Designing and implementing a secure machine learning workflow for processing sensitive data

PIERRE LE FEVRE EMIL KARLSSON
pierrelf@kth.se | emilk2@kth.se

December 20, 2023

Abstract

In the shifting sand that is machine learning research, there is an ongoing balancing act between keeping sensitive data and optimizing workflow. As the AI gold rush is in full force, the demand for secure and efficient workflows has been steadily increasing. Today, workflows are often based on either too restrictive practices that worsen the quality of research, e.g. complicated policies designed for large corporations with dedicated teams, or forgo the security aspect altogether, breaking regulations and the trust of data owners. This paper addresses ways to conduct machine learning training on real medical records in a safe environment. The result of the project is a prototype implementation of a secure workflow with a custom code runner which facilitates machine learning projects that use sensitive data while ensuring secure data storage. Field expert interviews provided insights into the workflow's effectiveness, highlighting the trade-offs between security and usability, and the need for future enhancements to address potential data leakage issues.

Contents

1 Introduction **3**
1.1 Theoretical framework . . . 3
1.2 Objectives, research questions and hypotheses . . . 4

2 Methods **4**
2.1 Research strategy . . . 4
2.2 Designing a secure system . . . 5
2.3 Designing the workflow . . . 5
2.4 Designing the code runner . . . 6

3 Results and Analysis **6**
3.1 Architecture . . . 6
3.2 User Workflow . . . 7
3.3 System workflow . . . 7
3.4 Caching results . . . 8
3.5 Database access . . . 8
3.5.1 Injected variables . . . 8
3.5.2 GPG key verification . . . 8
3.6 Prototype code and documentation . . . 10
3.6.1 Server . . . 10
3.6.2 PyPi Package . . . 10
3.6.3 Documentation . . . 10
3.7 Interview results . . . 10

3.7.1 Interview questions . . . 10
3.8 Quantitative results . . . 11
3.8.1 Time taken to complete tasks . . . 11

4 Discussion **11**
4.1 Unverified commits . . . 12
4.2 Dataset Exposure . . . 12
4.3 Ethical Considerations . . . 12
4.4 Evaluating goals . . . 12
4.4.1 Objectives . . . 12
4.4.2 Research questions . . . 12
4.4.3 Hypotheses . . . 13

5 Future work **13**

6 Conclusion **13**

A Hardware used **15**

B secd Configuration **16**

List of Acronyms and Abbreviations

GPG GNU Privacy Guard

IAM Identity and Access Management

K8s Kubernetes

RBAC Role-based Access Control

TRE Trusted Research Environment

ZT Zero Trust

ZTA Zero Trust Architecture

1 Introduction

Running machine learning models on sensitive data is a relatively new multifaceted problem [1]. While the idea has existed for a long time, it is not until recently such practices are being assessed [2], and it is not well received by the general population, 63% of adults being negative towards allowing personal data to be used to improve healthcare [1].

Trusted Research Environment (TRE) is an environment in which it is possible to facilitate sensitive data in a secure manner and allow users to extract meaningful data using analytics tools without compromising the privacy of the individual involved. Such an environment aims to alleviate the burden of a user by taking away the responsibility of managing the data [2]. However, currently Trusted Research Environments do not have clear guidelines about the practicality around model disclosure, meaning such an environment does not allow for the type of outputs a machine learning workflow has, such as trained machine learning models [3].

From our discussions with researchers at KTH [4], many of them are experiencing difficulties regarding security, compliance and ethics when storing datasets containing patient records and training machine learning models using these. While synthetic data generation has been thrown around as a silver bullet, it is clear that the technology is not adequate yet for research applications, with 92% of the tested models favoring real data [5].

1.1 Theoretical framework

In a series of studies conducted by Kavianpour S. et.al [2], the concept of a TRE emerges as a component in the realm of secure medical research. The authors define a TRE as an environment in which it is possible to provide access to patient data for research purposes while safeguarding the privacy of the individual involved. Such an environment imposes meticulous controls on data access, ensuring that researchers can work with the possibly sensitive data without jeopardizing the integrity of the individual. For instance, a researcher may be granted access to arbitrary data about a number of people, such as age, hospital visits, medication, but personal information remains securely in the Trusted Research Environment.

