# Security Analysis Report: GoNZ Travel Web Application

In this web application, we are saving personal data of all of the users – their full names, as well as employment details (position and department), and further personal information may be stored as the app becomes more complex, such as timetables or locations of certain employees via tours.

The Privacy Act 2020 provides certain rules and standards on collecting and securing personal data. It is important we comply with the act to remain lawful, but also ethically the act ensures we are responsible for keeping the personal data secure. If data was to be accessed by a bad actor, besides feeling personally violated a data breach could put our user’s identities at risk for use in fraud or other malicious activities.

One way to keep both our application and the personal data of our users safe is to use authentication and authorization. Authentication ensures we know who the user is when they are accessing the application so we can verify what information they are allowed to see and only let them access the information they are entitled to. We currently have a process in place to do this by having unique usernames and passwords for all of our users. If they want to access the app, they must sign in and we can authenticate who they are and control what data they are allowed to see. Django includes a set of built-in password validators which ensure any password being created is unique, uncommon, a minimum length, not all numeric and secure. Authorization is the process of checking if the user has certain permissions, we are using it in this application with Django’s built in groups. Different user groups (Agent, Admin, Management) all have different levels of permissions, such as admin having all permissions (superusers), and Agents only being able to edit their own posts. Authorization ensures that only certain people may be able to view information, and also that only certain people are permitted to change information.

Another security mechanism we can use to protect data privacy and the web application privacy is data validation. This helps prevent OWASP A1:2017-Injection, OWASP A4:2017 XML External Entities (XXE), and OWASP A7:2017-Cross-Site Scripting (XSS) vulnerabilities. Validation helps preserve data integrity, ensuring data is both accurate and consistent. Django has inbuilt validation methods for our form submissions where data is cleaned and validated to check for any invalid characters which could be malicious content. By limiting the length of certain fields or limiting the input to a selection we can stop bad actors injecting scripts as there is literally not enough space in the field. Using the http POST method on out submit in form handling also makes things more resistant to cross site forgery request attacks.

Cross site request forgery is a type of attack that tricks a user into submitting a malicious request. Depending on how it is done, it can put the individual victims information at risk, or if the victim is an admin or superuser it can jeopardize the integrity of the entire application. By using CSRF tokens in our html forms we can prevent these attacks. CSRF tokens are a unique, secret and predictable token generated server-side, and when a form is submitted it is checked against the token found in the user session and the request. If they are not the same, the session is invalid, and the event is logged as potential attack keeping the user’s information safe.

Data encryption is important to keep data secret and guarantee its integrity. Django uses PBKDF2 algorithm with a SHA256 hash as a default way of encrypting passwords. According to NIST(USA Govt organization) and the Django documentation, this should be secure enough for most users as it takes a lot of computing power to crack this type of encryption, however if we wanted we could have amore secure algorithm. However, I don’t think this application requires that level of security at this stage.