

Creating an AI bill of materials for secure, transparent, and compliant model training

Wiebe Vandendriessche

Student number: 02111026

Supervisors: Prof. dr. Bruno Volckaert, Dr. ing. Merlijn Sebrechts

Counsellor: ing. Jordi Thijsman

Master's dissertation submitted in order to obtain the degree of Master of Science in Information Engineering Technology

Academic year 2024-2025

Acknowledgment

First and foremost, I would like to sincerely thank everyone who has supported and guided me throughout the completion of my master's degree.

I am especially grateful to my thesis supervisors, Prof. Dr. ir. Bruno Volckaert and Dr. ing. Merlijn Sebrechts, for their dedicated guidance and insightful advice.

I would also like to thank my counsellor, ing. Jordi Thijsman for always being available to answer my questions, for his valuable feedback, and for the time he invested in evaluating this work.

I am thankful to all the lecturers and professors who have taught me in the past years.

I also thank imec Idlab for providing the opportunity and resources necessary for this research.

My gratitude goes to my fellow students for their companionship and the fascinating discussions we have shared.

A special thanks to my family and friends for their moral support and constant encouragement.

Finally, I would like to thank everyone who contributed in any way to the accomplishment of this thesis. Your help has been very valuable.

Statement on the Use of LLM¹

As the author is not a native English speaker, GPT models and Writefull have been utilized as language refinement tools throughout the writing of this thesis to:

- identify and correct grammatical and spelling errors,
- maintain consistency and clarity in language,
- refine the phrasing of certain sections,
- summarize and interpret source materials.

In addition to language assistance, LLMs were also used to support the development process by:

- enhancing the readability and structure of source code,
- providing suggestions and improvements for code fragments,
- generating explanatory comments for certain parts of the code.

For all summaries, interpretations, and code-related output, the generated content was carefully reviewed and cross-checked against the original sources and documentation to ensure accuracy and reliability.

¹A Large Language Model (LLM) is a type of machine learning model designed for natural language processing tasks such as language generation. LLMs are language models with many parameters and are trained with self-supervised learning on a large amount of text [1].

Explanation regarding the master's thesis and the oral presentation

This master's dissertation is part of an exam. Any comments formulated by the assessment committee during the oral presentation of the master's dissertation are not included in this text.

Abstract

Creating an AI bill of materials for secure, transparent, and compliant model training

Author: Wiebe Vandendriessche

Student number: 02111026

Academic year: 2024-2025

Master's dissertation submitted in order to obtain the degree of Master of Science in
Information Engineering Technology

Supervisors:

Prof. Dr. Bruno Volckaert
Dr. Merlijn Sebrechts

Counsellors:

ing. Jordi Thijsman

Keywords: AI Bill of Materials (AIBOM), Trustability, Transparency, AI Act Compliance, AI PaaS

It is clear that integration of Artificial Intelligence (AI) into modern IT systems continues to grow at a rapid pace. While this brings about much innovation, many challenges also emerge regarding transparency, security, and legal compliance. This thesis explores the concept of an AI Bill of Materials (AIBOM), an extension of the Software Bill of Materials (SBOM), as a standardized approach to documenting trained AI models. By capturing datasets, model metadata, training metadata, and the training environment, AIBOMs improve the transparency and traceability of models, and thus make compliance with regulations such as the AI Act easier. The dissertation focuses on methodologies and challenges for automating AIBOM generation while maintaining trust and security. A Proof-of-Concept (PoC) implementation, named AIBoMGen, demonstrates how these AIBOMs can promote trust and reduce risk. The system addresses critical threats, such as dataset manipulation and AIBOM tampering. Key countermeasures include hash-based integrity verification, restricted execution environments, and enforced AIBOM generation workflows. Evaluations of AIBoMGen included potential tampering scenarios, integrity checks, and workflow bypass attempts. Results show that AIBoMGen effectively detects unauthorized modifications and resists tampering. Performance measurements indicate that generating an AIBOM takes less than 0.4 seconds, even for large models. These results highlight the potential of AIBoMGen as a foundational step toward building secure and transparent AI ecosystems, enabling compliance with regulatory frameworks like the EU AI Act.

Creating an AI bill of materials for secure, transparent, and compliant model training

Wiebe Vandendriessche 
IDLab-Imec, Department of Information Technology
Ghent University
Ghent, Belgium
wiebe.vandendriessche@ugent.be

Supervisors:
Prof. Dr. Bruno Volckaert
Dr. Merlijn Sebrechts

Counsellors:
ing. Jordi Thijsman

Abstract—It is clear that integration of Artificial Intelligence (AI) into modern Information Technology (IT) systems continues to grow at a rapid pace. While this brings about much innovation, many challenges also emerge regarding transparency, security, and legal compliance. This thesis explores the concept of an AI Bill of Materials (AIBOM), an extension of the Software Bill of Materials (SBOM), as a standardized approach to documenting trained AI models. By capturing datasets, model metadata, training metadata, and the training environment, AIBOMs improve the transparency and traceability of models, and thus make compliance with regulations such as the AI Act easier. The dissertation focuses on methodologies and challenges for automating AIBOM generation while maintaining trust and security. A Proof-of-Concept (PoC) implementation, named AIBoMGen, demonstrates how these AIBOMs can promote trust and reduce risk. The system addresses critical threats, such as dataset manipulation and AIBOM tampering. Key countermeasures include hash-based integrity verification, restricted execution environments, and enforced AIBOM generation workflows. Evaluations of AIBoMGen included potential tampering scenarios, integrity checks, and workflow bypass attempts. Results show that AIBoMGen effectively detects unauthorized modifications and resists tampering. Performance measurements indicate that generating an AIBOM takes less than 0.4 seconds, even for large models. These results highlight the potential of AIBoMGen as a foundational step toward building secure and transparent AI ecosystems, enabling compliance with regulatory frameworks like the EU AI Act.

Index Terms—AIBOM, Trustability, Transparency, AI Act Compliance, AI PaaS

I. INTRODUCTION

Artificial Intelligence (AI) is being rapidly integrated into modern Information Technology (IT) systems, with applications spanning healthcare, finance, and public administration [1]. However, this rapid adoption brings significant challenges in security, transparency, and compliance. High-profile cases, such as the Clearview AI scandal [2], where biometric data were collected without consent, and Stanford University's findings on unethical datasets containing Child Sexual Abuse Material (CSAM) [3], highlight the risks of opaque AI systems.

Traditional cybersecurity methods are often insufficient for

AI-specific risks such as data poisoning¹ and adversarial manipulation² [6]. Trust and transparency must be established, particularly in high-risk AI systems.

Existing solutions, such as model cards [7] provided by platforms like Hugging Face [8], attempt to address transparency concerns but remain insufficient. While they offer descriptive metadata, they lack a comprehensive, verifiable record of the AI development process.

The AI Bill of Materials (AIBOM), an adaptation of the Software Bill of Materials (SBOM), has emerged as a potential solution. AIBOMs provide a structured inventory of all components involved in AI model development [9]. However, transparency alone is not enough. Trust is needed [10]. Without mechanisms to ensure the integrity and authenticity of the documented information, AIBOMs cannot effectively support compliance and security [11]. Many AI development environments and research-focused infrastructures do not have a standardized procedure in place to provide guarantees on these AIBOMs, let alone generate them [12].

To address these challenges, this research focuses on the following key questions:

- What are the current advancements and challenges in the research and development of AIBOMs?
- In what ways can AIBOMs support compliance with AI regulations, such as the AI Act, and other emerging standards?
- How can trust be effectively integrated into the AIBOM framework to ensure transparency, integrity, and ethical concerns?
- What methods and technologies can be used to automate the generation of AIBOMs while ensuring their reliability and security?

¹Data poisoning is a type of cyberattack where threat actors manipulate or corrupt the training data used to develop AI and Machine Learning (ML) models [4].

²Adversarial manipulation involves crafting inputs designed to deceive AI models, causing them to produce incorrect or unintended outputs [5].

II. EVOLVING REGULATORY LANDSCAPE

Upcoming regulations, such as the EU AI Act [13], underline the importance of verifiable and transparent model documentation. The AI Act adopts a risk-based approach, classifying AI systems into four levels of risk [14]: unacceptable, high, limited, and minimal or no risk. High-risk systems, often deployed in critical infrastructure like healthcare, must comply with strict requirements, including risk assessments, high data quality, technical documentation, and transparency measures (AIBOM). Providers must also establish a risk management system throughout the life cycle of the system.

III. CURRENT STATE OF AIBOM

Bill of Materials (BOM) standards like CycloneDX [15] and SPDX 3.0 [16] are beginning to adapt their BOM schemes for AI. CycloneDX, originally designed for SBOMs, has been extended to include AI-specific components, enabling structured and machine-readable AIBOMs [17]. Similarly, SPDX 3.0 introduces fields for AI-specific metadata [18] and compliance with regulations like the EU AI Act [19]. These standards provide a foundation for creating interoperable and verifiable AIBOMs, but their adoption remains limited.

At the moment, there are many AIBOM-related projects coming up with proposals to transparency issues. One proposal on AIBOM content, from Manifest Cyber [20], states that should be included:

- Metadata of the model, such as: name, version, description and authorship.
- Timestamp and generation tools.
- Detailed information about model architecture, datasets used and ethical considerations and intended usage.
- Dependencies required by the model such as: external sources and ML libraries.

While Manifest Cyber’s example is a strong starting point, it has several key limitations. The BOM relies on declarative documentation, which can be manually edited and is not cryptographically verifiable. There are no artifact hashes or digital signatures, making it impossible to verify the authenticity or integrity of the documented files. Key information, such as intended use, misuse scenarios, and limitations, relies solely on the developer’s honesty and accuracy. This manual approach increases the risk of inconsistencies or discrepancies between the documented information and the actual properties of the model.

IV. PROOF-OF-CONCEPT: AIBOMGEN

To address these limitations, AIBOMGen was developed as a Proof-of-Concept (PoC) platform for generating trusted CycloneDX AIBOMs [21]. The platform introduces the following key features:

- **Trusted attestations:** The PoC introduces in-toto attestations [22], providing cryptographic evidence of the training process and environment.
- **Artifact hashes for integrity:** These attestations include SHA-256 hashes of artifacts such as datasets, models,

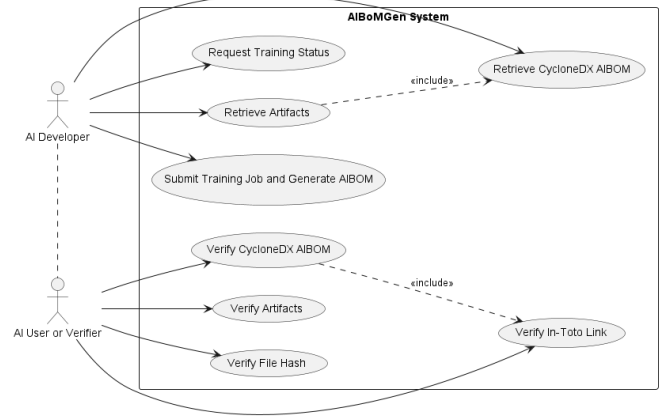


Fig. 1. Use case diagram of the AIBOMGen application

and metrics, enabling users to verify the integrity of each item.

- **Digital signatures:** The AIBOM itself is signed, and signature fields are embedded in the BOM.
- **Automation:** The process is automated, reducing the risk of human error or manipulation.

The platform acts as a scalable AI Platform as a Service (PaaS) and could be extended and deployed on cloud infrastructure, backed by a trusted institution like the EU government. However, the current state of the system is a PoC and is not intended for production use.

A. User flows

The system involves two primary actors: the AI Developer and the Verifier. Fig. 1 illustrates the main system use cases of these actors.

The AI Developer submits training jobs to the system through a secure API. These jobs include the model, dataset, and training parameters. Currently, the implementation supports classification and regression tasks on tabular and image data using TensorFlow. Once the training is complete, the developer retrieves resulting artifacts, including a trained model, metrics, and the generated AIBOM. The developer can make his AIBOM publicly available for transparency.

The verifier ensures the integrity and authenticity of the AIBOM and its associated artifacts. This is achieved by verifying cryptographic signatures and hashes recorded during the training process. The verifier can also validate artifact hashes against the hashes stored in the AIBOM to detect any tampering.

The PoC implementation has four verification endpoints:

- **Verify AIBOM and in-toto Link File:** The verifier uploads the signed CycloneDX AIBOM to the API Gateway. The endpoint validates the BOM’s schema, verifies its cryptographic signature, and checks the referenced

attestation file (in-toto link). This provides an end-to-end integrity and authenticity check for the entire training process and its outputs.

- **Verify in-toto Link File:** The verifier uploads the attestation file to the API endpoint, which checks the cryptographic signatures, layout validity, and artifact rules. This ensures the file is authentic and the recorded actions were performed as expected. Reference [23] provides more details about in-toto supply chain security.
- **Verify File Hash Against Link File:** The verifier uploads both the in-toto link file and a specific artifact (e.g., dataset, model, metrics ...) to the API. The endpoint computes the hash of the artifact and compares it to the hash recorded in the link file, confirming the artifact is the same as the one recorded.
- **Verify All Artifacts in Storage:** The verifier uploads the in-toto link file to the API, which then automatically downloads all referenced materials and products from storage. The endpoint computes their hashes and compares them to those in the link file, detecting any tampering in storage.

The verifier uses the results from the "Verify AIBOM" endpoint to make a trust decision. Optionally, they can further validate artifacts or certain artifact hashes and vulnerabilities to ensure the integrity of the training process.

B. Architecture

The AIBoMGen platform is built around a modular architecture designed for scalability, security, and transparency. Fig. 2 displays a simplified architecture diagram. The system consists of several key components, each playing a specific role in the AIBOM generation and verification process:

- **API Gateway:** The API Gateway serves as the central point of interaction for both developers and verifiers. It handles job submissions, artifact retrieval, and verification requests. By limiting interaction only through the gateway, remote code execution on the workers becomes

impossible, without a specific vulnerability elsewhere. The implementation used the FastAPI framework [24].

- **Task Scheduler:** A central task scheduler, implemented using Celery [25], coordinates the execution of training jobs and other tasks across the system. It uses RabbitMQ [26] as a message broker to ensure reliable communication. MySQL is used as a results backend [27].
- **Worker Nodes:** Worker nodes are responsible for executing training jobs in isolated Docker containers. Each worker processes tasks from the task scheduler and generates the AIBOM after training. They also perform environment extraction, artifact hashing, and in-toto attestation generation to ensure the integrity of the training process.
- **Data Storage:** MinIO, an S3-compatible object storage solution [28], is used to store datasets, models, logs and generated AIBOMs. It ensures secure and scalable storage for all artifacts. The system uses presigned URLs to provide time-limited access to stored artifacts.
- **Vulnerability Scanner:** A dedicated Trivy scanner [29] periodically checks the Docker images used by the workers for vulnerabilities. The results are also stored in the object store and a summary is included in the AIBOM.

V. EVALUATION AND RESULTS

AIBoMGen was evaluated on three key metrics: performance, storage, and security. The results demonstrate that the platform introduces minimal performance and storage overhead while providing robust security measures. This section provides an analysis of these metrics, along with additional insights into the threat evaluation. The experiment scripts and results are available on Zenodo [30].

A. Performance

The performance impact of AIBOM generation was measured by submitting training jobs with varying epoch counts. As shown in Table I, the AIBOM generation time remained constant regardless of the number of epochs, while the training time increased linearly. The table also illustrates that the overhead introduced by AIBOM generation becomes negligible for longer training jobs, confirming its scalability.

B. Storage Impact

The storage overhead of AIBOM generation was evaluated by analyzing the sizes of artifacts produced during training. Fig. 3 shows that the AIBOM size grows slightly with dataset size but remains small relative to the dataset and model sizes.

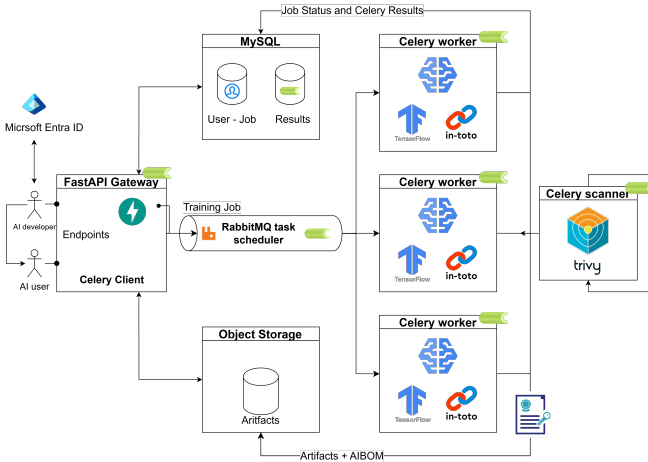


Fig. 2. Architecture diagram of AIBoMGen backend

TABLE I
TRAINING AND AIBOM GENERATION TIMES FOR DIFFERENT EPOCHS

Epochs	Training Time (s)	AIBOM Time (s)	Total Time (s)	AIBOM Overhead (%)
1	4.316	0.265	4.581	5.78
2	6.629	0.234	6.863	3.41
5	12.492	0.218	12.710	1.72
10	23.790	0.196	23.986	0.82
50	117.728	0.198	117.926	0.17
100	230.837	0.197	231.034	0.09

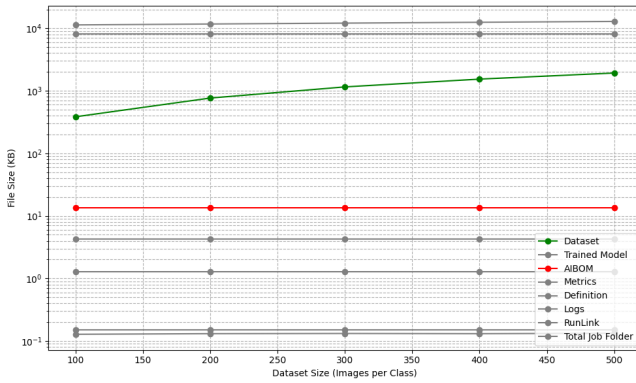


Fig. 3. Plot showing artifact file sizes for different dataset sizes

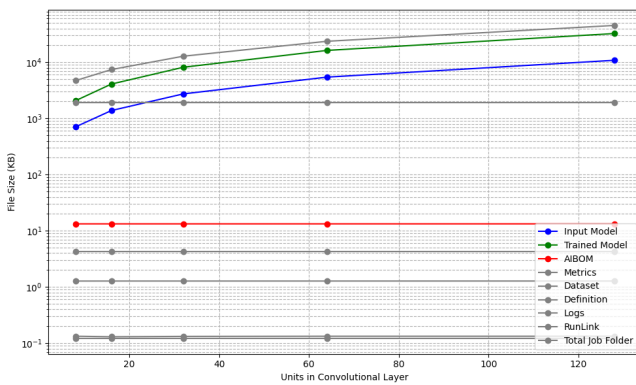


Fig. 4. Plot showing artifact file sizes for different input model sizes

Similarly, Fig. 4 demonstrates that the AIBOM size is minimal compared to the trained model size, even for larger models. These results confirm that the storage impact is negligible.

C. Threat Evaluation

A detailed threat analysis was conducted to evaluate the platform’s resilience against tampering and malicious activities. The evaluation focused on identifying potential threats, implementing countermeasures, and validating their effectiveness through manual testing and integrity verification.

The system addresses a wide range of tampering scenarios, including model manipulation, dataset tampering, training process interference, artifact modification, and attempts to bypass AIBOM generation. Key threats, their countermeasures, and the methods used to validate them are summarized below:

- **Model Manipulation:** Malicious actors could embed unsafe `Lambda` layers in models to execute arbitrary code [31].
 - **Countermeasures:**
 - * TensorFlow’s `safe_mode=True` option [32] is used to block unsafe deserialization.
 - * Models are loaded in isolated Docker containers to limit the impact of potential attacks.

- * All model artifacts are hashed and recorded in the AIBOM for integrity verification.

– Testing Methods:

- * Uploaded models with unsafe `Lambda` layers were rejected due to safe loading.
- * Verified model hashes in the AIBOM matched the uploaded model.

- **Dataset Tampering:** Attackers could inject malicious data or alter dataset structures.

– Countermeasures:

- * Validation of dataset structure and content (e.g., checks for `.csv` files).
- * Secure extraction of `.zip` files to prevent resource exhaustion attacks (e.g., zip bombs).

– Testing Methods:

- * Provided malformed datasets and verified that the system rejected them based on validation checks.

- **Training Process Interference:** Threats include modifying training parameters or injecting malicious code.

– Countermeasures:

- * Training jobs are executed in read-only Docker containers with restricted capabilities.
- * In-toto attestations are used to verify the integrity of the training process.
- * Environment metadata (e.g., Python and TensorFlow versions) is included in the AIBOM.

– Testing Methods:

- * Verified that training jobs could not modify their environment due to read-only container settings.
- * Tested in-toto attestations to confirm that all recorded actions were performed as expected.

- **Artifact Modification:** Attackers could tamper with generated artifacts (e.g., models, logs, metrics).

– Countermeasures:

- * Hash-based integrity verification is applied to all artifacts.
- * Artifacts are stored in MinIO buckets.
- * Cryptographic signing of AIBOMs and in-toto link files ensures authenticity.

– Testing Methods:

- * Modified artifacts post-generation and verified that hash mismatches were detected.
- * Tested verify MinIO storage endpoint.

- **Tampering with the AIBOM:** A malicious actor could tamper with the AIBOM to falsify information about the training process, inputs, or outputs.

– Countermeasures:

- * Cryptographic signing ensures that the AIBOM cannot be modified without detection.
- * Hash verification allows users to confirm the integrity of all input and output artifacts.

– Testing Methods:

- * Modified AIBOM files and verified that signature checks failed.
- **Unauthorized Access to Artifacts or System Components:** A malicious actor could gain unauthorized access to artifacts or system components, leading to data breaches or tampering.
 - **Countermeasures:**
 - * Azure OAuth is used for secure API calls, ensuring only authenticated and authorized users can access the system.
 - **Testing Methods:**
 - * Reviewed authentication and access control configurations.
 - * No penetration testing was performed, but manual reviews confirmed proper access restrictions.
- **Bypassing AIBOM Generation:** Malicious actors could attempt to bypass the AIBOM generation process.
 - **Countermeasures:**
 - * Enforced AIBOM generation for every training job.
 - * Controlled API interactions to prevent unauthorized modifications.
 - **Testing Methods:**
 - * The system successfully removes artifacts if the AIBOM generation fails.
 - * Reviewed API logs to confirm that all interactions adhered to the enforced workflows. If the layout does not match, the link file verification fails.

Remaining risks include potential vulnerabilities in dependencies, key compromise, and undetectable malicious content in valid models or datasets. Advanced attacks (e.g., exploits, key compromise, adversarial data, or privilege escalation) were not tested due to resource and scope constraints.

These results highlight the robustness of AIBoMGen in securing the AI development process.

VI. CONCLUSION

This research investigated how AI Bill of Materials (AIBOM) can be implemented as a standardized and trustworthy approach to ensure transparency in AI development. The findings address four key research questions:

A. What are the current advancements and challenges in the research and development of AIBOMs?

The research revealed that while AIBOMs are getting traction, existing standards like CycloneDX and SPDX are evolving to fully support AI-specific components. A major challenge lies in the lack of secure, attested, automated, and standardized workflows for generating AIBOMs during the training process. Current industry practices often prioritize flexibility over security, leaving gaps in trustworthiness and compliance.

B. In what ways can AIBOMs support compliance with AI regulations, such as the AI Act, and other emerging standards?

AIBOMs can play an important role in meeting transparency requirements of the AI Act. It provides a verifiable record of datasets, dependencies, and training processes, which enables organizations to demonstrate compliance. However, the lack of widespread adoption, and most of all, tooling for AIBOMs remains a barrier.

C. How can trust be effectively integrated into the AIBOM framework to ensure transparency, integrity, and ethical concerns?

The PoC demonstrated that trust can be integrated through cryptographic attestations, integrity checks, and controlled execution environments. A key lesson is that many industry-standard training workflows rely on custom Docker containers and training scripts, which are not inherently secure or attested. There is a need for secure-by-design AI development platforms.

D. What methods and technologies can be used to automate the generation of AIBOMs while ensuring their reliability and security?

The PoC showed that automation is feasible using tools like CycloneDX Python library, in-toto (or other attestation tools), and containerized workflows. The reliance on emerging standards and incomplete library support (e.g., CycloneDX Python library) also underscores the need for further development in this area.

In conclusion, this thesis highlights the potential of AIBoMGen (and similar platforms) as a foundational step toward building secure and transparent AI ecosystems, enabling compliance with regulatory frameworks like the EU AI Act.

FUTURE WORK

Future work will focus on expanding the platform's capabilities. This includes support for additional AI frameworks and integration with emerging standards like SPDX 3.0. Technologies such as Trusted Platform Modules (TPM) and Trusted Execution Environment (TEE) can also be explored. These technologies can provide hardware-based security guarantees, ensuring that the training environment and processes remain tamper-proof and verifiable while enabling secure execution of custom training scripts.

ACKNOWLEDGMENT

I would like to thank my thesis supervisors, Prof. Dr. Bruno Volckaert and Dr. Merlijn Sebrechts, for their guidance and advice, and my counsellor, ing. Jordi Thijsman, for his valuable feedback and support. I am also grateful to imec IDLab for providing the opportunity for this research.

REFERENCES

- [1] B. Ghosh, "AI Bill of Materials (AI BOM)," Jul. 2024. [Online]. Available: <https://medium.com/@bijit211987/ai-bill-of-materials-ai-bom-80d48f9d75e0>
- [2] R. Hart, "Clearview AI—Controversial Facial Recognition Firm—Fined \$33 Million For ‘Illegal Database’," 2024, section: Business. [Online]. Available: <https://www.forbes.com/sites/roberthart/2024/09/03/clearview-ai-controversial-facial-recognition-firm-fined-33-million-for-illegal-database/>
- [3] D. Thiel, "Investigation Finds AI Image Generation Models Trained on Child Abuse," Dec. 2023. [Online]. Available: <https://cyber.fsi.stanford.edu/news/investigation-finds-ai-image-generation-models-trained-child-abuse>
- [4] T. Krantz, "What Is Data Poisoning?" Dec. 2024. [Online]. Available: <https://www.ibm.com/think/topics/data-poisoning>
- [5] Dremio, "Adversarial Attacks in AI," 2025. [Online]. Available: <https://www.dremio.com/wiki/adversarial-attacks-in-ai/>
- [6] S. Creese, "Why we need to reflect on the need for cybersecurity of AI," Dec. 2023. [Online]. Available: <https://www.weforum.org/stories/2023/12/cybersecurity-ai-ethics-responsible-innovation/>
- [7] E. Ozoani, M. GerChick, and M. Mitchell, "Model Cards," 2022. [Online]. Available: <https://huggingface.co/blog/model-cards>
- [8] Hugging Face, "Hugging Face," May 2025. [Online]. Available: <https://huggingface.co/>
- [9] D. Bardenstein, "Manifest: AIBOMs, Transparency into AI," 2023. [Online]. Available: <https://www.manifestcyber.com/aibom>
- [10] TechWorks and NquiringMinds, "Trustworthy AI – Practical Collaborative Engineering," 2023. [Online]. Available: <https://www.techworks.org.uk/wp-content/uploads/2024/01/Engineering-Trustworthy-AI.pdf>
- [11] TechWorks, "Compliance with the EU AI Act The Techworks Trusted AI Bill Of Materials (TAIBOM) project," Jun. 2024. [Online]. Available: <https://www.techworks.org.uk/ai-taibom/compliance-with-the-eu-ai-act-the-techworks-trusted-ai-bill-of-materials-taibom-project>
- [12] imec, "GPULab — imec iLab.t documentation." [Online]. Available: <https://doc.ilabt.imec.be/ilabt/gpulab/>
- [13] EU, "AI Act | Shaping Europe's digital future," Sep. 2024. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai>
- [14] —, "High-level summary of the AI Act | EU Artificial Intelligence Act," Feb. 2024. [Online]. Available: <https://artificialintelligenceact.eu/high-level-summary/>
- [15] OWASP Foundation, "CycloneDX - Software Bill of Materials (SBOM)," Jul. 2024. [Online]. Available: <https://cyclonedx.org/>
- [16] Linux Foundation, "SPDX," 2023. [Online]. Available: <https://spdx.dev/>
- [17] OWASP Foundation, "CycloneDX - Machine Learning Bill of Materials (ML-BOM)," 2024. [Online]. Available: <https://cyclonedx.org/capabilities/mlbom/>
- [18] Linux Foundation, "SPDX AI Github Specification 3.0.1," 2024. [Online]. Available: <https://spdx.github.io/spdx-spec/v3.0.1/model/AI/AI/>
- [19] —, "Implementing AI Bill of Materials (AI BOM) with SPDX 3.0," 2024. [Online]. Available: <https://www.linuxfoundation.org/research/ai-bom>
- [20] D. Bardenstein, "Whitepaper: Driving AI Transparency: the AI Bill of Materials (Manifest)," 2023. [Online]. Available: manifestcyber.com
- [21] W. Vandendriessche, "AIBoMGen," May 2025. [Online]. Available: <https://zenodo.org/records/15536533>
- [22] in-toto, "in-toto," 2025. [Online]. Available: <https://in-toto.io/>
- [23] —, "in-toto 3.0.0 documentation," 2025. [Online]. Available: <https://in-toto.readthedocs.io/en/latest/>
- [24] FastAPI, "FastAPI," 2025. [Online]. Available: <https://fastapi.tiangolo.com/>
- [25] A. Solem and contributors, "Celery 5.5.2 documentation," 2023. [Online]. Available: <https://docs.celeryq.dev/en/stable/>
- [26] RabbitMQ, "RabbitMQ: One broker to queue them all," 2025. [Online]. Available: <https://www.rabbitmq.com/>
- [27] Oracle, "MySQL," 2025. [Online]. Available: <https://www.mysql.com/>
- [28] MinIO, "MinIO - S3 Compatible Storage for AI," 2025. [Online]. Available: <https://min.io>
- [29] Aqua Security, "Trivy," 2025. [Online]. Available: <https://trivy.dev/v0.62/>
- [30] W. Vandendriessche, "AIBoMGen-experiments Scripts & Results," May 2025. [Online]. Available: <https://zenodo.org/records/15505280>
- [31] T. Righi, "TensorFlow Remote Code Execution with Malicious Model," Oct. 2022. [Online]. Available: <https://splint.gitbook.io/cyberblog/security-research/tensorflow-remote-code-execution-with-malicious-model>
- [32] TensorFlow, "tf.keras.models.load_model," Jun. 2024. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/models/load_model

Het creëren van een AI bill of materials voor veilige, transparante en conforme modeltraining

Wiebe Vandendriessche 
IDLab-Imec, Afdeling Informatietechnologie
Universiteit Gent
Gent, België
wiebe.vandendriessche@ugent.be

Promotoren:
prof. dr. Bruno Volckaert
dr. Merlijn Sebrechts

Begeleiders:
ing. Jordi Thijsman

Abstract—Het is duidelijk dat de integratie van Artificiële Intelligentie (AI) in moderne Informatietechnologie (IT)-systemen in een snel tempo blijft groeien. Hoewel dit veel innovatie met zich meebrengt, ontstaan er ook veel uitdagingen op het gebied van transparantie, veiligheid en wettelijke naleving. Deze scriptie verkent het concept van een *AI Bill of Materials* (AIBOM), een uitbreiding van de *Software Bill of Materials* (SBOM), als een gestandaardiseerde aanpak voor het documenteren van getrainde AI-modellen. Door datasets, modelmetadata, trainingsmetadata en de trainingsomgeving vast te leggen, verbeteren AIBOMs de transparantie en traceerbaarheid van modellen, en maken ze naleving van regelgeving zoals de AI Act eenvoudiger. De scriptie richt zich op methodologieën en uitdagingen voor het automatiseren van AIBOM-generatie terwijl vertrouwen en veiligheid behouden blijven. Een *Proof-of-Concept* (PoC)-implementatie, genaamd AIBoMGen, demonstreert hoe deze AIBOMs vertrouwen kunnen bevorderen en risico's kunnen verminderen. Het systeem behandelt kritieke bedreigingen, zoals datasetmanipulatie en AIBOM-vervalsing. Belangrijke tegenmaatregelen omvatten hash-gebaseerde integriteitsverificatie, beperkte uitvoeringsomgevingen en afgedwongen AIBOM-generatieworkflows. Evaluaties van AIBoMGen omvatten potentiële vervalsingsscenario's, integriteitscontroles en pogingen tot workflowomzeiling. Resultaten tonen aan dat AIBoMGen ongeautoriseerde wijzigingen effectief detecteert en bestand is tegen vervalsing. Prestatiemetingen geven aan dat het genereren van een AIBOM minder dan 0,4 seconden duurt, zelfs voor grote modellen. Deze resultaten benadrukken het potentieel van AIBoMGen als een fundamentele stap richting het bouwen van veilige en transparante AI-ecosystemen, waarmee naleving van regelgevende kaders zoals de EU AI-wet mogelijk wordt gemaakt.

Index Terms—AIBOM, Vertrouwen, Transparantie, AI Act Naleving, AI PaaS

I. INLEIDING

Artificiële Intelligentie (AI) wordt snel geïntegreerd in moderne Informatietechnologie (IT)-systemen, met toepassingen in de gezondheidszorg, financiën en openbare administratie [1]. Deze snelle adoptie brengt echter aanzienlijke uitdagingen met zich mee op het gebied van veiligheid, transparantie en naleving. Incidenten, zoals het Clearview AI-schandaal [2], waarbij biometrische gegevens zonder toestemming werden verzameld, en de bevindingen van Stanford University over

onethische datasets met *Child Sexual Abuse Material* (CSAM) [3], benadrukken de risico's van ondoorzichtige AI-systemen.

Traditionele cybersecurity-methoden zijn vaak onvoldoende voor AI-specifieke risico's zoals *data poisoning*¹ en *adversarial manipulation*² [6]. Vertrouwen en transparantie moeten worden gevestigd, vooral in risicovolle AI-systemen.

Bestaande oplossingen, zoals modelkaarten [7] aangeboden door platformen zoals Hugging Face [8], proberen transparantieproblemen aan te pakken, maar blijven onvoldoende. Hoewel ze beschrijvende metadata bieden, missen ze een uitgebreide, verifieerbare registratie van het AI-ontwikkelingsproces.

De *AI Bill of Materials* (AIBOM), een aanpassing van de *Software Bill of Materials* (SBOM), is naar voren gekomen als een potentiële oplossing. AIBOMs bieden een gestructureerd overzicht van alle componenten die betrokken zijn bij de ontwikkeling van AI-modellen [9]. Transparantie alleen is echter niet genoeg. Vertrouwen is nodig [10]. Zonder mechanismen om de integriteit en authenticiteit van de gedocumenteerde informatie te waarborgen, kunnen AIBOMs niet effectief veiligheid en AI-Act-naleving ondersteunen [11]. Veel AI-ontwikkelingsomgevingen en onderzoeksgerichte infrastructuur hebben geen gestandaardiseerde procedure om garanties te bieden op deze AIBOMs, laat staan om ze te genereren [12].

Om deze uitdagingen aan te pakken, richt dit onderzoek zich op de volgende kernvragen:

- Wat zijn de huidige vooruitgangen en uitdagingen in het onderzoek en de ontwikkeling van AIBOMs?
- Op welke manieren kunnen AIBOMs naleving van AI-regelgeving, zoals de AI Act, en andere opkomende standaarden ondersteunen?
- Hoe kan vertrouwen effectief worden geïntegreerd in het AIBOM-kader om transparantie, integriteit en ethische

¹*Data poisoning* is een type cyberaanval waarbij actoren trainingsdata manipuleren of corrumperen die worden gebruikt om AI- en *Machine Learning* (ML)-modellen te ontwikkelen [4].

²*Adversarial manipulation* omvat het creëren van invoer die is ontworpen om AI-modellen te misleiden, waardoor ze onjuiste of onbedoelde uitvoer produceren [5].

zorgen te waarborgen?

- Welke methoden en technologieën kunnen worden gebruikt om de generatie van AIBOMs te automatiseren terwijl hun betrouwbaarheid en veiligheid worden gewaarborgd?

II. EVOLUERENDE REGELGEVINGEN

Opkomende regelgevingen, zoals de EU AI Act [13], onderstrepen het belang van verifieerbare en transparante model documentatie. De AI Act hanteert een op risico's gebaseerde benadering, waarbij AI systemen in vier risiconiveaus worden geclassificeerd [14]: onaanvaardbaar, hoog, beperkt en minimaal of geen risico. Systemen met een hoog risico, vaak ingezet in kritieke infrastructuur zoals de gezondheidszorg, moeten voldoen aan strikte eisen, waaronder risico-evaluaties, hoge gegevenskwaliteit, technische documentatie, en transparantiemaatregelen (AIBOM). Leveranciers moeten ook een risicobeheer systeem opzetten voor de levenscyclus van het systeem.

III. HUIDIGE STAND VAN ZAKEN VAN AIBOM

Bill of Materials (BOM) normen zoals CycloneDX [15] en SPDX 3.0 [16] beginnen hun BOM schema's aan te passen voor AI. CycloneDX, oorspronkelijk ontworpen voor SBOMs, is uitgebreid om AI-specifieke componenten op te nemen, waardoor gestructureerde en machine-leesbare AIBOMs mogelijk zijn [17]. Evenzo introduceert SPDX 3.0 velden voor AI-specifieke metadata [18] en naleving van regelgeving zoals de EU AI Act [19]. Deze normen bieden een basis voor het creëren van interoperabele en verifieerbare AIBOMs, maar hun adoptie blijft beperkt.

Op dit moment zijn er veel AIBOM-gerelateerde projecten die voorstellen doen om transparantieproblemen aan te pakken. Een voorstel over de inhoud van AIBOM, van Manifest Cyber [20], stelt dat het volgende moet worden opgenomen:

- Metadata van het model, zoals: naam, versie, beschrijving en auteurschap.
- Tijdstempel en generatiehulpmiddelen.
- Gedetailleerde informatie over modelarchitectuur, gebruikte datasets en ethische overwegingen en bedoeld gebruik.
- Afhankelijkheden die door het model vereist zijn, zoals: externe bronnen en ML bibliotheken.

