Team Secrets

for Jira

https://www.teamsecrets.io

Practice safe sharing

Team Secrets protects the sensitive files you share in Jira with end-to-end encryption and real recipient verification.

End-to-end encryption

Team Secrets uses encryption to protect attachments and text fields from the moment you start uploading until they're opened by the recipient. Snooping on the transfer or the file will result in something undecipherable.

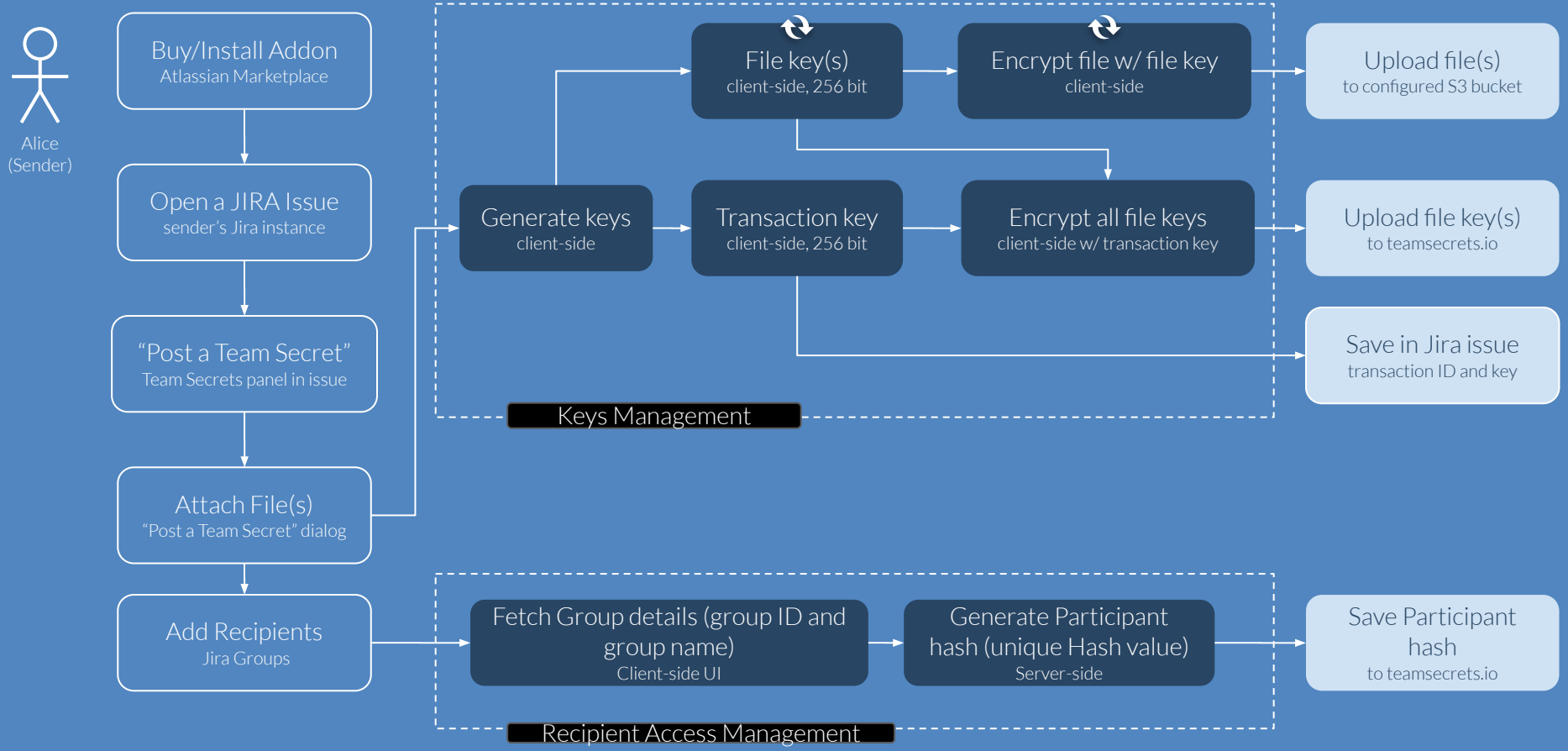
Real recipient verification

Team Secrets ensures that anyone opening your secret is authenticated using up to 3 forms of ID. Passphrase and mobile phone verification means only the people you choose can assemble the keys necessary to decrypt.