However, in a paper by Mansouri-Benssassi E [3], an issue within Trusted Research Environments is highlighted - the lack of clear guidelines around model disclosure. It was observed that due to the lack of guidelines, it was in practice not allowed to output types from a machine learning world flow, such as a trained model. Such limitation poses a challenge revolving the practicality of these environments, calling for further exploration and research.

Addressing the need for broader datasets for machine learning, Rankin D. et al [5] describes in a study the early stage result of using synthetic data to train machine learning models. Synthetic data serves as a means to anonymize and generate a more extensive training dataset for machine learning applications. While the results indicate that real data outperforms synthetic data in 92% of the cases, the models trained solely on synthetic data show promising results. However, the author emphasizes that the suitability of synthetic data varies across different model types and that it was not clear if the generated data is anonymized enough. Thus, underscoring the necessity of further research and refinement in the field synthetic data generation for machine learning models.

In recent years, information security has shifted from a traditional network-centric model towards the concept of Zero Trust (ZT). In a paper by Rose S. et. al. [6], ZT is presented as a concept where users, assets and resources are prioritized over static network rules. It assumes attackers are present in the system, and thus no implicit trust is given. The Zero Trust Architecture (ZTA) embodies the principles of ZT aiming to prevent data breaches.

1.2 Objectives, research questions and hypotheses

The aim of this research project is to provide a framework for machine learning research which respects the ethics and regulations around private and sensitive data such as medical records, while not impairing the researcher's productivity.

In order to accomplish this aim, a set of objectives has been outlined.

1. Providing storage for sensitive data that comply with relevant legal and ethical standards.
2. Allowing researchers to run their machine learning models and algorithms on sensitive data and extract meaningful insight without compromising the confidentiality of the involved individuals.
3. Creating a user-friendly and practical workflow, which does not burden the researchers.

This project will, in alignment with the objectives, address the following research questions:

1. How can a machine learning model extract meaningful insight without compromising the confidentiality of the individuals involved?
2. What are some current tools or security technologies that can help us achieve an end-to-end secure workflow?
3. What are some trade-offs between data confidentiality and research productivity? If any, how are they mitigated?

Based on the research questions, the following hypotheses has been formulated:

1. By leveraging existing tools and technologies, it is possible to create a secure and ethical privacy-preserving machine learning workflow that allows machine learning models to extract meaningful data without compromising the confidentiality of the individuals involved.
2. A well-designed workflow will boost research productivity by alleviating the burden on researchers of managing the sensitive data.

2 Methods

In this section, we aim to delve into the practical details of how we designed a secure environment capable of managing workflows on sensitive data. The following will be explained:

- **Strategy:** We detail the strategy that will be used to produce a desired prototype.
- **Workflow:** We present what we considered when designing how the system should operate on a higher level
- **Our Add-on:** We present how we designed a custom code runner that has access to sensitive data, and is capable of running user-requested task and later presenting the result to the users.

2.1 Research strategy

Due to the practical nature of the goal of this project, a mixed-methods approach was chosen. For the core prototyping and field expert evaluation, a quantitative method was applied with the following parts:

- Develop the prototype machine learning workflow with a focus on security measures and efficiency enhancements.

- Invite field experts to interact with the prototype.
- Data Collection (Gather quantitative data on user interaction, such as time taken to complete tasks.)
- Compare the performance of the prototype to a baseline or existing workflows.

Other aspects of the research were more qualitative:

- Expert Feedback: Focus on experiences, opinions, and suggestions regarding the prototype.
- Analysis: Analyze qualitative data to identify challenges and user perception related to the prototype's usefulness and security.
- Ethical Considerations: Explore ethical considerations and user concerns related to handling sensitive medical data in the workflow.
- Recommendations: Based on the qualitative insights, generate recommendations for improvements to the prototype and its security features.

2.2 Designing a secure system

When choosing how to design a secure environment hosting sensitive data, we primarily used the knowledge gained from our discussions with researchers at KTH [4]. The discussions gave insight into what kind of security technologies such as a system should employ, such as ZT. A ZTA, in practice, meant that we needed a system that could authenticate users at every interaction made, from the start of the workflow until it is finished. Therefore, to establish a secure system, we designed an architecture inspired by ZTA and the principles of TRE whose fundamental concept is that data is never accessed directly by the researcher but only through a proxy.