Hoewel het voorbeeld van Manifest Cyber een sterke basis biedt, heeft het verschillende belangrijke beperkingen. De BOM is afhankelijk van declaratieve documentatie, die handmatig kan worden bewerkt en niet cryptografisch verifieerbaar is. Er zijn geen *artifact hashes* of digitale handtekeningen, waardoor het onmogelijk is om de authenticiteit of integriteit van de gedocumenteerde bestanden te verifiëren. Belangrijke informatie, zoals *intended usage*, *misuse scenarios*, en *limitations*, is uitsluitend afhankelijk van de eerlijkheid en nauwkeurigheid van de ontwikkelaar. Deze handmatige benadering verhoogt het risico op inconsistenties of discrepanties tussen de gedocumenteerde informatie en de werkelijke eigenschappen van het model.

IV. PROOF-OF-CONCEPT: AIBOMGEN

Om deze beperkingen aan te pakken, is AIBOMGen ontwikkeld als een *Proof-of-Concept* (PoC) platform voor het genereren van betrouwbare CycloneDX AIBOMs [21]. Het platform introduceert de volgende belangrijke functies:

- **Vertrouwde attesten:** De PoC introduceert in-toto attesten [22], die cryptografisch bewijs leveren van het trainingsproces en de omgeving.
- **Artifact hashes voor integriteit:** Deze attesten bevatten *SHA-256-hashes* van *artifacts* zoals datasets, modellen en *metrics*, waarmee gebruikers de integriteit van elk item kunnen verifiëren.
- **Digitale handtekeningen:** De AIBOM zelf is ondertekend, en handtekeningsvelden zijn ingebed in de BOM.
- **Automatisering:** Het proces is geautomatiseerd, waardoor het risico op menselijke fouten of manipulatie wordt verminderd.

Het platform fungeert als een schaalbare *AI Platform as a Service* (PaaS) en kan worden uitgebreid en ingezet op cloud-infrastructuur, ondersteund door een vertrouwde instelling zoals de EU overheid. De huidige staat van het systeem is echter een PoC en is niet bedoeld voor productiegebruik.

A. Use cases

Het systeem omvat twee primaire actoren: de AI Ontwikkelaar en de Controleur. Fig. 1 illustreert de belangrijkste use cases van deze actoren.

De AI Ontwikkelaar dient trainingsopdrachten in bij het systeem via een veilige API. Deze opdrachten bevatten het model, de dataset en de trainings- parameters. Momenteel ondersteunt de implementatie classificatie- en regressietaken op tabel- en afbeeldingsgegevens met behulp van TensorFlow. Zodra de training is voltooid, haalt de ontwikkelaar de resulterende *artifacts* op, waaronder een getraind model, *metrics* en de gegenereerde AIBOM. De ontwikkelaar kan zijn AIBOM openbaar maken voor transparantie.

De Controleur zorgt voor de integriteit en authenticiteit van de AIBOM en de bijbehorende *artifacts*. Dit wordt bereikt door cryptografische handtekeningen en *hashes* te verifiëren die tijdens het trainingsproces zijn vastgelegd. De Controleur kan ook *artifact hashes* valideren tegen de *hashes* die in de AIBOM zijn opgeslagen om eventuele vervalsingen te detecteren.

De PoC implementatie heeft vier verificatie-eindpunten:

- **Verifieer AIBOM en in-toto Linkbestand:** De Controleur uploadt de ondertekende CycloneDX AIBOM naar de API Gateway. Het eindpunt valideert het schema van de BOM, verifieert de cryptografische handtekening en controleert het verwijzende attestatie- bestand (in-toto link). Dit biedt een end-to-end integriteits- en authenticiteitscontrole voor het gehele trainingsproces en de output.
- **Verifieer in-toto Linkbestand:** De Controleur uploadt het attestatiebestand naar het API eindpunt, dat de cryptografische handtekeningen, lay-outvaliditeit en *artifact-*

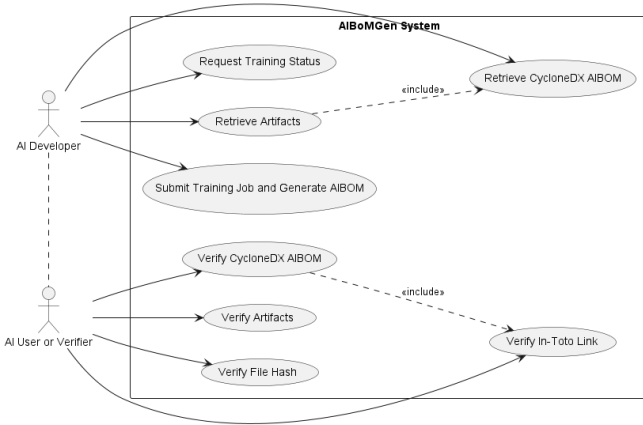


Fig. 1. Use case diagram van de AIBoMGen-toepassing

regels controleert. Dit zorgt ervoor dat het bestand authentiek is en dat de vastgelegde acties werden uitgevoerd zoals verwacht. Referentie [23] biedt meer details over in-toto *supply chain*-beveiliging.

- **Verifieer Bestands-hash Tegen Linkbestand:** De Controleur uploadt zowel het in-toto linkbestand als een specifiek *artifact* (bv. dataset, model, *metrics* ...) naar de API. Het eindpunt berekent de *hash* van het *artifact* en vergelijkt deze met de *hash* die in het linkbestand is vastgelegd, waarmee wordt bevestigd dat het *artifact* hetzelfde is als het geregistreerde.
- **Verifieer Alle Artifacts in Opslag:** De Controleur uploadt het in-toto linkbestand naar de API, die vervolgens automatisch alle verwijzende materialen en producten uit de opslag downloadt. Het eindpunt berekent hun *hashes* en vergelijkt deze met die in het linkbestand, waardoor eventuele vervalsingen in de opslag worden gedetecteerd.

De Controleur gebruikt de resultaten van het "Verifieer AIBOM" eindpunt om een vertrouwensbeslissing te nemen. Optioneel kunnen ze verdere validatie uitvoeren van *artifacts* of bepaalde *artifact hashes* en kwetsbaarheden om de integriteit van het trainingsproces te waarborgen.

B. Architectuur

Het AIBoMGen-platform is gebouwd rond een modulaire architectuur die is ontworpen voor schaalbaarheid, veiligheid en transparantie. Fig. 2 toont een vereenvoudigd architectuurdigram. Het systeem bestaat uit verschillende belangrijke componenten, die elk een specifieke rol spelen in het AIBOM generatie- en verificatieproces:

- **API Gateway:** De API Gateway dient als het centrale interactiepunt voor zowel ontwikkelaars als Controleurs. Het beheert de indiening van opdrachten, het ophalen van *artifacts* en verificatie- verzoeken. Door de interactie alleen via de gateway te beperken, wordt externe code-uitvoering op de *workers* onmogelijk, zonder een

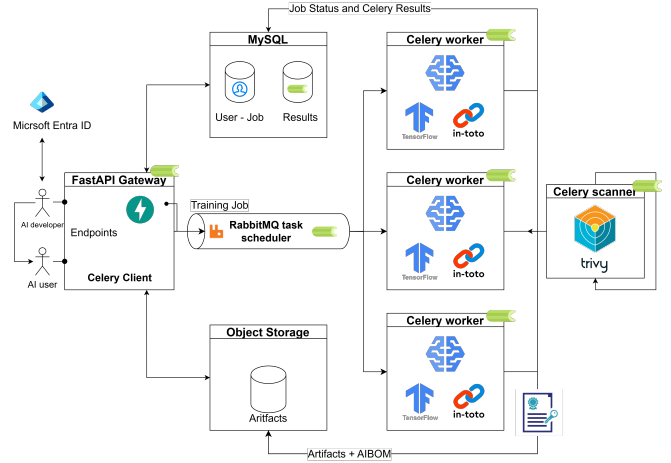


Fig. 2. Architectuurdigram van de AIBoMGen backend

specifieke kwetsbaarheid elders. De implementatie maakt gebruik van het FastAPI framework [24].

- **Taakplanner:** Een centrale taakplanner, geïmplementeerd met Celery [25], coördineert de uitvoering van trainingsopdrachten en andere taken in het systeem. Het gebruikt RabbitMQ [26] als berichtenbroker om betrouwbare communicatie te waarborgen. MySQL wordt gebruikt als resultaten-backend [27].
- **Worker Nodes:** *Worker nodes* zijn verantwoordelijk voor het uitvoeren van trainingsopdrachten in geïsoleerde Docker-containers. Elke *worker* verwerkt taken van de taakplanner en genereert de AIBOM na training. Ze voeren ook omgevingsextractie, *artifact hashing*, en in-toto attestatie-generatie uit om de integriteit van de training te waarborgen.
- **Gegevensopslag:** MinIO, een S3-compatibele *object store* [28], wordt gebruikt om datasets, modellen, logs en gegenereerde AIBOMs op te slaan. Het zorgt voor veilige en schaalbare opslag voor alle *artifacts*. Het systeem maakt gebruik van presigned URL's om tijdsgebonden toegang tot opgeslagen *artifacts* te bieden.
- **Kwetsbaarheidsscanner:** Een speciale Trivy-scanner [29] controleert periodiek de Docker- afbeeldingen die door de *workers* worden gebruikt op kwetsbaarheden. De resultaten worden ook opgeslagen in de objectopslag en een samenvatting is opgenomen in de AIBOM.

V. EVALUATIE EN RESULTATEN

AIBoMGen is geëvalueerd op drie belangrijke *metrics*: prestaties, opslag en beveiliging. De resultaten tonen aan dat het platform minimale prestaties en opslag overhead introduceert terwijl het robuuste beveiligingsmaatregelen biedt. Deze sectie biedt een analyse van deze *metrics*, samen met aanvullende inzichten in de bedreigingsanalyse. De experimentele scripts en resultaten zijn beschikbaar op Zenodo [30].

TABLE I
TRAINING EN AIBOM GENERATIETIJDEN VOOR VERSCHILLENDE
EPOCHS

Epochs	Training Time (s)	AIBOM Time (s)	Total Time (s)	AIBOM Overhead (%)
1	4.316	0.265	4.581	5.78
2	6.629	0.234	6.863	3.41
5	12.492	0.218	12.710	1.72
10	23.790	0.196	23.986	0.82
50	117.728	0.198	117.926	0.17
100	230.837	0.197	231.034	0.09

A. Prestaties

De prestatie-impact van AIBOM generatie is gemeten door trainingsoopdrachten in te dienen met variërende *epoch*-aantallen. Zoals weergegeven in Tabel I, bleef de tijd voor AIBOM generatie constant, ongeacht het aantal *epochs*, terwijl de trainingstijd lineair toenam. De tabel illustreert ook dat de *overhead* die door AIBOM generatie wordt geïntroduceerd verwaarloosbaar wordt voor langere trainingstaken, wat de schaalbaarheid bevestigt.

B. Opslagimpact

De opslagimpact van AIBOM generatie is geëvalueerd door de groottes van *artifacts* die tijdens de training zijn geproduceerd te analyseren. Fig. 3 toont aan dat de AIBOM grootte licht toeneemt met de datasetgrootte, maar klein blijft in verhouding tot de dataset- en modelgroottes. Ook toont Fig. 4 aan dat de AIBOM grootte minimaal is in vergelijking met de getrainde modelgrootte, zelfs voor grotere modellen. Deze resultaten bevestigen dat de opslagimpact verwaarloosbaar is.

C. Bedreigingsevaluatie

Een gedetailleerde bedreigingsanalyse is uitgevoerd om de weerstand van het platform tegen vervalsing en kwaadaardige activiteiten te evalueren. De evaluatie was gericht op het identificeren van potentiële bedreigingen, het implementeren van tegenmaatregelen, en het valideren van hun effectiviteit door middel van handmatige *tests* en integriteitsverificatie.

Het systeem adresseert een breed scala aan vervalsings-scenario's, waaronder modelmanipulatie, datasetmanipulatie,

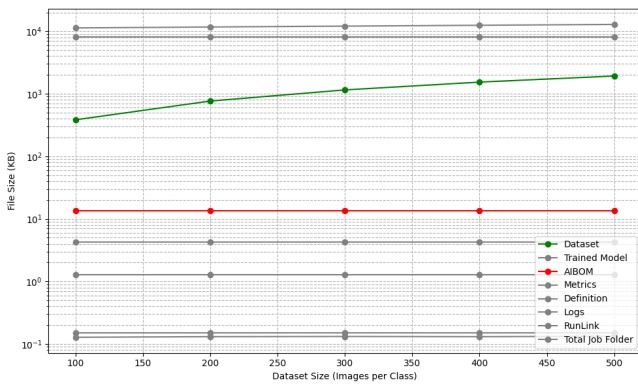


Fig. 3. Plot die artifact-bestandsgroottes voor verschillende datasetgroottes toont

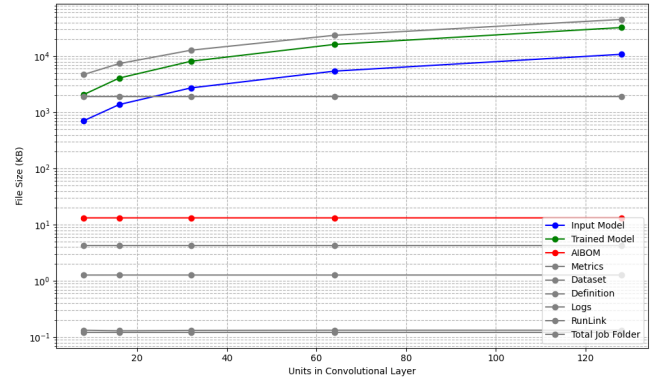


Fig. 4. Plot die artifact-bestandsgroottes voor verschillende invoermodel-groottes toont

interferentie met het trainingsproces, *artifact*-wijziging, en pogingen om de AIBOM generatie te omzeilen. Belangrijke bedreigingen, hun tegenmaatregelen, en de methoden die zijn gebruikt om ze te valideren, zijn hieronder samengevat:

- **Modelmanipulatie:** Kwaadaardige actoren kunnen onveilige Lambda lagen in modellen inbedden om willekeurige code uit te voeren [31].
 - **Tegenmaatregelen:**
 - * TensorFlow's `safe_mode=True` optie [32] wordt gebruikt om onveilige deserialisatie te blokkeren.
 - * Modellen worden geladen in geïsoleerde Docker-containers om de impact van potentiële aanvallen te beperken.
 - * Alle *modelartifacts* worden gehasht en vastgelegd in de AIBOM voor integriteitsverificatie.
 - **Testmethoden:**
 - * Geüploade modellen met onveilige Lambda lagen werden afgewezen vanwege veilige *model loading*.
 - * Geverifieerde *modelhashes* in de AIBOM kwamen overeen met het geüploade model.
- **Datasetmanipulatie:** Aanvallers kunnen kwaadaardige gegevens injecteren of datasetstructuren wijzigen.
 - **Tegenmaatregelen:**
 - * Validatie van datasetstructuur en inhoud (bv., controles op `.csv` bestanden).
 - * Veilige extractie van `.zip` bestanden om *resource*-uitputtingsaanvallen (bv. *zip-bombs*) te voorkomen.
 - **Testmethoden:**
 - * Uploaden van verkeerd gevormde datasets en geverifieerd dat het systeem deze afwees op basis van validatiecontroles.
- **Interferentie met het Trainingsproces:** Bedreigingen zijn onder andere het wijzigen van trainingsparameters of het injecteren van kwaadaardige code.

- **Tegenmaatregelen:**
 - * Trainingsopdrachten worden uitgevoerd in *read only*-Docker-containers met beperkte mogelijkheden.
 - * In-toto attestaties worden gebruikt om de integriteit van het trainingsproces te verifiëren.
 - * Omgevingsmetadata (bv. Python- en TensorFlow-versies) worden opgenomen in de AIBOM.
- **Testmethoden:**
 - * Geverifieerd dat trainingsopdrachten hun omgeving niet konden wijzigen vanwege *read only*-containerinstellingen.
 - * Geteste in-toto attestaties om te bevestigen dat alle vastgelegde acties werden uitgevoerd zoals verwacht.
- **Artifact-wijziging:** Aanvallers kunnen proberen om gegenereerde *artifacts* (bv. modellen, logs, *metrics*) te vervalsen.
 - **Tegenmaatregelen:**
 - * *Hash*-gebaseerde integriteitsverificatie wordt toegepast op alle *artifacts*.
 - * *Artifacts* worden opgeslagen in *MinIO-buckets*.
 - * Cryptografische ondertekening van AIBOMs en in-toto linkbestanden zorgt voor authenticiteit.
 - **Testmethoden:**
 - * Gewijzigde *artifacts* na generatie en geverifieerd dat *hash mismatches* werden gedetecteerd.
 - * Geteste *MinIO*-opslagverificatie-eindpunt.
- **Vervalsing van de AIBOM:** Een kwaadaardige actor zou de AIBOM kunnen vervalsen om valse informatie over het trainingsproces, invoer of uitvoer te verstrekken.
 - **Tegenmaatregelen:**
 - * Cryptografische ondertekening zorgt ervoor dat de AIBOM niet kan worden gewijzigd zonder detectie.
 - * *Hash*-verificatie stelt gebruikers in staat om de integriteit van alle invoer- en uitvoer-*artifacts* te bevestigen.
 - **Testmethoden:**
 - * Wijzigen van AIBOM bestanden en geverifieerd dat handtekeningcontroles faalden.
- **Ongeautoriseerde Toegang tot Artifacts of Systeemcomponenten:** Een kwaadaardige actor zou ongeautoriseerde toegang kunnen krijgen tot *artifacts* of systeemcomponenten, wat zou leiden tot datalekken of vervalsing.
 - **Tegenmaatregelen:**
 - * Azure OAuth wordt gebruikt voor veilige API oproepen, waardoor alleen geauthenticeerde en geautoriseerde gebruikers toegang tot het systeem kunnen krijgen.
 - **Testmethoden:**
 - * Beoordeelde authenticatie- en toegangscontrole-configuraties.

- * Er zijn geen penetratietests uitgevoerd, maar handmatige controles bevestigden de juiste toegangsbeperkingen.

- **Omzeilen van AIBOM Generatie:** Kwaadaardige actoren zouden kunnen proberen het AIBOM generatieproces te omzeilen.

- **Tegenmaatregelen:**

- * Afgedwongen AIBOM generatie voor elke trainingsopdracht.
- * Gecontroleerde API interacties om te voorkomen dat ongeautoriseerde wijzigingen plaatsvinden.

- **Testmethoden:**

- * Het systeem verwijdert met succes *artifacts* als de AIBOM generatie mislukt.
- * Beoordeelde API logs om te bevestigen dat alle interacties voldeden aan de afgedwongen workflows. Als de lay-out niet overeenkomt, faalt de verificatie van het linkbestand.

Overige risico's zijn onder andere potentiële kwetsbaarheden in afhankelijkheden, sleutelcompromittering, en ondetecteerbare kwaadaardige inhoud in geldige modellen of datasets. Geavanceerde aanvallen (bv. *exploits*, sleutelcompromittering, *adversarial data*, of privilege-escalatie) zijn niet getest vanwege beperkingen in middelen en scope.

Deze resultaten benadrukken de robuustheid van AIBOM-Gen bij het beveiligen van het AI ontwikkelingsproces.

VI. CONCLUSIE

Dit onderzoek heeft onderzocht hoe *AI Bill of Materials* (AIBOM) kan worden geïmplementeerd als een gestandaardiseerde en betrouwbare benadering om transparantie in AI ontwikkeling te waarborgen. De bevindingen beantwoorden vier belangrijke onderzoeksvragen:

A. Wat zijn de huidige vooruitgangen en uitdagingen in het onderzoek en de ontwikkeling van AIBOMs?

Het onderzoek heeft aangetoond dat hoewel AIBOMs aan terrein winnen, bestaande normen zoals CycloneDX en SPDX zich verder ontwikkelen om volledig AI-specifieke componenten te ondersteunen. Een grote uitdaging ligt in het ontbreken van veilige, geteste, geautomatiseerde, en gestandaardiseerde workflows voor het genereren van AIBOMs tijdens het trainingsproces. Huidige toepassingen geven vaak prioriteit aan flexibiliteit boven veiligheid, waardoor er hiaten ontstaan in betrouwbaarheid en naleving.

B. Op welke manieren kunnen AIBOMs ondersteuning bieden bij de naleving van AI-regelgeving, zoals de AI Act, en andere opkomende standaarden?

AIBOMs kunnen een belangrijke rol spelen bij het voldoen aan de transparantie-eisen van de AI Act. Het biedt een verifieerbaar overzicht van datasets, afhankelijkheden en trainingsprocessen, waarmee organisaties naleving kunnen aantonen. Het gebrek aan wijdverspreide adoptie, en vooral *tooling* voor AIBOMs blijft echter een obstakel.

C. Hoe kan vertrouwen effectief worden geïntegreerd in het AIBOM-kader om transparantie, integriteit en ethische zorgen te waarborgen?

De PoC heeft aangetoond dat vertrouwen kan worden geïntegreerd door middel van cryptografische attestaties, integriteitscontroles en gecontroleerde uitvoerings-omgevingen. Een belangrijke les is dat veel industriestandaard trainingsworkflows afhankelijk zijn van *custom* Docker-containers en trainingsscripts, die niet van nature veilig of getest zijn. Er is behoefte aan *secure-by-design* AI ontwikkelingsplatforms.

D. Welke methoden en technologieën kunnen worden gebruikt om de generatie van AIBOMs te automatiseren terwijl hun betrouwbaarheid en veiligheid worden gewaarborgd?

De PoC toonde aan dat automatisering haalbaar is met behulp van tools zoals CycloneDX Python-bibliotheek, in-toto (of andere attestatietools), en gecontaineriseerde workflows. De afhankelijkheid van opkomende normen en onvolledige bibliotheek-ondersteuning (bijv. CycloneDX Python-bibliotheek) benadrukt ook de noodzaak van verdere ontwikkeling op dit gebied.

Ter conclusie benadrukt deze scriptie het potentieel van AIBOMGen (en soortgelijke platforms) als een fundamentele stap richting het bouwen van veilige en transparante AI ecosystemen, waarmee naleving van regelgevende kaders zoals de EU AI Act mogelijk wordt.

TOEKOMSTIG WERK

Toekomstig werk zal zich richten op het uitbreiden van de mogelijkheden van het platform. Dit omvat ondersteuning voor aanvullende AI frameworks en integratie met opkomende normen zoals SPDX 3.0. Technologieën zoals *Trusted Platform Modules* (TPM) en *Trusted Execution Environment* (TEE) kunnen ook worden verkend. Deze technologieën kunnen hardware-gebaseerde beveiliging garanties bieden, waarmee wordt gewaarborgd dat de trainingsomgeving en processen vervalsingsbestendig en verifieerbaar blijven, terwijl veilige uitvoering van *custom* trainingsscripts mogelijk wordt gemaakt.

ERKENNING

Ik wil mijn promotoren, prof. dr. Bruno Volckaert en dr. Merlijn Sebrechts, bedanken voor hun begeleiding en advies, en mijn begeleider, ing. Jordi Thijsman, voor zijn waardevolle feedback en ondersteuning. Ik ben ook dankbaar aan imec ID-Lab voor het bieden van de mogelijkheid voor dit onderzoek.

REFERENCES

- [1] B. Ghosh, “AI Bill of Materials (AI BOM),” Jul. 2024. [Online]. Available: <https://medium.com/@bijit211987/ai-bill-of-materials-ai-bom-80d48f9d75e0>
- [2] R. Hart, “Clearview AI—Controversial Facial Recognition Firm—Fined \$33 Million For ‘Illegal Database’,” 2024, section: Business. [Online]. Available: <https://www.forbes.com/sites/roberthart/2024/09/03/clearview-ai-controversial-facial-recognition-firm-fined-33-million-for-illegal-database/>
- [3] D. Thiel, “Investigation Finds AI Image Generation Models Trained on Child Abuse,” Dec. 2023. [Online]. Available: <https://cyber.fsi.stanford.edu/news/investigation-finds-ai-image-generation-models-trained-child-abuse>
- [4] T. Krantz, “What Is Data Poisoning?” Dec. 2024. [Online]. Available: <https://www.ibm.com/think/topics/data-poisoning>
- [5] Dremio, “Adversarial Attacks in AI,” 2025. [Online]. Available: <https://www.dremio.com/wiki/adversarial-attacks-in-ai/>
- [6] S. Creese, “Why we need to reflect on the need for cybersecurity of AI,” Dec. 2023. [Online]. Available: <https://www.weforum.org/stories/2023/12/cybersecurity-ai-ethics-responsible-innovation/>
- [7] E. Ozoani, M. GerChick, and M. Mitchell, “Model Cards,” 2022. [Online]. Available: <https://huggingface.co/blog/model-cards>
- [8] Hugging Face, “Hugging Face,” May 2025. [Online]. Available: <https://huggingface.co/>
- [9] D. Bardenstein, “Manifest: AIBOMs, Transparency into AI,” 2023. [Online]. Available: <https://www.manifestcyber.com/aibom>
- [10] TechWorks and NquiringMinds, “Trustworthy AI – Practical Collaborative Engineering,” 2023. [Online]. Available: <https://www.techworks.org.uk/wp-content/uploads/2024/01/Engineering-Trustworthy-AI.pdf>
- [11] TechWorks, “Compliance with the EU AI Act The Techworks Trusted AI Bill Of Materials (TAIBOM) project,” Jun. 2024. [Online]. Available: <https://www.techworks.org.uk/ai-taibom/compliance-with-the-eu-ai-act-the-techworks-trusted-ai-bill-of-materials-taibom-project>
- [12] imec, “GPULab — imec iLab.t documentation.” [Online]. Available: <https://doc.ilabt.imec.be/ilabt/gpulab/>
- [13] EU, “AI Act | Shaping Europe’s digital future,” Sep. 2024. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai>
- [14] —, “High-level summary of the AI Act | EU Artificial Intelligence Act,” Feb. 2024. [Online]. Available: <https://artificialintelligenceact.eu/high-level-summary/>
- [15] OWASP Foundation, “CycloneDX - Software Bill of Materials (SBOM),” Jul. 2024. [Online]. Available: <https://cyclonedx.org/>
- [16] Linux Foundation, “SPDX,” 2023. [Online]. Available: <https://spdx.dev/>
- [17] OWASP Foundation, “CycloneDX - Machine Learning Bill of Materials (ML-BOM),” 2024. [Online]. Available: <https://cyclonedx.org/capabilities/mlbom/>
- [18] Linux Foundation, “SPDX AI Github Specification 3.0.1,” 2024. [Online]. Available: <https://spdx.github.io/spdx-spec/v3.0.1/model/AI/AI/>
- [19] —, “Implementing AI Bill of Materials (AI BOM) with SPDX 3.0,” 2024. [Online]. Available: <https://www.linuxfoundation.org/research/ai-bom>
- [20] D. Bardenstein, “Whitepaper: Driving AI Transparency:

- the AI Bill of Materials (Manifest)," 2023. [Online]. Available: manifestcyber.com
- [21] W. Vandendriessche, "AIBoMGen," May 2025. [Online]. Available: <https://zenodo.org/records/15536533>
- [22] in-toto, "in-toto," 2025. [Online]. Available: <https://in-toto.io/>
- [23] —, "in-toto 3.0.0 documentation," 2025. [Online]. Available: <https://in-toto.readthedocs.io/en/latest/>
- [24] FastAPI, "FastAPI," 2025. [Online]. Available: <https://fastapi.tiangolo.com/>
- [25] A. Solem and contributors, "Celery 5.5.2 documentation," 2023. [Online]. Available: <https://docs.celeryq.dev/en/stable/>
- [26] RabbitMQ, "RabbitMQ: One broker to queue them all," 2025. [Online]. Available: <https://www.rabbitmq.com/>
- [27] Oracle, "MySQL," 2025. [Online]. Available: <https://www.mysql.com/>
- [28] MinIO, "MinIO - S3 Compatible Storage for AI," 2025. [Online]. Available: <https://min.io>
- [29] Aqua Security, "Trivy," 2025. [Online]. Available: <https://trivy.dev/v0.62/>
- [30] W. Vandendriessche, "AIBoMGen-experiments Scripts & Results," May 2025. [Online]. Available: <https://zenodo.org/records/15505280>
- [31] T. Righi, "TensorFlow Remote Code Execution with Malicious Model," Oct. 2022. [Online]. Available: <https://splint.gitbook.io/cyberblog/security-research/tensorflow-remote-code-execution-with-malicious-model>
- [32] TensorFlow, "tf.keras.models.load_model," Jun. 2024. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/models/load_model

Contents

Abstract	v
List of Figures	xxiii
List of Tables	xxv
List of Acronyms	xxvi
List of Code Fragments	xxx
1 Introduction	1
1.1 Context and Motivation	1
1.2 Research Objectives and Questions	3
1.3 Thesis Structure	4
2 Foundations of AI Bill Of Materials	5
2.1 AI Bill Of Materials (AIBOM)	5
2.1.1 The SBOM	5
2.1.2 Model Cards on Hugging Face	6
2.1.3 The Users	7
2.1.4 Manifest example: AIBOM for an Open-Source Hugging Face Model	7
2.1.5 Example from Proof-of-Concept application	9
2.1.6 Essential Components of an AIBOM	10
2.1.7 What Should Not Be Included in an AIBOM?	13
2.1.8 Model Lineage	14
2.2 Existing BOM Standards	15
2.2.1 CycloneDX	15
2.2.2 SPDX	19
2.3 AI Compliance	22
2.3.1 The AI Act	22
2.3.2 AI Regulation: EU vs U.S.	24

2.3.3	Implementation of AIBOM compliance with AI Act	25
2.4	Trustable AI projects	26
3	System Design & Architecture	29
3.1	Naive approach	30
3.2	Ensuring Trust-ability	31
3.2.1	Confidential Computing as a potential solution	32
3.2.2	AIBoMGen's Approach: Trustability through Trusted APIs	33
3.3	High-level Architecture	34
3.3.1	Central Task Scheduler	35
3.3.2	Data Storage	35
3.3.3	API Gateway	36
3.3.4	Worker Nodes	36
3.3.5	Vulnerability Scanner	36
3.3.6	Monitoring and Management Interface	36
3.4	User flows	37
3.4.1	AI developer flow	37
3.4.2	AI verifier flow	39
4	Technologies	42
4.1	Technologies used for the PoC, AIBoMGen PaaS	42
4.1.1	Containerization: Docker	42
4.1.2	Backend programming language: Python	42
4.1.3	Task scheduler: Celery	43
4.1.4	API Gateway: FastAPI	44
4.1.5	Authentication: Microsoft Entra ID (Azure Active Directory)	44
4.1.6	Object Storage: MinIO	45
4.1.7	ML framework: TensorFlow	45
4.2	Alternative technologies for scalable deployments	45
4.2.1	Kubeflow	45
4.2.2	MLflow	46
4.2.3	Firecracker	46
4.2.4	Kata Containers	46
4.2.5	Confidential computing platforms	47
4.2.6	Ray	47
4.3	Technologies used for the trusted AIBOM generation pipeline	47
4.3.1	CycloneDX python library	47

4.3.2	Trivy for vulnerability scanning	48
4.3.3	in-toto	48
4.3.4	Cryptography hazmat library	48
5	Implementation	49
5.1	Implementation details of the PoC, AIBoMGen PaaS	49
5.1.1	Celery configuration	49
5.1.2	Worker configuration	52
5.1.3	FastAPI configuration	55
5.1.4	Scanner configuration	60
5.1.5	MinIO utilities	61
5.1.6	Zip utilities	62
5.2	Implementation of the AIBOM generation pipeline	63
5.2.1	Metadata extraction	63
5.2.2	Environment details extraction	64
5.2.3	Building the AIBOM	64
5.3	Implementation details of the Integrity and Security Measures	65
5.3.1	Use of in-toto for supply chain integrity and verification	66
5.3.2	BOM signature and verification	67
5.3.3	Logging	68
5.3.4	Authentication with Microsoft Entra ID	69
5.3.5	Summary of security measures	69
5.4	Explanation of the AIBOM verification endpoints	70
5.4.1	Verify in-toto link file	70
5.4.2	Verify AIBOM	71
5.4.3	Verify MinIO artifacts	72
5.4.4	Verify uploaded file hash	73
5.5	Front-end dashboard for demo purposes	75
5.5.1	Front end in React (Next.js)	75
5.5.2	Front-end functionalities	75
6	Evaluation	78
6.1	Basic Evaluation	78
6.1.1	Successful Training Jobs	78
6.2	Performance	80
6.3	Storage Impact	82
6.3.1	Impact of Dataset Size	82

6.3.2	Impact of Model Size	84
6.3.3	Impact of Epoch Count	85
6.4	Security & Threat Evaluation	88
6.4.1	Testing Approach:	88
6.4.2	Summary of Findings:	88
Conclusion		89
Future Work		91
Societal Reflection		93
References		94
Appendices		101
A	SPDX 3.0 Compliance with EU Artificial Intelligence Act	102
B	Example of CycloneDX-Like AIBOM for RedPajama-INCITE-Instruct-3B . .	108
C	Detailed Threat Analysis	111
D	in-toto Layout and Link File Example	121
E	AIBoMGen PoC Proposal and Resulting AIBOM Example	124

List of Figures

2.1	Example of model lineage, showing the relationships between a base model, fine-tuning steps, and modifications [2]	15
2.2	CycloneDX Object Model with object types arranged in order and with their object properties [3]	18
2.3	SPDX v2.3 document overview [4]	19
2.4	SPDX 3.0 profile overview [5]	20
2.5	SPDX 3.0 AIBOM structure of an example [5]	21
2.6	Risk classification of AI systems under the AI Act	23
3.1	Use case diagram: the AIBoMGen system	29
3.2	Naive AIBoMGen architecture	31
3.3	AIBoMGen architecture	35
3.4	Sequence diagram: AI Developer flow	39
3.5	Sequence diagram: AI Verifier flow	41
5.1	Celery Setup: Interaction between FastAPI, RabbitMQ, MySQL, and Celery workers	50
5.2	AIBOM and link file generation	68
5.3	State diagram: Verification of an in-toto link file	71
5.4	State diagram: Verification of BOM and associated in-toto link file	72
5.5	State diagram: Verification of MinIO artifacts against the link file	73
5.6	State diagram: Verification of an uploaded file's hash against the link file	74
5.7	Dashboard Landing Page: Welcome screen	76
5.8	Job Submission Page: Users can submit new training jobs with required files and parameters	76
5.9	My Jobs Page: Overview of user's submitted jobs and their statuses	77
5.10	Worker Statistics Page: Real-time statistics of active Celery workers	77
6.1	Training and validation accuracy and loss for the MNIST image classification task	80

6.2	Training time and AIBOM overhead for different epochs	81
6.3	AIBOM overhead as percentage of total time	82
6.4	File sizes of all artifacts versus dataset size (log scale)	83
6.5	File sizes of all artifacts versus model size (log scale)	85
6.6	File sizes of all artifacts versus number of epochs (log scale)	86
6.7	AIBOM and metrics file sizes versus number of epochs	87

List of Tables

6.1	Training and AIBOM Generation Times for Different Epochs	81
6.2	Artifact Sizes for Different Dataset Sizes - Part 1	83
6.3	Artifact Sizes for Different Dataset Sizes - Part 2	83
6.4	Artifact Sizes for Different Model Sizes - Part 1	84
6.5	Artifact Sizes for Different Model Sizes - Part 2	84
6.6	Artifact Sizes for Different Epoch Counts - Part 1	85
6.7	Artifact Sizes for Different Epoch Counts - Part 2	86
A.1	EU AI Act Compliance with SPDX 3.0	103
C.1	Threats and their testing coverage in the evaluation	118

List of Acronyms

AAD Azure Active Directory. , 44, 57, 69

AI Artificial Intelligence. v, , 1–8, 10, 13, 14, 18–32, 34, 37–40, 43, 47, 55, 89, 90, 93, 124

AIBOM AI Bill of Materials. v, xxiii–xxv, , xxx, 1–10, 12–14, 19, 21, 22, 25, 26, 28–40, 45, 47–49, 52, 54, 55, 60, 63–65, 67, 69–71, 78–85, 87–91, 93, 108–112, 114–117, 119, 120, 123–127

API Application Programming Interface. , 16, 30–34, 36–40, 42–47, 55–57, 59, 61, 62, 67, 69, 70, 75, 78, 91, 114–116, 119

ASGI Asynchronous Server Gateway Interface. , 56

AWS Amazon Web Services. , 35, 46, 111

BOM Bill of Materials. xxiii, , 5, 14–16, 19, 21, 27, 40, 47, 48, 54, 57, 64, 65, 67, 71, 72, 84, 91, 112, 113, 124

BSD Berkeley Software Distribution. , 43

CIFAR Canadian Institute for Advanced Research. , 80

CLI Command Line Interface. , 27, 60

CORS cross-origin resource sharing. , 44, 57

CPU Central Processing Unit. , 52, 64, 78

CRA Cyber Resilience Act. , 22

CSAM Child Sexual Abuse Material. , 2

CSV comma-separated values. , 53, 62, 79

DA Data Act. , 22

DGA Data Governance Act. , 22

DMA Digital Markets Act. , 22

DSA Digital Services Act. , 22

ECMA European Computer Manufacturers Association. , 15

EU European Union. v, , 22–25, 90, 93

GDPR General Data Protection Regulation. , 22

GPAI General-Purpose AI. , 23

GPT Generative Pre-training Transformer. , 10

GPU Graphics Processing Unit. , 30, 32, 36, 45, 52, 64, 78

HSM Hardware Security Modules. , 119

HTTP Hypertext Transfer Protocol. , 57

IEC International Electrotechnical Commission. , 19

ISO International Standardization Organization. , 19

IT Information Technology. v, , 93

JPG Joint Photographic experts Group. , 62

JSON JavaScript Object Notation. , 8, 13, 53, 54, 56, 65

JWT JSON Web Token. , 69

Llama Large Language Model Meta AI. , 10, 14

LLM Large Language Model. iii, , 8, 22

LoRA Low-rank adaptation. , 14

ML Machine Learning. , 1, 6, 20, 42, 45, 46, 124

ML-BOM Machine Learning Bill of Materials. , 18, 19

MNIST Modified National Institute of Standards and Technology. , 79

MSAL Microsoft Authentication Library. , 75

NIS2 Network and Information Systems Directive 2022. , 22

NLP Natural Language Processing. , 6

NVML NVIDIA Management Library. , 64

OWASP Open Worldwide Application Security Project. , 15

PaaS Platform as a Service. , 31, 45, 49

png Portable Network Graphics. , 62, 79

PoC Proof-of-Concept. v, , xxx, 3, 4, 9, 10, 13, 28, 29, 31–33, 42, 43, 45, 46, 48, 89, 90, 93, 124–127

RAM Random Access Memory. , 78

RCE Remote Code Execution. , 111

SBOM Software Bill of Materials. v, , 1, 4–6, 15, 16, 26

SCA Software Component Analysis. , 15

SDG Sustainable Development Goal. , 93

SDGs Sustainable Development Goals. , 93

SDK Software Development Kit. , 27

SEV Secure Encrypted Virtualization. , 33

SGX Software Guard Extensions. , 33

SME Small and Medium-sized Enterprise. , 24

SPDX System Package Data Exchange. xxiii, , 14, 19–21, 25, 26, 91

SSO single sign-on. , 44

TAIBOM Trustable AI Bill of Materials. , 26

TEE Trusted Execution Environment. , 32, 33, 47

TPM Trusted Platform Module. , 27, 33, 92, 119

U.S. United States. , 24, 25

UI User Interface. , 43, 46, 75

URI Uniform Resource Identifier. , 16

URL Uniform Resource Locator. , 57, 59

VM Virtual Machine. , 46, 47

W3C World Wide Web Consortium. , 27, 92

List of Code Fragments

5.1	Celery broker and backend configuration in docker-compose	51
5.2	Celery broker and backend configuration	51
5.3	Celery queues and routing configuration	52
5.4	Key Celery worker configuration options	53
5.5	Minimal worker service configuration in docker-compose	55
5.6	FastAPI Dockerfile for the API service	56
5.7	FastAPI lifespan handler for startup initialization	56
5.8	Developer endpoint for job submission	58
5.9	Developer endpoint for job status	59
5.10	Developer endpoint for artifact retrieval	59
5.11	Periodic scanner task for vulnerability scanning	61
5.12	Zip utility for safe validation and extraction	63
5.13	Building, signing, and uploading the AIBOM	65
5.14	Generating and signing an in-toto link file	66
B.1	CycloneDX-Like AIBOM for RedPajama-INCITE-Instruct-3B - Part 1	108
B.2	CycloneDX-Like AIBOM for RedPajama-INCITE-Instruct-3B - Part 2	109
B.3	CycloneDX-Like AIBOM for RedPajama-INCITE-Instruct-3B - Part 3	110
D.1	in-toto Layout Example - Part 1	121
D.2	in-toto Layout Example - Part 2	122
D.3	in-toto Link File Example	123
E.1	CycloneDX AIBOM generated by AIBoMGen PoC - Part 1 (Training Environment and Dataset)	125
E.2	CycloneDX AIBOM generated by AIBoMGen PoC - Part 2 (Trained Model and Dependencies)	126
E.3	CycloneDX AIBOM generated by AIBoMGen PoC - Part 3 (Metadata and Tooling)	127

1

Introduction

1.1 Context and Motivation

Artificial Intelligence (AI) is becoming increasingly integrated in modern IT systems, with companies across industries embedding AI into their products and services [6]. However, alongside this rapid adoption comes an urgent need to address security, transparency, and compliance challenges. Traditional cybersecurity methods are often insufficient for AI-specific risks such as data poisoning¹ and adversarial manipulation² [9]. The lack of structured documentation, opaque decision-making processes, and vulnerabilities to malicious attacks underscore the need for new mechanisms to ensure AI integrity and trustworthiness.