Secure fields and files

Your team is already sharing sensitive files and text fields in Jira so it's time to get them secured. Encrypt passwords, SSH keys, secret tokens, AWS credentials - any text or file you need to keep secret.

Team Secrets: **Sender**



Key Management

Sender side

Generate Key:

- Files key(s):
  - Each file is encrypted with the dedicated file key.
  - The file key(s) gets encrypted with the transaction key.
  - The encrypted file key(s) are stored into the ‘teamsecrets.io’ database.
- Transaction key:
  - Every transaction has one transaction key.
  - The transaction key gets stored in Jira issue properties.

Sender side

Recipient Access Management for Group-level Recipients

- Fetch group details: Fetch group details (group ID and group name) from the selected recipient group list.
- Generate participant hash: The group details cause the participant hash to be generated. The participant hash is a unique hash value. After generation, it is stored in an RDS made by teamsecrets.io.

Encryption

Sender

For each Team Secret created, a random 256 bit key ("**transaction key**") is generated client-side

For each file within a Team Secret, a random 256 bit key ("**file key**") is generated client-side

Each file is encrypted client-side with a different **file key** using 'xsalsa20-poly1305'

All **files keys** are encrypted client-side with the **transaction key** using 'xsalsa20-poly1305'

Encrypted **file keys** are uploaded to teamsecrets.io via HTTPS, stored in Team Secrets AWS RDS

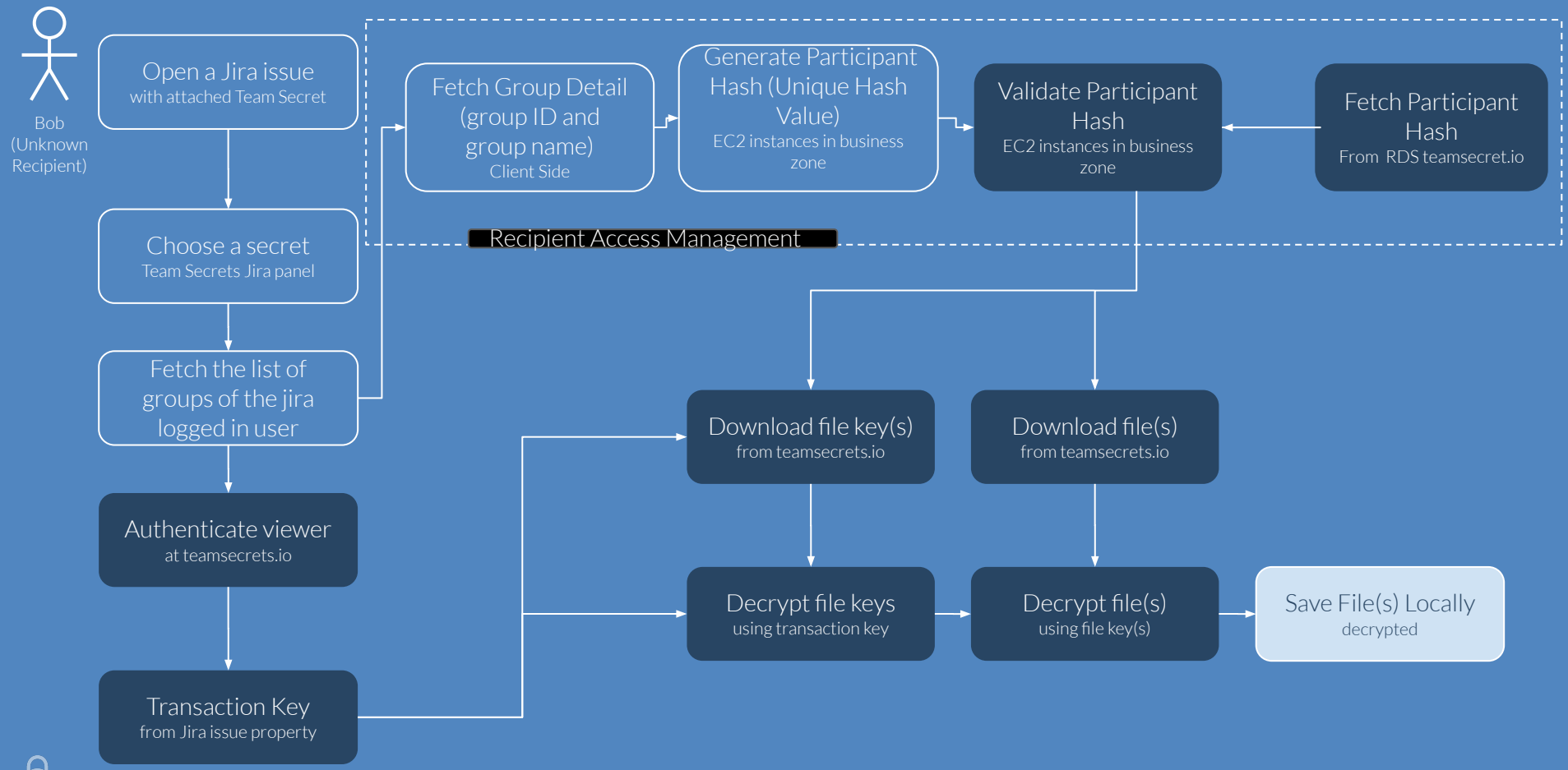
Encrypted **files** are uploaded to teamsecrets.io via HTTPS, stored in Team Secrets AWS S3 buckets

