Aito Data and Security Policy

This is an initial draft of the Aito Data Policy. It is not yet finalised, and there will be changes in the future. We will notify the users when material changes are done to the information written here.

1. Aito.ai service description

1.1. Offering maturity

Aito.ai is still in pilot phase, meaning that we do not offer fixed pricing or SLAs yet, but work in co-operation with our customers to provide a best-effort service to selected customers.

1.2. Service description

Aito.ai is a software-as-a-service company (**SaaS**), which offers a database-like solution with Machine Learning- (**ML**) and Artificial Intelligence- (**AI**) functionalities. Aito runs on cloud infrastructure provided by Amazon Web Services (**AWS**), and does not operate own hardware.

Aito offers a comprehensive solution for data indexing and querying. Aito operates under a SaaS-model, which means that Aito offers an API-interface (Application Programming Interface) that customers can use to upload, query and manage the data. Aito provides the API as a REST-API, with JSON as the data-interchange protocol. The customer gets a custom endpoint (**https://<customer>.api.aito.ai**) and needs a custom API-key, used for authentication and authorisation, and with these the customer can use the Aito service.

Aito handles all the server infrastructure, encryption certificates, data storage and backups on behalf of the customer. The service could thus be seen as a turn-key service, with the customer being responsible for building the integration to the provided API, and Aito taking care of all the management and running the service.

2. General operational guidelines

2.1. Account policy

Aito.ai production environment runs on AWS infrastructure. Production and development accounts are separated and independent. This means that separate login and access keys are needed in order to access these accounts. The accounts are not linked and do no share user accounts.

2.2. Usernames and access management

AWS offers different means of provisioning and managing the infrastructure, namely the AWS Console for interactive UI-based usage, and API-based access.

- **API-based access**
  API-based access is handled either through the AWS-command line tools or some of the provided programming interface libraries. The libraries are provided at least for Java, node.js and Python. The AWS-command line tools are built using the Python libraries

- **UI-based access**
  The AWS console can be used to manage every aspect of the infrastructure. Internally the console is built using the javascript libraries, and thus it uses the same API- endpoints as are being used by the various programming interfaces.

All users having console-access to the production, i.e. users having 'PowerUser'-privileges, are created through the scripted environment. Access to the console is then granted separately to these users by administrator users. Administrator and PowerUser access keys are not deemed safe to upload to any external systems, like an external CI/CD-system or provisioning scripts.

2.2.1. API-based access

API-access is enforced based on AWS access keys. These are generated from the console, and require a separate access key and a secret access key. The keys are automatically generated by AWS Identity and Access Management (**IAM**). The access keys for users are not shared, but generated separately for each user. Access keys are available only at generation or later to the administrator user. Normal and PowerUsers cannot later download these keys, but they must instead be recreated.

A set of keys can be generated and revoked individually for each user at any given time.

Programmatic access to AWS is handled in two different ways. Services and servers running in AWS are granted access based on service roles. A service role is a restricted role, managed through IAM. The permissions for each role are granted on a need-to-have basis, and each resource is only allowed roles which are necessary for the functionality. Hence, each service role gets minimal access rights, and each service only has access to roles it strictly needs.

Service roles are managed through the scripted environment, and all changes can be tracked back to a user. See the section Server infrastructure for more details.

2.2.2. Console access

The access to the UI-console is restricted to Aito-employees. There are no shared accounts, but every user has a private account, which is terminated as soon as employees leave Aito.

We enforce a strict policy of two-factor authentication for all users with console access. Two-factor auth for console access is implemented by AWS, and follows industry best practices.

AWS accounts are terminated as soon as a user leaves the company.

Should external parties require access to parts of the console or functionality provided through the console, we restrict the access permissions with a minimal-required-functionality or provide other means of access, e.g. a programmatic dashboard with only the specific functionality available, as described in the API-based access section.

2.3. Server infrastructure

The Aito service is built using the Infrastructure-as-code principle, meaning that all changes to the infrastructure are done through means that are possible to replicate:

1. There are separate, separately managed accounts for production use and test use. These do not share users.
2. All changes are managed through scripts, and manual changes are avoided where possible
3. All change scripts are version controlled in a private repository
4. The VCS is access controlled, and every change can be tracked back to an individual developer
5. All changes are applied automatically and tested in a separate test environment before implemented in production
6. Infrastructure supports rollback

2.4. Aito service region and GDPR compliance

Aito operates and is registered as a legal entity in Finland. The Aito infrastructure is provisioned in the AWS Region **eu-west-1**.

The eu-west-1 data centres physically reside in Dublin, Ireland. Like Finland, Ireland is also a member of the European Union (EU) and hence we comply to the laws and regulations mandated by Finland and the EU.

The most prominent requirement is the General Data Protection Regulation, the **GDPR**. Under the GDPR Aito operates as the Processor of data, whereas our customers are the Controller. As a data processor we are committed to help our customers comply with the GDPR regulations.

The GDPR applies to all stored data, which can be used to identify individual persons. The GDPR is not applied in cases where the data is anonymised in a way that prevents linking it back to individuals. Our customer, in the capacity of Controller, is ultimately responsible for any data stored into the system. Aito does not parse or use any semantic information about the data, and therefore we have no way of distinguishing between compliant and non-compliant data. We expect our customer to take necessary precautions when parsing, processing and storing data that is under the GDPR.

In order to comply with the GDPR Aito will enter into a data processing agreement with our customer before accepting any GDPR regulated data into the Aito service.