2.3 Designing the workflow

To maintain productivity while upholding security principles, we aimed to establish a workflow that minimizes necessary user interactions and maximizing automation. To achieve this, we settled for an isolated environment accessible solely through a VPN that required authentication. Further, since the users were not permitted to access the sensitive data, it was crucial to decouple what the user could access and where the sensitive data is stored. However, some components of the system still required access to databases containing sensitive data. Therefore, the isolated environment needed to be divided up into multiple areas: one for user interaction and another for system processes required in order to execute the user-requested tasks, as depicted in fig. 1.

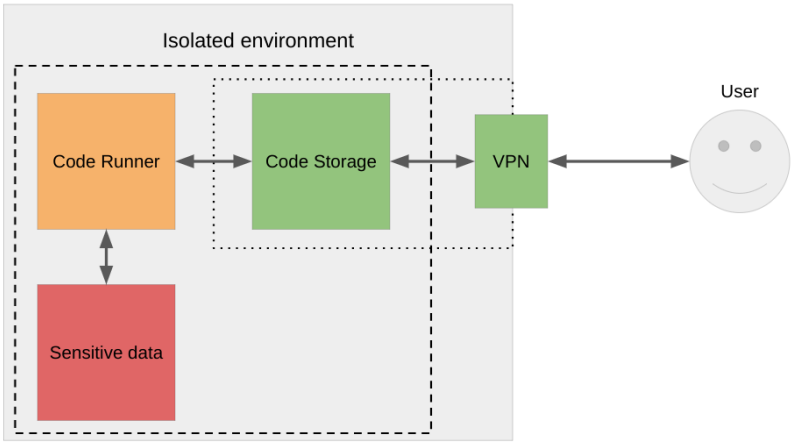

Figure 1: Workflow of the secure cluster

2.4 Designing the code runner

Given the crucial importance for accessing the sensitive data within the proposed solution, we opted to address it ourselves by designing and developing a custom code runner. We outlined the requirements for the program as the following:

- Verifying authenticity of the user-requested task
- Granting access to the dataset for the specific task
- Creating an isolated execution environment for the task
- Managing the result of the task in order to present it to the user

Furthermore, when discussing a potential solution, researchers expressed a desire for a workflow closely resembling that of a Jupyter Notebook [7]. The primary functionality sought after was the ability to retain previously computed code blocks in cached form, thereby avoiding their executing on each subsequent run. For instance, when running machine learning workflows, several data cleaning procedures are typically required, necessitating the loading of entire datasets to perform specific operations. By incorporating caching functionality for results of these operations in our code runner, we ensured that steps that have been computed in an earlier run can be omitted in subsequent runs.

3 Results and Analysis

In this section, we will discuss the results from the project, both the code written, infrastructure built and workflow designed for this project, as well as the interview results with our field experts.

3.1 Architecture

The final architecture ended up including distinct services for crucial feature such as Git, SSO and VPN. The following is a summary.

- We selected a self-hosted instance of GitLab [8] as our code storage primarily for its capacity to integrate ZT principles, including GNU Privacy Guard (GPG) signed commits, but also to act as the storage for the researchers' code. It primarily provides version control, productivity tools such as issues and discussions, and is available open-source. Finally, it support system-wide webhooks that our code runner can subscribe to.
- For the storage of sensitive data, we employed various databases such as MySQL [9] and Neo4j [10]. Access to these databases is enabled through the use of temporary user accounts, according to the principles of ZT. The supported databases are likely to be extended to suit the researcher's requirements.
- To manage Identity and Access Management (IAM), we used Keycloak [11]. This ensures that GitLab and other service can independently enforce authorization within their respective system, in accordance with the ZT principles. Keycloak, in particular, was chosen due to open integration with other systems (such as GitLab), and since we had previous experience working with it.
- To safeguard the system from unauthorized access, we encapsulated the entire infrastructure behind a OpenVPN [12] server. Consequently, authentication serves as the earliest step for any user of the system, according to the Zero Trust principles. OpenVPN was chosen over other alternatives due to it's ease of use for the researchers.

- For the purposes of verifying GPG-signed commits, securely running code, and delivering results (such as trained models), we developed our own open-source service, as described in 2.4, written in Python called *secd*. We implemented it as a program that integrates with a Kubernetes cluster to create isolated execution environments, see 3.3.

See figure 2 for final design of the architecture.