One solution to this challenge is the AI Bill of Materials (AIBOM), an adaptation of the Software Bill of Materials (SBOM), which provides a structured inventory of all components involved in AI model development. By documenting datasets, dependencies, configurations, and other key factors, AIBOMs enhance transparency and traceability, enabling organizations to secure their AI pipelines [10].

¹Data poisoning is a type of cyberattack where threat actors manipulate or corrupt the training data used to develop AI and ML models [7].

²Adversarial manipulation involves crafting inputs designed to deceive AI models, causing them to produce incorrect or unintended outputs [8].

1 Introduction

However, transparency alone is insufficient without trust [11]. The value of an AIBOM depends on its reliability, which requires mechanisms to ensure the integrity and authenticity of the documented information. Without trust, AIBOMs cannot effectively support compliance, security, or accountability [12]. Establishing trust involves implementing secure workflows and mechanisms, such as cryptographic attestations, to prevent manipulation and guarantee data integrity. This is particularly critical in addressing risks where malicious actors can compromise the training process.

High-profile cases, such as the Clearview AI scandal [13][14], where biometric data were collected without consent, highlight the risks of opaque AI systems. Similarly, a Stanford University investigation [15] found that certain AI models were trained on datasets containing Child Sexual Abuse Material (CSAM). These incidents reinforce the need for verifiable and trustworthy documentation of AI components.

Many AI development environments, such as cloud-based platforms, and research-focused infrastructures, such as GPULab³, do not have a standardized procedure in place to ensure model transparency and trustability. This leads to a lack of structured and trusted documentation for training data, software tools, and hardware configurations. As a result, it becomes difficult to demonstrate that AI models meet the requirements of the AI Act [17][18], a proposed European regulation that aims to ensure the safe and ethical development and deployment of AI systems. Organizations must adopt new methodologies to ensure that their models meet the expectations of transparency and accountability [19].

Existing solutions, such as model cards provided by platforms like Hugging Face, attempt to address transparency concerns but remain insufficient. Model cards offer descriptive metadata on the intended use, limitations, and biases, but do not provide a comprehensive, verifiable record of the entire AI development process [20]. A more robust solution is needed, one that ensures AI models are trained, documented, and deployed in a structured and secure manner and provides trust.

³GPULab is a research infrastructure provided by imec, offering GPU resources for machine learning and AI experiments [16].

1.2 Research Objectives and Questions

This research investigates how AIBOMs can be implemented as a standardized approach to ensure transparency, trust, and security in the development of AI. The research includes a Proof-of-Concept (PoC) application to demonstrate possible methods for generating trustable AIBOMs while training. The key research questions are as follows:

- **What are the current advancements and challenges in the research and development of AIBOMs?**

Assessing existing methodologies, technologies, and gaps in the field.

- **In what ways can AIBOMs support compliance with AI regulations, such as the AI Act, and other emerging standards?**

Examining the role of AIBOMs in ensuring legal, ethical, and regulatory alignment in AI development.

- **How can trust be effectively integrated into the AIBOM framework to ensure transparency, integrity, and ethical concerns?**

Exploring mechanisms to ensure the trustworthiness and accountability of AI models documented via AIBOMs. Implementing a PoC application.

- **What methods and technologies can be used to automate the generation of AIBOMs while ensuring their reliability and security?**

Investigating approaches to streamline the creation of AIBOMs in a trustworthy, scalable, and automated manner. Implementing a PoC application.

1.3 Thesis Structure

The main content of this thesis is organized into the following chapters.

- **Chapter 2: Foundations of AIBOM** – Explores the concept of AIBOM, its relationship with SBOM, and existing documentation practices such as model cards, highlighting their limitations. Discusses the regulatory landscape, including the AI Act and its implications for AI development.
- **Chapter 3: System Design and Architecture** – Discusses the "Trust" issue. Describes the proposed framework for the AIBoMGen Proof-of-Concept (PoC), including its components, use cases, and structure.
- **Chapter 4: Technologies** – Provides an overview of the key technologies used in the development of AIBoMGen, including alternative technologies for more scalable and production-ready versions.
- **Chapter 5: Implementation** – Details the technical implementation of the AIBoMGen PoC, including tools, technologies, and methodologies used.
- **Chapter 6: Evaluation** – Presents the evaluation of the AIBoMGen PoC through experiments with tampering, assessing its effectiveness. Also evaluates storage and performance overhead of the system.

2

Foundations of AI Bill Of Materials

2.1 AI Bill Of Materials (AIBOM)

As mentioned in the introduction, an AIBOM, which expands the traditional scope of SBOM, is becoming essential for the management of more complex systems and the broader adoption of AI. Capturing all relevant design decisions and development dependencies in AI system development is now considered a best practice by industry leaders and regulatory bodies, such as the European Commission under the AI Act [21][17].

2.1.1 The SBOM

The SBOM is a strong foundation on which to build. It is often described as a “list of ingredients” for software products. They provide transparency into the software and the risks in that software that can lead to cyber breaches or copyright threats. SBOMs help organizations identify and fix risks before selling them to customers. Teams can evaluate and buy more secure software. Regulators and auditors have the ability to accurately assess the maturity, security, and compliance of a given product [2]. Ultimately, SBOMs build trust in the underlying software. Transparency yields trust, and trust yields security [22].

SBOMs have already been used very widely in both Europe, India, and the United States, such as organizations from the White House to the food and drug administration [2]. Recently, organizations responsible for SBOM standards have started to adapt them to AI systems. Both of the leading SBOM formats now include AI-specific components [23][24], allowing for the development of structured AIBOM. The Section 2.2 about BOM standards will explore these developments in more detail.

2 Foundations of AI Bill Of Materials

It is important to emphasize that AIBOMs differ significantly from SBOMs. While SBOMs are derived from source code, AIBOM are inherently linked to machine learning models and can often be generated during the training process, although no actual standards for generation exist. Unlike software components, the properties of AI models do not directly correspond to executable code, making them less suitable for identifying vulnerabilities directly or indirectly [2].

2.1.2 Model Cards on Hugging Face

Hugging Face is a leading open-source platform for AI/ML models, offering a vast repository of pre-trained models, datasets, and tools for developers and researchers. It has become a central hub for the AI community, enabling collaboration, sharing, and innovation in machine learning. The platform supports a wide range of applications, from Natural Language Processing (NLP) to computer vision, and provides user-friendly interfaces for deploying and fine-tuning models [25].

The Model cards structure essential information describing an AI/ML model. Developers typically provide these model cards alongside the source code and model weights. They are widely used in industry and research, particularly in Hugging Face. However, model cards have four major limitations:

- **Lack of a standard format:** Different model developers follow different documentation practices.
- **Not machine-readable:** Model cards are typically written in human-readable formats, making automated processing difficult.
- **Lack of automation:** Model cards are created manually, which introduces inconsistencies and potential errors.
- **Cannot be trusted:** Due to the lack of automation and standardization, there is no guarantee that the information in model cards accurately reflects the true state of the model.

These limitations could be addressed by adopting AIBOMs as a standardized, machine-readable, and automated documentation method [2].

Efforts to improve model documentation have already begun. Hugging Face has introduced an updated model card template and a Model Card Creator Tool to ensure consistency and streamline documentation [20]. Although these initiatives are a step in

2 Foundations of AI Bill Of Materials

the right direction, they should be considered a stepping stone towards fully standardized AIBOMs. By integrating model cards into AIBOMs, documentation can become more structured, transparent, and compliant with regulatory and ethical standards.

2.1.3 The Users

The primary stakeholders of AIBOMs can be categorized into three groups [2]:

- **AI Model Developers:** Developers are responsible for maintaining AIBOMs for the models they create or use. This provides a clear inventory of dependencies, training data, and other critical components, enabling developers to ensure compliance with regulatory standards and industry best practices. Additionally, AIBOMs can streamline debugging and troubleshooting when issues arise due to upstream dependencies or unexpected behaviors in the model.
- **Consumers of AI Models:** Organizations and individuals using AI models can leverage AIBOMs to select the most reliable, transparent, and well-documented tools. Beyond technical considerations, AIBOMs play a crucial role in addressing ethical concerns. For example, understanding the training data and methodologies used in a model can help consumers identify and mitigate potential biases, ensuring that the AI system is suitable for sensitive applications such as hiring, healthcare, or legal decision-making. Transparency in AIBOMs empowers consumers to make informed choices and promotes trust in AI technologies.
- **Policymakers and Regulators:** Regulatory bodies can use AIBOMs to enforce compliance with AI-related laws and policies. By providing a standardized, machine-readable format, AIBOMs enable efficient audits and assessments of AI systems. This is particularly important for high-risk applications, where transparency and accountability are critical to safeguarding public trust and ensuring ethical use of AI.

2.1.4 Manifest example: AIBOM for an Open-Source Hugging Face Model

To illustrate how an AIBOM can be structured, this section presents an example from the Manifest Cyber whitepaper and github [2] [26]. The example uses the open-source RedPajama-INCITE-Instruct-3B model, which is hosted on Hugging Face. The AIBOM follows the CycloneDX format, incorporating key metadata, dependencies, licensing,

2 Foundations of AI Bill Of Materials

and ethical considerations.

Manifest Cyber

Manifest Cyber is a cybersecurity company specializing in AI governance, risk assessment, and compliance solutions [2]. The company focuses on ensuring transparency and security in AI development by providing frameworks and tools that help organizations document, audit, and manage AI models throughout their life cycle. Their research and white papers contribute to the establishment of industry standards for AI security and compliance.

Overview of the Model

The RedPajama-INCITE-Instruct-3B is a 2.8 billion parameter Large Language Model (LLM) developed by Together Computer. It is designed for text generation tasks and is fine-tuned from the RedPajama-INCITE-Base-3B-v1 model. The model is licensed under Apache 2.0 and includes documentation on intended use, limitations, and potential risks.

CycloneDX-Like AIBOM Representation

The complete JSON structure representing the AIBOM for RedPajama-INCITE-Instruct-3B is included in Appendix B, divided into three parts. Below is an explanation of the fields in each part.

Part 1: General Metadata B - This section includes the general metadata of the model, such as its name, version, description, and authorship. It also specifies the timestamp of the AIBOM creation and the tools used to generate it.

Part 2: Model Details and Ethical Considerations B - This section provides detailed information about the model, including its architecture, datasets used for training, input and output formats, and ethical considerations. The ethical considerations highlight the responsibility of the end user to ensure the model is used in a responsible and ethical manner.

Part 3: Dependencies and External References B - This section lists the dependencies required by the model, such as PyTorch and Transformers libraries, along with their descriptions. It also includes external references, such as the model source, website, and community links.

2 Foundations of AI Bill Of Materials

Limitations of the Manifest Example Proposal

While the Manifest example is a strong starting point, it has several key limitations:

- **Lack of Trusted Provenance and Integrity:** The manifest relies on declarative documentation, which can be manually edited and is not cryptographically verifiable. There are no artifact hashes or digital signatures, making it impossible to verify the authenticity or integrity of the documented files.
- **Insufficient Training Environment and Attestation:** The example does not capture the full training environment (hardware, container, software versions) or provide any attestation of the environment's integrity.
- **Dependence on Manual Developer Input:** Key information, such as intended use, misuse scenarios, and limitations, relies solely on the developer's honesty and accuracy. This manual approach increases the risk of inconsistencies or discrepancies between the documented information and the actual properties of the model.

2.1.5 Example from Proof-of-Concept application

To address the limitations identified in the Manifest proposal, a Proof-of-Concept (PoC) platform was developed, that also generates AIBOMs, AIBoMGen. The resulting AIBOM, shown in Appendix E, demonstrates several improvements:

- **Trusted Attestation:** The PoC integrates in-toto attestation, providing cryptographic evidence of the training process and environment.
- **Training Environment Capture:** The AIBOM includes detailed hardware, OS, container, and software version information, as well as job metadata.
- **Artifact Hashes for Integrity:** All datasets, models, and metrics are hashed (SHA-256), enabling users to verify the integrity of each artifact.
- **Digital Signatures:** The AIBOM itself is signed, and signature fields are included (as properties due to CycloneDX python library limitations).
- **Dependencies:** The AIBOM encodes dependencies between the trained model, dataset, and training environment.
- **Automated Generation:** The process is automated, reducing the risk of human error or manipulation.

2 Foundations of AI Bill Of Materials

Note: The modelCard component used in the Manifest example was not used in the PoC AIBOM because it is not yet implemented in the CycloneDX python library. Properties were used instead. More details can be found in Section 4.3.1 and 5.2.3.

2.1.6 Essential Components of an AIBOM

This section outlines the key components of a standardized AIBOM, based on the Manifest Cyber proposal [26], with clear indication of additional elements introduced in the AIBoMGen PoC (*PoC addition*). These additions address the limitations identified earlier and are recommended for future AIBOM standards.

A standardized AIBOM should include structured data that capture the critical aspects of an AI model:

Model Details

- **Model Name:** The official name of the model.
- **Model Version:** The version number to track updates.
- **Model Type:** The category of AI model (e.g., text generation, image classification).
- **Model Developer:** The organization or individual responsible for developing the model.
- **Source:** A link to where the model can be accessed (e.g., Hugging Face, GitHub).
- **References:** Related documentation, articles, or technical reports.

Model Architecture

- **Base Model:** The foundational model upon which the AI system is built (if applicable).
- **Parent Model:** The model from which the current model was fine-tuned.
- **Model Architecture:** The technical framework used, such as GPT-4, Llama, or Stable Diffusion.
- **Architecture Family:** The broader family of architectures to which the model belongs.

2 Foundations of AI Bill Of Materials

- **Input/Output Data Format:** The expected data format for input and the output generated by the model.

Training Data and Datasets

- **Dataset Names:** The datasets used to train the model, including versions.
- **Dataset Sources:** Where the data was obtained, with links if available.

Training Environment (*PoC addition*)

- **Operating System:** The OS used for training (e.g., Ubuntu 22.04, Windows 11).
- **Hardware Specifications:** Information about CPUs, GPUs, RAM, and storage used.
- **Python Version:** The version of Python used during training.
- **Framework & Library Versions:** Versions of AI frameworks such as PyTorch or TensorFlow.
- **Containerization & Virtualization:** Whether Docker or other virtual environments were used.

Software Dependencies and Supply Chain Risks

- **Software Libraries:** A list of external libraries required to run the model.
- **Potential Vulnerabilities:** Any known security vulnerabilities related to dependencies. This can be a summary or reference to a vulnerability database or file.

Licensing and Intellectual Property

- **Model License:** The license that governs the use of the model (e.g., Apache 2.0).
- **Dataset License:** The licensing terms for the datasets used.
- **Terms of Use:** Restrictions on how the model can be used, including commercial limitations.

2 Foundations of AI Bill Of Materials

Intended Usage and Limitations (*Optional*)

This was added by Manifest, but as said earlier, it is not a good idea to rely on the honesty of the developer. Instead the AIBOM, should be automatically generated on a trusted platform and not rely on the developer to provide this information.

- **Intended Use Cases:** The intended applications of the model.
- **Out-of-Scope Usage:** Scenarios where the model should not be used.
- **Potential Misuse:** Examples of harmful applications or adversarial risks.

Ethical and Environmental Considerations (*Optional*)

- **Bias and Fairness:** Documentation on how biases were addressed.
- **Environmental Impact:** Estimated energy consumption and carbon footprint of training and running the model.

Artifact Hashes and Integrity (*PoC addition*)

- **Cryptographic Hashes:** SHA-256 or similar hashes for datasets, models, and metrics to ensure integrity and verifiability.

Digital Signatures and Attestation (*PoC addition*)

- **Digital Signature:** The AIBOM document is digitally signed to guarantee authenticity.
- **Attestation References:** Links or references to in-toto or other attestation artifacts, providing cryptographic evidence of the training process and environment.

Automated Generation (*PoC addition*)

- **Automation:** The AIBOM should be generated automatically as part of the training pipeline to minimize manual intervention and reduce the risk of human error.

Explicit Model Lineage and Dependencies (*Optional*)

- **Lineage Graph:** All relationships between models, datasets, and environments should be somehow included.

2 Foundations of AI Bill Of Materials

The Manifest proposal provides a strong foundation for AIBOMs, but the additions from the PoC are necessary to ensure trust, verifiability, and compliance in modern AI systems.

2.1.7 What Should Not Be Included in an AIBOM?

While AIBOMs aim to improve transparency, not all information from traditional model documentation (such as model cards) should be included. The following elements are not necessary in an AIBOM.

Implementation details

- **Example Code:** Code snippets to run the model are useful to developers, but are not relevant to risk assessment.
- **Detailed Implementation Notes:** Extensive details on implementation are not required in an AIBOM, as the focus is on security and compliance rather than usability.

Lengthy or Visual Data Representations

- **Performance Metrics in Graphical Form:** While performance benchmarks are useful for evaluation, tables and graphs are difficult to represent in structured JSON formats. Instead, a reference to external evaluation reports or a summary can be provided.
- **Raw Training Logs:** Detailed logs of model training are not necessary in an AIBOM and would add unnecessary complexity. However it's not a bad idea to keep the logs in storage and reference them in the AIBOM for extra traceability and transparency.

Subjective Risk Assessments

- **Statements on Model Trustworthiness:** The AIBOM should provide objective facts about the construction of the model, rather than subjective assessments of its safety.
- **Security Risk Ratings:** Instead of assigning risk scores, an AIBOM should document vulnerabilities and dependencies in a structured format, allowing users to

2 Foundations of AI Bill Of Materials

conduct their own risk assessments.

2.1.8 Model Lineage

Model lineage refers to the traceability of an AI model's origins, including its foundational model, the fine-tuning steps, and any modifications applied throughout its lifecycle. Ensuring transparency in the model lineage is essential for compliance, reproducibility, and understanding dependencies within AI systems.

According to Manifest Cyber [2], an AIBOM should, at a minimum, include information about the current model and its foundational model. However, a key question remains:

How important is it to show the entire lineage from the current model to the foundational model to the data?

To answer this question, showing the entire lineage is crucial for traceability, as it allows stakeholders to understand the origins and dependencies of an AI model. This is particularly important for compliance, reproducibility, and risk assessment of high-risk systems. By documenting the relationships between models, datasets, and dependencies, organizations can identify potential vulnerabilities, biases, or licensing issues.

The SPDX (BOM standard) AI Profile addresses this by defining relationships between models, datasets, and other components in a structured yet manageable way. It avoids making the AIBOM overly complex by focusing on key relationships, such as "derived from" or "uses", while omitting unnecessary details. For more information, refer to Section 2.2.2.

An example is illustrated in Figure 2.1. The model GPT4A11-13B-Snoozy-SuperHOT-8K-GPTQ is based on the 13B Llama model by Meta AI. The Llama model was trained on a collection of datasets, and to develop GPT4A11-13B-Snoozy, further fine-tuning was performed using the Snoozy dataset with Low-rank adaptation (LoRA). Additionally, SuperHot interpolation was applied by introducing adapter weights trained on a separate dataset to enhance context length. The final model was weight-quantized, optimizing storage efficiency at the cost of some precision.

2 Foundations of AI Bill Of Materials

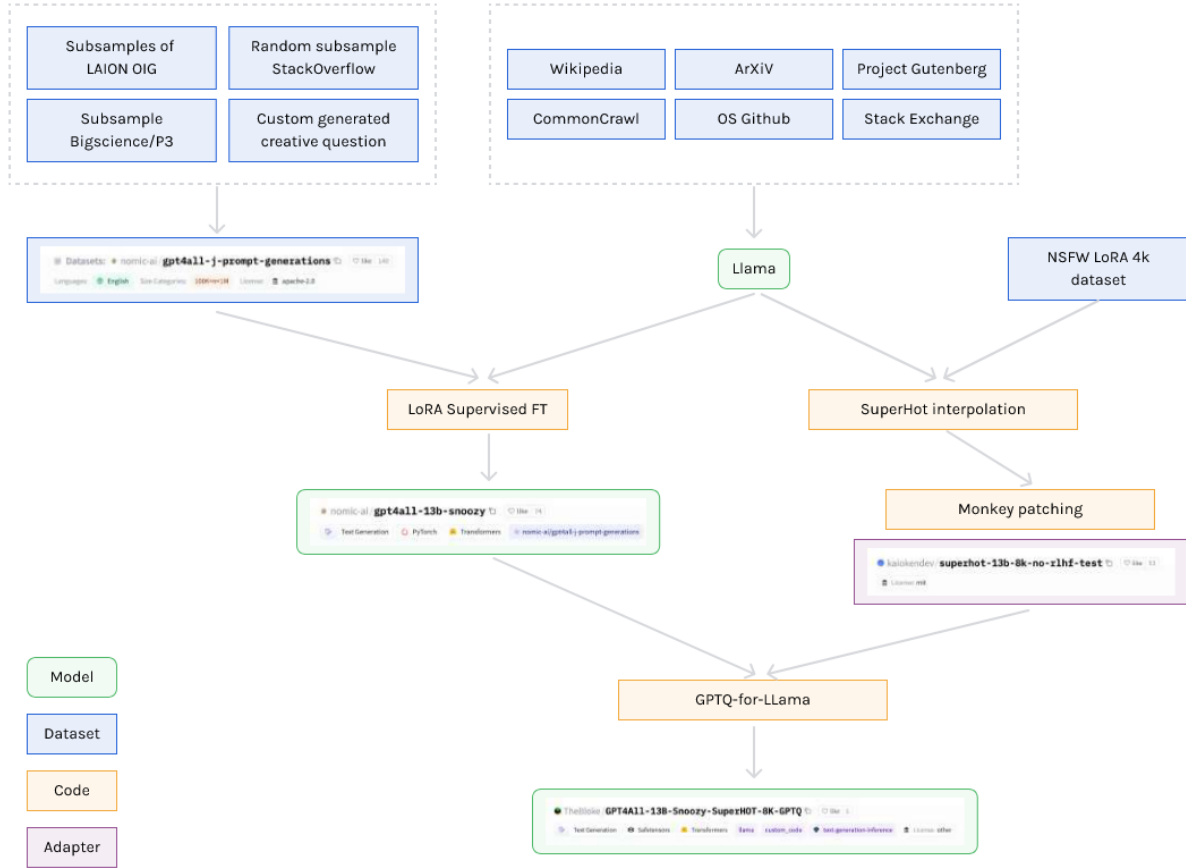


Figure 2.1: Example of model lineage, showing the relationships between a base model, fine-tuning steps, and modifications [2]

2.2 Existing BOM Standards

2.2.1 CycloneDX

CycloneDX is an industry standard framework for software supply chain management, developed as an open-source project by the Open Worldwide Application Security Project (OWASP) [3] [24]. It is officially recognized as the ECMA-424 standard, defining the CycloneDX v1.6 Bill of Materials (BOM) specification [27]. This standard enhances transparency by identifying and documenting components within a software application.

Modern software often combines proprietary and open-source components, some of which may have known vulnerabilities. CycloneDX integrates with Software Component Analysis (SCA) tools to automate component identification and generate a Soft-

2 Foundations of AI Bill Of Materials

ware Bill of Materials (SBOM). An SBOM lists all components in a codebase, including their versions, licenses, and known vulnerabilities. Additionally, CycloneDX maps relationships between components, making it possible to detect transitive dependencies, indirect dependencies introduced through other components.

Beyond open-source software, CycloneDX also supports proprietary software. If a vulnerability is found in a proprietary component, it helps identify all applications that use that component, enabling timely patching. It also tracks software licenses, helping to detect and resolve potential conflicts when an application includes components with incompatible licenses.

The CycloneDX Object Model

The CycloneDX object model is a data model that defines the structure of a CycloneDX BOM [3]. The object model is based on the concept of a component, which is a software component that is included in a software application. Each component has a number of properties, such as its name, version, and license. The CycloneDX object model also defines the relationships between components. For example, a component may have a dependency on another component.

In Figure 2.1, the CycloneDX BOM is organized into several key components that provide a detailed inventory and context of the described items [3]. The most important ones are:

- **Metadata:** This section provides crucial information about the BOM itself, such as the supplier, manufacturer, and target component, as well as details of the tools used to create the BOM and license information.
- **Components:** Components describe the complete inventory of first- and third-party components. The specification can represent software, hardware devices, machine learning models, source code, and configurations, along with the manufacturer's information, license and copyright details, and complete pedigree and provenance for every component.
- **Services:** Services represent external APIs that the software may call. They describe endpoint URIs, authentication requirements, and trust boundary traversals. The data flow between software and services can also be described, including the data classifications and the flow direction of each type.
- **Dependencies:** CycloneDX provides the ability to describe the components and

2 Foundations of AI Bill Of Materials

their dependence on other components. The dependency graph is capable of representing both direct and transitive relationships. Components that depend on services can be represented in the dependency graph, and services that depend on other services can be represented as well.

This comprehensive structure allows CycloneDX to represent a wide array of components and their relationships, supporting various use cases from vulnerability management to license compliance.

2 Foundations of AI Bill Of Materials



Figure 2.2: CycloneDX Object Model with object types arranged in order and with their object properties [3]

CycloneDX's ML-BOM

In addition to the core CycloneDX specification, there are a number of extensions that add support for other features. One such extension is the Machine Learning Bill of Materials (ML-BOM). ML-BOM is a CycloneDX extension for machine learning systems [3]. CycloneDX ML-BOM facilitates transparency in AI and machine learning systems by representing critical information about models, datasets, and their dependencies.

2 Foundations of AI Bill Of Materials

This includes the provenance of datasets, training methodologies, and the configuration of AI frameworks. This detailed visibility allows organizations to assess risks related to bias, data integrity, or model security, ensuring AI systems align with ethical and regulatory standards [24]. The AIBOM example from Manifest in Section 2.1.4 uses components and properties from CycloneDX's ML-BOM, like the modelCard property.

2.2.2 SPDX

Another BOM standard is System Package Data Exchange (SPDX) [4] [28]. It is an open standard designed to help organizations share information about software components, including details about their licenses, security, and origins, just like CycloneDX, described in Section 2.2.1.

The SPDX standard is recognized internationally and is even approved as an International Standardization Organization (ISO) standard (ISO/IEC 5962:2021)[29]. This means that it is a widely accepted way to document software components. SPDX is managed by the Linux Foundation and is a collaborative effort that involves many different organizations, including software vendors, system integrators, and legal experts.

In Figure 2.3 we can see how the structure of an SPDX document is defined in v2.3. There are all kinds of optional information components, such as: Package information, File information, but also Relationships and Annotations are possible just like in CycloneDX.

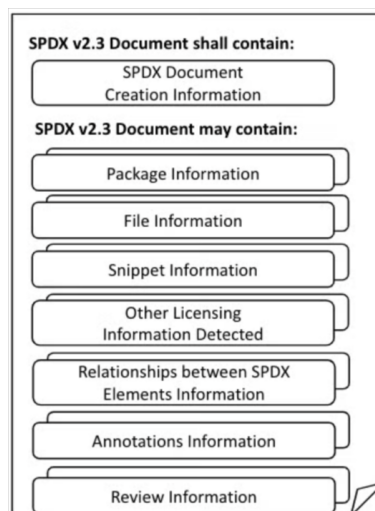


Figure 2.3: SPDX v2.3 document overview [4]

2 Foundations of AI Bill Of Materials

AI Profile with SPDX 3.0

To accommodate the diverse needs of different industries and use cases, SPDX 3.0 made a big upgrade structured around profiles, which are compliance points that define specific areas of focus within the standard, as shown in Figure 2.4. SPDX defines eight profiles, each addressing a different aspect of software and system documentation. The Core Profile is mandatory, while the others are optional, allowing organizations to choose the profiles that best fit their needs [23].

 SPDX SECURITY	Security information - vulnerability details related to software
 SPDX BUILD	Build related information - provenance and reproducible builds
 SPDX AI	Information about AI models - ethical, security, and model data
 SPDX DATA	Information about datasets - AI and other data use cases
 SPDX LITE	Minimal subset to support industry supply chain workflows
 SPDX LICENSING	Information about copyrights and licenses - supports compliance
 SPDX SOFTWARE	Information specific to software
 SPDX CORE	Information used across all profiles

Figure 2.4: SPDX 3.0 profile overview [5]

SPDX 3.0 introduces two key profiles that are particularly relevant for AI systems: the AI Profile and the Data Profile [23]. These profiles allow organizations to document AI systems in a structured and standardized way, addressing the unique challenges posed by AI and ML technologies.

- **AI Profile:** This profile focuses on the AI models and software components used in an AI system. It includes fields for documenting the type of model (for example, neural network, reinforcement learning), how it was trained, and the energy consumption associated with training and inference. It also captures information on performance metrics, limitations, and ethical considerations, such as bias or fairness issues.

2 Foundations of AI Bill Of Materials

- **Dataset Profile:** This profile focuses on the data used to train and test AI models. It includes fields to document the source of the data, how it was collected, and whether it contains sensitive or personal information. This is crucial to ensure compliance with data privacy regulations and identify potential biases in the data.

By separating the AI and Dataset Profiles, SPDX 3.0 provides a more granular view of AI systems, making it easier to track and manage the different components. For example, if an AI model is trained on a dataset that contains biased information, the Dataset Profile can help identify this issue, while The AI Profile can document how model performance might be affected [5].

The different profiles and relations can then be used to build an AIBOM that contains all components. This also enables tracking of the model lineage. As shown in Figure 2.5, the concept of Build and Source BOMs is also incorporated. Together with AI and Dataset Profiles, they effectively meet the requirements of a comprehensive AIBOM.

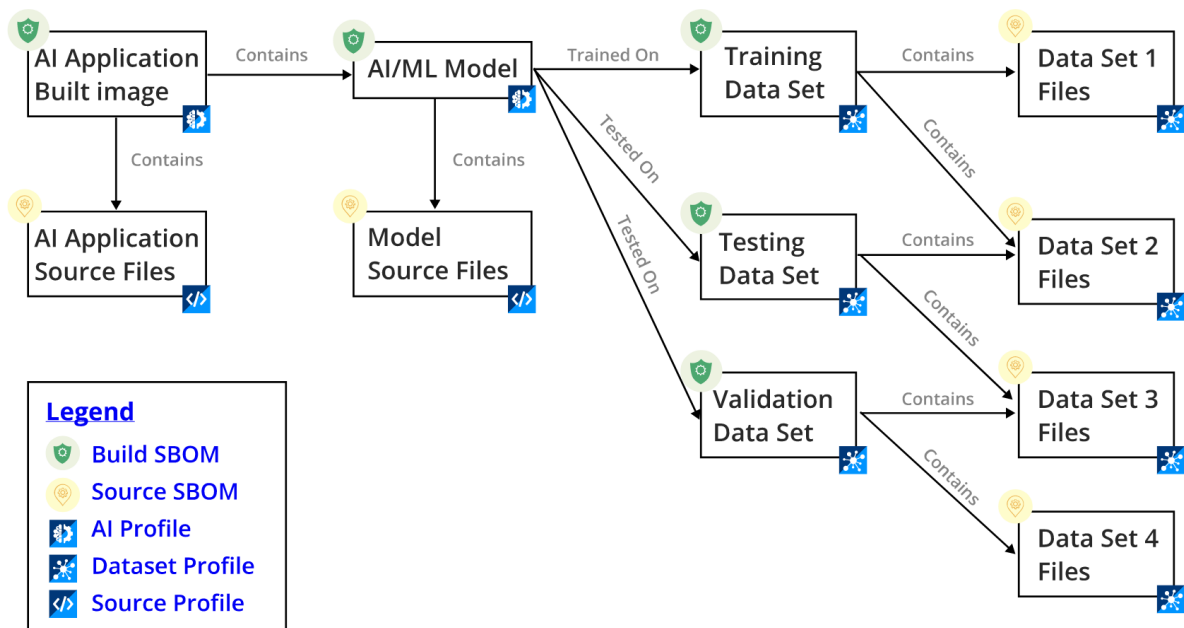


Figure 2.5: SPDX 3.0 AIBOM structure of an example [5]

2 Foundations of AI Bill Of Materials

2.3 AI Compliance

2.3.1 The AI Act

The European Union has positioned itself as a global leader in digital regulation, introducing a series of legislative frameworks to enhance security, competition, and transparency in the digital landscape. It began with the General Data Protection Regulation (GDPR), which set a global standard for data protection, followed by the Digital Markets Act (DMA) and Digital Services Act (DSA) to regulate online markets and digital services. Other key regulations, such as the Data Governance Act (DGA), Data Act (DA), Cyber Resilience Act (CRA), and Network and Information Systems Directive 2022 (NIS2), focus on data sharing, cybersecurity, and resilience against cyber threats [30].

Building on this foundation, the AI Act is the EU's most ambitious step toward regulating artificial intelligence [18][31]. As AI becomes more widely used in sensitive areas such as healthcare, finance, and law enforcement, concerns about transparency, fairness, and accountability have grown. The AI Act aims to address these risks by introducing a risk-based framework that ensures AI systems are developed and used responsibly.

Core principles of the AI Act

At its core, the AI Act adopts a risk-based approach [17] [32]. It classifies AI systems into four levels of risk: unacceptable, high, limited, and minimal or no risk (see Figure 2.6 for a visual representation).

- **Unacceptable risk:** This includes systems that are strictly prohibited. These are systems that compromise people's safety and rights, for example, government-operated social scoring systems and manipulative AI.
- **High-risk:** These are systems that are often deployed in critical infrastructure, such as law enforcement. This is where very strict and clear rules apply. Providers must perform risk assessments, achieve very high data quality, maintain technical documentation and transparency measures (AIBOM), employ mandatory human oversight and guarantee accuracy. On top of that, high-risk AI providers must establish a risk management system throughout the lifecycle of the system.
- **Limited-risk:** This includes applications such as chatbots. These fall under transparency obligations to ensure that users are aware that they are interacting with an AI. Providers of generative AI, for example LLMs or Diffusion models, must en-

2 Foundations of AI Bill Of Materials

sure that AI-generated content is identifiable, with specific labeling requirements for deepfakes and content published to inform the public on matters of public interest.

- **Minimal or no-risk:** These systems, such as AI-driven video games and spam filters, are free of regulatory requirements. The broad majority of AI systems used in the EU fall into this category [17].