NEW Customers can optionally configure a custom S3 bucket to store and control encrypted **files**

If passphrase is required, the **transaction key** is encrypted client-side using AES-256 and passphrase

The **transaction key** is saved in a Jira issue entity property and never uploaded to teamsecrets.io

Team Secrets: **Recipient (Jira logged in User)**



Recipient Access Management

- Fetch Group details: Fetch Group details i.e group ID and group name from the logged in Jira user.
- Generate Participant Hash: A participant hash is generated from the group details. This is a unique hash value.
- Fetch Participant Hash : Fetch participant hash from teamsecret.io.
- Validate Participant Hash: Logged in user will be able to access the secret when “Generated Participant Hash” and “Stored Participant Hash” are the same.

Recipient side

Key Management:

- Transaction Key: Fetch Transaction Key from the Jira issue properties.
- Files key(s): Retrieve file key(s) from the database for the team secret.
- Decrypt all the file key(s) with the “Transaction key”.
- Finally all the file(s) are decrypted with with the file key(s) and file is downloaded on recipient’s machine.

Decryption

Recipient

An authenticated Jira user that passes mobile phone and/or passphrase verification can download:

- The encrypted **files** from teamsecrets.io
- The encrypted **file keys** from teamsecrets.io
- The **transaction key** from the local Jira issue entity property

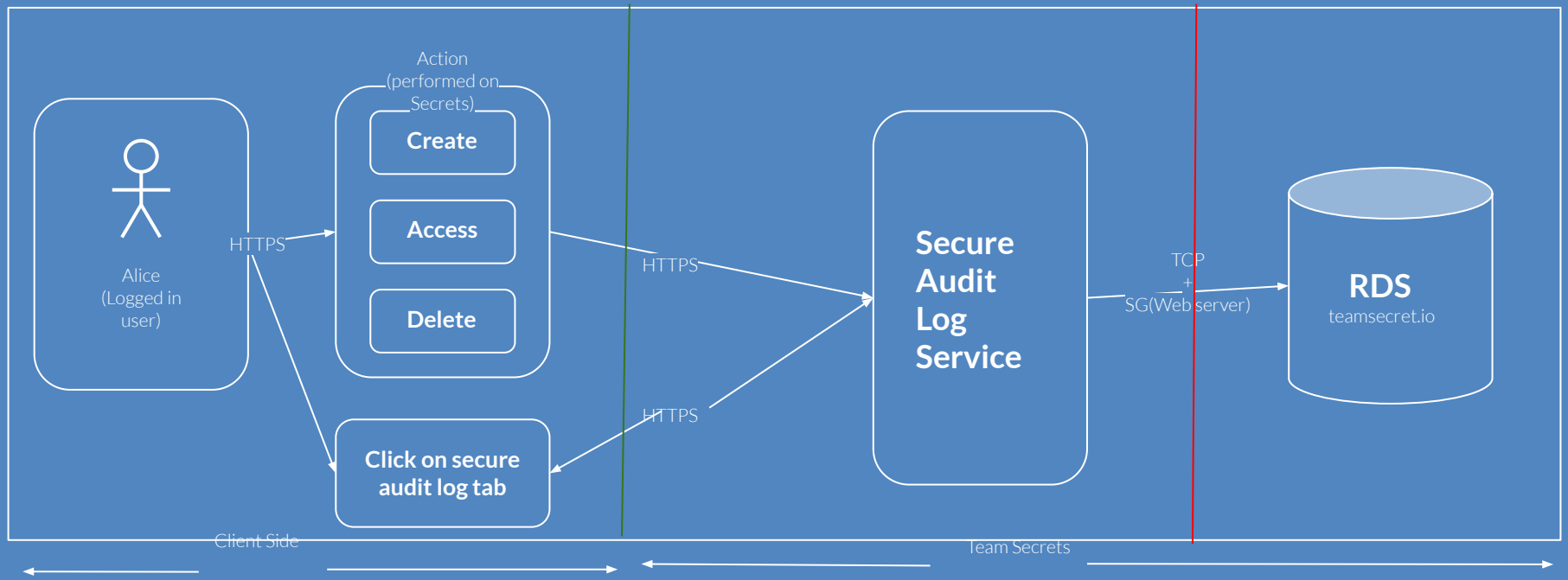
The **file keys** are decrypted client-side using the **transaction key**