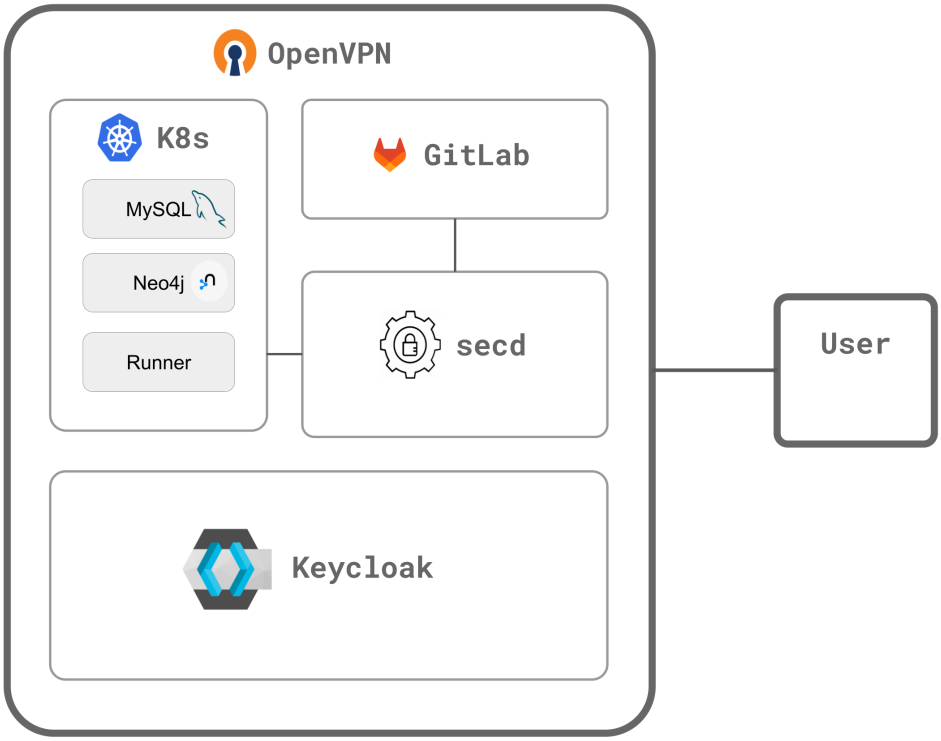

Figure 2: Architecture of secure cluster

3.2 User Workflow

We designed a workflow based on the chosen architecture, prioritizing efficiency wherever possible. The final design for the workflow from a users perspective is specified below:

1. Acquire credentials for Keycloak and OpenVPN
2. Connect to the OpenVPN server
3. Upload personal GPG-key to GitLab
4. Create a GitLab project and push verified commits, add configuration if needed, see appendix B
5. System workflow runs, defined at 3.3
6. Fetch results from the newly created branch in the GitLab repository

3.3 System workflow

1. GitLab system hook triggers the secd service
2. Code is fetched by the *secd* service
   (a) GPG keys are verified for all pushed commits

(b) Configuration file is read
(c) Temporary database user is created
(d) Container is built
(e) Kubernetes (K8s) resources are prepared and environment variables injected
(f) Code runs
(g) Results are collected and K8s resources are expunged
(h) Results are pushed to GitLab repository as a new branch

3. Researcher can view results on GitLab.

3.4 Caching results

We used Python function decorators [13] to designate which functions should be cached. The caching function generates a hash of both the source code and the input parameters, and saves the result as a Pickle file [14]. This allows Python to load it as native data types in subsequent runs. Hashing both the code and input parameters enables the caching function to detect any changes that would warrant recomputing the function's result.

3.5 Database access

We implemented a granular Role-based Access Control (RBAC) system, where roles are defined and managed within the Keycloak instance. Access privileges are granted by the project administrators.

For instance, when granting read access to the MySQL table super_secret_dataset, the corresponding group would be called `mysql-super_secret_dataset-read`. This nomenclature ensures clarity for project administrators in Keycloak, and simplifies the backend server's application of permission for the temporary user.

A temporary database user is issued each time a run is submitted, with the permissions determined by the Keycloak groups to which the researcher belongs. Using a temporary user over a permanent user for each researcher ensures that the database access is limited to the duration of a specific code execution, and cannot be saved for later access.