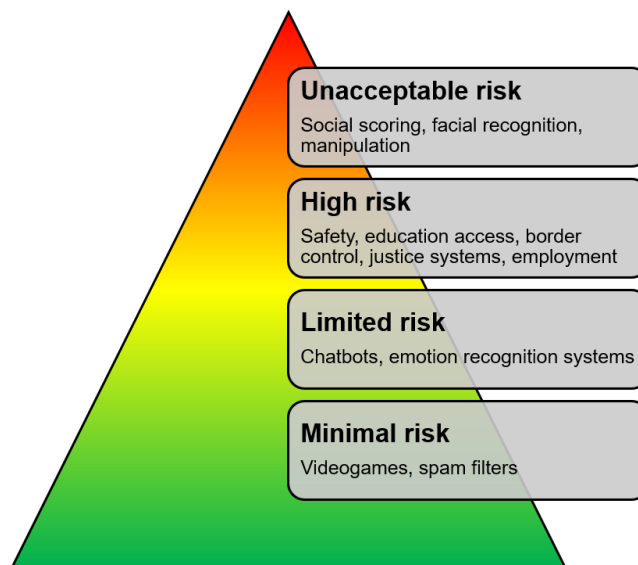


Figure 2.6: Risk classification of AI systems under the AI Act

Governance and Implementation of the AI Act

The AI Act is enforced and supervised at both the European and national levels. The main responsibility lies with the European AI Office, which was established within the European Commission in February 2024. In addition, each member state is responsible for having its own supervisory entity [32] [33].

A key role of the European AI Office is to oversee the implementation of the AI Act and to supervise General-Purpose AI (GPAI)¹ models, which are often among the most powerful AI systems and likely require extra attention [32].

The governance framework of the AI Act is supported by three advisory bodies [33]:

¹General-Purpose AI (GPAI) encompasses AI systems designed to perform a wide range of tasks across various domains, rather than being specialized for a single application [34].

2 Foundations of AI Bill Of Materials

- **The European Artificial Intelligence Board:** This board consists of representatives from all EU Member States and is responsible for ensuring a consistent implementation of the AI Act across the EU.
- **The Scientific Panel:** A team of independent AI experts who provide scientific and technical guidance on AI-related issues.
- **The Advisory Forum:** A diverse group of stakeholders from both the commercial and non-commercial sectors. They ensure that different perspectives are considered in AI regulation.

This structure provides a collaborative and regulated AI ecosystem within the European Union.

Potential Impact

AI legislation is designed around the importance of people and trusted AI, with an eye toward a high level of security for health, safety, and basic rights. The EU wants to establish a global standard for AI, similar to the enriching impact of the GDPR [17].

However, it is important to realize that this also has an impact on innovation. The legislation comes with many costs, especially for Small and Medium-sized Enterprises (SMEs) [35][36]. This could greatly delay the market entry of new AI solutions in Europe. Those costs for SMEs are estimated between 1% and 2.7% of their revenues [30].

Companies that do not comply with the regulations could face large fines of up to 7% of their global annual income for the most serious violations [35]. The EU is encouraging high-risk system companies to comply voluntarily with key obligations during the transition period, which is expected to last until 2 August 2026 [17].

2.3.2 AI Regulation: EU vs U.S.

There is a clear difference between EU and United States (U.S.) approaches. The EU's AI Act is clear legislation drafted in a risk-based manner with a preventive approach. In contrast, the U.S. aims to regulate AI without hindering the growth of its industry. Their focus leans more toward protecting national security, disclosure mandates, and election integrity, differing from the EU's emphasis on safeguarding individual rights [35].

The U.S. has proposed three key frameworks for AI regulation [35]:

2 Foundations of AI Bill Of Materials

- **SAFE Innovation Framework:** emphasizes transparency, accountability, and innovation in AI development, promoting security while avoiding restrictive regulations that could hinder growth.
- **Bipartisan Framework:** for U.S. AI Act focuses on consumer protection, transparency, and national security, proposing licensing requirements for AI systems and the removal of Section 230 immunity for AI companies.
- **National AI Commission Act:** proposes the creation of a commission to develop a risk-based AI regulatory framework, similar to the EU's approach but with more industry experts to balance regulation with growth.

2.3.3 Implementation of AIBOM compliance with AI Act

Compliance with the AI Act requires a structured way of working with documentation, transparency, and risk management. Integrating AIBOM is very important in this process. As mentioned earlier in Section 2.1, the AIBOM provides a detailed table of contents of the components, dependencies, and datasets used in AI systems. This allows stakeholders to assess compliance with regulatory requirements.

However, to make AIBOM also effective in terms of compliance and risk management, it must be generated with trust. The AIBOM itself should not be trusted and loses its value as a compliance and security measure. Without a trusted mechanism, there is no guarantee that the AIBOM accurately reflects the true state of the AI system. Currently, there is no standardized, widely used and trusted AIBOM generation tool. This creates a critical gap in the AI regulatory landscape. However, at the moment efforts are being made to develop trustable AIBOM generation tools. Section 2.4 described two of these projects.

Measures for compliance

To align with the AI Act and establish trust in AI documentation, the following measures should be implemented:

- **Structured Documentation:** Utilize standardized formats such as SPDX 3.0 or CycloneDX
- **Transparency Measures:** Ensure that high-risk AI systems have clear audit trails, including metadata about training, external data sources, and evaluation criteria.

2 Foundations of AI Bill Of Materials

- **Trusted AIBOM Generation:** Develop secure and verifiable AIBOM generation tools that operate within a controlled environment. The information should be accurate and resistant to tampering.
- **Risk Management:** Establish a compliance monitoring system that continuously tracks AI system modifications and maintains up-to-date AIBOMs.
- **Integration with the AI Act:** Map key fields to the AI Act requirements, particularly for high-risk AI systems. For example this is already done in the SPDX standard.

A detailed comparison of SPDX 3.0 fields and their correspondence with the AI Act's registration requirements is provided in Appendix A. This is an exact copy of the table in the SPDX AIBOM paper [5]. This mapping shows an almost seamless integration of compliance measures, reducing the administrative burden for AI providers while complying with regulatory standards.

2.4 Trustable AI projects

TAIBOM project

The Trustable AI Bill of Materials (TAIBOM) project [37] [12], developed by TechWorks and NQMinds, is an ongoing initiative to create a standardized and trustworthy approach for generating AIBOMs. TAIBOM is specifically designed to describe and attest the trustworthiness of full AI systems, including both training and inference [38]. In contrast, this thesis focuses primarily on the training phase, where provenance and transparency are most critical.

TAIBOM addresses two fundamental challenges [37]:

- **Versioning:** The versioning challenge in AI is rooted in the extreme volatility and complexity of AI systems. Even a minor change, such as a single training parameter, a poisoned dataset, or an insecure training environment, can drastically alter system behavior, often in subtle ways. TAIBOM tackles this by adapting and extending SBOM best practices (CycloneDX/SPDX) to AI, explicitly manifesting dependencies not only on code, but also on training data, algorithms, validation data, and the hardware/software environment. The goal is to provide a language and structure to describe a stable AI system in terms of its components and dependencies, enabling all stakeholders to deterministically describe and reference

2 Foundations of AI Bill Of Materials

both components and assembled systems.

- **Attestations (Annotations):** TAIBOM implements a flexible system for making verifiable claims about AI systems and their components. Claims are made using W3C Verifiable Credentials, which are digitally signed and distributed certificates. This approach ensures that each claim is explicit about the subject, the claimant, and the context, and can be aggregated across components to reason about the trustworthiness of the entire system.

TAIBOM's technical implementation includes an SDK and CLI for generating, signing, and verifying TAIBOMs. The SDK supports identity creation, BOM generation, and the creation of TAIBOMs for code, data, systems, and configurations. All claims are cryptographically validated against their associated hashes, ensuring integrity and non-repudiation.

It is important to note that TAIBOM is a work in progress and has evolved significantly during the period of this research. The project continues to adapt as industry standards and best practices develop.

AICert PoC

A closely related open-source proof-of-concept is AICert [39], developed by Mithril Security in collaboration with the Future of Life Institute. AICert demonstrates how cryptographic guarantees can be used to provide verifiable provenance for AI model training, addressing the limitations of traditional documentation-based approaches.

Key features of AICert include [40]:

- **Cryptographic Binding:** a Trusted Platform Module (TPM)² measures the entire hardware and software stack, ensuring that any tampering is detectable. The hashes of code, data, and resulting weights are recorded and bound in a certificate.
- **Transparency and Traceability:** Auditors and users can verify that a model was trained as claimed, with specific data and code, and can detect unauthorized changes.
- **Open Source:** The AICert framework is available under an Apache-2.0 license.

²A TPM is a secure cryptoprocessor designed to carry out cryptographic operations and ensure hardware integrity. For more information, see Ref [41].

2 Foundations of AI Bill Of Materials

While AICert currently focuses on fine-tuning rather than full training from scratch [42], and does not audit code or data for threats (such as backdoors or poisoned data) [40], it provides a strong technical foundation for verifiable AIBOMs. This approach goes beyond declarative documentation to cryptographically provable claims, which are essential for compliance and trustworthy AI development.

Both TAIBOM and AICert represent important steps toward a secure and transparent AI supply chain, and both are rapidly evolving as the field matures. Similar techniques to both projects will be used in the AIBoMGen PoC, which is described in Chapters 3 and 5.

3

System Design & Architecture

This chapter delves into the PoC: AIBoMGen, which demonstrates how trustable and verifiable AI Bill of Materials (AIBOM) can be generated during the training phase of the AI life cycle. The implementation of AIBoMGen is publicly available on GitHub¹ and published on Zenodo [43]. Figure 3.1 illustrates the use case diagram for the AIBoMGen system, highlighting the interactions between the AI developer, verifier, and the system.

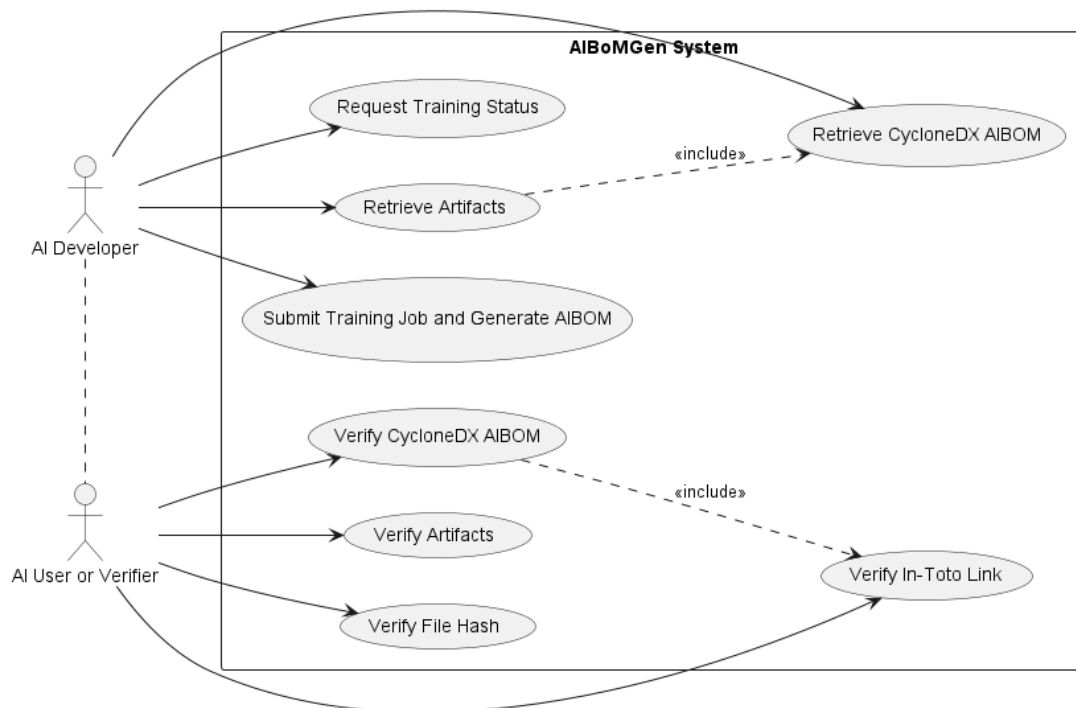


Figure 3.1: Use case diagram: the AIBoMGen system

¹<https://github.com/idlab-discover/AIBoMGen>

3 System Design & Architecture

The diagram illustrates the interaction between two primary actors, the AI developer and the verifier, with the system. The AI developer is responsible for submitting a training job to the system, which trains the model and generates the AIBOM. The developer can retrieve artifacts and check training status.

The AI User or Verifier can verify the integrity and authenticity of the AIBOM, which is made public for transparency. The AIBOM includes a reference to the int-toto Link file, which is a signed file containing the attestations of the artifacts. This is a recording of the training process, that can also be verified. The verifier can also check hashes of artifacts with the ones recorded on the AIBOM.

Trustability is a core aspect of the AIBoMGen system. The mechanisms to ensure it, such as cryptographic attestations, secure APIs, and controlled execution, are discussed in later sections, starting with Section 3.2. First in Section 3.1 a naive approach is examined. Detailed implementations and attestation mechanisms are explained in Chapter 5.

3.1 Naive approach

The initial version of AIBoMGen used a naive method to generate AIBOMs. It executed a user-provided training script, `train.py`, within a container built from a user-defined Dockerfile. This script controlled the entire training process. AIBoMGen captured its output and attempted to extract metadata about the model, dataset, and environment using runtime logs and container inspection.

This approach prioritized flexibility: users could define their architecture, dataset, and training logic, as well as the execution environment. AIBoMGen passively observed the process, collecting information such as installed packages (`pip freeze`), Docker and GPU info, timestamps, and log outputs. These logs included framework-specific metrics, which the system would try to parse.

Although this approach successfully proved that automated data collection during training was possible, it lacked integrity guarantees. No one could verify the authenticity of the collected data. Malicious or faulty scripts could bypass logging or provide misleading information. Additionally, using `exec()` to run arbitrary code introduced serious security issues like arbitrary code execution.

Ultimately this approach served as an exploratory phase: it revealed the kind of data that could be collected, how it might be structured into a CycloneDX-compatible AIBOM,

3 System Design & Architecture

and what gaps existed in trustability.

Figure 3.2 shows the first naive system: a user uploads a Docker image, mounts `train.py` and a dataset, and AIBoMGen launches the container using `exec()` to run the script. Logs and environment snapshots are then used to generate an AIBOM.

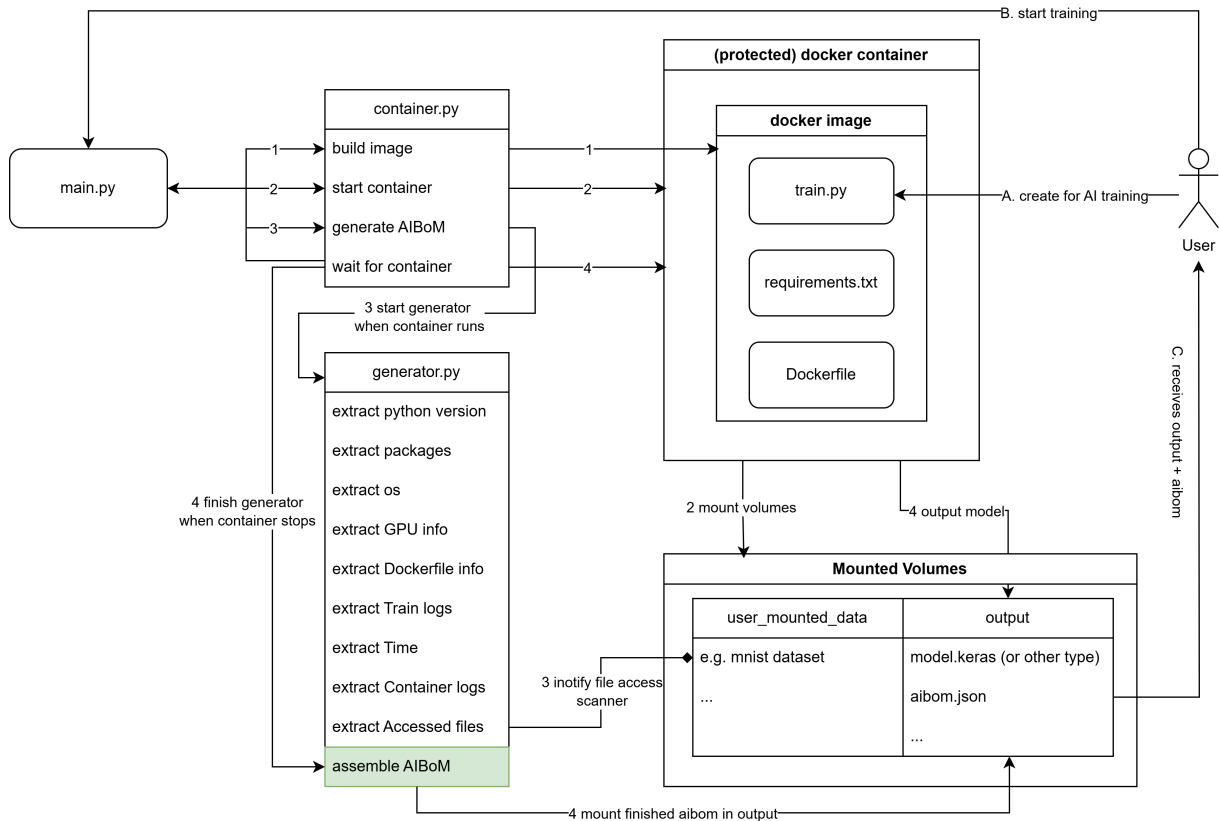


Figure 3.2: Naive AIBoMGen architecture

3.2 Ensuring Trust-ability

The naive approach revealed critical gaps in trustability and security. Without guarantees of integrity and authenticity of the collected data, and with arbitrary code execution, the system was vulnerable to tampering and manipulation.

To address these issues, AIBoMGen introduces cryptographic attestations, secure APIs, and isolated container execution. These will make sure that training is verifiable and tamper-proof. The PoC is not intended as a production-ready system, but as a demonstration of how these concepts can be applied to create a secure and trustable AI development environment. The ultimate goal is to deploy a similar system as an AI Platform

3 System Design & Architecture

as a Service (PaaS) on cloud infrastructure, backed by a trusted institution such as the EU government. Such platform would provide a trusted AI development environment, enable transparency, facilitate compliance with regulations, and foster trust between developers, users and regulatory bodies.

It's important to mention that this AIBOM generation tool only provides verifiable information about the model, data, and training part of the AI development lifecycle and not about the inference or deployment part. Full AI systems can't be verified with this tool, as these parts are not included in the AIBOM generation process. However, the AIBoMGen PoC can be extended to include model deployment and inference in the future, similar to projects like TAIBOM and AICert discussed in Section 2.4.

While platforms like GPULab [16] provide powerful infrastructure for GPU-based model training, they are not suitable for generating trusted AIBOMs. GPULab allows users to run arbitrary code via custom Docker images or Jupyter notebooks [44], which introduces the same integrity risks as the naive approach. This lack of control makes it incompatible with the security and trustability requirements of AIBoMGen.

A central challenge in designing a trustable system is the trade-off between developer flexibility and the trustability of the generated AIBOM. On one hand, allowing developers to run arbitrary training scripts maximizes flexibility but undermines trust, as the system cannot guarantee the authenticity or integrity of the process. On the other hand, enforcing strict workflows and controls can ensure trustability but limits developer freedom. The following sections examine two different approaches to addressing this trade-off: confidential computing (Section 3.2.1) and AIBoMGen's trusted API solution (Section 3.2.2).

3.2.1 Confidential Computing as a potential solution

Confidential computing offers a promising way to prevent tampering during model training by executing code inside a Trusted Execution Environment (TEE). A TEE is a hardware-protected enclave that isolates computation from the rest of the system, ensuring that data and code are protected even from privileged software like the host OS or hypervisor.

Training scripts executed within a TEE benefit from strong security guarantees. The enclave ensures confidentiality and integrity, and generates cryptographic attestations: signed proofs that specific code was executed on untampered data in a trusted environment. These attestations could be incorporated into the AIBOM, offering verifiable

3 System Design & Architecture

evidence that the training process was secure and unmodified.

Underlying these attestations is the concept of a chain of trust. This chain starts from a hardware root of trust embedded in the CPU, which verifies the integrity of the TEE itself at boot. Once the enclave is verified, the code loaded into it is measured (hashed), and this measurement is included in the cryptographic attestation. This means that anyone verifying the attestation can confirm that only approved, untampered code executed within the protected environment. The chain of trust thus links the hardware, the execution environment, and the training script together in a verifiable manner.

This approach allows developers to run full user-defined training scripts without conforming to predefined workflows, preserving flexibility while maintaining trustability. The cryptographic attestations and chain of trust provide strong guarantees that the training process was secure and tamper-free.

However, confidential computing requires specialized hardware (e.g., Intel SGX, AMD SEV) and additional software layers, increasing complexity and resource requirements. While promising, it is not practical for the lightweight, containerized design of the AIBoM-Gen PoC. Also using this technology would require more research on how to integrate AIBOM generation in the TEE. This is out of scope for this thesis.

A related but distinct approach is used in AICert which is described in Section 2.4, it leverages Trusted Platform Module (TPM) to attest the integrity of the software stack, training code, data, and model outputs. While this does not constitute confidential computing, since the training does not occur in a hardware-isolated enclave, it achieves similar goals by cryptographically binding model artifacts to the execution environment, offering strong, hardware-backed provenance without relying on TEEs.

In conclusion, while both TEEs and TPMs provide valuable mechanisms for ensuring trustability, their reliance on specialized hardware and increased complexity make them less suitable for the lightweight, containerized design of AIBoMGen. This is why a more practical solution, based on trusted APIs, was chosen to address the trustability challenge.

3.2.2 AIBoMGen’s Approach: Trustability through Trusted APIs

AIBoMGen addresses the trustability challenge without relying on confidential computing or TPMs by implementing a secure and trusted API. This ensures integrity and auditability through several mechanisms. Trusted API is meant in the sense of enforc-

3 System Design & Architecture

ing strict workflows and requiring users to interact with the system solely through it, ensuring AIBOM generation is always executed securely.

Key features of the trusted API include:

- **Controlled Execution:** Developers submit training jobs through the API, which ensures that all operations are executed within containerized environments. This prevents arbitrary code execution and unauthorized modifications.
- **Automated AIBOM Generation:** The API integrates seamlessly with the AIBOM generation process, ensuring that every training job produces a cryptographically signed AIBOM. Users cannot bypass or alter this step.
- **Audit Logging:** All interactions are logged, providing a transparent record of the training process. This enhances trustability and supports compliance with regulatory requirements.
- **Attestations:** The AIBOM generation process generates cryptographic attestations for input and output data, ensuring that the AIBOM is verifiable and tamper-proof. These attestations can be used to prove the integrity of the training process.
- **Data Validation:** The API validates all inputs, such as datasets and model files, to ensure consistency and prevent tampering before the training process begins.

By leveraging trusted APIs, AIBoMGen achieves a high level of trustability while maintaining a lightweight and practical design. Although this approach imposes some restrictions on developer flexibility, it ensures that the system remains secure, reliable, and compliant without the need for complex technologies like confidential computing.

3.3 High-level Architecture

Figure 3.3 provides a visual representation of the system's architecture, highlighting the relationships between its core components. These components, introduced in Sections 3.3.1 through 3.3.6, each play a specific role in ensuring secure, scalable, and verifiable AI model training. The architecture is designed to be modular for future enhancements and has much in common with a microservices system.

3 System Design & Architecture

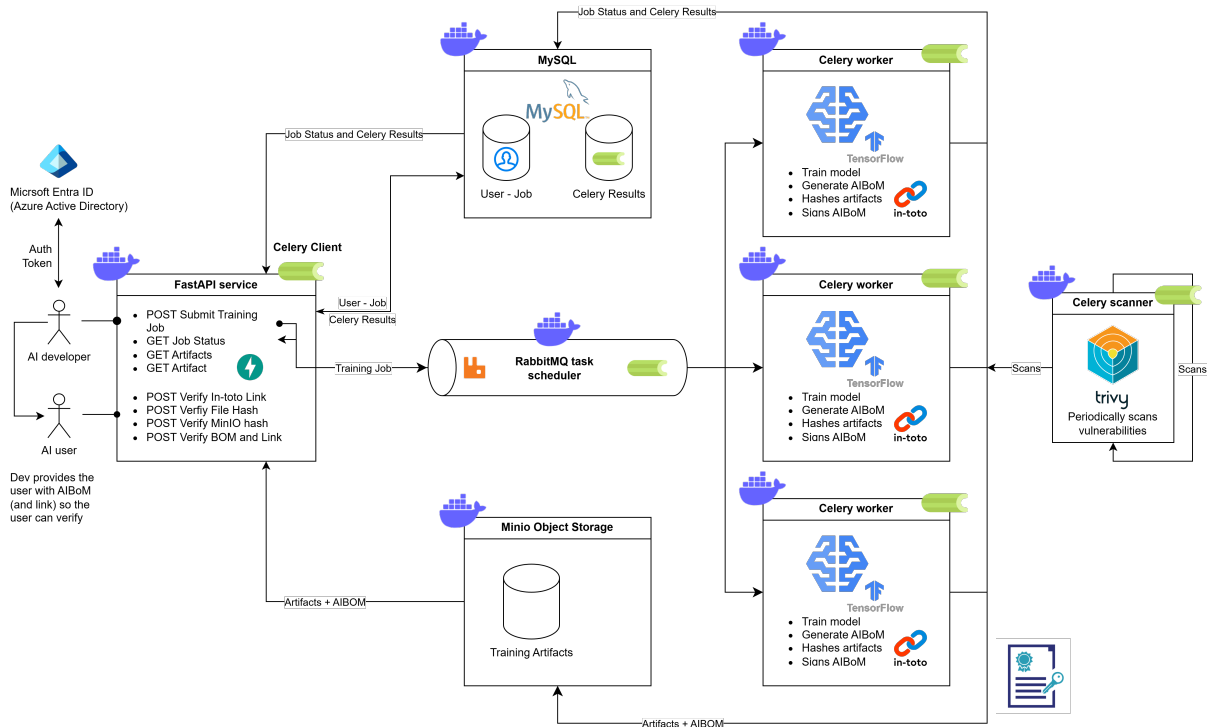


Figure 3.3: AIBoMGen architecture

3.3.1 Central Task Scheduler

A task scheduling service coordinates the execution of training jobs and ensures that tasks are distributed efficiently across the different workers. It acts as the backbone of the platform. It uses a message queue to manage the flow of tasks and responses between components. This allows for asynchronous processing and ensures that the system can handle multiple training jobs concurrently.

3.3.2 Data Storage

A secure data storage system is used to manage training artifacts, such as datasets, models, and generated AIBOMs. This ensures that all data is stored reliably and can be accessed or verified when needed. The data storage system is designed to be compatible with Amazon Web Services (AWS). More about the specific technologies used can be found in Chapter 4.

Object storage is chosen for this system because it is highly scalable and ideal for storing large, unstructured files such as datasets, model binaries, logs, and AIBOMs.

3 System Design & Architecture

3.3.3 API Gateway

A central web API provides developers and users with a single point of interaction with the platform. It handles job submissions, status queries, and artifact retrieval. Important is that it also provides several different verification endpoints to verify the integrity of the AIBOM and the training process. The API is designed to be user-friendly and provides clear documentation for developers. Users must authenticate with a microsoft account to access the API. This ensures that only authorized users can submit jobs or access sensitive data.

3.3.4 Worker Nodes

A pool of scalable worker nodes processes training jobs. These nodes handle both model training and AIBOM generation. Workers have host GPU access for efficient training and are responsible for executing the training scripts within isolated environments. Each worker is designed to be lightweight and can be deployed on various cloud platforms, allowing for easy scaling based on demand and available GPUs.

3.3.5 Vulnerability Scanner

A dedicated worker node is responsible for scanning the docker images of the workers as well as its own image for vulnerabilities. It can be extended to scan all the images including the API. This ensures that the system is transparent and that the platform and users are aware of the vulnerabilities in the system. A summary of vulnerabilities is added to the AIBOM as well.

3.3.6 Monitoring and Management Interface

A management interface provides real-time insights into the system's operations. Different components have their own monitoring interfaces. These dashboards are discussed with their technology in more detail in Chapter 4.

By combining these components, the AIBoMGen platform ensures a secure, scalable,

3 System Design & Architecture

and auditable environment for AI model training.

3.4 User flows

This section describes the main user flows in the AIBoMGen platform. Both the actions of the primary users (AI developers and AI verifier) and the interactions between system components such as the API gateway, task queue, worker nodes, job database, vulnerability scanner and object storage, are explained.

3.4.1 AI developer flow

The AI developer flow covers the process from submitting a training job to retrieving the resulting artifacts and AIBOM. A sequence diagram is shown in Figure 3.4. The steps below highlight both high-level user actions and internal system interactions:

1. **Authentication:** The AI developer needs to be authenticated with a Microsoft account to access certain endpoints in the API Gateway. A token is requested from the authentication service and passed to the API gateway.
2. **Job Submission:** The AI developer prepares the required input files (model, dataset, dataset definition) and submits a training job via the API gateway, also providing training parameters and optional metadata.
3. **Input Validation and Storage:** The API gateway validates the submission, stores job-user details in the job database and uploads the input files to the object storage.
4. **Task Scheduling:** The API gateway enqueues a new training task in the central task scheduler (message queue).
5. **Worker Assignment:** An available worker node fetches the training task from the task scheduler.
6. **Training Execution:** The worker node downloads the input files from object Storage, validates them, and executes the training process in the isolated worker container environment. Training logs, a trained model file and metrics are generated.
7. **AIBOM Generation:** Upon completion, the worker node computes hashes for all

3 System Design & Architecture

relevant artifacts, generates the AIBOM (in CycloneDX format) (including references to in-toto link files, Section 4.3.3), and cryptographically signs it.

8. **Artifact Upload:** The worker node uploads the trained model, logs, metrics, AIBOM, and in-toto link file to object storage.
9. **Vulnerability Scanning:** The vulnerability scanner may scan the worker's Docker image and add a summary of vulnerabilities to the AIBOM. It also scans its own image for vulnerabilities and uploads all the resulting files to object storage.
10. **Status Update:** The worker node updates the task scheduler and API gateway with the job status and artifact locations.
11. **Result Retrieval:** The AI developer queries the API gateway for job status and downloads the resulting artifacts and AIBOM from object storage.

3 System Design & Architecture

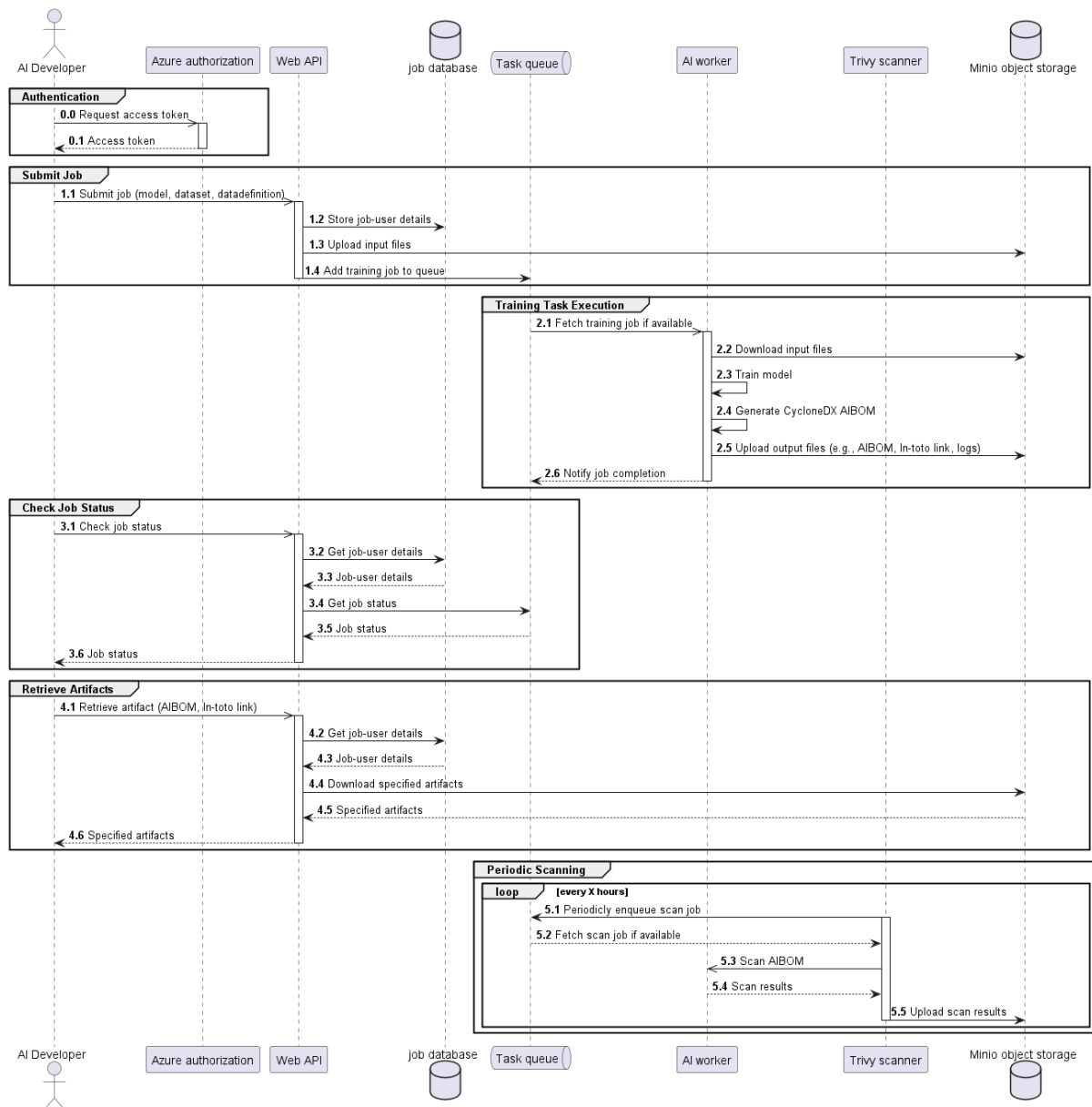


Figure 3.4: Sequence diagram: AI Developer flow

3.4.2 AI verifier flow

The AI verifier flow describes how a user (verifier) can check the authenticity and integrity of a trained model and its associated AIBOM using dedicated API endpoints. The process leverages cryptographic signatures, hash verification, and artifact integrity checks, ensuring that the artifacts have not been tampered with and originate from a trusted training process. The main steps and corresponding API endpoints are:

3 System Design & Architecture

1. **Obtain Artifacts:** The verifier retrieves the AIBOM (CycloneDX BOM), attestation file (in-toto link), and/or relevant artifacts (such as the trained model or metrics), either directly from the AI developer for transparency purposes, or from the API gateway if they have the necessary permissions.
2. **Verify AIBOM and Linked Artifacts (/verifier/verify_bom_and_link):** The verifier uploads the signed CycloneDX AIBOM to the API. The endpoint validates the BOM's schema, verifies its cryptographic signature, and checks the referenced attestation file (in-toto link, downloaded from storage). This provides an end-to-end integrity and authenticity check for the entire training process and its outputs.
3. **Verify in-toto Link File (/verifier/verify_in-toto_link):** The verifier uploads the attestation file to the API endpoint, which checks the cryptographic signatures, layout validity, and artifact rules. This ensures the file is authentic and the recorded actions were performed as expected. How in-toto works is explained in Section 5.3.1.
4. **Verify File Hash Against Link File (/verifier/verify_file_hash):** The verifier uploads both the in-toto link file and a specific artifact (e.g., dataset, model or metrics file) to the API. The endpoint computes the hash of the artifact and compares it to the hash recorded in the link file, confirming the artifact from the user is the same as the one recorded. This endpoint requires the in-toto link file to be verified first.
5. **Verify All Artifacts in storage (/verifier/verify_minio_artifacts):** The verifier uploads the in-toto link file to the API, which then automatically downloads all referenced materials and products from storage. The endpoint computes their hashes and compares them to those in the link file, detecting any tampering in storage. This endpoint requires the in-toto link file to be verified first.
6. **Trust Decision:** The verify AIBOM endpoint is enough for the verifier to make a well grounded trust decision. The verifier can then optionally check artifacts or certain artifact hashes and vulnerabilities to further validate the training process.

A more detailed explanation of the verification endpoints along with their state diagrams is available in Section 5.4.