Each file is decrypted client-side using a **file key**

The browser presents a standard file save dialog for each file so the recipient can save files locally

Secure Audit Log

- All the action performed by logged in user on secrets under UI components like create, delete and access are recorded under the **Secure Audit Log.**
- **Secure audit log** is a service and all the audit based data is stored inside Teamsecrets.io's RDS .



Facts

- We use the **xsalsa20-poly1305** algorithm with **256 bit keys** for each transfer and each file within a transfer
- If a secure passphrase is used, we encrypt the secret key with **AES-256** as well

- All encryption and decryption of the files protected with Team Secrets happens using JavaScript on the **client-side in the browser**

- Since we never have the transaction key, **we cannot decrypt** your keys or files
- Someone accessing your Jira instance alone **cannot decrypt** the files
- Decryption requires **transaction key** in Jira, **files** AND **keys** from our servers
- If you **lose the transaction key** we can't help you decrypt
- If you **forgot the passphrase**, we can't help you decrypt

Logical Trust Zone/boundaries:

We have divided the complete network under AWS infrastructure into three logical trust zone/boundaries and we maintain different security level/policies for each:

Trust Zone

Perimeter Zone

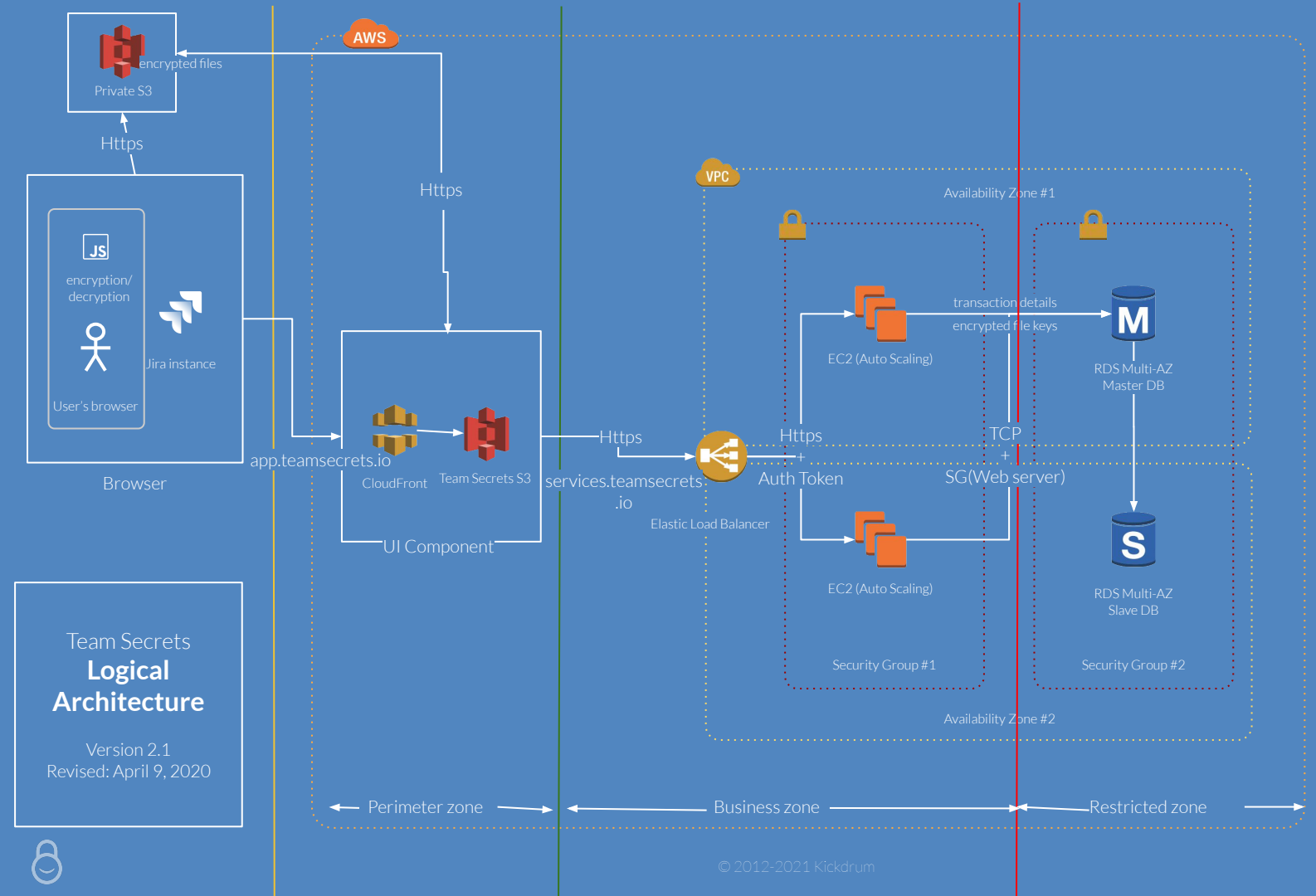
The perimeter zone is front facing and can be accessed through Jira with a web browser. It hosts the user interface(UI) component of Team Secrets for each Jira instance.

Business Zone

The business zone is on the next level and it consists of all our core business logic. All our service logic is inside this zone. When the logged-in user uses the Jira UI with the embedded team secret panel, the business layer is called from the team secret panel.

Restricted zone

The restricted zone is secure and can only be accessed from the internal dedicated server in the “Business Zone”. All the data and metadata resides in this zone.