3.5.1 Injected variables

Temporary database credentials and output location is injected by secd, including the following variables:

- `DB_USER`: The database user.
- `DB_PASS`: The database password.
- `DB_HOST`: The database URL.
- `OUTPUT_PATH`: The local output path within the container; data residing in this directory will be exported back to the user.
- `SECD=PRODUCTION`: Allows the PyPi package, see 3.6.2, to detect when it is running inside our cluster, and use previous caches.

3.5.2 GPG key verification

The signature of every commit in each push is verified by secd by querying GitLab. If and only if each of the multiple commits that make up a given push are GPG-signed will code be executed.

```
for commit in push do
  if not is_verified_in_gitlab(commit) then
    return ERROR
```

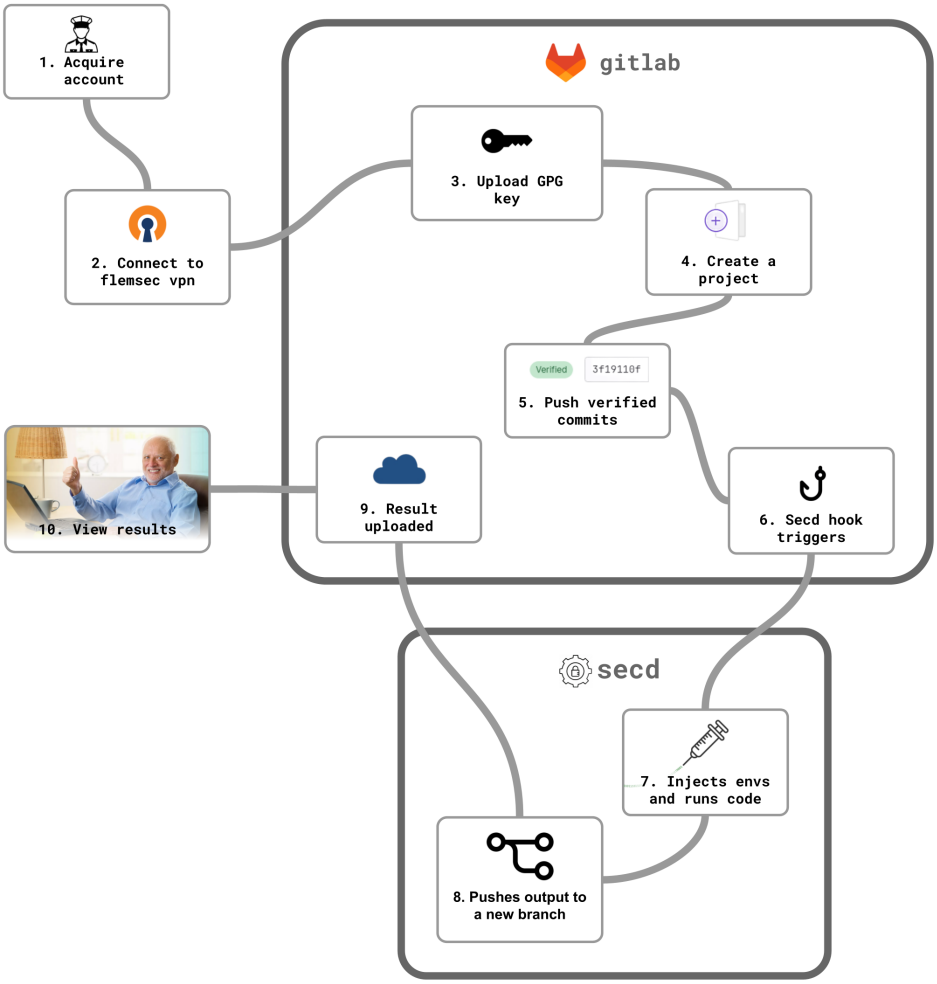

Figure 3: Workflow of secure cluster as a flowchart

3.6 Prototype code and documentation

3.6.1 Server

To run the workflow designed in this project, where code is committed and pushed to a GitLab, and the result is seamlessly uploaded to the same repository as a new branch, quite a lot of work was needed under the hood. The workflow is described in more detail in section 3.3.

The code can be viewed at `https://github.com/kthcloud/secd`

3.6.2 PyPi Package

For the ease of researchers using the workflow, file caching was needed (see Section 3.4). The caching functions have been abstracted to a PyPi package which can be downloaded here: `https://pypi.org/project/secd/`

3.6.3 Documentation

The workflow is documented, and example projects provided on a wiki page at `https://wiki.cloud.cbh.kth.se/index.php/Secd`

3.7 Interview results

To ensure we got the most valuable answers and feedback, interviews were conducted with a small set of field experts at the KTH Centre for Data Driven Health [15].

