCAN in identifying security risks compared to traditional rule-based methods?

5.1. RQ1: Identifying Suspicious Risks in Configuration Files on Hugging Face.

Datasets. In this part, we employed a web crawler to gather data from 150,000 repositories on Hugging Face. Due to the substantial storage demands of repository data, we limited our collection to repository names. We developed a system that automatically retrieves and extracts the file contents of a repository using its name as the primary identifier.

Methods. To address the first research question, we utilized a rule-based analysis method to extract suspicious elements from configuration files. We focused on three distinct risk categories: file, website and repository operation risk. As described in Section 4.1, we used CONFIGSCAN to help detect these elements through a rule-based approach.

Results. As shown in Table 1, we identified 13,091, 1,324, and 35,761 suspicious repositories on Hugging Face that contained configuration files associated with potential risks. Within these repositories, we found 13,901, 1,215, and 31,373

TABLE 1. RULE-BASED ANALYSIS

Potential Risk	Suspicious Repositories	Suspicious Config Files
File Operation	13,091	13,091
Website Operation	1,324	1,215
Repository Operation	35,761	31,373

suspicious configuration files, each corresponding to File Operation Risk, Website Operation Risk, and Repository Operation Risk, respectively, demonstrating the potential risks in configuration files. However, this approach alone did not uncover the root causes of these issues, highlighting the significant limitations of rule-based methods in addressing such challenges effectively.

5.2. RQ2: Evaluating the Effectiveness of CONFIGSCAN.

Datasets. Due to API constraints and time limitations, we sampled 1,000 repositories from the results of Section 5.1 to evaluate the effectiveness of CONFIGSCAN.

Methods. In this part, we utilized CONFIGSCAN to scan the 1,000 selected repositories. After the scan, we manually evaluated these repositories to confirm the effectiveness of CONFIGSCAN.

Results. In our experiment, CONFIGSCAN identified two new repositories (hauson-fan/RagRetriever and Hugging-Worm/RagRetriever), both containing references to unknown repositories in their configuration files. These pose potential risks to users who execute them, risks that cannot be detected using rule-based methods alone. Additionally, CONFIGSCAN confirmed that other analyzed repositories did not present security risks, which was also verified by our manual check. In contrast to rule-based methods, CONFIGSCAN uses a dynamic and adaptive approach, integrating LLM-based reasoning to analyze configuration structures beyond simple pattern matching. This approach enables CONFIGSCAN to uncover hidden risks, such as the inclusion of unverified repositories, which traditional methods fail to detect. The discovery of hauson-fan/RagRetriever and HuggingWorm/RagRetriever underscores CONFIGSCAN's ability to identify latent security threats, reducing false positives.

6. Conclusion

In conclusion, our study reveals significant security risks from malicious configuration files on AI model hosting platforms like Hugging Face. We identified three attack scenarios—file, website, and repository operations—and introduced CONFIGSCAN, an LLM-based tool that analyzes configuration files alongside their runtime code to detect vulnerabilities with high accuracy and low false positives. Our findings underscore the urgent need for enhanced security measures in AI supply chains.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[2] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[3] H. Qian, W. Liu, Z. Ding, W. Sun, and C. Fang, "Abstract syntax tree for method name prediction: How far are we?" in *2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS)*. IEEE, 2023, pp. 464–475.

[4] Y. Liu, K. Zhang, Y. Li, Z. Yan, C. Gao, R. Chen, Z. Yuan, Y. Huang, H. Sun, J. Gao *et al.*, "Sora: A review on background, technology, limitations, and opportunities of large vision models," *arXiv preprint arXiv:2402.17177*, 2024.

[5] Y. Qu, P. Liu, W. Song, L. Liu, and M. Cheng, "A text generation and prediction system: pre-training on new corpora using bert and gpt-2," in *2020 IEEE 10th international conference on electronics information and emergency communication (ICEIEC)*. IEEE, 2020, pp. 323–326.

[6] HuggingFace, "https://huggingface.co/," 2024, https://huggingface.co/.

[7] Pypi, "https://pypi.org/," 2024, https://pypi.org/.

[8] T. Han, W. Sun, Z. Ding, C. Fang, H. Qian, J. Li, Z. Chen, and X. Zhang, "Mutual information guided backdoor mitigation for pre-trained encoders," *arXiv preprint arXiv:2406.03508*, 2024.

[9] S. Qiu, Q. Liu, S. Zhou, and C. Wu, "Review of artificial intelligence adversarial attack and defense technologies," *Applied Sciences*, vol. 9, no. 5, p. 909, 2019.

[10] J. Zhao, S. Wang, Y. Zhao, X. Hou, K. Wang, P. Gao, Y. Zhang, C. Wei, and H. Wang, "Models are codes: Towards measuring malicious code poisoning attacks on pre-trained model hubs," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024, pp. 2087–2098.

[11] W. Jiang, N. Synovic, M. Hyatt, T. R. Schorlemmer, R. Sethi, Y.-H. Lu, G. K. Thiruvathukal, and J. C. Davis, "An empirical study of pre-trained model reuse in the hugging face deep learning model registry," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 2463–2475.

[12] P. Zhou, "https://i.blackhat.com/asia-24/presentations/asia-24-zhou-howtomakehuggingface.pdf," 2024, https://i.blackhat.com/Asia-24/Presentations/Asia-24-Zhou-HowtoMakeHuggingFace.pdf.

[13] M. Walker and A. Wood, "https://i.blackhat.com/asia-24/presentations/asia-24-zhou-howtomakehuggingface.pdf," 2024, https://i.blackhat.com/Asia-24/Presentations/Asia-24-Wood-Confused-Learning.pdf.

[14] R. Zhu, G. Chen, W. Shen, X. Xie, and R. Chang, "My model is malware to you: Transforming ai models into malware by abusing tensorflow apis," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 12–12.

[15] fickling, "https://github.com/trailofbits/fickling," 2024, https://github.com/trailofbits/fickling.

[16] protectai, "https://github.com/protectai/modelscan," 2024, https://github.com/protectai/modelscan.

[17] ——, "https://github.com/protectai/vulnhuntr," 2024, https://github.com/protectai/vulnhuntr.

[18] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, "Poisoning web-scale training datasets is practical," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 407–425.