3 System Design & Architecture

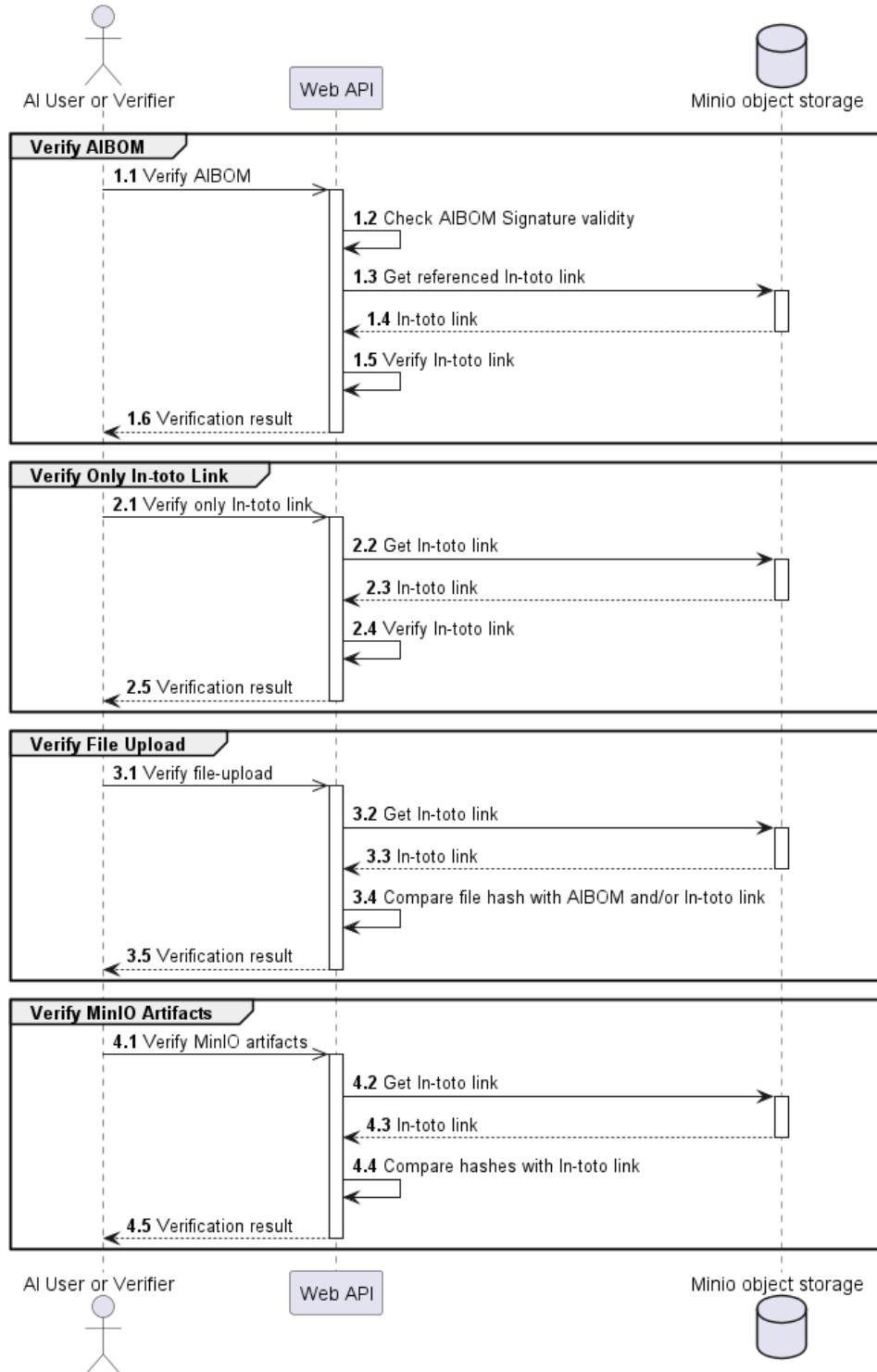


Figure 3.5: Sequence diagram: AI Verifier flow

4

Technologies

This chapter outlines the main technologies that form the foundation of the platform. Each technology was chosen based on its maturity, community support, and suitability for the specific challenges. Additionally, alternative technologies are discussed for future scalability and robustness.

4.1 Technologies used for the PoC, AlBoMGen PaaS

4.1.1 Containerization: Docker

Docker [45] was selected as the primary containerization platform for the PoC. By encapsulating each service in its own Docker container, the platform achieves consistent and reproducible deployments across different environments and system components. The PoC needed to be lightweight and deployable on a single machine for demonstration purposes, and Docker's simplicity and ease of use made it an ideal choice. For the same reasons docker-compose was used to orchestrate the containers [46] instead of a technology like Kubernetes.

On top of that Docker enables rapid scaling and isolation of services, which is essential for security and maintainability. The official Docker documentation was referenced extensively throughout the implementation process, ensuring best practices were followed, like dropping unnecessary capabilities [47].

4.1.2 Backend programming language: Python

Python was chosen as the main backend programming language due to its extensive ecosystem and strong support for automation, data processing, and Machine Learning (ML) [48]. The backend services, including the API and worker logic, are implemented in

4 Technologies

Python 3.11 due to this being the latest stable python version that works with TensorFlow 2.16.1, which is the version used in the PoC. Why this version was chosen is explained in Section 4.1.7. Python's rich set of libraries, such as FastAPI for the web API and Celery for task management, make it fast and easy to develop core functionalities.

4.1.3 Task scheduler: Celery

Celery is a simple, flexible, and reliable distributed system for processing large amounts of messages [49]. It's a task queue with real-time processing, while also supporting task scheduling [50]. It has a large community and is Open Source under the BSD license. Because of its simplicity to setup and its flexibility, Celery was chosen as the task queue for AI training jobs. Celery allows easy distribution of training jobs across multiple workers, which is essential for scaling the platform. While it is not the most ideal solution for distributing AI training jobs, it was chosen for the PoC due to its simplicity. More advanced AI intended solutions like Ray or Kubeflow, are mentioned in Section 4.2. These were not used in the PoC due to the complexity and the fact that the application is not intended for production use.

Celery's modular design allows it to integrate with various components to enhance its functionality. The key components used in this implementation are:

- **Beat:**

Celery Beat is a scheduler that kicks off tasks at regular intervals, which are then executed by available worker nodes in the cluster [50]. It is used to schedule periodic tasks, such as the vulnerability scanner.

- **Flower:**

Flower is a real-time monitoring tool for Celery [51]. It provides a web-based UI to monitor task progress, worker status, and other metrics. Flower was used to make debugging easier and provide insights on what is actually happening on the system.

For production use, a more advanced solution like Ray or Kubeflow with Prometheus and Grafana monitoring would be more suitable.

4 Technologies

- **Broker (RabbitMQ):**

RabbitMQ is used as the message broker for Celery. It facilitates reliable communication between the API and worker nodes. RabbitMQ was chosen for its robustness and proven scalability in production environments [52]. It ensures that tasks are queued and delivered efficiently, even under high load.

The management dashboard provided by RabbitMQ allows for real-time monitoring of queues, exchanges, and messages. This is valuable for debugging.

- **Backend (MySQL):**

For persistent storage of task results and metadata, the Celery backend is configured to use a MySQL database. This enables tracking of task states, results, and error messages, which is essential for debugging and auditing asynchronous workflows. MySQL was selected due to its maturity, performance, and integration possibility with Celery [53]. The database schema is managed automatically by Celery.

4.1.4 API Gateway: FastAPI

FastAPI serves as the main API gateway, exposing endpoints for job submission, status queries, and artifact retrieval. FastAPI was selected for its high performance, automatic OpenAPI documentation, and native support for asynchronous programming [54]. The API is containerized and deployed with CORS and OAuth2 middleware enabled for security. FastAPI's dependency injection and type validation features help ensure robust code.

4.1.5 Authentication: Microsoft Entra ID (Azure Active Directory)

Authentication and authorization are important features of the system as security and trustability are key requirements. Microsoft Entra ID (previously known as Azure Active Directory (AAD)) was chosen for its easy integration with FastAPI and its OAuth2 and OpenID Connect support. This integration provides secure, enterprise-grade identity management and single sign-on (SSO) capabilities if configured with the Ghent University's tenant. The gateway validates tokens issued by Entra ID, enforcing role-based access control on sensitive endpoints. For integration `fastapi-azure-auth` was used, which is a library that simplifies the integration of FastAPI with Microsoft Entra ID [55].

4 Technologies

4.1.6 Object Storage: MinIO

For storing datasets and other AI model artifacts, a object storage solution is needed. MinIO provides S3-compatible object storage for the platform, used to store generated AIBOMs, logs, vulnerability scan results, datasets, models and other artifacts. MinIO was chosen for its ease of docker deployment, high performance, and compatibility with S3 APIs [56]. The platform interacts with MinIO using the boto3 library. Access policies and bucket configurations can be easily managed through the MinIO dashboard, which is a user-friendly web interface.

4.1.7 ML framework: TensorFlow

TensorFlow [57] is used as the primary machine learning framework within the platform. The API currently provides only job submissions for TensorFlow jobs, but this can be extended to other frameworks like PyTorch or SciKit-learn in the future. TensorFlow was chosen for its GPU ready docker images and layer based model files (`model.keras`), which can be used as a basis for the job submission. The PoC uses TensorFlow 2.16.1, which has the latest stable docker image for Python 3.11 that also supported the authors GPU for testing [58]. TensorFlow's Keras API is leveraged for model definition and training, while its built-in utilities are used for exporting model metadata to be included in the AIBOM.

4.2 Alternative technologies for scalable deployments

While the PoC implementation of AIBoMGen PaaS focuses on simplicity and ease of deployment, several alternative technologies were considered for future scalability, robustness, and advanced production use cases. These technologies are not part of the current PoC but are relevant for organizations aiming to scale up, harden, or automate their machine learning and DevSecOps pipelines.

4.2.1 Kubeflow

Kubeflow is an open-source platform designed to make machine learning workflows on Kubernetes simple, portable, and scalable [59]. It provides a suite of tools for orchestrating complex ML pipelines, including training, hyperparameter tuning, and model serving. Its adoption is recommended for organizations with existing Kubernetes infrastructure or those requiring advanced ML workflow automation.

4 Technologies

Importantly, Kubeflow Pipelines can be configured to mitigate arbitrary code execution risks by restricting pipeline creation to only use pre-approved, trusted components and disabling the execution of custom user code. This approach enhances security. Adopting Kubeflow would represent a significant step up in complexity and capability compared to the current PoC, which relies on Celery for task management and a custom training API that would require substantial further development to support multiple frameworks and algorithm types in a secure and flexible manner.

4.2.2 MLflow

MLflow is an open-source platform for managing the end-to-end machine learning lifecycle, including experiment tracking, model packaging, and deployment [60]. It is framework-agnostic and can be integrated with a variety of ML libraries and cloud services. MLflow provides a central UI for tracking experiments, storing artifacts, and managing model versions, making it easier to collaborate and reproduce results in large teams. While not used in the PoC, MLflow could be integrated for enhanced experiment management and model governance and was used in the first naive version of the platform.

4.2.3 Firecracker

Firecracker is a lightweight virtualization technology developed by AWS, designed for running secure container workloads at scale [61]. It uses microVMs to provide strong isolation with minimal overhead, making it more suitable for untrusted code execution scenarios. Firecracker could be considered for securely running user-submitted jobs or isolating workloads, providing a higher level of security than traditional containers. However, it would add complexity to the deployment and management of the platform.

4.2.4 Kata Containers

Kata Containers is an open-source project that combines the performance of containers with the security of lightweight virtual machines [62]. Similar to Firecracker, it provides stronger isolation than traditional container runtimes by running each container inside its own lightweight VM.

For secure AI training and AIBOM generation, Kata Containers could be used to isolate workloads, ensuring that sensitive data and models are protected from potential attacks

4 Technologies

originating from other containers. While both Kata Containers and Firecracker aim to enhance workload isolation, Kata Containers is more tightly integrated with existing container ecosystems, such as Kubernetes, making it a better fit for organizations already leveraging container orchestration tools. Like Firecracker, Kata Containers adds a layer of complexity compared to standard container runtimes like Docker.

4.2.5 Confidential computing platforms

Confidential computing platforms, such as Azure Confidential VMs or Google Confidential Computing, leverage hardware-based Trusted Execution Environment (TEE)s to protect data and code during processing. These platforms are particularly relevant for sensitive workloads, such as those involving proprietary models or regulated data. By running critical components inside TEEs, organizations can ensure confidentiality and integrity even in the presence of privileged attackers. This setup would have the benefits discussed in Section 3.2.1.

4.2.6 Ray

Ray is an open-source framework for building and running distributed applications, especially those involving machine learning and data processing [63]. It provides simple primitives for parallel and distributed computing, and supports scalable training, hyperparameter tuning, and model serving. Ray is designed for high performance and can be integrated with existing Python codebases. It is a strong candidate for organizations seeking to scale AI workloads beyond the capabilities of Celery or traditional task queues.

4.3 Technologies used for the trusted AIBOM generation pipeline

4.3.1 CycloneDX python library

The CycloneDX Python library [64] is used to generate AIBOMs in CycloneDX format. This library provides a simple API for creating and manipulating CycloneDX BOMs, and gives the ability to include various types of metadata, such as dependencies, licenses, and vulnerabilities. This official CycloneDX library is actively maintained and widely used in the industry, making it a reliable choice for generating BOMs. It provides also

4 Technologies

methods for validating the generated BOMs against the CycloneDX schema, ensuring that they conform to the specification.

However, it is important to note that the library does not yet support all the features of the most recent CycloneDX v1.6 schema specification. Important features like the attestations, embedded signatures, and the modelCard field for machine learning components are not yet supported [65]. This means that workarounds are needed to include these features in the generated AIBOMs. Because of that actual BOMs generated by the PoC at the moment do not fully match with the CycloneDX v1.6 specification. The library is still under active development, and it is expected that these features will be added in future releases. The workarounds are explained in Section 5.2 and 5.3

4.3.2 Trivy for vulnerability scanning

Trivy is an open-source vulnerability scanner for containers [66]. It is used to periodically (Celery Beat) scan the Docker images used in the workers for vulnerabilities. Trivy is fast, easy to use, and provides detailed reports. A summary of the vulnerabilities found is included in the AIBOM. The results are stored in MinIO.

4.3.3 in-toto

In-toto is an open-source framework for securing the integrity of software supply chains [67]. It is used in the pipeline to record, sign, and verify each step of the AIBOM and artifact generation process in a specific worker. More details on how in-toto is used in the PoC are provided in Section 5.3.1.

4.3.4 Cryptography hazmat library

The cryptography library's hazmat (hazardous materials) primitives [68] are used for secure signing and verification of the AIBOMs. For signing in-toto link files (Sections 4.3.3 and 5.3.1), `securesystemslib`'s `CryptoSigner` [69] is used, because it was recommended by the in-toto documentation and is interoperable with the cryptography hazmat library.

5

Implementation

This chapter provides a focused overview of the most relevant implementation details of the AIBoMGen PaaS application [43]. While it is not feasible to cover every aspect of the system, the following sections highlight key decisions, interesting technical challenges, and necessary workarounds encountered during development. Special attention is given to the configuration of the core components and to the signing and verification system which is central to the integrity and trustability of the AIBOM generation process.

5.1 Implementation details of the PoC, AIBoMGen PaaS

5.1.1 Celery configuration

As explained in Chapter 4, the Celery task manager is a central component of the AIBoMGen architecture, enabling distributed, asynchronous task execution for training jobs, and scanning. The main components involved are the FastAPI application (task producer), Celery workers (task consumers), a dedicated scanner service, RabbitMQ as the message broker, and a MySQL database as the result backend. Figure 5.1 provides a close-up of the interactions between these components.

5 Implementation

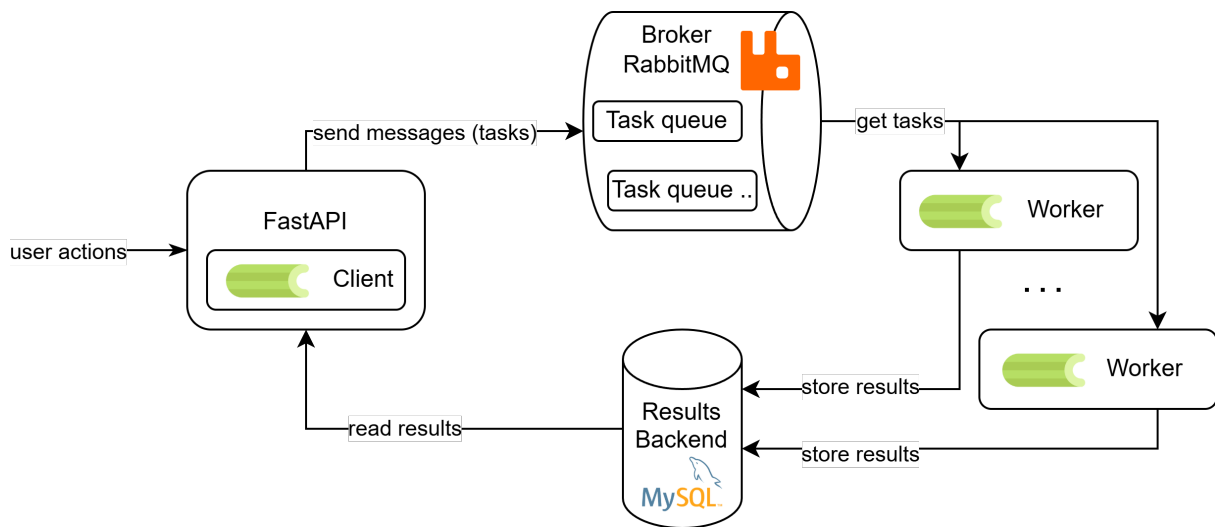


Figure 5.1: Celery Setup: Interaction between FastAPI, RabbitMQ, MySQL, and Celery workers

RabbitMQ is used as the message broker, providing robust and reliable task queuing. MySQL serves as the result backend, allowing persistent storage and retrieval of task results and metadata. Both services are defined in the `docker-compose.yml` file and configured via environment variables for credentials and connection details. A part of the `docker-compose.yml` is shown in Listing 5.1.

For the MySQL database, a custom initialization script is used to create two databases: one for the Celery backend and one for user-job related data (used for user-resource restriction). The initialization script is mounted as a read-only volume in the MySQL container. This ensures that the database is set up correctly when the container starts.

5 Implementation

```
services:
  rabbitmq:
    image: rabbitmq:3-management
    ports:
      - "5672:5672"
      - "15672:15672"
    environment:
      RABBITMQ_DEFAULT_USER: $RABBITMQ_USER
      RABBITMQ_DEFAULT_PASS: $RABBITMQ_PASSWORD
  mysql:
    image: mysql:8.0
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    volumes:
      - mysql_data:/var/lib/mysql
      - ./mysql/init-multiple-dbs.sql:/docker-entrypoint-initdb.d/init-multiple-dbs.s
      ↪ ql:ro
```

Code Fragment 5.1: Celery broker and backend configuration in docker-compose

Each component that interacts with Celery (FastAPI, worker, and scanner) initializes its own Celery application instance, specifying the broker and backend URLs. That means they all need to have their own Celery configuration. As example, a part of the worker's configuration¹ is presented in Listing 5.2.

```
celery_app = Celery(
    "aibomgen_worker",
    broker=os.getenv("CELERY_BROKER_URL"),
    backend=os.getenv("CELERY_RESULT_BACKEND"),
)
```

Code Fragment 5.2: Celery broker and backend configuration

Celery is configured to use named queues for different types of tasks, improving scalability and separation of concerns. For instance, training jobs are routed to the training queue, while scanning tasks are routed to the scanning queue. This can be extended

¹https://github.com/idlab-discover/AIBoMGen/blob/main/aibomgen-platform/worker/celery_config.py

5 Implementation

by using different queues when you have for example different types of workers each with their own dedicated GPUs and CPUs. Queues and routes (i.e., which task uses which queue) are configured in the worker, as shown in Listing 5.3. The worker only uses the training queue.

```
celery_app.conf.update(  
    task_routes={  
        'tasks.run_training': {'queue': 'training_queue'},  
    },  
    task_queues=(  
        Queue('training_queue', routing_key='training.#'),  
    ),  
    task_default_queue='training_queue',  
    task_default_routing_key='training.default',  
)
```

Code Fragment 5.3: Celery queues and routing configuration

There are lots of other configuration options available for Celery, such as task timeouts, retries, worker concurrency and making sure a worker can only process one task at a time. These are all set in the worker configuration file and can be adjusted based on the needs of your system and resource availability.

5.1.2 Worker configuration

The Celery worker is a critical component responsible for executing background tasks such as model training and AIBOM generation. Each worker runs in its own Docker container, based on a custom image that extends the official `tensorflow/tensorflow:2.16.1-gpu` image [58]. This ensures GPU support for deep learning workloads and provides a consistent environment for all training jobs.

The worker Docker image installs all necessary dependencies, including Celery, TensorFlow, cryptography, CycloneDX, in-toto, and MinIO utilities, as specified in the `requirements` file². The entrypoint script dynamically sets the worker name based on the container hostname and starts the Celery worker process, subscribing it to the training queue on RabbitMQ.

The worker's Celery configuration (`worker/celery_config.py`) is designed for reliability and matches the resources provided on the authors host machine. Key settings include:

²<https://github.com/idlab-discover/AIBoMGen/blob/main/aibomgen-platform/worker/requirements.txt>

5 Implementation

- **Concurrency:** `worker_concurrency=1` ensures each worker processes one task at a time, which is important for GPU resource management and reproducibility.
- **Task reliability:** Tasks are retried up to three times on failure (`task_max_retries=3`), and are requeued if a worker crashes (`task_reject_on_worker_lost=True`).
- **Serialization:** Only JSON-serialized tasks and results are accepted for security and compatibility.
- **Result persistence:** Results are stored persistently in MySQL.
- **Database setup:** Celery is configured to automatically create the necessary tables in the backend database.

A selection of these configurations is displayed in Listing 5.4.

```
celery_app.conf.update(  
    worker_concurrency=1,  
    task_max_retries=3,  
    task_reject_on_worker_lost=True,  
    result_persistent=True,  
    accept_content=["json"],  
    task_serializer="json",  
    result_serializer="json",  
    database_create_tables_at_setup=True,  
    ...  
)
```

Code Fragment 5.4: Key Celery worker configuration options

The main training task (`tasks.run_training`) is defined in `worker/tasks.py`³. When a task is received, the worker:

1. Downloads the model, dataset, and dataset definition from MinIO.
2. Loads and validates the dataset using helper functions from `training_logic.py`⁴, supporting CSV, image, and TFRecord formats.
3. Loads the Keras model and validates its compatibility with the dataset.

³<https://github.com/idlab-discover/AIBoMGen/blob/main/aibomgen-platform/worker/tasks.py>

⁴https://github.com/idlab-discover/AIBoMGen/blob/main/aibomgen-platform/worker/training_logic.py

5 Implementation

4. Depending on submission splits the dataset into training and validation sets, then trains the model using parameters provided in the task.
5. Saves the trained model and training metrics, and uploads them back to MinIO.

After training, the worker generates an AIBOM for the job. This involves:

- Extracting environment details and artifact metadata.
- Generating a CycloneDX BOM.
- Creating an in-toto link file that records the provenance of the training step, including input and output artifacts, and signing it with the worker's private key.
- Referencing the in-toto link file in the AIBOM, signing it, and serializing it to JSON.
- Uploading the BOM and in-toto link file to MinIO, making it available for verification.

This ensures that every training job is fully traceable and its integrity can be independently verified.

Worker replication is managed through the Docker Compose `deploy.replicas` setting, allowing the system to scale horizontally by simply increasing the number of worker containers. Each worker is resource-constrained using Docker's `resources.limits` and `resources.reservations`, as visible in Listing 5.5, to prevent overconsumption of CPU and memory. The worker containers run with minimal privileges (`cap_drop: [ALL]` [47]) and in read-only mode [70] for improved security. The Docker socket is mounted to allow the worker to interact with the Docker daemon if needed (e.g., for environment extraction).

5 Implementation

```
worker:
  build:
    context: .
    dockerfile: ./worker/Dockerfile
  deploy:
    replicas: 3
    resources:
      limits:
        memory: 4g
        cpus: "2.0"
      reservations:
        memory: 2g
        cpus: "1.0"
  runtime: nvidia
  read_only: true
  cap_drop:
    - ALL
  networks:
    - internal_network
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
```

Code Fragment 5.5: Minimal worker service configuration in docker-compose

This worker configuration ensures that the AIBoMGen platform can reliably execute distributed, resource-intensive training jobs, while maintaining strong isolation, traceability, and security. The integration of AIBOM and in-toto provenance generation at the end of each task provides a robust foundation for trustworthy and auditable AI workflows. This is explained in Section 5.2 (AIBOM) and 5.3 (Integrity measures).

5.1.3 FastAPI configuration

The FastAPI application serves as the main API gateway for the platform, and is the only component that should be publicly accessible when deployed in production. It is responsible for orchestrating job submissions, authentication, artifact retrieval, and communication with Celery workers and MinIO object storage.

FastAPI runs in a dedicated Docker container based on the official `python:3.11-slim` image. All dependencies are specified in a `requirements.txt` file just like most components. They include libraries like FastAPI, Uvicorn, Celery, SQLAlchemy and FastAPI-

5 Implementation

Azure-Auth. The container is started using Uvicorn as the Asynchronous Server Gateway Interface (ASGI) server. A Dockerfile example is presented in Listing 5.6.

```
FROM python:3.11-slim
WORKDIR /app
COPY ./api/requirements.txt .
RUN pip install -r requirements.txt

# shared utility code for both API and worker (e.g. MinIO utils)
COPY ./shared /app/shared
COPY ./api /app

CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "8000"]
```

Code Fragment 5.6: FastAPI Dockerfile for the API service

The app is initialized with a custom lifespan handler to ensure that authentication, MinIO buckets, and the MySQL database are ready before serving requests. Authentication can be enabled or disabled. The lifespan function is given in Listing 5.7.

```
@asynccontextmanager
async def lifespan(app: FastAPI):
    if AUTH_ENABLED:
        from auth_utils import azure_scheme
        await azure_scheme.openid_config.load_config()
    create_bucket_if_not_exists()
    initialize_database_with_retry()
    yield
```

Code Fragment 5.7: FastAPI lifespan handler for startup initialization

The API integrates with Celery for asynchronous task submission and status tracking. The Celery configuration (`api/celery_config.py`) ensures that only JSON-serialized tasks are accepted, results are persisted in MySQL, and tables are created automatically if needed. This is very similar to the worker configuration. The API submits training jobs to the `training_queue` and provides endpoints for querying task status and results.

The API uses SQLAlchemy for database access, with models defined in `models.py` and connection management in `database.py`. A retry mechanism ensures the database is available before the app starts. As said in Section 5.1.1 and visible in the architecture

5 Implementation

diagram 3.3, there are two databases: one for the Celery backend and one for user-job related data. The API uses the latter to store user-job metadata after authentication. That way the API can make sure only authenticated users can access their own jobs and data. The database is initialized with a custom script that creates the necessary table.

CORS is enabled to allow secure cross-origin requests from the front-end dashboard and other clients. The API uses the `slowapi` library [71] for rate limiting, protecting endpoints from abuse and accidental overload. Middleware is configured to allow credentials, all HTTP methods, and all headers from trusted origins.

The API is organized into modular routers for developer endpoints (job submission, artifact management), verifier endpoints (artifact and BOM verification), and Celery utilities (task monitoring). Each router is included in the main app, and dependencies such as authentication and database sessions are injected using FastAPI's dependency system.

The developer endpoints in the FastAPI application provide authenticated users with the ability to submit training jobs, monitor their status, and retrieve resulting artifacts. These endpoints are grouped under the `/developer` prefix and are protected by AAD authentication, ensuring that only authorized users can interact with their own jobs and data.

The primary endpoint for submitting a training job is `POST /developer/submit_job_by_model_and_data`. This endpoint accepts model files, datasets, and dataset definitions as file uploads, along with metadata and training parameters as form fields. The endpoint performs several steps:

1. Validates and saves the uploaded files to a temporary directory.
2. Validates the dataset (e.g., checks image zip files for safety).
3. Uploads the files to MinIO object storage.
4. Submits a Celery task to the `training_queue` with all relevant URLs and parameters.
5. Stores job metadata (including the user ID and unique job directory) in the SQL job-user database.
6. Returns the Celery job ID and status to the user.

5 Implementation

A simplified function for this endpoint is shown in Listing 5.8.

```
@developer_router.post("/submit_job_by_model_and_data",
↳ dependencies=[Depends(get_current_user)])
async def submit_job(
    ...,
    model: UploadFile = File(...),
    dataset: UploadFile = File(...),
    dataset_definition: UploadFile = File(...),
    ...,
    user: User = Depends(get_current_user),
    db: Session = Depends(get_db),
    ...
):
    # Save files, validate, and upload to MinIO
    ...
    # Send Celery task
    task = celery_app.send_task(
        'tasks.run_training',
        args=[unique_dir, model_url, dataset_url, dataset_definition_url, {...},
↳ {...}],
        queue='training_queue'
    )
    # Store job metadata in the database
    job = Job(id=task.id, user_id=user_id, unique_dir=unique_dir)
    db.add(job)
    db.commit()
    return {"job_id": task.id, "status": "Training started", "unique_dir": unique_dir}
```

Code Fragment 5.8: Developer endpoint for job submission

The GET `/developer/job_status/{job_id}` endpoint allows users to query the status and result of their submitted jobs. It checks that the requesting user owns the job and then queries Celery for the current task status. A simplified endpoint is presented in Listing 5.9.

5 Implementation

```
@developer_router.get("/job_status/{job_id}",
↳ dependencies=[Depends(get_current_user)])
async def job_status(job_id: str, user: User = Depends(get_current_user), db: Session
↳ = Depends(get_db)):
    ...
    celery_task = AsyncResult(job_id, app=celery_app)
    return {"status": celery_task.status, "result": celery_task.result}
```

Code Fragment 5.9: Developer endpoint for job status

Later in the development, Celery endpoints were added to the API Gateway to allow for more detailed monitoring of the Celery tasks. They are preferred when trying to access task status and results and are also used by the front-end dashboard to display the status of the jobs.

Users can list and retrieve artifacts generated by their jobs using the following endpoints:

- GET /developer/job_artifacts/{job_id}: Lists all artifacts for a given job.
- GET /developer/job_artifacts/{job_id}/{artifact_name}: Retrieves a specific artifact, returning a presigned MinIO URL or redirecting the user for download.

These endpoints ensure that only the job owner can access the artifacts, and use MinIO presigned URLs for secure, time-limited access. The example code on how to retrieve a specific artifact is displayed in Listing 5.10.

```
@developer_router.get("/job_artifacts/{job_id}/{artifact_name}",
↳ dependencies=[Depends(get_current_user)])
async def download_artifact(job_id: str, artifact_name: str, user: User =
↳ Depends(get_current_user), db: Session = Depends(get_db), redirect: bool = True):
    ...
    presigned_url = generate_presigned_url(object_name, TRAINING_BUCKET,
↳ expiration=3600)
    if redirect:
        return RedirectResponse(url=presigned_url)
    else:
        return {"artifact_name": artifact_name, "url": presigned_url}
```

Code Fragment 5.10: Developer endpoint for artifact retrieval

5 Implementation

The developer endpoints provide a secure and user-friendly interface for submitting training jobs, tracking their progress, and retrieving results and artifacts. Verification endpoints are also available for users to verify the integrity of their AIBOMs and artifacts. These are described in Section 5.3

5.1.4 Scanner configuration

The scanner is a dedicated Celery worker responsible for periodically scanning both the worker and scanner Docker images for vulnerabilities using Trivy. It runs in its own Docker container, based on a minimal `python:3.11-slim` image with Docker CLI and Trivy dependencies installed. The scanner uses its own Celery app instance, configured to listen to the `scanner_queue` and is scheduled via Celery Beat to execute the scan task every hour.

The main scanning task (`tasks.scan_worker_and_self_images`) retrieves the relevant image names, runs Trivy in a subprocess to scan for vulnerabilities, and uploads the resulting reports to MinIO in dedicated buckets. The scanner dynamically determines its own image and the worker image, ensuring up-to-date scans even after image updates. All results are stored in MinIO for later retrieval and inclusion in the AIBOM.

A simplified code fragment for the scan task performed is presented in Listing 5.11.

5 Implementation

```
def perform_scan(image_name, bucket_name, bucket_prefix):
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    output_file = f"/tmp/{image_name.replace(':',
    ↪ '_')}_vulnerabilities_{timestamp}.json"

    network_name = "internal_network"
    container_name = f"trivy_scanner_{timestamp}"

    trivy_result = subprocess.run(
        [
            "docker", "run", "--rm",
            "-v", "/var/run/docker.sock:/var/run/docker.sock",
            "-v", f"{os.path.expanduser('~')}/.cache/trivy:/root/.cache/",
            "aquasec/trivy:latest", "image", "--scanners", "vuln", "--format",
            ↪ "json", image_name
        ],
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE,
        text=True,
    )
    if trivy_result.returncode != 0:
        raise Exception(
            f"Trivy scan failed for {image_name}: {trivy_result.stderr.strip()}
            ↪ ")

    vulnerabilities = json.loads(trivy_result.stdout)
    with open(output_file, "w") as f:
        json.dump(vulnerabilities, f, indent=4)
```

Code Fragment 5.11: Periodic scanner task for vulnerability scanning

5.1.5 MinIO utilities

The platform uses a set of shared MinIO utility functions to abstract and simplify interactions with the MinIO object storage service. These utilities are implemented in `shared/minio_utils.py` and are used by the API, worker, and scanner components. Key functionalities include:

- **Uploading files:** `upload_file_to_minio(file_path, object_name, bucket_name)` uploads files to a specified bucket and returns the object URL.
- **Downloading files:** `download_file_from_minio(object_name, download_path, bucket_name)` retrieves files from MinIO for local processing.

5 Implementation

- **Bucket management:** `create_bucket_if_not_exists()` ensures all required buckets (for training jobs, worker scans, and scanner scans) exist before use.
- **Listing files:** `list_files_in_bucket(prefix, bucket_name)` lists all files under a given prefix in a bucket, supporting artifact discovery.
- **Presigned URLs:** `generate_presigned_url(object_name, bucket_name, expiration)` generates secure, time-limited URLs for downloading artifacts, enabling secure access without exposing credentials.

5.1.6 Zip utilities

To ensure the integrity and safety of uploaded datasets, the platform includes robust zip file validation and extraction utilities in `shared/zip_utils.py`. These utilities are shared by the API and worker to validate and safely extract user-uploaded zip files, especially for image datasets.

- **Size limits:** Enforces maximum zip file size, total uncompressed size, and individual file size to prevent resource exhaustion attacks.
- **Extension whitelisting:** Only allows files with approved extensions (e.g., JPG, png, CSV) to be extracted.
- **Path traversal protection:** Ensures that extracted files cannot escape the intended extraction directory, preventing directory traversal attacks.
- **Custom exceptions:** Uses a custom `ZipValidationError` to provide clear error messages for invalid or malicious zip files.

A code fragment for validating and extracting a zip file is shown in Listing 5.12.

5 Implementation

```
def validate_and_extract_zip(zip_path, extract_to):
    validate_zip_file(zip_path)
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        total_uncompressed_size = 0
        for member in zip_ref.infolist():
            member_path = os.path.realpath(os.path.join(extract_to, member.filename))
            if not member_path.startswith(os.path.realpath(extract_to)):
                raise ZipValidationError("Path traversal detected in .zip file!")
            # ... (other checks)
        zip_ref.extractall(extract_to)
```

Code Fragment 5.12: Zip utility for safe validation and extraction

5.2 Implementation of the AIBOM generation pipeline

This section describes how the AIBOM is generated at the end of each training job. The process is modular and consists of three main steps: metadata extraction, environment details extraction, and building the AIBOM document itself.

5.2.1 Metadata extraction

Metadata extraction is the first step in the AIBOM generation pipeline. After a training job completes, the worker collects detailed metadata about all relevant artifacts. This includes:

- **Materials:** Input artifacts such as the original model file, dataset, and dataset definition. For each, the worker computes SHA-256 hashes and records their MinIO paths and local paths.
- **Products:** Output artifacts such as the trained model and metrics file, again with hashes and storage locations.
- **Training parameters:** Both required (`fit_params`) and optional parameters (e.g., model name, version, description, framework, license) are included.
- **Attestations:** The MinIO path to the in-toto link file generated during the training step is included as an attestation reference.

It is important to mention that developer-specified (optional) data is included for completeness, but this information cannot be verified by the platform and thus is not trusted.

5 Implementation

A developer could choose not to fill in these parameters carefully. The platform only guarantees the integrity of what the developer has entered in these fields. Independent audits for these parameters are therefore still necessary.

This metadata is grouped into a dictionary structure using the `generate_basic_bom_data` function⁵, which is then passed to the next stage.

5.2.2 Environment details extraction

To ensure full transparency and reproducibility, the worker extracts detailed information about the environment in which the training job was executed. This is handled by the `extract_environment_details` function⁶, which collects:

- **Operating system, Python, and TensorFlow versions**
- **CPU, memory, and disk usage**
- **GPU** (using NVIDIA Management Library (NVML), if available)
- **Celery task information** (task ID, name, queue, unique directory)
- **Docker container and image details**
- **Vulnerability scan results** (summarized from the latest Trivy scan in MinIO)
- **Timestamps** for key events (task start, training start, AIBOM generation start)

5.2.3 Building the AIBOM

The final step is to build the AIBOM document itself, using the CycloneDX Python library. The `transform_to_cyclonedx` function⁷ takes the grouped metadata and environment details and constructs a CycloneDX BOM with the following structure:

- **Metadata:** Includes generator information, authors, supplier, and references to the AIBOMGen platform.
- **Components:**

⁵https://github.com/idlab-discover/AIBOMGen/blob/main/aibomgen-platform/worker/bom_data_generator.py

⁶https://github.com/idlab-discover/AIBOMGen/blob/main/aibomgen-platform/worker/environment_extractor.py

⁷https://github.com/idlab-discover/AIBOMGen/blob/main/aibomgen-platform/worker/transform_to_cyclonedx.py

5 Implementation

- **Environment component:** Captures all environment properties.
- **Data component:** Describes the dataset and dataset definition, including hashes and preprocessing details.
- **Model component:** Describes the trained model, architecture summary, metrics, and training parameters.
- **External references:** Attestations such as the in-toto link file are referenced as external artifacts.
- **Dependencies:** Registers relationships between model, data, and environment components.

The BOM is then serialized to JSON and validated against the CycloneDX v1.6 schema. A digital signature is generated using the worker's private key and included as a property in the BOM metadata. The signed BOM is then uploaded to MinIO for later artifact retrieval and verification.

A simplified code fragment for building and signing the AIBOM is displayed in Listing 5.13.

```
bom = transform_to_cyclonedx(bom_data)
serialize_bom(bom, output_path)
sign_and_include_bom_as_property(bom, private_key_path)
upload_file_to_minio(output_path, minio_object_name, TRAINING_BUCKET)
```

Code Fragment 5.13: Building, signing, and uploading the AIBOM

This pipeline ensures that every training job produces a verifiable, standards-based AIBOM, supporting transparency.

5.3 Implementation details of the Integrity and Security Measures

This section details the integrity and security measures implemented in the AIBoMGen platform, focusing on supply chain verification, artifact signing, authentication, logging, and container isolation. These measures are designed to counter a wide range of tampering and attack scenarios, as outlined in the threat analysis Appendix C.