- Jayanth Ragothama, professor in machine learning and data science
- Reine Bergström, lecturer in machine learning and software engineering

3.7.1 Interview questions

1. How effective do you believe the proposed workflow is in ensuring the security and confidentiality of sensitive medical data during model training? What are some of the main challenges you encountered?

   Jayanth Ragothama I think it secures the data. I am not sure it protects the confidentiality. Depends on your interpretation of confidential, but it does nothing to stop me from reading the data and creating a csv dump, and then pulling it into my computer. Unless you restrict some actions such that it can only be computed on the server, I believe the line is between secure and confidential. Unless of course you can claim that ONLY trusted users can perform these actions.

   Reine Bergström Think it will be very effective security wise, but as always, better security can lead to harder to use.

2. What are the primary challenges and trade-offs between maintaining data confidentiality and maximizing research productivity within the secure environment we've developed?

   Jayanth Ragothama Workflow is cumbersome, error logs are necessary, and without testing on large complex workflows not qualified to answer.

   Reine Bergström The balance between security and ease of use. And making GPG work with GitLab was tricky.

3. Can you offer insights into the usability and user-friendliness of the workflow? How does it impact the productivity of researchers working with sensitive medical data?

Jayanth Ragothama User friendly enough for people who work with these technologies everyday. Setup is cumbersome, and would prefer more stability and opportunities to resolve technical problems on our own.

Reine Bergström For people with less computer knowledge it might be a bit hard at first. There are some steps that people might know what it is and maybe question why.

4. What's your opinion on how the workflow addresses the issue of model disclosure (leaking information through the output model)? Are there any improvements you'd suggest in this regard? *Note: we are not doing anything in that regard - but what would you like to see, or not see?*

   Jayanth Ragothama It does absolutely nothing to solve those issues. And like I mentioned earlier, it also doesn't stop trusted users from abusing workarounds.

   Reine Bergström It's not that user friendly at all. It can impact unless you get much help starting.

5. What is your perspective on the use of synthetic data for training machine learning models in research applications. Do you think it could enhance our workflow?

   Jayanth Ragothama Makes no difference.

   Reine Bergström At the moment no change, but this is of course a weak point we might think about in the future. Not even sure there is a good solution, maybe force output into a strict format to make it harder.

3.8 Quantitative results

3.8.1 Time taken to complete tasks

While the researchers made clear that some slow down due to the tighter security of secd was acceptable [4], it should not be so slow that productivity is impacted negatively. We devised some test projects aimed at testing the latency between pressing run and getting results, comparing results on secd to running locally.

Local environment was run using docker build and docker run, where the first run was not counted as the image is downloaded and cached on first run. This is the same on secd as all images are proxied through the private registry, and as such are cached upon first run.

small-docker is a simple dockerized Python script that generates some paragraphs of text (lorem ipsum) and saves to file.

machine-learning is a dockerized Python script that installs PyTorch and probes available GPU devices.

pytorch-examples is a docker container that runs the example scripts from the PyTorch GitHub repository.

Task	Laptop	secd
small-docker	$<$ 1 second	22 seconds
machine-learning	4 seconds	18 seconds
pytorch-examples	566 seconds	860 seconds

Table 1: Time taken for each task on a Dell XPS 15 9500 laptop versus secd.

4 Discussion

In this section we present some disclaimers, and discuss and evaluate goals using the results of our prototype and the interviews.

4.1 Unverified commits

While secd is responsible for verifying every commits in a given push, some responsibility is intentionally left out. For instance, given a scenario where a user pushes commits with unverified, and potentially unsafe, commits. When secd receives this specific task, it would not be run. However, any subsequent updates to that repository would still be run given the latest push only contains signed commits (even though previous commits were not signed). Thus, in this particular scenario, the responsibility of ensuring the legitimacy of the previous commit falls upon the user.

4.2 Dataset Exposure

As discussed in Section 1.1, and mentioned in 3.7 the challenge of model disclosure was a significant concern when utilizing machine learning on sensitive data within a TRE. This issue persisted within our system as well. It is theoretically possible for an authenticated user to retrieve sensitive data from an internal database and output the data directly from the program, thereby exposing the sensitive data beyond the confines of the secure environment.