2.5. Coding conventions

The source code of the service is version controlled, and each change can be traced back to an individual developer. There are no shared accounts, so each user can be identified individually.

Source code is managed in a third party service, as is continuous integration. Since distributed version control (git) is being used, changes cannot readily be made to the sources without being noticed by the system.

Continuous integration is also handled by a 3rd party provider. It needs to access the Version Control System and the infrastructure, in order to be able to implement CI/CD, but the privileges are restricted to the minimum possible.

3. Network traffic

3.1. In-transit encryption

Aito.ai uses SSL/TLS-encrypted traffic for all customer data. Aito does not provide endpoints that would not be protected through SSL/TLS.

SSL/TLS is the default industry standard for client-server communication and the major benefits are twofold:

1. It enforces the server identity by using a certificate authority. This means that customers of the APIs are guarded against man-in-the-middle attacks, the most common type of data theft and forgery attacks.
2. It manages a key-exchange and encryption protocol that makes sure that all the traffic between client and host is encrypted, with a session based key, leaving it immune to eavesdropping.

SSL/TLS is colloquially referred to 'https'-traffic. Aito certificates are generated by the AWS, and use the best commonly available algorithms and protocols. Aito does not use self-signed certificates, or certificates signed by untrusted entities.

3.2. Certificate management

The reliability of the system is dependant on safe storage and management of the certificates and must offer a way revoke compromised certificates. The Aito-service uses the AWS Certificate Manager (CM), which offers all the required feature for such a system. The certificates used by Aito are issued through the Certificate Manager. This guarantees a level of safety according to the service level provided by the CM. Aito does not manage or handle the certificates through other means, nor does Aito export the encryption keys from the CM, as it might result compromise of the certificate security.

3.3. API endpoint management

API endpoints are provided using AWS API Gateway. Each customer gets a separate subdomain based endpoint, with a separate usage policy. This means that customers cannot intentionally or un-intentionally cause DoS for the other users due to lack of server resources.

Access restrictions to the endpoint based on e.g. IP or client certificate is not possible at the moment.

4. Index data

Aito stores user data both in transient and permanent (data-at-rest) fashion.

4.1. Environment separation

Each customer environment is separated from other environments on multiple levels. The data is stored in dedicated directories, and each respective environment manages it's own data. Environments are also incapable of seeing or accessing the data of other environments.

Each customer environment is in effect an own process, and is hence completely separated from all other environments. The queries are parsed and resolved locally in the environment, and no state is shared between environments.

4.2. Permanent storage

The permanent storage is managed on disk volumes. Each customer environment operates only on data stored through this environment and has no access to the data of other customer environments.

4.3. Transient data storage

Transient data storage in Aito is used for caching. The caches are specific to the environment, and hence only the customer specific environment has access to the cache. The same restrictions to access apply as with data at rest.

4.4. System level access to stored data

Aito personnel with access to the running system, and thus the data at rest, is limited to an absolute need-to-have basis. All maintenance operations to the systems, as well as any operations to the live system is handled through managed scripts and trusted path access, e.g. maintenance tools and scripts created for this purpose. Tools and scripts are version controlled and treated in similar fashion as all production code.

4.5. Data backups

In order to prevent catastrophic loss of indexed data, Aito executes an hourly backup of all the index data. The backups are performed for all customer automatically. Backups are retained until the subscription is terminated, after which all the backups are removed on a best-effort basis. The removal is not automated, so this might occur only at regular maintenance windows.

5. Data use

All data stored in the Aito system is owned by the end-user/customer. Aito manages the data on behalf of the customer, but the data ownership resides with the customer. In order to provide best-effort service to customer, Aito will occasionally log queries and responses to the system.

Logs are retained for 2 months (60 days), after which they are deleted from the system. Local log dumps are deleted once the purpose for accessing the logs is no longer valid.

Aito will not directly use the customer data for any internal or external purpose. The data is not shared with any external parties. Aito does not parse the data or use any of the semantics of the data for internal purposes. Data access behaviour and metadata about performance or query characteristics are, however, monitored and used for improving the level of the service. This data is anonymous and cannot be used to deduce information about service users.

6. Data access

Data access on Aito is authorised and restricted using a pair of customer specific API-keys. The keys are separated into a read-only key, with access to the query interfaces of the API, and a read-write key, which can be used to change the schema or the data in the database.

API-keys are generated on environment setup. The generation is done by the AWS API Gateway. The key is validated both on the API GW, as well as on the server level when accessing the data, preventing attacks by other authorised users.

API keys cannot, as of now, be generated by the end-user. Changing the API key must be done by Aito-administration. This feature is planned to be implemented as soon as the Aito management UI is online.

API keys are not sent to the customer over public and unencrypted channels. Getting hold of the API-key requires access to the AWS Console, and is thus protected as described in Console access.

6.1. Internal authorisation and authentication

It is not possible to limit access or to compartmentalise the data inside the Aito database. Aito is primarily meant for ML-operations over large datasets, and thus optimised for efficiency over such data. Should there be need for anonymising, hashing or limiting the access to parts of the data, this must be handled outside of the Aito software, on the client side.

Compartmentalisation can be handled by using a separate Aito-instance for this purpose. These two instances will share no state or data, and hence access can be managed independently. These environments can, however, not be linked and hence it is not possible to run queries over multiple environments in the Aito service.

6.2. Query validation

All the Aito APIs accept JSON-encoded payloads. The JSON format is parsed using pre-existing libraries and the messages are parsed according to the query schema in strict fashion. This means that invalid queries are rejected and thus not applied to the data.