5 Implementation

5.3.1 Use of in-toto for supply chain integrity and verification

In-toto is used to provide end-to-end supply chain integrity for the training pipeline. At the start of the project, a trusted in-toto layout is generated and signed using a dedicated Ed25519 key pair⁸. The layout defines the expected steps, required materials (inputs), and expected products (outputs) for each job. The public key and signed layout of the platform are distributed as trusted root of verification. Appendix D contains an example layout file.

During each training job, the worker generates an in-toto link file [72] using the `in_toto_link_generator.py` script⁹. This file records the hashes of all input (materials) and output (products) artifacts, the executed command, and relevant metadata. The link file is then signed with the worker's private key and uploaded to MinIO. Appendix D contains an example link file.

Generating a link file is done using the `Link` and `Metablock` classes of in-toto. A simplified code fragment is presented in Listing 5.14.

```
link = Link(
    name=task_name,
    materials=materials_dict,
    products=products_dict,
    byproducts={},
    command=command,
)
metablock = Metablock(signed=link)
metablock.sign(signer)
metablock.dump(link_file_path)
```

Code Fragment 5.14: Generating and signing an in-toto link file

When a user or verifier wants to check the integrity of a training job, they can use the in-toto verification process: the link file is compared against the trusted layout, and the signature is verified using the public key. This ensures that the training process followed the expected steps and that all artifacts are unmodified and included.

⁸https://github.com/idlab-discover/AIBoMGen/blob/main/aibomgen-platform/utils/generate_in-toto_signed_layout.py

⁹https://github.com/idlab-discover/AIBoMGen/blob/main/aibomgen-platform/worker/in_toto_link_generator.py

5 Implementation

5.3.2 BOM signature and verification

In addition to in-toto, the CycloneDX AIBOM itself is cryptographically signed using the same Ed25519 private key as the in-toto link files. The signature is embedded as a property in the BOM metadata. This provides a second layer of integrity: users can verify that the BOM has not been tampered with and that it was generated by a trusted worker. The verification endpoints in the API allow users to check both the in-toto link file and the BOM signature.

An embedded signature in the AIBOM was chosen to ensure that the AIBOM document itself serves as the single source of truth for both the artifact metadata and its integrity. By embedding the signature directly within the BOM, anyone retrieving the AIBOM from storage can immediately verify its authenticity and integrity without needing to reference or manage a separate signature file. This approach simplifies distribution, reduces the risk of mismatched or lost signatures, and aligns with the principle of self-contained, verifiable artifacts.

However, the CycloneDX Python library does not yet support embedded signatures in the BOM [65], which is a feature of the v1.6 specification [24]. To work around this limitation, the BOM is first serialized and signed using the platform's private key. The resulting signature is then added as a custom property in the BOM metadata section. When verifying the BOM, this signature property is first extracted and removed from the BOM, after which the signature is verified against the rest of the BOM content using the platform's public key.

The full AIBOM generation pipeline including the generation and signing of the referenced in-toto link file is presented in Figure 5.2.

5 Implementation

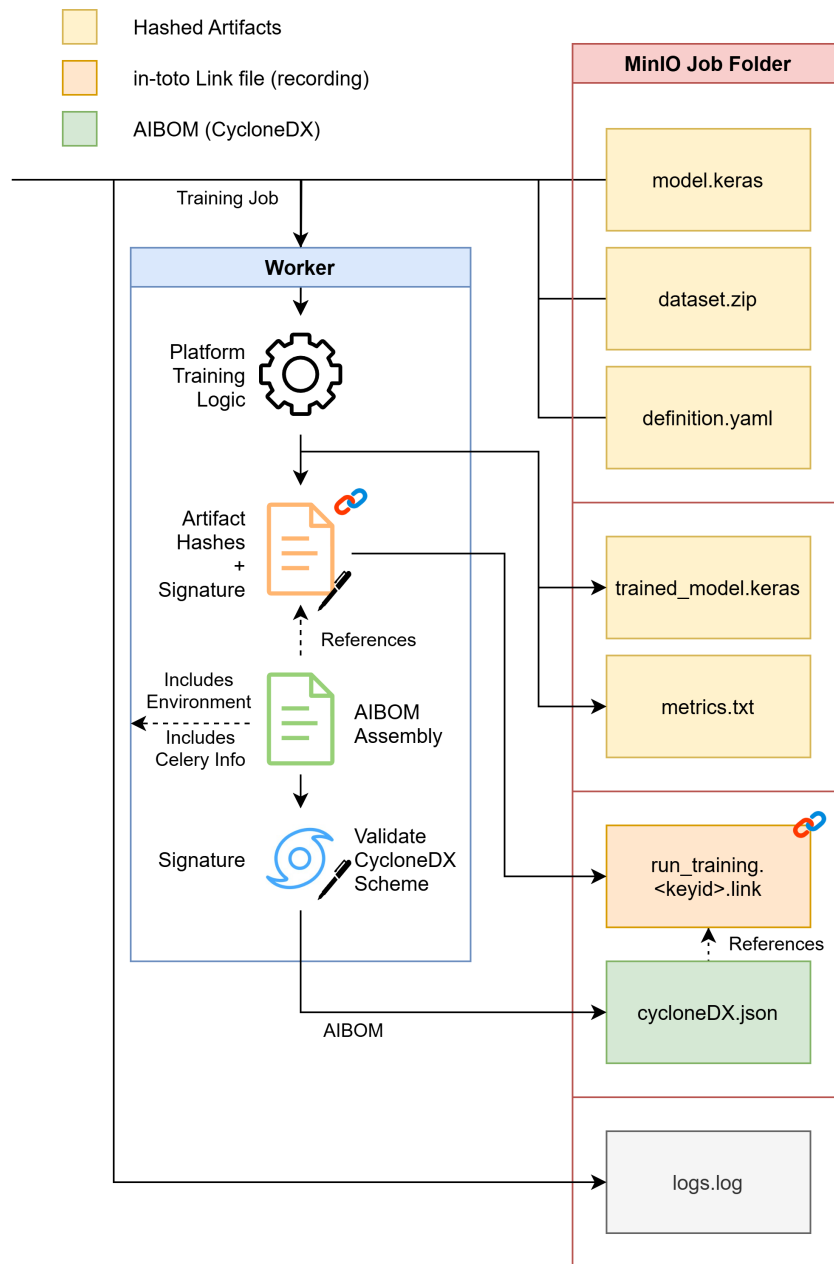


Figure 5.2: AIBOM and link file generation

5.3.3 Logging

Comprehensive logging is implemented throughout the platform, especially in the worker. All significant events, errors, and process steps are logged using Python's logging framework. Logs are written both to the console (for real-time monitoring) and to files, which are then uploaded to MinIO for persistent storage and later auditing. This ensures traceability and supports analysis in case of incidents.

5 Implementation

5.3.4 Authentication with Microsoft Entra ID

Authentication and authorization are enforced using Microsoft Entra ID (formerly AAD). The platform requires three app registrations in Azure:

- **Frontend registration:** Used by the React/Next.js front end for user authentication and acquiring tokens.
- **FastAPI OpenAPI registration:** Allows secure access to the OpenAPI docs and interactive API testing.
- **API backend registration:** Protects the main API endpoints, ensuring only authenticated users and services can access sensitive operations.

How this is setup is done, is explained in the README of the project.

As said before, the FastAPI backend uses the `fastapi-azure-auth` library to validate JWT tokens, enforce scopes, and extract user identities. Endpoints are protected using dependency injection, and user-job associations are enforced at the database level to prevent unauthorized access to artifacts or job metadata.

5.3.5 Summary of security measures

In summary, several security measures are implemented to further harden the platform:

- **Container isolation:** All worker containers run with minimal privileges (`cap_drop: [ALL]`) [47], in read-only mode, and as non-root users. Temporary storage is provided via `tmpfs` mounts [70]. The Docker socket is only mounted where necessary.
- **Safe file validation and extraction:** All uploaded zip files are validated for size, structure, and allowed extensions, and are extracted using secure routines that prevent path traversal and zip bomb attacks (`shared/zip_utils.py`).
- **TensorFlow safe mode:** Models are loaded using `tf.keras.models.load_model(..., safe_mode=True)` [73], which prevents arbitrary code execution via malicious Lambda layers in model files [74].
- **Vulnerability scanning:** All worker and scanner images are regularly scanned for vulnerabilities using Trivy, and results are stored in MinIO and included in the AIBOM.

5 Implementation

- **Hash-based integrity:** All artifacts (inputs and outputs) are hashed and their hashes are included in both the in-toto link file and the AIBOM, enabling independent verification of artifact integrity.
- **Access control:** MinIO buckets can be configured with strict access controls, and all API endpoints are protected by authentication and authorization checks.

These security and integrity measures ensure that the AIBoMGen platform provides strong guarantees against tampering, unauthorized access, and supply chain attacks, as detailed in the threat analysis, Appendix C.

5.4 Explanation of the AIBOM verification endpoints

This section explains the verification endpoints provided by the API for end-users and auditors to independently verify the integrity of training jobs and their artifacts. Each endpoint corresponds to a specific aspect of the supply chain and artifact verification process. The following subsections describe the logic and flow of each endpoint, supported by their state diagrams, Figures 5.3, 5.4, 5.5 and 5.6.

5.4.1 Verify in-toto link file

The `/verify_in-toto_link` endpoint verifies the integrity of an uploaded in-toto link file against the signed layout. This ensures that the link file is valid, correctly signed, and that the recorded steps, materials, and products match the platform's supply chain policy.

5 Implementation

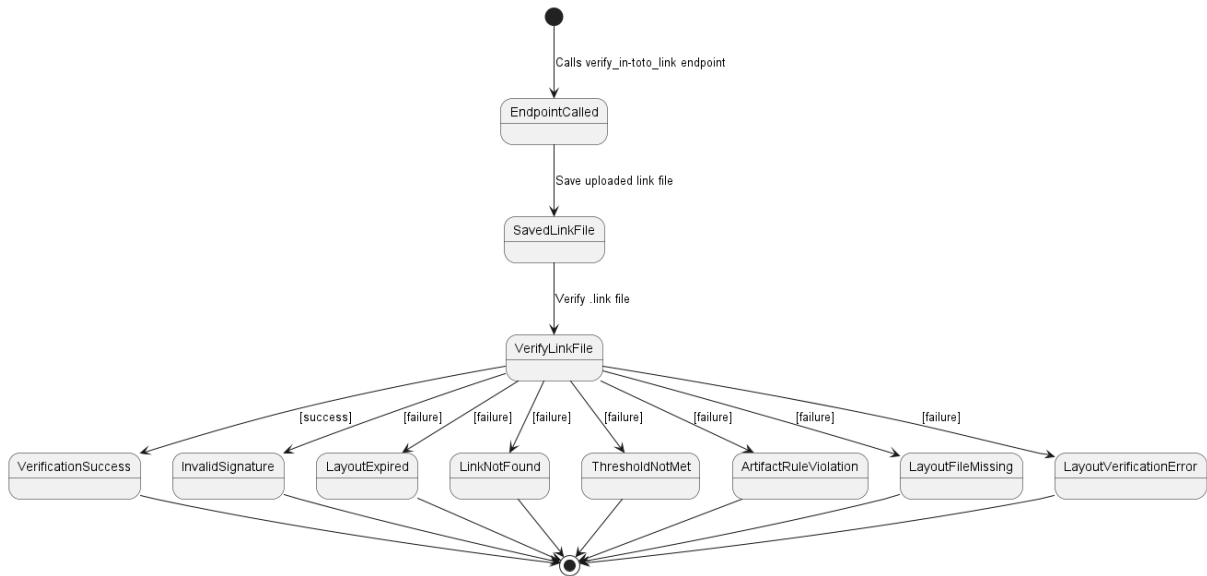


Figure 5.3: State diagram: Verification of an in-toto link file

The process involves saving the uploaded link file, verifying its signature, and checking it against the signed layout. Possible outcomes include successful verification, invalid signature, layout expiration, missing layout, or artifact rule violations.

5.4.2 Verify AIBOM

The `/verify_bom_and_link` endpoint verifies both the AIBOM (CycloneDX BOM) and its associated in-toto link file. The endpoint first validates the BOM against the CycloneDX schema, then extracts and verifies the embedded signature. If successful, it extracts the reference to the link file and downloads it from MinIO, then verifies the link file as described above.

5 Implementation

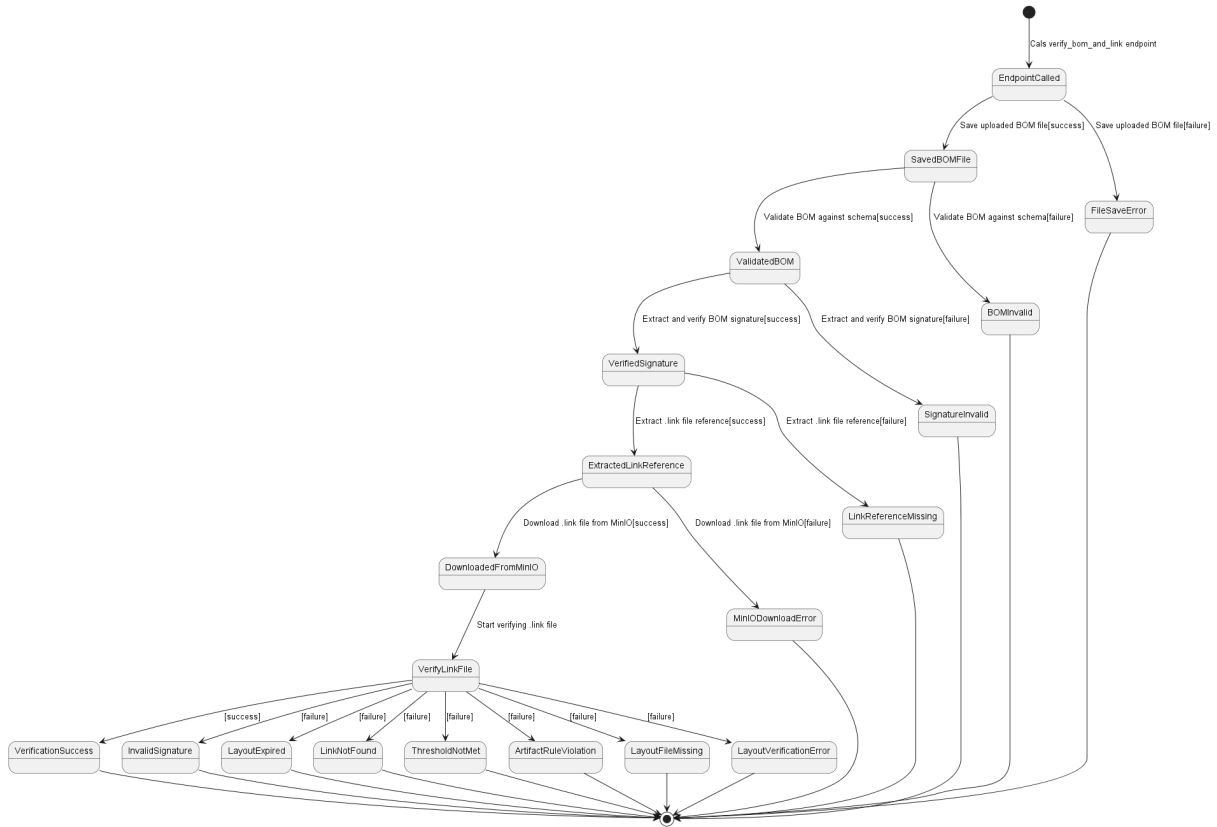


Figure 5.4: State diagram: Verification of BOM and associated in-toto link file

This endpoint ensures that the BOM is authentic, unmodified, and that its generation (as recorded in the link file) is valid and trustworthy.

5.4.3 Verify MinIO artifacts

The `/verify_minio_artifacts` endpoint allows users to verify that the actual artifacts stored in MinIO (such as models, datasets, and metrics) match the hashes recorded in a previously verified in-toto link file. The endpoint checks both materials and products, reporting any mismatches.

5 Implementation

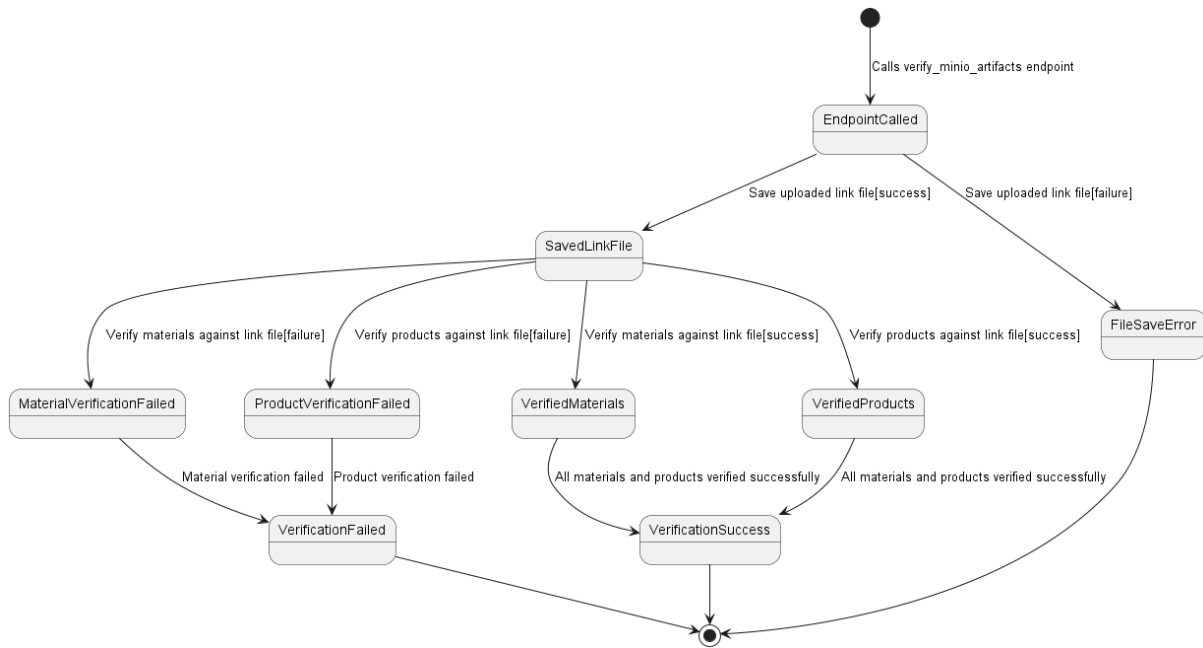


Figure 5.5: State diagram: Verification of MinIO artifacts against the link file

This endpoint is crucial for end-to-end integrity, ensuring that the files available for download are exactly those produced and recorded by the trusted training process. It does not check the link file integrity, so before this endpoint is called, the `/verify_in-toto_link` endpoint should be successful.

5.4.4 Verify uploaded file hash

The `/verify_file_hash` endpoint allows users to upload a file (e.g., a model or dataset) and verify its hash against the values recorded in a previously verified in-toto link file. This is useful for confirming that a local copy of an artifact matches the official, trusted version. It does not check the link file integrity, so before this endpoint is called, the `/verify_in-toto_link` endpoint should be successful.

5 Implementation

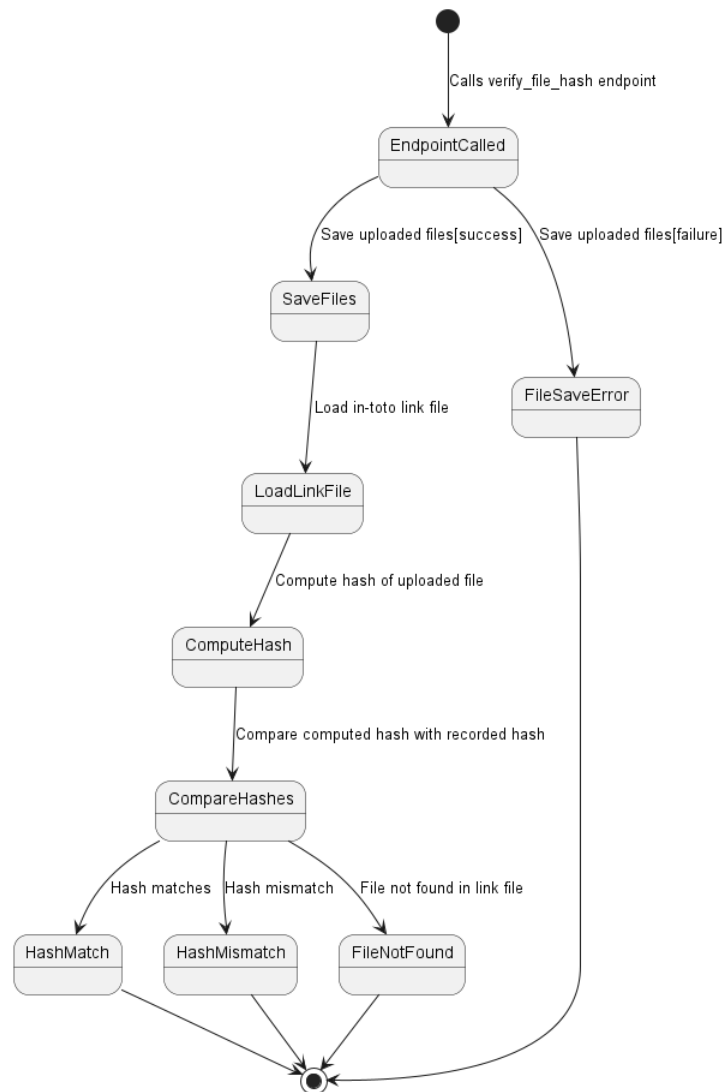


Figure 5.6: State diagram: Verification of an uploaded file's hash against the link file

The endpoint computes the hash of the uploaded file, compares it to the recorded hash, and reports whether it matches, mismatches, or is not found in the link file.

5 Implementation

5.5 Front-end dashboard for demo purposes

5.5.1 Front end in React (Next.js)

The front-end dashboard for AlBoMGen is meant to enhance the demo experience and provide a cohesive, visual showcase of the platform's capabilities. It is a modern single-page application using React with the Next.js framework [75]. It is styled using TailwindCSS [76] and Shadcn [77] components, ensuring a consistent and responsive design. The use of Shadcn also enabled very fast development, because of the use of practical, ready-to-use UI components.

Authentication and authorization are handled via Microsoft Entra ID using the Microsoft Authentication Library (MSAL) [78]. The dashboard communicates with the FastAPI backend through API calls.

5.5.2 Front-end functionalities

Key features of the dashboard include:

- Job Submission and monitoring
- Artifact Retrieval
- Viewing Worker Statistics
- User Profile and Settings: Authenticated users can view their profile information (fetched from Microsoft Graph) and adjust settings such as theme and language.
- Internationalization
- Integrated verification endpoints

Figures 5.7, 5.8, 5.9 and 5.10 display some main pages of the dashboard.

5 Implementation

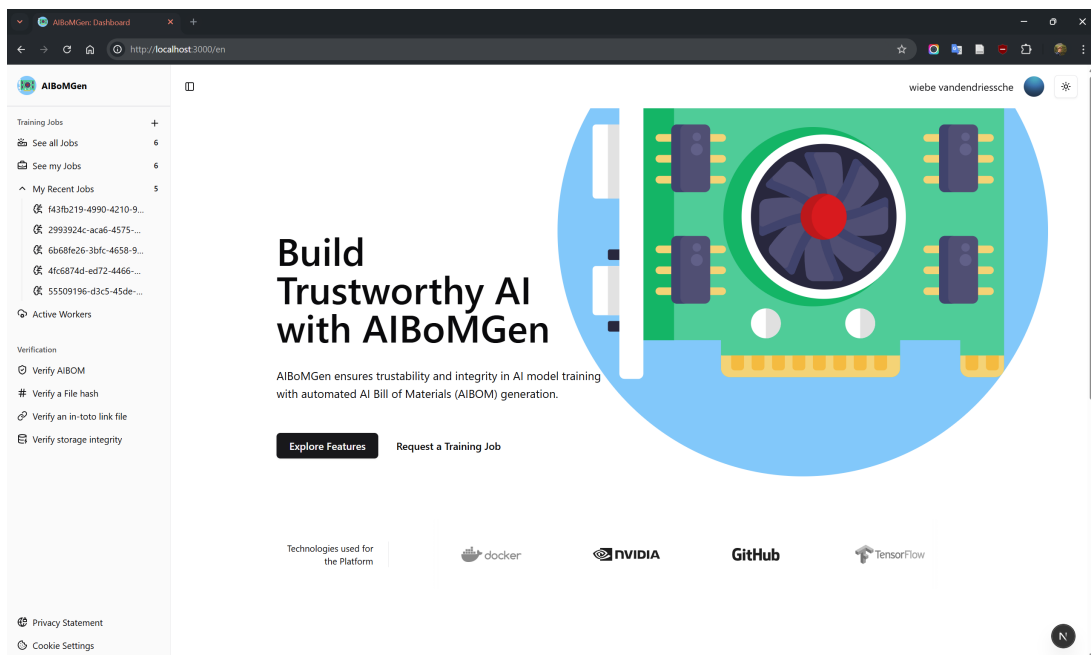


Figure 5.7: Dashboard Landing Page: Welcome screen

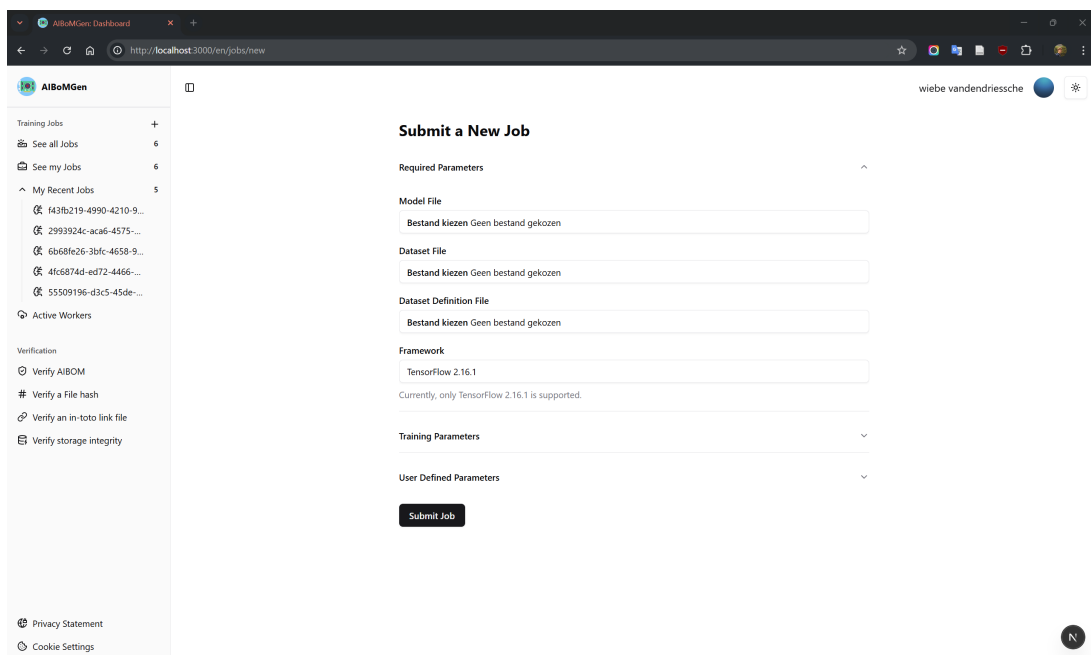
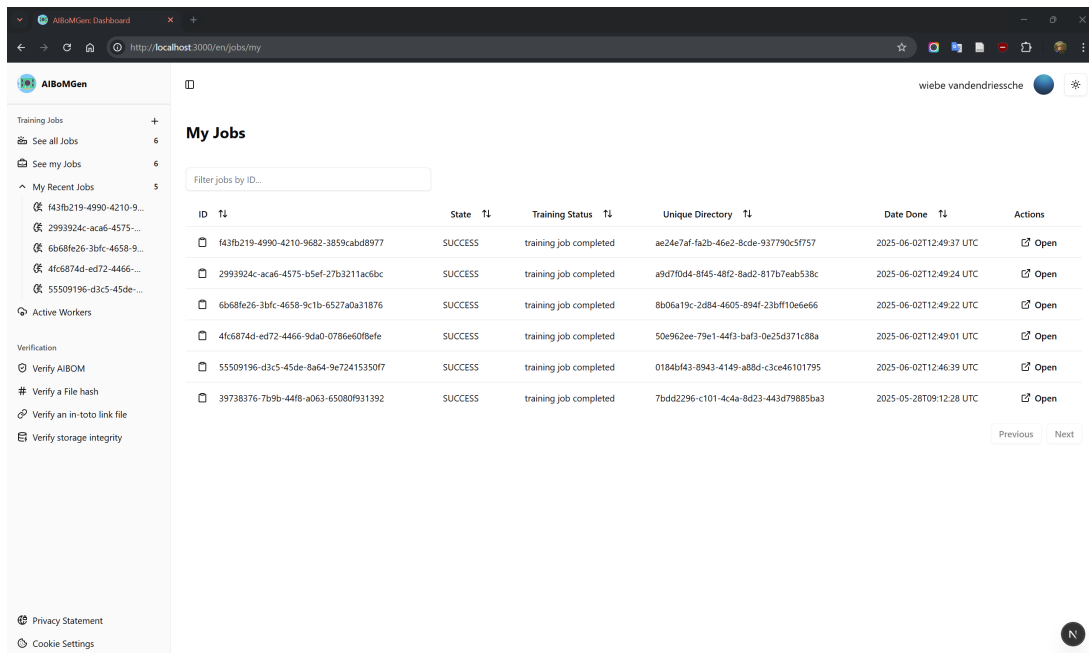


Figure 5.8: Job Submission Page: Users can submit new training jobs with required files and parameters

5 Implementation



The screenshot shows the 'My Jobs' page in the AIBoMGen dashboard. The page displays a table of submitted jobs with the following columns: ID, State, Training Status, Unique Directory, Date Done, and Actions. The table lists 7 jobs, all of which are in a 'SUCCESS' state and have a 'training job completed' status. The unique directories are long alphanumeric strings. The completion dates range from May 28, 2025, to June 2, 2025. The left sidebar contains navigation links for Training Jobs, Verification, and Active Workers. The top right shows the user's name 'wiebe vandendriessche' and a search icon.

ID	State	Training Status	Unique Directory	Date Done	Actions
f43fb219-4990-4210-9...	SUCCESS	training job completed	ae24e7af-fa2b-46e2-8cde-937790c5f757	2025-06-02T12:49:37 UTC	Open
2993924c-acaf-4575-...	SUCCESS	training job completed	a9d7f0d4-8f45-48f2-8ad2-817b7eab538c	2025-06-02T12:49:24 UTC	Open
6b68fe26-3bfc-4658-9...	SUCCESS	training job completed	8b06a19c-2d84-4605-894f-23bf10e6e66	2025-06-02T12:49:22 UTC	Open
4fc6874d-ed72-4466-...	SUCCESS	training job completed	50e962ee-79e1-44f3-baf3-0e25d371c88a	2025-06-02T12:49:01 UTC	Open
55509196-d3c5-45de-8a64-9e72415350f7	SUCCESS	training job completed	0184bf43-8943-4149-a88d-c3ce46101795	2025-06-02T12:46:39 UTC	Open
39738376-7b9b-44f8-a063-65080f931392	SUCCESS	training job completed	7bdd2296-c101-4c4a-8d23-443d79885ba3	2025-05-28T09:12:28 UTC	Open

Figure 5.9: My Jobs Page: Overview of user's submitted jobs and their statuses

AIBoMGen: Dashboard

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

Figure 5.10: Worker Statistics Page: Real-time statistics of active Celery workers

6

Evaluation

After the design and implementation of the AIBOM generation system, it was essential to verify its functionality, robustness, and performance under realistic conditions. This chapter presents an evaluation of the system based on a series of test cases, performance benchmarks and finally a threat analysis. The goal is to quantify the overhead introduced by AIBOMs and assess the system's resilience against malicious actors.

6.1 Basic Evaluation

Due to the complex infrastructure of the system, which consists of multiple Docker components, it was challenging to fully automate the testing process while developing. As a result, most tests during development were performed manually by submitting training jobs to the system and verifying the different outputs the system produced. For the final evaluation, experiments were conducted using a series of Jupyter Notebooks, which are published together with results on Zenodo [79]. The latest version of the experiments are available on GitHub¹. The notebooks include detailed code and instructions for replicating the experiments described in this chapter. During testing and evaluation, the backend was run on a host machine equipped with an AMD Ryzen 7 5800H CPU, 32GB RAM, and a NVIDIA GeForce RTX 3060 GPU.

6.1.1 Successful Training Jobs

Since developing a flexible API was not the main focus of this thesis, the API is limited to tabular and image classification or regression tasks. The API is designed to be extensible, allowing for the addition of new tasks in the future. The following sections describe the successful types of training jobs that were executed.

¹<https://github.com/idlab-discover/AIBoMGen-experiments>

6 Evaluation

Tabular Classification with CSV

A tabular classification task was evaluated using the Wine Quality dataset in CSV format [80]. The dataset consists of physicochemical properties of wine samples, such as acidity, sugar content, pH, and alcohol level. For this evaluation, the quality score of the wine, a numerical value ranging from 0 (lowest quality) to 10 (highest quality), was explicitly chosen as the target label to predict. A simple layered model was used with some dense layers. More details on the models and other input artifacts used for the tests can be found in the GitHub repository. The system successfully ingested the CSV file, applied the specified preprocessing steps, and executed the training job, resulting in a valid AIBOM. The implementation of this experiment is available in the notebook `tabular_test.ipynb`.

Tabular Classification with TFRecord

To test support for TFRecord, the Wine Quality dataset was also converted to TFRecord format [57]. The system correctly parsed the TFRecord files, handled the feature and label mappings as defined in the YAML specification, and produced a valid AIBOM for the training run. This experiment is implemented in `tfrecord_test.ipynb`. Training results were not as good as with the CSV dataset, which can be attributed to the training logic in the worker container. The logic was not optimized as this was not the focus of this thesis.

Image Classification with MNIST zip dataset

An image classification task was performed using the MNIST dataset [81], provided as a zip folder of png images organized by class. The system processed the images, applied normalization, and trained a convolutional neural network. To prove that training on the platform is actually feasible, Figure 6.1 illustrates the results from the metrics file included in the output of the training. These show positive training results for this specific task. The resulting AIBOM for this evaluation run is included in Appendix E as an example. The implementation details are available in `mnist_test.ipynb`.

6 Evaluation

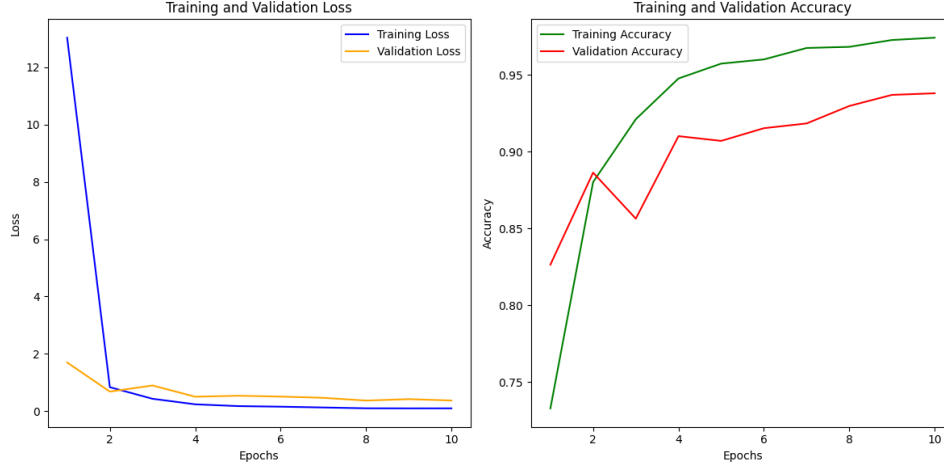


Figure 6.1: Training and validation accuracy and loss for the MNIST image classification task

Image Classification with CIFAR10 zip dataset

The CIFAR-10 dataset [82], also provided as a zipped folder of images, was used to evaluate image classification capabilities on a more complex dataset. The system successfully handled the multi-class image data, executed the training job, and generated a corresponding AIBOM. This experiment is implemented in `cifar_test.ipynb`.

6.2 Performance

To analyze the performance impact of AIBOM generation, a series of training jobs were submitted with increasing numbers of epochs. For each job, the time spent on model training and the time required for AIBOM generation were measured. The results are summarized in Table 6.1.

The performance evaluation was conducted using the CIFAR-10 dataset and a convolutional neural network. The training jobs were executed for different epoch counts: 1, 2, 5, 10, 50, and 100 epochs. The training time, AIBOM generation time, and total time were recorded for each configuration. The implementation of this experiment is available in the notebook `increasing_epochs_test.ipynb`.

As shown in Table 6.1, the training time increases with the number of epochs, as expected, while the AIBOM generation time remains nearly constant regardless of the number of epochs. Consequently, the percentage overhead of AIBOM generation decreases as the training time increases. This trend is visualized in Figure 6.2, which plots

6 Evaluation

both the training time and the AIBOM overhead for different epoch counts. Figure 6.3 further illustrates that the AIBOM overhead becomes negligible for longer and more realistic training jobs. Overall, these results confirm that AIBOM generation overhead is independent of training time and does not increase with longer training jobs.