Any trivial attempts to detect and block this behavior should be possible to circumvent by a sufficiently skilled user, such as encoding or encrypting the output, or even training a simple machine learning model which can then be queried for the original data.

However, our approach adopts a different perspective, assuming researchers are benevolent as they are the ones inserting the sensitive data in the first place. Thus, in the event of data leaking outside the secure environment, it would ultimately be in the researchers' best interested to maintain its confidentiality.

4.3 Ethical Considerations

When laying the groundwork for this project, ethical considerations played a pivotal role. In any endeavor involving sensitive data, particularly patient records, it is crucial to ensure that the data remains secure and confidential. It is therefore necessary for the data storage and processing systems employ robust encryption techniques to prevent unauthorized access. Access to sensitive data is restricted to authorized accounts only, and rigorous access controls are enforced to prevent data breaches.

4.4 Evaluating goals

In this section we aim to answer the objectives, research questions and hypothesis that we defined in the beginning of this paper, see 1.2.

4.4.1 Objectives

1. We have not evaluated the legal and ethical standards ourselves, however we have followed precedent set by the ZT paper [6], and the recommendations from our peers at KTH [4].

2. Whether this has been completed or not depends on what level of confidentiality is expected. The researchers are still able to see the data, and as stated in section 4.2, it is up to the researcher to follow best practices.

3. While the proposed workflow is promising in this regard, the implemented prototype turned out to be neither user-friendly nor practical.

4.4.2 Research questions

1. By implementing a secure environments where users are not accessing the sensitive data directly, but rather through a proxy, meaningful insight can be extracted by the researcher in the form of a for example trained machine learning model. However, as described in 4.2, our proposed solution is not

bullet-proof, as it relies on the benevolence of researchers. Therefore, while it secures the sensitive data storage-wise, it does not directly ensure the confidentiality of the individuals involved.

2. When designing the secure environment we identified several useful technologies such as GitLab's integration of GPG-signed commits to verify the authenticity of a user. Another example is ZTA that we used as a basis when designing the authentication parts of the environment.

3. Researchers will certainly observe a slight decrease in productivity when solely examining the time from code submission to result retrieval as opposed to executing the same code on the sensitive data hosted locally. However, some aspects are improved by using our prototype. For instance, its centralized design makes caching Docker images more efficient, potentially resulting in significant amount time savings if the image is large. Other aspects to consider are stability and reliability. When executing time-intensive tasks, researcher need not to be concerned about the state of the hardware used during the task execution, and could in the interim divert their attention to other tasks.

4.4.3 Hypotheses

1. We believe this is possible, however issues with model disclosure remain unsolved. Our prototype shows a possible workflow which, while usable, can be improved.

2. According to the interview results, the workflow ended up feeling cumbersome due to a complicated setup. Therefore, it appears that the workflow does not boost research productivity.

5 Future work

Additional database support could be added to facilitate the execution of a broader range of tasks. Databases such as MongoDB and PostgreSQL are worth considering.

The automation of sensitive dataset imports could address the question of how data is securely transported from a sensitive data provider to our secure cluster.

Although we have designed the secure environment to be accessible only through a VPN, it could potentially be taken another step further by preventing any internal traffic from leaving the environment, except through the VPN. Nonetheless, this would introduce new challenges, such as certificate management and software updates.

6 Conclusion

This project has successfully implemented a secure machine learning workflow for processing sensitive data. However, as the field expert indicated in the interviews, the workflow was not user-friendly and needs improvements to be seen as a useful tool.

Finally, as described in 4.2, our solution is not bullet-proof regarding sensitive dataset exposure. Therefore, a system that is capable of running machine learning workflows while preventing any form of sensitive data to be exposed, despite bad actors, is yet to be designed.