Team Secrets
Logical Architecture

Version 2.1
Revised: April 9, 2020

Logical Zone/Trust boundaries and there level of securities:

Perimeter Zone: HTTPS, TLS, SSL, IPv6

Business Zone: HTTPS, Auth Token, Security zone policies

Restricted Zone: TCP, Special Security Zone Policy

Team Secrets: **Threat Model analysis**

We use STRIDE model to analysis/evaluation of level of trust

	Threat	**Property Violated**	**Threat Defence provide by Team Secrets**
S	Spoofing Identity	Authentication	HTTPS, SSL, TLS, Multi-factor Authentication (Jira Credential, Mobile verification and pass phrase)
T	Tempering with Data	Integrity	All data are inside the restricted zone/trust boundary and computed at run time.
R	Repudiation	Non-Repudiation	Every action we record along with associated metadata, we maintain Audit logs.
I	Information Disclosure	Confidentiality	We assure all the secrets are kept confidential, We use 256 bit key to encrypt all the secrets.
D	Denial of Service	Availability	We use DDOS protection provided by AWS.
E	Elevation of Privilege	Authorization	For every access of service, it requires a special AUTH TOKEN (teamSecrets.io generated). It is refreshed after a certain time interval. We maintain Role based access for the system administrator to access the resource with multi-factor authentication.

Encryption algorithm: xsalsa20-poly1305

XSalsa20 is a stream cipher based upon Salsa20 but with a much longer nonce: 192 bits instead of 64 bits.

XSalsa20 uses a 256-bit key as well as the first 128 bits of the nonce in order to compute a subkey. This subkey, as well as the remaining 64 bits of the nonce, are the parameters of the Salsa20 function used to actually generate the stream.

We achieve “Forward Secrecy” by using a different key for each transfer/file.

Libraries

https://github.com/dchest/tweetnacl-js

https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/crypto-js/CryptoJS%20v3.1.2.zip

Questions?

https://www.teamsecrets.io/support