Table 6.1: Training and AIBOM Generation Times for Different Epochs

Epochs	Training Time (s)	AIBOM Time (s)	Total Time (s)	AIBOM Overhead (%)
1	4.316	0.265	4.581	5.78
2	6.629	0.234	6.863	3.41
5	12.492	0.218	12.710	1.72
10	23.790	0.196	23.986	0.82
50	117.728	0.198	117.926	0.17
100	230.837	0.197	231.034	0.09

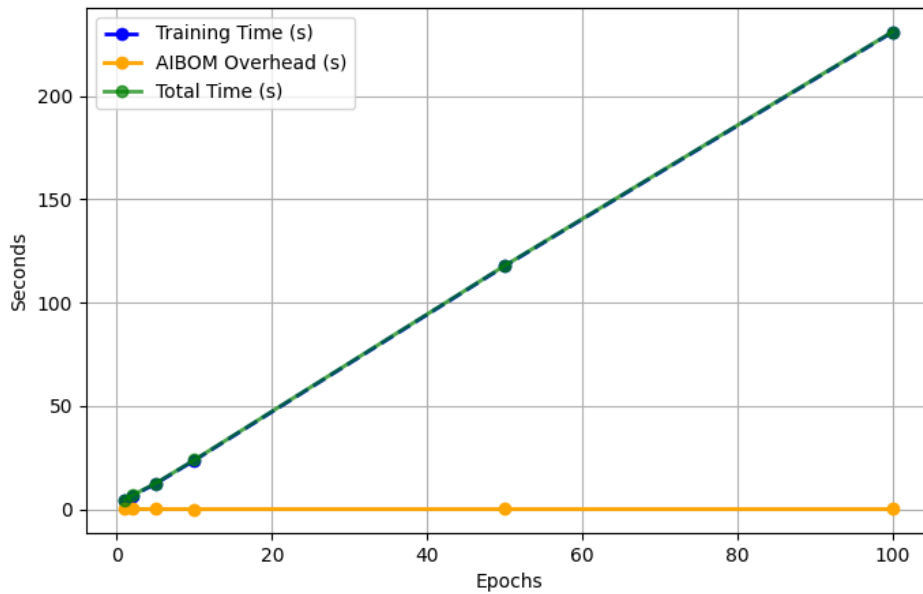


Figure 6.2: Training time and AIBOM overhead for different epochs

6 Evaluation

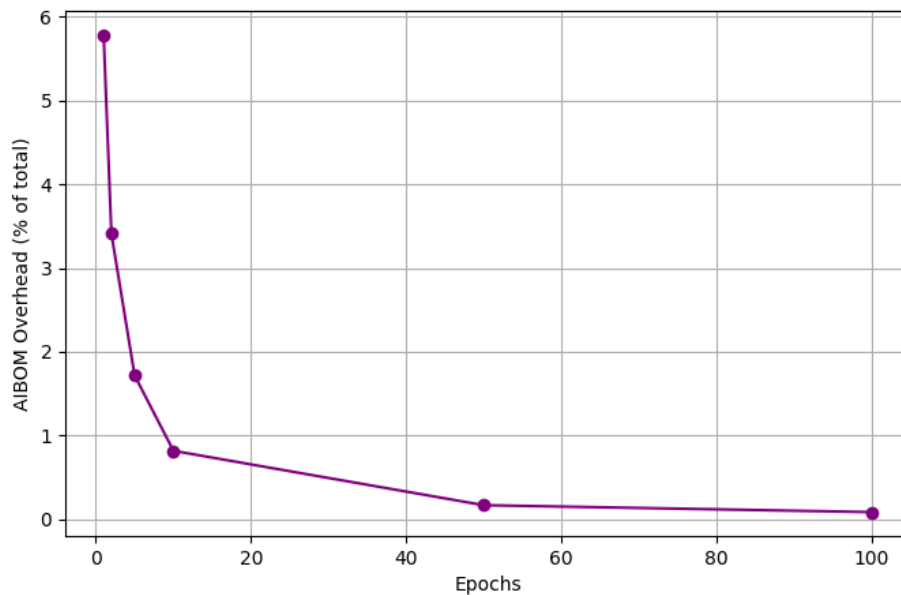


Figure 6.3: AIBOM overhead as percentage of total time

6.3 Storage Impact

To assess the storage impact of AIBOM generation, the sizes of all relevant artifacts produced during training jobs were measured for different dataset sizes, model sizes, and epoch counts. The focus was on the size of the generated AIBOM compared to other files such as the trained model, dataset, and logs.

6.3.1 Impact of Dataset Size

The MNIST dataset was used with varying numbers of images per class (100, 200, 300, 400, and 500). A simple convolutional neural network was used. The sizes of the dataset, trained model, AIBOM, and other artifacts were recorded and analyzed. The implementation details are available in `increasing_dataset_test_storage.ipynb`.

Table 6.2 and 6.3 show the file sizes for jobs with increasing dataset sizes. As the dataset size increases, the size of the AIBOM also increases slightly, since it contains dataset-specific metadata such as hashes, file names, and references. However, this increase is marginal compared to the size of the dataset, as the AIBOM does not include the dataset itself.

6 Evaluation

Table 6.2: Artifact Sizes for Different Dataset Sizes - Part 1

Dataset Size	Dataset Zip (KB)	Definition (KB)	Input Model (KB)	Trained Model (KB)	AIBOM (KB)
100	384.12	0.15	2,738.69	8,163.10	13.474
200	765.18	0.15	2,738.69	8,163.10	13.477
300	1,153.45	0.15	2,738.69	8,163.10	13.478
400	1,537.13	0.15	2,738.69	8,163.10	13.477
500	1,923.44	0.15	2,738.69	8,163.10	13.478

Table 6.3: Artifact Sizes for Different Dataset Sizes - Part 2

Dataset Size	Logs (KB)	Metrics (KB)	RunLink (KB)	Total Folder (KB)	AIBOM% of Trained Model
100	4.25	0.13	1.28	11,305.19	0.17
200	4.25	0.13	1.28	11,686.25	0.17
300	4.25	0.13	1.28	12,074.52	0.17
400	4.25	0.13	1.28	12,458.20	0.17
500	4.25	0.13	1.28	12,844.52	0.17

The relationship between dataset size and artifact sizes is visualized in Figure 6.4, which shows that the AIBOM size remains small relative to the dataset and model sizes, even as the dataset grows.

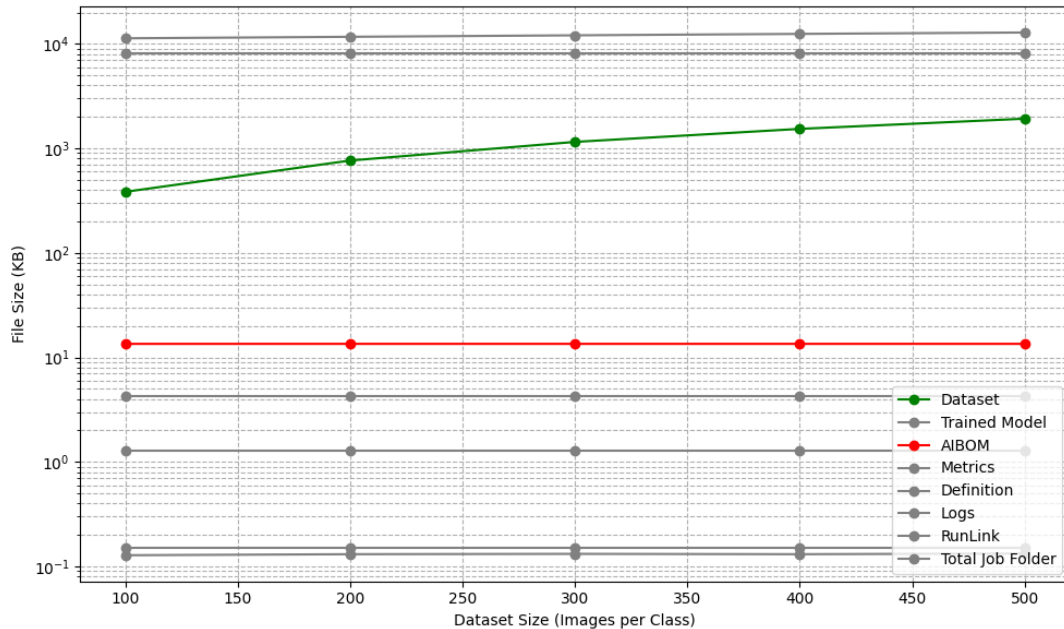


Figure 6.4: File sizes of all artifacts versus dataset size (log scale)

6 Evaluation

6.3.2 Impact of Model Size

A convolutional neural network with varying numbers of units in the first convolutional layer (8, 16, 32, 64, and 128) was trained using the MNIST dataset. The sizes of the input model, trained model, AIBOM, and other artifacts were measured. This experiment is implemented in `increasing_model_test_storage.ipynb`.

Table 6.4, 6.5 and Figure 6.5 show the impact of increasing model size on artifact sizes. The AIBOM size increases only slightly with model size, remaining much smaller than the trained model file. This can be attributed to the fact that the AIBOM includes a layer summary and other model metadata, but does not store the entire model weights or architecture, as this would significantly increase BOM sizes for large models.

Table 6.4: Artifact Sizes for Different Model Sizes - Part 1

Model Units	Dataset (KB)	Definition (KB)	Input Model (KB)	Trained Model (KB)	AIBOM (KB)
8	1,923.44	0.12	708.06	2,071.22	13.287
16	1,923.44	0.12	1,384.95	4,101.86	13.286
32	1,923.44	0.12	2,738.70	8,163.11	13.287
64	1,923.44	0.12	5,446.20	16,285.61	13.289
128	1,923.44	0.12	10,861.21	32,530.61	13.291

Table 6.5: Artifact Sizes for Different Model Sizes - Part 2

Model Units	Logs (KB)	Metrics (KB)	RunLink (KB)	Total Folder (KB)	AIBOM% of Trained Model
8	4.25	0.13	1.28	4,721.79	0.64
16	4.25	0.13	1.28	7,429.32	0.32
32	4.25	0.13	1.28	12,844.32	0.16
64	4.25	0.13	1.28	23,674.33	0.08
128	4.25	0.13	1.28	45,334.33	0.04

6 Evaluation

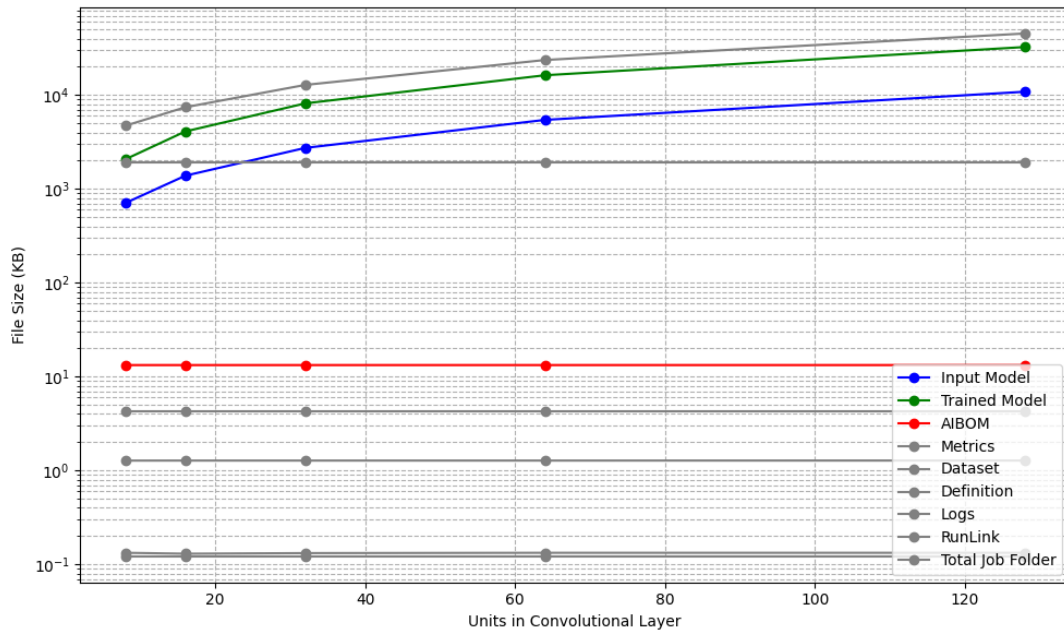


Figure 6.5: File sizes of all artifacts versus model size (log scale)

6.3.3 Impact of Epoch Count

The storage impact was also analyzed for different epoch counts. The MNIST dataset was used to train a fixed model for counts (1, 2, 5, 10, and 50). The implementation details are available in `increasing_epochs_test_storage.ipynb`.

Figure 6.6 and Tables 6.6 6.7 show that the AIBOM size increases slightly with the number of training epochs, as additional metrics from each epoch are included in the AIBOM. This effect is further illustrated in Figure 6.7, which shows that both the AIBOM and metrics file sizes grow gradually as the number of epochs increases.

Table 6.6: Artifact Sizes for Different Epoch Counts - Part 1

Epochs	Dataset (KB)	Definition (KB)	Input Model (KB)	Trained Model (KB)	AIBOM (KB)
1	1,923.44	0.12	2,738.69	8,163.1	13.46
2	1,923.44	0.12	2,738.69	8,163.1	13.54
5	1,923.44	0.12	2,738.69	8,163.1	13.78
10	1,923.44	0.12	2,738.69	8,163.1	14.17
50	1,923.44	0.12	2,738.69	8,163.1	17.31

6 Evaluation

Table 6.7: Artifact Sizes for Different Epoch Counts - Part 2

Epochs	Logs (KB)	Metrics (KB)	RunLink (KB)	Total Folder (KB)	AIBOM% of Trained Model
1	4.25	0.13	1.27	12,844.47	0.165
2	4.25	0.21	1.27	12,844.63	0.166
5	4.25	0.44	1.27	12,845.09	0.169
10	4.25	0.83	1.27	12,845.87	0.174
50	4.25	3.97	1.27	12,852.15	0.212

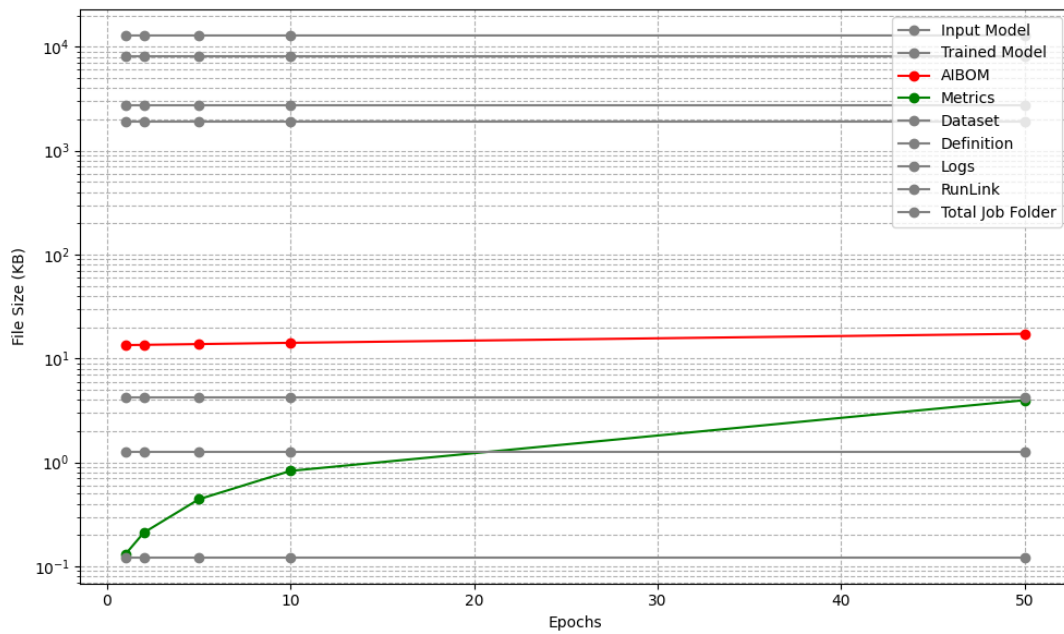


Figure 6.6: File sizes of all artifacts versus number of epochs (log scale)

6 Evaluation

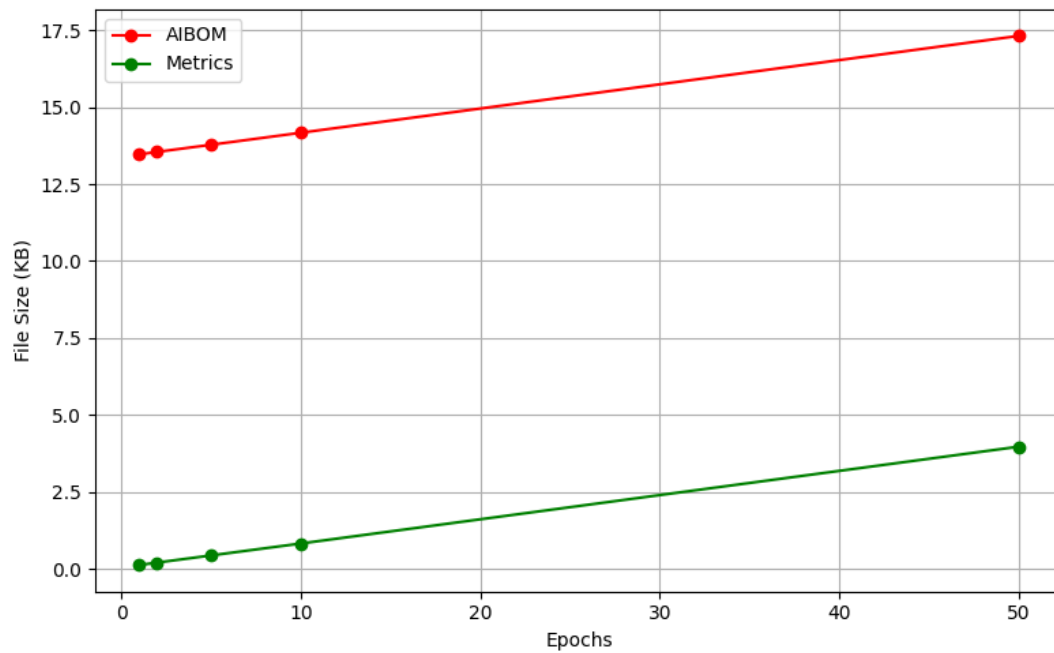


Figure 6.7: AIBOM and metrics file sizes versus number of epochs

Overall, these results demonstrate that the storage overhead introduced by AIBOM generation is minimal compared to the size of the trained model and dataset, and scales only slightly with increasing dataset or model complexity and epoch count.

6.4 Security & Threat Evaluation

A threat analysis was conducted to evaluate the trustworthiness and the effectiveness of its integrity verification mechanisms (Appendix C). The evaluation focused on tampering scenarios, including model and dataset manipulation, training process interference, artifact modification, and attempts to bypass AIBOM generation.

6.4.1 Testing Approach:

- **Manual Testing:** Key countermeasures were validated through manual tests, such as attempting to upload models with unsafe Lambda layers, providing malformed datasets, and modifying artifacts post-generation.
- **Integrity Verification:** Hashes and cryptographic signatures of artifacts and AIBOMs were checked to ensure tampering could be detected.
- **Limitations:** Some advanced attack scenarios (e.g., exploits, key compromise, or sophisticated adversarial attacks) were not tested due to resource and scope constraints.

For a full breakdown on what and how the tests were performed, refer to Table C.1 in Appendix C.

6.4.2 Summary of Findings:

- The implemented countermeasures (safe model loading, container isolation, hash/signature verification) were effective against common tampering attempts.
- The system reliably detected and prevented basic integrity violations and unauthorized modifications.
- Remaining risks include potential vulnerabilities in dependencies, key management, and undetectable malicious content in valid models or datasets.

For a full breakdown of threat scenarios, countermeasures, and further improvements, refer to Appendix C.

Conclusion

This thesis set out to investigate how AI Bill of Materials (AIBOM) can be implemented as a standardized and trustworthy approach to ensure transparency in AI development. The research was done with four primary questions in focus:

- **What are the current advancements and challenges in the research and development of AIBOMs?**

The research revealed that while AIBOMs are getting traction, existing standards like CycloneDX and SPDX are evolving to fully support AI-specific components. A major challenge lies in the lack of secure, attested, automated, and standardized workflows for generating AIBOMs during the training process. Current industry practices often prioritize flexibility over security, leaving gaps in trustworthiness and compliance.

- **In what ways can AIBOMs support compliance with AI regulations, such as the AI Act, and other emerging standards?**

AIBOMs can play an important role in meeting transparency requirements of the AI Act. It provides a verifiable record of datasets, dependencies, and training processes, which enables organizations to demonstrate compliance. However, the lack of widespread adoption, and most of all, tooling for AIBOMs remains a barrier.

- **How can trust be effectively integrated into the AIBOM framework to ensure transparency, integrity, and ethical concerns?**

The PoC demonstrated that trust can be integrated through cryptographic attestations, integrity checks, and controlled execution environments. A key lesson is that many industry-standard training workflows rely on custom Docker containers and training scripts, which are not inherently secure or attested. There is a need for secure-by-design AI development platforms.

- **What methods and technologies can be used to automate the generation of AIBOMs while ensuring their reliability and security?**

The PoC showed that automation is feasible using tools like CycloneDX Python library, in-toto (or other attestation tools), and containerized workflows. The reliance on emerging standards and incomplete library support (e.g., CycloneDX Python library) also underscores the need for further development in this area.

Conclusion

Key Insights and Lessons Learned:

The research highlights that while AIBOMs are a promising tool for enhancing transparency and compliance, their adoption faces significant technical and cultural challenges. The lack of secure and attested training workflows in many AI development environments is a critical gap that must be addressed. Additionally, the reliance on manual input for certain AIBOM components, such as ethical considerations, limits their trustworthiness.

The PoC demonstrated that it is possible to generate verifiable and tamper-resistant AIBOMs with minimal performance overhead. However, scaling this approach to real-world, production-grade systems will require further advancements in tooling, standardization, and adoption.

Main Contributions:

- Analysis of the **state of the art** in **AIBOMs**
- Analysis of the requirements for **trusted AIBOMs**
- **The development of a PoC** for automated, standards-based, machine-readable AIBOM generation
- **Evaluation** of its performance, efficiency and security

In conclusion, this thesis highlights the potential of AIBoMGen (and similar platforms) as a foundational step toward building secure and transparent AI ecosystems, enabling compliance with regulatory frameworks like the EU AI Act.

Future Work

While the current state of the application demonstrated the possibility of generating and verifying AIBOMs, there are several areas for improvement:

Application and format expansion

- **Lack of different machine learning algorithms and model and data formats:** An expansion of supported algorithms and formats would be a great improvement. One way could be by utilising an open-source library/API that already supports a wide range of algorithms and formats, but still enforces strict workflows and data formats. KubeFlow Pipelines and Ray's library could be possible competitors. It's important when using this, remote code execution is avoided, to maintain trustability and security.
- **Scalability and resource management:** The use of orchestration tools like KubeFlow and Ray would also be a good way to improve scalability and resource management. These are more fit for a production environment.
- **Integration of SPDX 3.0:** Integration of SPDX 3.0 would also be a promising direction, as this is a very recent standard supporting AIBOM. A CycloneDX/SPDX converter or SPDX generator would improve interoperability.
- **Immutable and transparent storage:** Another area that could be looked into is the use of a blockchain-based ledger for AIBOM storage. This allows for immutable and transparent storage of AIBOMs. The BOMs can be directly retrieved and verified from the blockchain. This would also allow for a more decentralized approach to AIBOM storage, which could be beneficial in terms of security and trustability.

Compliance and Automation

- **Mapping to regulatory requirements:** The mapping of certain AIBOM fields to regulatory requirements could be automated.
- **Dataset quality assessment and bias detection:** Integration of tools for dataset quality assessment and bias detection could be beneficial, as for the moment, this is a manual process.

Future Work

- **Model lineage tracking:** A mechanism for model lineage tracking is also an important feature that should definitely be added. This allows for better tracking of models and datasets. If, for example, a dataset was deemed to be biased, it would be possible to track which models were trained on this dataset.

Security and Trustability

- **Custom training scripts:** A thorough evaluation should be made of methods to allow custom training scripts without exposing the platform to arbitrary code execution. This would mitigate the need for restricted workflows and strict APIs and allow for more flexibility.
- **Confidential computing:** A great start would be to investigate confidential computing or TPM-based approaches, like in the AICert project [39].
- **Attestation methods:** Another area of improvement is the use of different attestation methods, such as Sigstore [83] [84] or the use of certificates, like W3C certificates [85].
- **Threat countermeasures:** Additional recommendations on threat countermeasures and security are available at the end of Appendix C.

Societal Reflection

The integration of AI into modern IT systems brings exciting opportunities and much innovation, but it also introduces important ethical considerations. At the moment AI systems are being used more and more for decision-making in high-risk applications like healthcare, finance and public administration. This raises questions about the transparency, accountability and fairness of these systems. This dissertation provides an overview of the concept of AIBOMs and how they can be used to improve that transparency.

A central motivation for this work is the EU's AI Act, which sets strong regulations, especially for high-risk AI systems, including transparency. The system documents the lineage of AI models and their datasets, training environment and code, to ensure transparency. This makes it possible to verify datasets and if ethical and legal standards are met.

Ethical considerations are embedded in the PoC design. The platform addresses issues such as potential malicious tampering by implementing cryptographic attestations and other security measures. The application empowers stakeholders to be transparent about their AI systems and address ethical concerns. This supports the broader goal of trustworthy and human-centric AI.

From a societal perspective, the adoption of AIBOMs contributes to the digital transformation by promoting responsible AI and innovation. Public trust in AI systems is essential for the widespread adoption of those systems.

This work also aligns with several Sustainable Development Goals (SDGs), particularly SDG 9 (Industry, Innovation and Infrastructure) and SDG 16 (Peace, Justice and Strong Institutions). Systems like AIBoMGen contribute to building a resilient digital infrastructure and boost ethical innovation. Furthermore, the focus on lineage tracking and data quality supports SDG 10 (Reduced Inequalities) by ensuring biased or poisoned datasets do not perpetuate through new AI models.

In conclusion, the development of AIBOMs and the AIBoMGen platform is a significant step toward addressing the ethical and societal challenges posed by AI.

References

- [1] Wikipedia, “Large language model,” Apr. 2025, page Version ID: 1288064259. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Large_language_model&oldid=1288064259
- [2] D. Bardenstein, “Whitepaper: Driving AI Transparency: the AI Bill of Materials (Manifest),” 2023. [Online]. Available: manifestcyber.com
- [3] OWASP Foundation, “Authorative Guide to SBOM,” OWASP Foundation, Tech. Rep., 2024. [Online]. Available: https://cyclonedx.org/guides/OWASP_CycloneDX-Authoritative-Guide-to-SBOM-en.pdf
- [4] Linux Foundation, “SPDX,” 2023. [Online]. Available: <https://spdx.dev/>
- [5] —, “Implementing AI Bill of Materials (AI BOM) with SPDX 3.0,” 2024. [Online]. Available: <https://www.linuxfoundation.org/research/ai-bom>
- [6] B. Ghosh, “AI Bill of Materials (AI BOM),” Jul. 2024. [Online]. Available: <https://medium.com/@bijit211987/ai-bill-of-materials-ai-bom-80d48f9d75e0>
- [7] T. Krantz, “What Is Data Poisoning?” Dec. 2024. [Online]. Available: <https://www.ibm.com/think/topics/data-poisoning>
- [8] Dremio, “Adversarial Attacks in AI,” 2025. [Online]. Available: <https://www.dremio.com/wiki/adversarial-attacks-in-ai/>
- [9] S. Creese, “Why we need to reflect on the need for cybersecurity of AI,” Dec. 2023. [Online]. Available: <https://www.weforum.org/stories/2023/12/cybersecurity-ai-ethics-responsible-innovation/>
- [10] D. Bardenstein, “Manifest: AIBOMs, Transparency into AI,” 2023. [Online]. Available: <https://www.manifestcyber.com/aibom>
- [11] TechWorks and NquiringMinds, “Trustworthy AI – Practical Collaborative Engineering,” 2023. [Online]. Available: <https://www.techworks.org.uk/wp-content/uploads/2024/01/Engineering-Trustworthy-AI.pdf>
- [12] TechWorks, “Compliance with the EU AI Act The Techworks Trusted AI Bill Of Materials (TAIBOM) project,” Jun. 2024. [Online]. Available: <https://www.techworks.org.uk/ai-taibom/compliance-with-the-eu-ai-act-the-techworks-trusted-ai-bill-of-materials-taibom-project>

References

- [13] C. Jasserand, “Clearview AI: illegally collecting and selling our faces in total impunity? (Part I),” Apr. 2022. [Online]. Available: <https://www.law.kuleuven.be/citip/blog/clearview-ai-illegally-collecting-and-selling-our-faces-in-total-impunity-part-i/>
- [14] R. Hart, “Clearview AI—Controversial Facial Recognition Firm—Fined \$33 Million For ‘Illegal Database’,” 2024, section: Business. [Online]. Available: <https://www.forbes.com/sites/roberthart/2024/09/03/clearview-ai-controversial-facial-recognition-firm-fined-33-million-for-illegal-database/>
- [15] D. Thiel, “Investigation Finds AI Image Generation Models Trained on Child Abuse,” Dec. 2023. [Online]. Available: <https://cyber.fsi.stanford.edu/news/investigation-finds-ai-image-generation-models-trained-child-abuse>
- [16] imec, “‘GPULab and JupyterHub introduction’ slidedeck.” [Online]. Available: https://doc.ilabt.imec.be/ilabt/_downloads/a94a14498e29a054885db6019351534d/GPULab%20and%20JupyterHub%20introduction.pdf
- [17] EU, “AI Act | Shaping Europe’s digital future,” Sep. 2024. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai>
- [18] —, “EU Artificial Intelligence Act | Up-to-date developments and analyses of the EU AI Act,” 2024. [Online]. Available: <https://artificialintelligenceact.eu/>
- [19] Securin, “How Evolving AI Regulations Impact Cybersecurity - Securin,” Jul. 2024. [Online]. Available: <https://www.securin.io/articles/how-evolving-ai-regulations-impact-cybersecurity/>
- [20] E. Ozoani, M. GerChick, and M. Mitchell, “Model Cards,” 2022. [Online]. Available: <https://huggingface.co/blog/model-cards>
- [21] Securin, “Bringing in the BoM Squad: Defining and Generating AI Bill of Materials,” Jun. 2024. [Online]. Available: <https://www.securin.io/articles/bringing-in-the-bom-squad-defining-and-generating-ai-bill-of-materials/>
- [22] D. Bardenstein, “Webinar: AIBoM (Manifest),” 2023.
- [23] Linux Foundation, “AI – SPDX,” 2023. [Online]. Available: <https://spdx.dev/learn/areas-of-interest/ai/>
- [24] OWASP Foundation, “CycloneDX - Software Bill of Materials (SBOM),” Jul. 2024. [Online]. Available: <https://cyclonedx.org/>

References

- [25] Hugging Face, “Hugging Face,” May 2025. [Online]. Available: <https://huggingface.co/>
- [26] D. Bardenstein, “manifest-cyber/aibom,” Nov. 2024, original-date: 2023-11-02T15:10:30Z. [Online]. Available: <https://github.com/manifest-cyber/aibom>
- [27] ECMA international, “ECMA-424 CycloneDX - Bill of Materials (BOM) specification,” 2021. [Online]. Available: https://ecma-international.org/wp-content/uploads/ECMA-424_1st_edition_june_2024.pdf
- [28] Linux Foundation, “SPDX Github Specification 3.0.1,” 2024. [Online]. Available: <https://spdx.github.io/spdx-spec/v3.0.1/>
- [29] I. O. for Standardization and International Electrotechnical Commission, “ISO/IEC 5962:2021 Software package data exchange (SPDX),” 2021. [Online]. Available: <https://www.iso.org/standard/81870.html>
- [30] EU Made Simple, “The EU’s AI Act Explained,” Sep. 2024. [Online]. Available: https://www.youtube.com/watch?v=s_rxOnCt3HQ
- [31] EU, “The AI Act Explorer | EU Artificial Intelligence Act,” 2024. [Online]. Available: <https://artificialintelligenceact.eu/ai-act-explorer/>
- [32] —, “High-level summary of the AI Act | EU Artificial Intelligence Act,” Feb. 2024. [Online]. Available: <https://artificialintelligenceact.eu/high-level-summary/>
- [33] —, “Governance and enforcement of the AI Act | Shaping Europe’s digital future,” 2025. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/policies/ai-act-governance-and-enforcement>
- [34] M. Tambiama, “General-purpose artificial intelligence,” 2023. [Online]. Available: [https://www.europarl.europa.eu/RegData/etudes/ATAG/2023/745708/EPRS_ATA\(2023\)745708_EN.pdf](https://www.europarl.europa.eu/RegData/etudes/ATAG/2023/745708/EPRS_ATA(2023)745708_EN.pdf)
- [35] S. Gracias, “Comparing the EU AI Act to Proposed AI-Related Legislation in the US |,” 2024. [Online]. Available: <https://businesslawreview.uchicago.edu/print-archive/comparing-eu-ai-act-proposed-ai-related-legislation-us>
- [36] EU, “EU AI Act Compliance Checker | EU Artificial Intelligence Act,” 2024. [Online]. Available: <https://artificialintelligenceact.eu/assessment/eu-ai-act-compliance-checker/>

References

- [37] TechWorks, “TAIBOM,” 2025. [Online]. Available: <https://taibom.nqminds.com/docs/>
- [38] —, “SDK for Creating & Verifying TAIBOMs,” 2025. [Online]. Available: <https://taibom.nqminds.com/sdk/sdk/>
- [39] Mithril Security, “AICert,” 2025. [Online]. Available: <https://aicert.mithrilsecurity.io/en/latest/>
- [40] —, “mithril-security/aicert,” May 2025, original-date: 2023-07-04T07:24:29Z. [Online]. Available: <https://github.com/mithril-security/aicert>
- [41] vinaypamnani-msft, “Trusted Platform Module Technology Overview,” Jul. 2024. [Online]. Available: <https://learn.microsoft.com/en-us/windows/security/hardware-security/tpm/trusted-platform-module-overview>
- [42] Mithril Security, “AICert fine-tuning demo,” Jun. 2024. [Online]. Available: <https://www.youtube.com/watch?v=-GQeXQyunYc>
- [43] W. Vandendriessche, “AIBoMGen,” May 2025. [Online]. Available: <https://zenodo.org/records/15536533>
- [44] imec, “GPULab — imec iLab.t documentation.” [Online]. Available: <https://doc.ilabt.imec.be/ilabt/gpulab/>
- [45] Docker, “Docker,” Apr. 2025. [Online]. Available: <https://www.docker.com/>
- [46] —, “Docker Compose,” 2025. [Online]. Available: <https://docs.docker.com/compose/>
- [47] —, “Services top-level elements,” 2025. [Online]. Available: <https://docs.docker.com/reference/compose-file/services/>
- [48] Python, “Python Release Python 3.11.0,” 2022. [Online]. Available: <https://www.python.org/downloads/release/python-3110/>
- [49] A. Solem and contributors, “Celery 5.5.2 documentation,” 2023. [Online]. Available: <https://docs.celeryq.dev/en/stable/>
- [50] Ask Solem and contributors, “Celery 5.5.2 documentation: Periodic Tasks,” 2023. [Online]. Available: <https://docs.celeryq.dev/en/stable/userguide/periodic-tasks.html>

References

- [51] M. Movsisyan and contributors, “Flower 2.0.0 documentation,” 2023. [Online]. Available: <https://flower.readthedocs.io/en/latest/>
- [52] RabbitMQ, “RabbitMQ: One broker to queue them all,” 2025. [Online]. Available: <https://www.rabbitmq.com/>
- [53] Oracle, “MySQL,” 2025. [Online]. Available: <https://www.mysql.com/>
- [54] FastAPI, “FastAPI,” 2025. [Online]. Available: <https://fastapi.tiangolo.com/>
- [55] J. K. Svensson, “FastAPI-Azure-Auth,” 2025. [Online]. Available: <https://intility.github.io/fastapi-azure-auth/>
- [56] MinIO, “MinIO - S3 Compatible Storage for AI,” 2025. [Online]. Available: <https://min.io>
- [57] TensorFlow, “TensorFlow v2.16.1: tf.data.TFRecordDataset,” 2024. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/data/TFRecordDataset
- [58] Docker and TensorFlow, “Image Layer Details - tensorflow/tensorflow:2.16.1-gpu,” 2024. [Online]. Available: <https://hub.docker.com/layers/tensorflow/tensorflow/2.16.1-gpu/images/sha256-4ab9ffddd6ffacc9251ac6439f431eb38d66200d3f52397b5d77f9bc3298c4e9>
- [59] Kubeflow, “Kubeflow,” 2025. [Online]. Available: <https://www.kubeflow.org/>
- [60] MLflow, “MLflow,” 2025. [Online]. Available: <https://mlflow.org/>
- [61] Amazon Web Services, “Firecracker,” 2025. [Online]. Available: <https://firecracker-microvm.github.io/>
- [62] OpenInfra Foundation, “Kata Containers - Open Source Container Runtime Software,” 2025. [Online]. Available: <https://katacontainers.io/>
- [63] Ray, “Ray by Anyscale,” 2025. [Online]. Available: <https://ray.io>
- [64] OWASP Foundation, “CycloneDX Python Library 10.0.1 documentation,” 2025. [Online]. Available: <https://cyclonedx-python-library.readthedocs.io/en/latest/>
- [65] —, “CycloneDX Python Library 10.0.1 documentation: v1.6 schema Support,” 2025. [Online]. Available: <https://cyclonedx-python-library.readthedocs.io/en/latest/schema-support.html>
- [66] Aqua Security, “Trivy,” 2025. [Online]. Available: <https://trivy.dev/v0.62/>

References

- [67] in-toto, “in-toto,” 2025. [Online]. Available: <https://in-toto.io/>
- [68] A. Gaynor and P. Kehrer, “cryptography/hazmat,” 2025. [Online]. Available: <https://github.com/pyca/cryptography/blob/main/src/cryptography/hazmat/primitives/ciphers/algorithms.py>
- [69] L. Pühringer, “secure-systems-lab/securesystemslib,” May 2025, original-date: 2016-12-06T21:21:30Z. [Online]. Available: <https://github.com/secure-systems-lab/securesystemslib>
- [70] Docker, “tmpfs mounts,” 2025. [Online]. Available: <https://docs.docker.com/engine/storage/tmpfs/>
- [71] L. Savaete, “slowapi,” May 2025, original-date: 2020-02-21T12:56:20Z. [Online]. Available: <https://github.com/laurentS/slowapi>
- [72] in-toto, “in-toto 3.0.0 documentation,” 2025. [Online]. Available: <https://in-toto.readthedocs.io/en/latest/>
- [73] TensorFlow, “tf.keras.models.load_model,” Jun. 2024. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/models/load_model
- [74] T. Righi, “TensorFlow Remote Code Execution with Malicious Model,” Oct. 2022. [Online]. Available: <https://splint.gitbook.io/cyberblog/security-research/tensorflow-remote-code-execution-with-malicious-model>
- [75] Vercel, “Next.js by Vercel,” 2025. [Online]. Available: <https://nextjs.org/>
- [76] Tailwind Labs, “Tailwind CSS,” 2025. [Online]. Available: <https://tailwindcss.com/>
- [77] shadcn, “shadcn/ui,” 2025. [Online]. Available: <https://ui.shadcn.com/>
- [78] Microsoft, “AzureAD/microsoft-authentication-library-for-js,” May 2025, original-date: 2017-02-24T21:49:17Z. [Online]. Available: <https://github.com/AzureAD/microsoft-authentication-library-for-js>
- [79] W. Vandendriessche, “AIBoMGen-experiments Scripts & Results,” May 2025. [Online]. Available: <https://zenodo.org/records/15505280>
- [80] UC Irvine, “winequality-red.csv,” Oct. 2009. [Online]. Available: <https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv>
- [81] Keras Team, “Keras documentation: MNIST digits classification dataset,” 1994. [Online]. Available: <https://keras.io/api/datasets/mnist/>

References

- [82] —, “Keras documentation: CIFAR10 small images classification dataset,” 2009. [Online]. Available: <https://keras.io/api/datasets/cifar10/>
- [83] Sigstore Community, “Sigstore,” 2025. [Online]. Available: <https://www.sigstore.dev/undefined/>
- [84] C. Kennedy, “Comparing in-toto and Sigstore: Two Approaches to Software Supply Chain Security,” 2025. [Online]. Available: <https://testifysec.com/blog/sigstore-vs-in-toto/>
- [85] World Wide Web Consortium, “W3C,” 2025. [Online]. Available: <https://www.w3.org/>
- [86] Docker, “Running containers,” 2025. [Online]. Available: <https://docs.docker.com/engine/containers/run/>
- [87] —, “Isolate containers with a user namespace,” 2025. [Online]. Available: <https://docs.docker.com/engine/security/userns-remap/>

Appendices

A SPDX 3.0 Compliance with EU Artificial Intelligence Act

This table is from the SPDX white paper: "Implementing AI Bill of Materials (AI BOM) with SPDX 3.0" [5].