References

[1] E. Vayena, A. Blasimme, and I. G. Cohen, "Machine learning in medicine: Addressing ethical challenges," *PLOS Medicine*, vol. 15, no. 11, p. e1002689, 2018. doi: 10.1371/journal.pmed.1002689. [Online]. Available: https://doi.org/10.1371/journal.pmed.1002689

[2] S. Kavianpour, J. Sutherland, E. Mansouri-Benssassi, N. Coull, and E. Jefferson, "Next-generation capabilities in trusted research environments: Interview study," *J Med Internet Res*, vol. 24, no. 9, p. e33720, 2022. doi: 10.2196/33720. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9533202/

[3] E. Mansouri-Benssassi, S. Rogers, J. Smith, F. Ritchie, and E. Jefferson, "Machine learning models disclosure from trusted research environments (tre), challenges and opportunities," *arXiv preprint arXiv:2111.05628*, 2021. doi: 10.48550/arXiv.2111.05628. [Online]. Available: https://doi.org/10.48550/arXiv.2111.05628

[4] J. Raghothama, R. Bergström, and H. Krishna, "Private communication," Private communication, 2023.

[5] D. Rankin, M. Black, R. Bond, J. Wallace, M. Mulvenna, and G. Epelde, "Reliability of supervised machine learning using synthetic data in health care: Model to preserve privacy for data sharing," *JMIR Med Inform*, vol. 8, no. 7, p. e18910, 2020. doi: 10.2196/18910. [Online]. Available: https://medinform.jmir.org/2020/7/e18910/

[6] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," National Institute of Standards and Technology, Special Publication (NIST SP), 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-207

[7] Jupyter, "Project jupyter — home." [Online]. Available: https://jupyter.org/

[8] GitLab, "The first single application for the entire devops lifecycle - gitlab." [Online]. Available: https://about.gitlab.com/

[9] Oracle, "Mysql." [Online]. Available: https://www.mysql.com/

[10] Neo4j, "Neo4j graph platform – the leader in graph databases." [Online]. Available: https://neo4j.com/

[11] Keycloak. [Online]. Available: https://www.keycloak.org/

[12] OpenVPN, "Openvpn." [Online]. Available: https://openvpn.net/

[13] "Pep 318 – decorators for functions and methods — peps.python.org." [Online]. Available: https://peps.python.org/pep-0318/

[14] Pickle, "pickle — python object serialization — python 3.12.0 documentation." [Online]. Available: https://docs.python.org/3/library/pickle.html

[15] KTH, "Centre for data driven health (cddh)." [Online]. Available: https://www.kth.se/cddh

A Hardware used

Hardware is required in order to run the machine learning workflows. Three supermicro servers previously used by KTH PDC were graciously donated for this project, with specs shown at 2.

Part	Part number	Total
CPU	2x Intel Xeon E5-2690 @ 2.60GHz	48 cores
RAM	16x Samsung 39F259FE512 32GB DDR4 2133 MHz	512 GB
GPU	NVIDIA Tesla K80	24 GB VRAM
OS disk	ADATA SU650 SSD	240GB

Table 2: Supermicro compute server specs

For storage, a Dell Precision 7920 Rack with specs 3 was purchased and running TrueNAS Core. Disk redundancy was ensured with ZFS, and performance ensured with RAM cache and write log SSDs.

Part	Part number	Total
CPU	Intel Xeon Bronze 3204 @ 1.90GHz	6 cores
RAM	8x SK Hynix HMA82GR7DJR8N 16GB DDR4 2133 MHz	128 GB
OS disk	KIOXIA KXG80ZNV512G 512GB NVMe SSD	512 GB
Write log	2x Dell 1.92TB MZ7L31T9HBLTAD3 PCIe SSD	1.92 TB (redundant)
NAS disks	4x WD Ultrastar DC HC520 HUH721212ALE600	48 TB (23 TB usable redundant)

Table 3: Dell NAS specs

Miscellaneous hardware included:

- HP server running PFsense as Firewall/VPN server
- StarTech 22U Server Rack Cabinet with secure locking door RK2236BKF
- Aten MasterView USB VGA KVM CS1308
- Philips VGA monitor
- Dell USB Keyboard
- Power strip

B secd Configuration

secd accepts a configuration file in YAML format allowing the user to define certain parameters.

```
runfor: 0.1
gpu: false
cache_dir: "small-docker"
mount_path: "/app/cache"
```

Figure 4: Example secd.yml config file, in YAML format

- runfor - number of hours before the run is terminated, can be a float - example 0.1 = 6 minutes. Default is 3
- gpu - choose whether a GPU is attached during run (may delay start if all GPUs are in use). Default is false
- cache_dir - path inside storage bucket. If not specified, caching will not be used.
- mount_path - mount path inside container, default is ”/cache”