This table details the relationship between SPDX 3.0 fields and critical components of the EU Artificial Intelligence Act (EU AI Act), focusing specifically on the EU database registration mandate outlined in Article 49. This registration must be completed prior to marketing or testing high-risk AI systems in the real world. Effective from 1 August 2024, the EU AI Act is designed to regulate artificial intelligence by implementing precise guidelines to ensure AI systems are developed and used in accordance with existing laws and fundamental human rights. The legislation classifies AI systems by their risk categories, applying stricter requirements to high-risk systems to boost transparency and accountability. Utilizing SPDX 3.0 enables organizations to comprehensively map and document their AI systems, ensuring adherence to the EU AI Act's stipulations and easing the registration process. This integration not only streamlines regulatory compliance but also encourages the creation of safe and ethical AI advancements.

Table A.1: EU AI Act Compliance with SPDX 3.0

Categories	Subcategories	EU AI Act Clause	In AI BOM?	Matching Field in AI BOM	SPDX Profile
System, Provider, and Deployer Identification	System	"A Union-wide unique single identification number of the testing in real world conditions" - Annex IX (1) "URL of the entry of the AI system in the EU database by its provider" - Annex VIII Section C (3) "Any additional unambiguous reference allowing the identification and traceability of the AI system" - Annex VIII Section A (4) - Annex VIII Section B (4)	yes	spdxId, externalIdentifier, packageUrl in Package	Core
	unique ID				
	Provider				
	contact				
Deployer	System name	"AI system trade name" - Annex VIII Section A (4) - Annex VIII Section B (4)	yes	name in Package (AIPackage)	Core
	Provider	"name, address, and contact details of the provider" - Annex VIII Section A (1) - Annex VIII Section B (1) - Annex VIII Section C (1)	yes	createdBy in CreationInfo, externalIdentifier in Person/Organization	Core
	contact				
System Details	Intended purpose	- Annex VIII Section A (5) - Annex VIII Section B (5) - Annex IX (3)	yes	suppliedBy, externalIdentifier in Person/Organization	Core

Continued on next page

Continued from previous page

Categories	Subcategories	EU AI Act Clause	In AI BOM?	Matching Field in AI BOM	SPDX Profile
System Details	Information used by system	"Information used by system (data, inputs) and its operating logic" - Annex VIII Section A (6)	yes	informationAboutApplication, informationAboutTraining, testedOn, Dataset profile	AI, trainedOn, Dataset
	System status	"The status of the AI system (on the market, or in service; no longer placed on the market/in service, recalled)" - Annex VIII Section A (7)	yes	validUntilTime, supportLevel	Core
	System classification	"A short summary of the grounds on which the AI system is considered to be not-highrisk in application of the procedure under Article 6(3)" - Annex VIII Section B (7)	yes	informationAboutApplication, documentation relationship	AI, Core
	Instructions for use	"instructions for use for the employer, and a basic description of the user-interface provided to the employer, where applicable" - Annex IV (1)(h)	yes	informationAboutApplication, documentation relationship	AI, Core

Continued on next page

Appendix A

Continued from previous page

Categories	Subcategories	EU AI Act Clause	In AI BOM?	Matching Field in AI BOM	SPDX Profile
System Details	Impact assessment	"A summary of the findings of the fundamental rights impact assessment conducted in accordance with Article 27" - Annex VIII Section C (4) "A summary of the data protection impact assessment carried out in accordance with Article 35 of Regulation (EU) 2016/679 or Article 27 of Directive (EU) 2016/680 as specified in Article 26(8) of this Regulation, where applicable." - Annex VIII Section C (5)	yes	hasDocumentation, relationship description	Core
	Certification	"A copy of the EU declaration of conformity referred to in Article 47" - Annex IV (8) "The type, number and expiry date of the certificate issued by the notified body and the name or identification number of that notified body, where applicable" - Annex VIII Section A (8)	yes	standardCompliance, validUntilTime, hasDocumentation, relationship description	AI, Core

Verification Details	Testing plan	"A summary of the main characteristics of the plan for testing in real world conditions;" - Annex IX (4)	yes	hasDocumentation, relationship description	Core
----------------------	--------------	--	-----	--	------

Continued on next page

Appendix A

Continued from previous page

Categories	Subcategories	EU AI Act Clause	In AI BOM?	Matching Field in AI BOM	SPDX Profile
Verification Details	Users involved in testing	in- “The name and contact details of the provider or prospective provider and of the deployers involved in the testing in real world conditions” - Annex IX (2)	no	There is no specific relationship type for this. However, a Relationship with “to” as Agent (Person and/or Organization) and “relationshipType” as “other” can be used, together with a standardized description value. An externalIdentifier in Person and Organization will hold the contact details.	Core
Application Details	Market	“Any Member States in which the AI system has been placed on the market, put into service or made available in the Union” - Annex VIII Section A (10)	yes	Organization with externalIdentifier as country code	Core

Appendix A

B Example of CycloneDX-Like AIBOM for RedPajama-INCITE-Instruct-3B

```

{
  "bomFormat": "CycloneDX",
  "specVersion": "1.5",
  "serialNumber": "urn:uuid:3e671687-395b-41f5-a30f-a58921a69b79",
  "metadata": {
    "component": {
      "type": "machine-learning-model",
      "name": "RedPajama-INCITE-Instruct-3B",
      "version": "1.4",
      "description": "A 2.8B parameter pretrained language model."
    },
    "timestamp": "2023-03-28T19:16:23Z",
    "authors": {
      "name": "Firstname Lastname"
    },
    "tools": [
      {
        "name": "manifest-cli",
        "version": "1.0.0"
      }
    ]
  },
  "components": [
    {
      "bom-ref": "ml-id-ref",
      "type": "machine-learning-model",
      "author": "Together Computer",
      "name": "RedPajama-INCITE-Instruct-3B-v1",
      "version": "1.4",
      "description": "A 2.8B parameter pretrained language model.",
      "properties": {
        "category": "text-generation",
        "baseModel": "RedPajama-INCITE-Base-3B-v1",
        "baseModelSource": "https://huggingface.co/togethercomputer/RedPajama-INCITE-Base-3B-v1",
        "intended-use": "Excluded uses are described below.",
        "out-of-scope": [
          ↪ "RedPajama-INCITE-Instruct-3B-v1 is a languagemodel and may not perform well for other use cases..."
          ↪ ,
          ↪ "misuse-or-malicious":
          ↪ "RedPajama-INCITE-Instruct-3B-v1 is designed for language modeling. Misuse of the model..."
        ]
      },
      "licenses": [
        {
          "license": {
            "id": "Apache-2.0"
          }
        }
      ]
    }
  ],

```

Code Fragment B.1: CycloneDX-Like AIBOM for RedPajama-INCITE-Instruct-3B - Part

```

"modelCard": {
  "modelParameters": {
    "tasks": [
      {
        "task": "text inference"
      }
    ],
    "architectureFamily":
      ↪ "GPT-NeoX (https://together.ai/blog/redpajama-models-v1)",
    "modelArchitecture": "GPTNeoXForCausalLM",
    "datasets": [
      {
        "type": "dataset",
        "name": "RedPajama-Data-1T",
        "contents": {
          "url": ↪
            ↪ "https://huggingface.co/datasets/togethercomputer/RedPajama-Data-1T"
        },
        "classification": "public"
      }
    ],
    "inputs": [{ "format": "text" }],
    "outputs": [{ "format": "text" }]
  },
  "considerations": {
    "ethicalConsiderations": [
      {
        "name": "It is the responsibility of the end user to ensure that
          the model is used in a responsible and ethical manner."
      }
    ]
  }
},

```

Code Fragment B.2: CycloneDX-Like AIBOM for RedPajama-INCITE-Instruct-3B - Part 2

```

    "externalReferences": [
      {
        "comment": "Model source",
        "type": "vcs",
        "url": "https://huggingface.co/togethercomputer/RedPajama-INCITE-Base-3B-v1"
      },
      {
        "comment": "Model website",
        "type": "website",
        "url": "https://together.ai/blog/redpajama-models-v1"
      },
      {
        "name": "Together Discord",
        "type": "chat",
        "url": "https://discord.gg/6ZVDU8tTD4"
      }
    ]
  },
  {
    "type": "library",
    "bom-ref": "pytorch-id-ref",
    "name": "PyTorch",
    "description":
      ↪ "PyTorch is a Python package that provides two high-level features:
      Tensor computation (like NumPy) with strong GPU acceleration;
      Deep neural networks built on a tape-based autograd system"
  },
  {
    "type": "library",
    "bom-ref": "transformers-id-ref",
    "name": "transformers",
    "description":
      ↪ "Transformers provides thousands of pretrained models to perform tasks
      on different modalities such as text, vision, and audio."
  }
],
  "dependencies": [
    {
      "ref": "ml-id-ref",
      "dependsOn": ["pytorch-id-ref", "transformers-id-ref"]
    }
  ]
}

```

Code Fragment B.3: CycloneDX-Like AIBOM for RedPajama-INCITE-Instruct-3B - Part

C Detailed Threat Analysis

This appendix contains an explanation of most tampering possibilities and how they are (or are not) countered in the application.

Threats

1. Tampering inside the uploaded `model.keras` or `model.h5` file

Threat: A malicious actor could embed a Lambda layer inside a TensorFlow/Keras model file (`.keras` or `.h5`). The Lambda layer allows arbitrary Python code to be serialized and executed when the model is loaded. This could lead to Remote Code Execution (RCE), enabling attackers to execute harmful commands on the system, such as creating reverse shells or deleting files.

In the context of this platform, such an attack could be used to tamper with or disable the measurement tools responsible for generating a trustworthy AIBOM. By compromising these tools, an attacker could falsify the recorded artifacts, manipulate the training process metadata, or bypass integrity checks, undermining the trustworthiness of the entire system.

Countermeasure in the application:

- **Safe Mode Loading:** The application uses TensorFlow's `safe_mode=True` option when loading models. According to the TensorFlow documentation [73], this option ensures that unsafe Lambda deserialization is disallowed, preventing arbitrary code execution during model loading [74]. This argument is only applicable to the Keras v3 model format, so only these types of model files are allowed. The application enforces this setting by default:

```
model = tf.keras.models.load_model(model_path, safe_mode=True)
```

- **Restricted Model Loading Environment:** Models are loaded in restricted Docker containers configured with only the necessary permissions. This minimizes the impact of potential malicious code by isolating the model loading process. Additionally, this setup can be extended with the use of sandboxes, virtual machines, or lightweight virtualization technologies such as AWS Firecracker to further enhance security and isolation.

Appendix C

- **Model Integrity Verification:** When the AIBOM is generated, all materials, including the model, are hashed and included in the BOM. This ensures that any injected malicious code in the model file is also hashed and recorded.

Remaining risks:

- If a malicious actor finds a vulnerability in TensorFlow's `safe_mode` implementation, they could potentially bypass the safeguard.
- The system cannot detect malicious intent embedded in the model's architecture or weights (e.g., backdoors or adversarial triggers). External auditing of the model is required.

2. Tampering with the dataset (.zip or .csv) or definition file

Threat: A malicious actor could tamper with the dataset files (e.g., .zip or .csv) or the dataset definition file. This could involve injecting malicious data, altering the dataset structure, or modifying the dataset definition to cause unexpected behavior during training. Such tampering could lead to corrupted models, biased results, or even system crashes.

Countermeasure in the application:

- **Dataset Validation:** The application validates the structure and content of the dataset files before use. For example:
 - .zip files are carefully validated and extracted securely to ensure they contain the expected files and structure. This includes checks for malicious content such as zip bombs to prevent resource exhaustion attacks.
 - .csv files are checked for consistency with the dataset definition (e.g., column names, data types).
- **Dataset Definition Validation:** The dataset definition file is parsed and validated to ensure it adheres to the expected schema. Any inconsistencies or unexpected modifications are flagged, and the process is halted.
- **Controlled Environment:** Dataset processing is performed in an isolated Docker container, ensuring that any potential malicious data cannot affect the broader system. This isolation limits the scope of any potential attack and protects the integrity of the platform.

Remaining risks:

- If the input files themselves are malicious but valid (e.g., a biased dataset), your system cannot detect this. This requires external auditing or dataset validation mechanisms.

3. Tampering with the Training Process

Threat: A malicious actor could interfere with the training process by modifying training parameters, injecting malicious code into the training logic, or altering the environment in which the training occurs. This could lead to corrupted models, biased results, or compromised outputs.

Countermeasure in the application:

- **Container Isolation:** Training jobs are executed in isolated Docker containers, preventing developers from directly accessing the system or modifying the training process.
- **In-Toto Verification:** The `run_training` step is attested using in-toto, and the `.link` file includes hashes of all input and output artifacts. Users can verify the `.link` file against the signed layout to ensure the training process followed the expected steps.
- **CycloneDX BOM:** The CycloneDX BOM includes metadata about the training environment (e.g., Python version, TensorFlow version), ensuring the environment is consistent and trusted.

Remaining risks:

- If the developer gains access to the worker container (e.g., through a vulnerability in the container runtime), they could tamper with the training process. This risk is partially mitigated by:
 - **Read-Only Containers with Temporary Storage:** The `docker-compose.yml` file specifies `read_only: true` for workers to limit write access within the container. Additionally, a `tmpfs` mount is used to allow temporary file storage in memory, ensuring that necessary temporary files can still be created without compromising the container's read-only nature. For more details, refer to the Docker Compose documentation on services [46] and the Docker documentation on `tmpfs` storage [70].

Appendix C

- **Capability Restrictions:** Dropping all capabilities (`cap_drop: ALL`) and only allowing `NET_BIND_SERVICE` reduces the attack surface. See the Docker documentation on capabilities [86] for more information.
- **Non-Root User:** Worker containers are configured to run as a non-root user by specifying the `user` directive in the `docker-compose.yml` file. This minimizes the impact of potential exploits by restricting access to privileged operations. For more details, refer to the Docker documentation on user namespaces [87].

4. Tampering with the Generated Artifacts

Threat: A malicious actor could tamper with the generated artifacts (e.g., trained models, logs, metrics, AIBOM files) after they are created but before they are shared with users. This could lead to compromised models, falsified metrics, or altered AIBOMs.

Countermeasure in the application:

- **Hash-Based Integrity Verification:** All generated artifacts are hashed, and these hashes are included in the AIBOM and in-toto `.link` files. Users can verify the integrity of the artifacts by comparing their hashes with the recorded values.
- **Secure Storage:** Artifacts are stored in a secure MinIO bucket with restricted access to prevent unauthorized modifications.
- **Cryptographic Signing:** The AIBOM and in-toto `.link` files are cryptographically signed to ensure their authenticity and integrity.

Remaining risks:

- If the storage system (e.g., MinIO) is compromised, attackers could tamper with the artifacts despite secure storage. This is partially mitigated by:
 - The API provides an endpoint to check the hashes of the MinIO artifacts with the recorded hashes. If they fail that means someone tampered in the MinIO storage. However, they cannot update the link file to match their tampered artifacts because they do not have access to the platform private key to correctly resign the link file.
- The system does not currently support real-time monitoring of artifact access or modifications.

5. Tampering with the AIBOM

Threat: A malicious actor could tamper with the AIBOM to falsify information about the training process, inputs, or outputs. This could mislead users into trusting a compromised model.

Countermeasure in the application:

- **Cryptographic Signing:** The AIBOM is signed using the system's private key. Users can verify the signature using the public key to ensure the AIBOM has not been tampered with.
- **Hash Verification:** The AIBOM includes hashes of all input and output artifacts, allowing users to verify their integrity.

Remaining risks:

- If the private key used for signing the AIBOM is compromised, attackers could generate falsified AIBOMs that appear legitimate.
- The system does not currently enforce key rotation or hardware-based key storage, which could reduce the risk of key compromise.

6. Unauthorized Access to Artifacts or System Components

Threat: A malicious actor could gain unauthorized access to artifacts (e.g., datasets, models, logs) or system components (e.g., API, workers, storage). This could lead to data breaches, tampering, or system compromise.

Countermeasure in the application:

- **Authentication and Authorization:** The system uses Azure OAuth for secure API calls, ensuring that only authenticated and authorized users can access the system.
- **Access Control:** MinIO buckets and other system components can be configured with strict access controls when deployed to prevent unauthorized access.

Remaining risks:

- If an attacker exploits a zero-day vulnerability in the authentication or authorization mechanisms, they could gain unauthorized access.

Appendix C

- The system does not yet have alerting for suspicious activity implemented.

7. Bypassing AIBOM generation

Threat: A malicious actor could attempt to bypass the AIBOM generation process to avoid recording the training process or its artifacts. This could lead to untraceable or unverified models.

Countermeasure in the application:

- **Enforced AIBOM Generation:** The system automatically generates an AIBOM for every training job. Users cannot bypass this step.
- **Controlled API:** All interactions with the system are performed through a secure API, ensuring that users cannot interfere with the AIBOM generation process.

Remaining risks:

- If an attacker exploits a vulnerability in the API or worker logic, they could potentially bypass the AIBOM generation process.

8. Exploiting Vulnerabilities in the Platform

Threat: A malicious actor could exploit vulnerabilities in the platform (e.g., API, worker containers, storage) to gain unauthorized access, execute malicious code, or compromise the system.

Countermeasure in the application:

- **Regular Vulnerability Scanning:** The system scans container images and dependencies for vulnerabilities using tools like Trivy.
- **Read-Only Containers:** Worker containers are configured as read-only to limit the impact of potential exploits.
- **Capability Restrictions:** Containers drop all capabilities (`cap_drop: ALL`) and only allow minimal capabilities (e.g., `NET_BIND_SERVICE`) to reduce the attack surface.

Remaining risks:

- New vulnerabilities in dependencies or container images could be exploited before they are patched.

9. Providing a Fake `.link` and/or AIBOM file

Threat: A malicious actor could provide a fake `.link` file or AIBOM to mislead users into trusting a compromised model or falsified training process.

Countermeasure in the application:

- **Cryptographic Verification:** Users can verify the authenticity of `.link` files and AIBOMs using the system's public key. Any tampering or forgery will result in a failed verification.
- **Hash Verification:** The `.link` file and AIBOM include hashes of all input and output artifacts, ensuring that users can verify their integrity.

Remaining risks:

- If the public key used for verification is replaced or tampered with, users could be misled into trusting fake files.
- The system does not currently provide a mechanism for users to verify the authenticity of the public key itself (e.g., through a trusted certificate authority).

Threat Testing Coverage

The following table summarizes how each threat was addressed during evaluation, indicating whether it was explicitly tested and how:

Appendix C

Threat	How Tested	Tested?
Tampering inside uploaded model file (e.g., unsafe Lambda layers)	Attempted to upload models with Lambda layers and observed system rejection due to safe loading.	Yes (manual)
Tampering with dataset or definition file	Uploaded malformed and wrong datasets/definitions to verify rejection.	Yes (manual)
Tampering with the training process	Training jobs run in isolated containers; container permissions and isolation were inspected, but no advanced container escape was attempted.	Partially (config review)
Tampering with generated artifacts	Modified artifacts post-generation and checked if hash/signature verification detected changes.	Yes (manual)
Tampering with the AIBOM	Modified AIBOM files and verified that signature checks failed.	Yes (manual)
Unauthorized access to artifacts or system components	Authentication and access controls were reviewed; no penetration testing was performed.	Partially (manual)
Bypassing AIBOM generation	System enforced AIBOM creation for all jobs, due to API.	Partially (no issues found)
Exploiting platform vulnerabilities	Container images scanned for vulnerabilities; no advanced exploitation performed.	Partially (automated scan)
Providing fake .link or AIBOM files	Provided fake files and verified that signature/hash checks failed.	Yes (manual)

Table C.1: Threats and their testing coverage in the evaluation

Note: Advanced attacks (e.g., exploits, key compromise, adversarial data, or privilege escalation) were not tested due to resource and scope constraints.

Possible Improvements on Trustability

Recommendations for further improving trustability:

- **Use of Hardware Security Modules (HSM):** Store private keys in HSMs to enhance security and reduce the risk of key compromise. HSMs are dedicated hardware devices designed specifically for secure key storage and cryptographic operations, typically managed in data centers or cloud environments.
- **Use of Trusted Platform Module (TPM):** Leverage TPMs to provide hardware-based security at the device level. TPMs are integrated chips on the system motherboard that can securely generate, store, and use cryptographic keys, supporting device identity, secure boot, and attestation. Unlike HSMs, TPMs are intended for individual devices rather than centralized key management. They can be utilized like in the AICert project 2.4.
- **Key Rotation and Management:** Implement regular key rotation policies and integrate with secure key management systems to minimize the impact of key compromise.
- **Code Integrity Verification:** Include hashes of the worker container code, API code, and other critical components in the AIBOM to detect unauthorized modifications.
- **Expand the Scanner Container:** Extend the scanner container's functionality to scan not only worker containers but also the API, storage systems, and other platform components for vulnerabilities.
- **Real-Time Monitoring and Alerting:** Implement real-time monitoring and alerting for suspicious activity, such as unauthorized access to containers, storage systems, or API endpoints.
- **Public Key Authenticity Verification:** Provide a mechanism for users to verify the authenticity of the public key (e.g., through a trusted certificate authority or blockchain-based verification).
- **Runtime Monitoring of Containers:** Introduce runtime monitoring for containers to detect and respond to active exploitation attempts in real time.
- **Enhanced Dataset Validation:** Develop advanced mechanisms for detecting malicious but valid datasets (e.g., biased or poisoned data) using external auditing

Appendix C

or AI-based validation tools.

- **User Education and Documentation:** Provide detailed documentation and training for users on how to verify AIBOMs, `.link` files, and artifacts to ensure proper usage of the system's trustability features.

D in-toto Layout and Link File Example

Layout Example: The layout file below specifies that the `run_training` step must be executed, with specific expected input materials (model, dataset, and dataset definition) and output products (trained model and metrics). The layout is signed and includes the public keys authorized to perform the step. These keys can be extended if different keys for different workers are needed or if key rotations is needed.

```
{
  "signatures": [
    {
      "keyid": "8a98b0773b596abdb6a3357c...e90c9746fab2633162qrgf0",
      "sig": "1c95ec74de892f16a40c44b2c...3ac34ff1e0e468330ja0a04"
    }
  ],
  "signed": {
    "_type": "layout",
    "expires": "2025-07-28T11:39:28Z",
    "inspect": [],
    "keys": {
      "8a98b0773b596abdb6a3c5dc1afc620ee37a2cf5357ce90c9746fab2633162f0": {
        "keyid": "8a98b0773b596abdb6a3c5dc1afc620ee37a2cf5357ce90c9746fab2633162f0",
        "keytype": "ed25519",
        "keyval": {
          "public": "e9c1aa9a86573dcf9ba921e9aeaa08df4bb72db3d055c1733f70984723c67f64"
        },
        "scheme": "ed25519"
      }
    },
    "readme": "This layout defines the steps for the run_training task.",
    "steps": [
      {
        "_type": "step",
        "expected_command": [
          "python",
          "tasks.py",
          "run_training"
        ],

```

Code Fragment D.1: in-toto Layout Example - Part 1

```

    "expected_materials": [
      [
        "MATCH",
        "model",
        "WITH",
        "PRODUCTS",
        "FROM",
        "api"
      ],
      [
        "MATCH",
        "dataset",
        "WITH",
        "PRODUCTS",
        "FROM",
        "api"
      ],
      [
        "MATCH",
        "dataset_definition",
        "WITH",
        "PRODUCTS",
        "FROM",
        "api"
      ]
    ],
    "expected_products": [
      [
        "CREATE",
        "trained_model.keras"
      ],
      [
        "CREATE",
        "metrics.json"
      ]
    ],
    "name": "run_training",
    "pubkeys": [
      "8a98b0773b596abdb6a3c5dc...357ce90c9746fab2633162f0"
    ],
    "threshold": 1
  }
}

```

Code Fragment D.2: in-toto Layout Example - Part 2

Link File Example: The link file records the actual execution of the `run_training` step, including the command run, the hashes of all input materials, and the hashes of the resulting products. This file is also signed to ensure its integrity. As explained above, the `environment` field is empty because environment details are captured in the

AIBOM.

```

{
  "signatures": [
    {
      "keyid": "8a98b0773b596abc...46fab2633162f0",
      "sig": "ee1a299b84b203ac...ff04b6396a60b"
    }
  ],
  "signed": {
    "_type": "link",
    "byproducts": {
      "stdout": "Task completed successfully."
    },
    "command": [
      "python",
      "tasks.py",
      "run_training"
    ],
    "environment": {},
    "materials": {
      "11552122-b274-4940-b767-f6a18a933c8b/dataset/image_dataset.zip": {
        "sha256": "90971f884523eaf...f731f56eb0458652467"
      },
      "11552122-b274-4940-b767-f6a18a933c8b/definition/image_definition.yaml": {
        "sha256": "eb434d36856dd04...f62f1c720390c7a54d2"
      },
      "11552122-b274-4940-b767-f6a18a933c8b/model/image_model.keras": {
        "sha256": "4b819acb156ed71...78a396bf0d7f8563b34f"
      }
    },
    "name": "run_training",
    "products": {
      "11552122-b274-4940-b767-f6a18a933c8b/output/metrics.json": {
        "sha256": "7143bb3c289f9078...a1bebdd1bca1e1be956"
      },
      "11552122-b274-4940-b767-f6a18a933c8b/output/trained_model.keras": {
        "sha256": "722934b6186d745e...cf5758e751ec4133b97"
      }
    }
  }
}

```

Code Fragment D.3: in-toto Link File Example

E AIBoMGen PoC Proposal and Resulting AIBOM Example

This appendix presents the proposal and resulting CycloneDX AIBOM generated by the AIBoMGen PoC. The AIBOM includes detailed provenance for the training environment, dataset, and trained model, as well as metadata and signature workarounds. This demonstrates how a trusted, machine-readable BOM can be produced for a typical ML training workflow.

Overview

The AIBoMGen PoC aims to automate the generation of a CycloneDX-compliant AIBOM for AI/ML workflows. It captures:

- The full training environment (hardware, software, container, and job metadata)
- Dataset and definition, with hashes for integrity
- Trained model, architecture, metrics, and hashes
- Tooling and provenance, including signature and attestation references (in-toto)

Workarounds are used for modelCard and signature fields to fit within CycloneDX 1.6.

Resulting CycloneDX AIBOM

```

{
  "components": [
    {
      "bom-ref": "training-environment@1.0",
      "description": "Details of the environment used for training",
      "name": "Training Environment",
      "properties": [
        {"name": "CPU Count", "value": "16"},
        {"name": "Celery Queue", "value": "training_queue"},
        {"name": "Celery Task ID", "value": "4618204e-d29b-4c4f-b14e-1d62f27b3d38"},
        {"name": "Celery Task Name", "value": "tasks.run_training"},
        {"name": "Disk Usage (MB)", "value": "1031018"},
        {"name": "Docker Container ID", "value": "cc8c5b582d02"},
        {"name": "Docker Image ID", "value": "sha256:f2197ebee77d20f3f3de92aff68dd1fb8d2a423d689b850709b66adbcd5a8b7e"},
        {"name": "Docker Image Name", "value": "aibomgen-platform-worker:latest"},
        {"name": "GPU Memory Total (MB)", "value": "6144"},
        {"name": "GPU Memory Used (MB)", "value": "5728"},
        {"name": "GPU Name", "value": "NVIDIA GeForce RTX 3060 Laptop GPU"},
        {"name": "Job ID", "value": "4618204e-d29b-4c4f-b14e-1d62f27b3d38"},
        {"name": "Memory Total (MB)", "value": "15922"},
        {"name": "OS", "value": "Linux 5.15.167.4-microsoft-standard-WSL2"},
        {"name": "Python Version", "value": "3.11.0rc1"},
        {"name": "Request Time", "value": "2025-05-20 15:32:40"},
        {"name": "Start AIBOM Time", "value": "2025-05-20 15:42:19"},
        {"name": "Start Training Time", "value": "2025-05-20 15:32:54"},
        {"name": "TensorFlow Version", "value": "2.16.1"},
        {"name": "Training Time (seconds)", "value": "564.33296251297"},
        {"name": "Unique Directory", "value": "492c5d53-7d5d-49f6-9288-e88a779f2ce7"},
        {"name": "Vulnerability Scan Error", "value": "No vulnerability scan files found."}
      ],
      "type": "container"
    },
    {
      "bom-ref": "training-dataset@1.0",
      "description": "Dataset and dataset definition used for training",
      "hashes": [
        {"alg": "SHA-256", "content": "b31dd54667af4265de773abd5957f75bc212d06685358a015a4dfc17866cd538"},
        {"alg": "SHA-256", "content": "dc3ed037b98e47915138be6af85feda04cd31836e090401ee0205383e8346087"}
      ],
      "name": "Training Dataset",
      "properties": [
        {"name": "Dataset Definition Hash", "value": "b31dd54667af4265de773abd5957f75bc212d06685358a015a4dfc17866cd538"},
        {"name": "Dataset Hash", "value": "dc3ed037b98e47915138be6af85feda04cd31836e090401ee0205383e8346087"},
        {"name": "Dataset MinIO Path", "value": "492c5d53-7d5d-49f6-9288-e88a779f2ce7/dataset/mnist_dataset.zip"},
        {"name": "Input Shape", "value": "[28, 28, 3]"},
        {"name": "Output Shape", "value": "[10]"},
        {"name": "Preprocessing", "value": "{\n  \"normalize\": true\n}"}
      ],
      "type": "data"
    }
  ],
}

```

Code Fragment E.1: CycloneDX AIBOM generated by AIBoMGen PoC - Part 1 (Training Environment and Dataset)

```
{
  "bom-ref": "trained-model@1.0",
  "description": "Simple CNN for MNIST digit classification",
  "hashes": [
    {"alg": "SHA-256", "content": "2341d9699a1663854797041346f6959b0d7cb51d05d95fcdac50b7b860d54f69"},
    {"alg": "SHA-256", "content": "62c22e6dd78a762bb9fbaaa7d6b4ac5ce56c42b72ef1eecd797a994e36322dia"}
  ],
  "name": "mnist_cnn",
  "properties": [
    {"name": "Architecture Summary", "value": "Name\\tType\\tShape\\nconv...dense_1\\tDense\\t(None, 10)"},
    {"name": "Fit Param: batch_size", "value": "32"},
    {"name": "Fit Param: epochs", "value": "30"},
    {"name": "Fit Param: initial_epoch", "value": "Unknown"},
    {"name": "Fit Param: steps_per_epoch", "value": "Unknown"},
    {"name": "Fit Param: validation_freq", "value": "1"},
    {"name": "Fit Param: validation_split", "value": "0.1"},
    {"name": "Fit Param: validation_steps", "value": "Unknown"},
    {"name": "Framework", "value": "TensorFlow 2.16.1"},
    {"name": "License", "value": "Unknown"},
    {"name": "Metric: accuracy", "value": "[0.8730, 0.9461, ...]"},
    {"name": "Metric: loss", "value": "[1.5865, 0.2148, ...]"},
    {"name": "Metric: val_accuracy", "value": "[0.9121, 0.9594, ...]"},
    {"name": "Metric: val_loss", "value": "[0.3355, 0.1721, ...]"},
    {"name": "Metrics Hash", "value": "2341d9699a1663854797041346f6959b0d7cb51d05d95fcdac50b7b860d54f69"},
    {"name": "Optional Param: author", "value": "Wiebe Vandendriessche"},
    {"name": "Optional Param: framework", "value": "TensorFlow 2.16.1"},
    {"name": "Optional Param: license_name", "value": "Unknown"},
    {"name": "Optional Param: model_description", "value": "Simple CNN for MNIST digit classification"},
    {"name": "Optional Param: model_name", "value": "mnist_cnn"},
    {"name": "Optional Param: model_version", "value": "1"},
    {"name": "Trained Model Hash", "value": "62c22e6dd78a762bb9fbaaa7d6b4ac5ce56c42b72ef1eecd797a994e36322dia"}
  ],
  "type": "machine-learning-model",
  "version": "1"
},
{
  "dependencies": [
    {
      "dependsOn": [
        "training-dataset@1.0",
        "training-environment@1.0"
      ],
      "ref": "trained-model@1.0"
    },
    {"ref": "training-dataset@1.0"},
    {"ref": "training-environment@1.0"}
  ],
  "externalReferences": [
    {
      "comment": "in-toto .link file for artifact integrity verification",
      "type": "attestation",
      "url": "492c5d53-7d5d-49f6-9288-e88a779f2ce7/output/run_training.8a98b077.link"
    }
  ]
},
]
```

Code Fragment E.2: CycloneDX AIBOM generated by AIBoMGen PoC - Part 2 (Trained Model and Dependencies)

```

"metadata": {
  "authors": [
    {
      "email": "wiebe.vandendriessche@ugent.be",
      "name": "AIBoMGen by Wiebe Vandendriessche"
    }
  ],
  "manufacturer": {
    "name": "IDLab from Imec, Ghent University",
    "url": ["https://www.idlab.ugent.be"]
  },
  "properties": [
    {
      "name": "BOM Signature",
      "value": "BWclh3mKzg0CQ13bHs/dmc0o9ioYUurxQg4sNaecZMLHPmXMMTo0NIaaFF1o2Bkpb/X6sIGLBXHagrJ1iCBBg=="
    }
  ],
  "supplier": {
    "name": "IDLab from Imec, Ghent University",
    "url": ["https://www.idlab.ugent.be"]
  },
  "tools": {
    "components": [
      {
        "description": "Python library for CycloneDX",
        "externalReferences": [
          {"type": "build-system", "url": "https://github.com/CycloneDX/cyclonedx-python-lib/actions"},
          {"type": "distribution", "url": "https://pypi.org/project/cyclonedx-python-lib/"},
          {"type": "documentation", "url": "https://cyclonedx-python-library.readthedocs.io/"},
          {"type": "issue-tracker", "url": "https://github.com/CycloneDX/cyclonedx-python-lib/issues"},
          {"type": "license", "url": "https://github.com/CycloneDX/cyclonedx-python-lib/blob/main/LICENSE"},
          {"type": "release-notes", "url": "https://github.com/CycloneDX/cyclonedx-python-lib/blob/main/CHANGELOG.md"},
          {"type": "vcs", "url": "https://github.com/CycloneDX/cyclonedx-python-lib"},
          {"type": "website", "url": "https://github.com/CycloneDX/cyclonedx-python-lib/#readme"}
        ],
        "group": "CycloneDX",
        "licenses": [
          {"license": {"acknowledgement": "declared", "id": "Apache-2.0"}}
        ],
        "name": "cyclonedx-python-lib",
        "type": "library",
        "version": "10.0.1"
      },
      {
        "bom-ref": "aibomgen@0.1.0",
        "description": "A platform for AI training and generating trusted AIBOMs",
        "group": "IDLab from Imec and Ghent University",
        "licenses": [
          {"license": {"id": "MIT"}}
        ],
        "name": "AIBoMGen",
        "supplier": {
          "contact": [
            {
              "email": "wiebe.vandendriessche@ugent.be",
              "name": "Wiebe Vandendriessche",
              "phone": "+32 9 264 92 00"
            }
          ],
          "name": "IDLab from Imec, Ghent University",
          "url": ["https://www.idlab.ugent.be"]
        },
        "type": "platform",
        "version": "0.1.0"
      }
    ]
  },
  "serialNumber": "urn:uuid:40769855-7a76-4c44-852f-7135815be328",
  "version": 1,
  "$schema": "http://cyclonedx.org/schema/bom-1.6.schema.json",
  "bomFormat": "CycloneDX",
  "specVersion": "1.6"
}

```

Code Fragment E.3: CycloneDX AIBOM generated by AIBoMGen PoC - Part 3 (Meta-data and Tooling)

Appendix E

Note: The signature and modelCard fields are included as properties due to CycloneDX python library limitations [65].