Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Pentest-Report Enpass Vault Features 05.2023

Cure53, Dr.-Ing. M. Heiderich, S. Possegger, Dipl.-Ing. D. Gstir, Dipl.-Ing. C. Rottermanner

Index

Introduction

"A vault is a secure space (encrypted database) that stores your Enpass data. By default, Enpass creates a single vault, but additional vaults can be created if you need to some of your data sequestered or stored separately. Vaults in Enpass can be shared and synced with family, co-workers, or other collaborators who need to access shared credentials."

From https://support.enpass.io/app/vault/using_vaults_in_enpass.htm

This documentation offers a comprehensive overview of the scope, findings, and final viewpoints concerning a Cure53 penetration test and source code audit against multiple Enpass features. Enpass Technologies Inc. requested this security assessment in early May 2023 for their upcoming new component named Enpass Hub.

The request was subsequently fulfilled by four senior members of the Cure53 team throughout CW20 and CW21 May 2023, selected for their specific know-how and experience in this field. The auditors honed in on a plethora of primary scope areas during their procedures, in adherence with a white-box penetration testing methodology.

Enpass Hub is a new feature about to be released to the market for the first time for Enpass business customers to assist with vault password recovery and controlled sharing via Hub. The work was grouped into three distinct work packages (WPs) for ease of execution, as defined below:

- **WP1**: Pentests, deep-dives & code audits against Enpass Vault password recovery
- **WP2**: Pentests, deep-dives & code audits against Enpass Vault Sharing feature
- **WP3**: Pentests, deep-dives & source code audits against Enpass Hub feature

Notably, Enpass attributes have already been subjected to analysis by Cure53. This report therefore represents a follow-up exercise for ENP-02 (development stage), with a particular focus on deep-dives for certain features.

The client handed over a host of assisting materials to aid the testing team's examinations, including sources, URLs, credentials, documentation, and other miscellaneous entities. These items also assisted with the preparations required ahead of the active assessment window, which were enacted in CW19 May 2023 and served to resolve any problems or delays in advance.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

In terms of the communication process, the teams adopted a dedicated and shared Slack channel. All relevant personnel from Enpass Technologies and Cure53 were invited to collaborate using this format. Communication during the test was seamless and few inquiries needed to be made. The scope was thoroughly planned and unambiguous, and there were no significant obstacles encountered throughout the process.

Cure53 provided frequent updates regarding the testing progress and corresponding findings. The team also enacted live reporting by furnishing expansive write-ups of each problem whilst the project was ongoing.

With respect to the findings, the Cure53 team's satisfactory coverage over the three work packages yielded a total of six. Half of those six findings were considered to exhibit noteworthy security threats and were assigned to the *Identified Vulnerabilities* section, whilst the remaining three offered negligible risk of exploitation and were documented under *Miscellaneous Issues*. To extrapolate Cure53's viewpoint concerning the total volume of security concerns detected, one can certainly argue that the security posture of the audited features has been established to a commendable standard. Despite the presence of a *High*-severity vulnerability that requires urgent nullification, the overall impression of the features' degree of resilience was positive. This opinion is corroborated by the web application area's effective safeguarding against traditional, associated security compromise vectors such as XSS, CSRF, injection-based attacks, and similar. The developer team's due diligence in establishing a meticulously implemented and performant codebase that successfully nullifies the majority of commonly encountered weaknesses is evident.

Nonetheless, Cure53 noted a number of opportunities for supplementary hardening. These miscellaneous findings do not constitute vulnerabilities per se, but should be addressed at the earliest possible convenience to imbue optimum security paradigms for the components in scope. Notably, however, one of these miscellaneous issues was deemed a false positive. The audit examined specific features based on the findings of the previous ENP-02 assessment carried out during the development phase of the Enpass Hub feature. As such, the meritorious outcome encountered for this iteration further reinforces the company's continued commitment to security.

To summarize, with the caveat of the aforementioned *High* impact flaw, the Enpass features application demonstrates a generally strong security posture. The Enpass Technologies team should allocate enough resources to initiate follow-up mitigations for all areas of concern outlined in this report, which will undoubtedly contribute to supplying first-rate defense-in-depth for the aspects in question if correctly administered.

From a structural perspective, the report is divided into a number of key sections moving forward. Firstly, the scope, test setup, and available materials are all defined in the bullet points enumerated below.

After that, all findings are outlined in ticket format and by chronological order of detection, starting with the vulnerabilities unearthed and culminating with the general weaknesses. Each ticket provides a unique identifier, an advanced technical description, a Proof-of-Concept (PoC) or steps to reproduce, and Cure53's suggested solution(s) for negation.

To finalize proceedings, the *Conclusions* section elucidates Cure53's final thoughts concerning the scope and exercise in general, before underlining the perceived security posture of the scope comprising the Enpass Vault Password Recovery, Vault Sharing, and Hub features.

Scope

- **Penetration tests, deep dives & code audits against Enpass Vault features**
    - **WP1**: Pentests, deep-dives & code audits against Enpass Vault Password Recovery
        - **Sources:**
            - *enpass-core-audit-1369.zip*
            - *enpass-aux-server-source.zip*
    - **WP2**: Pentests, deep-dives & code audits against Enpass Vault Sharing feature
        - **Sources:**
            - *enpass-core-audit-1369.zip*
            - *enpass-aux-server-source.zip*
    - **WP3**: Pentests, deep-dives & source code audits against Enpass Hub feature
        - **Sources:**
            - *enpass-aux-server-source.zip*
    - **Provided documentation:**
        - *Enpass Hub Security.docx*
        - *Enpass Hub Api Docs.zip*
    - **Relevant URLs:**
        - **Hub:**
            - https://hub-pt.enpass.io
        - **License server:**
            - https://license-pt.enpass.io
        - **Admin console:**
            - https://console-pt.enpass.io
    - **Test-supporting material was shared with Cure53**
    - **All relevant sources were shared with Cure53**

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, all tickets are given a unique identifier (e.g., *ENP-03-001*) to facilitate any future follow-up correspondence.

ENP-03-001 WP3: Lack of *python-requests* timeouts *(Low)*

***Fix note:** This issue was mitigated during active testing and fix-verified by Cure53.*

Whilst auditing the *enpass-aux-server* repository, Cure53 verified that the application utilizes the *python-requests* module. Timeouts in the *python-requests* module refer to the maximum duration a request will wait for a response before raising an exception. By default, any *python-requests* calls will wait until the connection is closed. Owing to this behavior, in the event the timeout has not been configured or has been set to an excessively high value, the application may assume an indefinite state and wait for a response that may never arrive.

Affected files:

- *App/enpass-aux-server/accesskey/tasks.pyPython*
- *App/enpass-aux-server/auth_token/utils.py*
- *App/enpass-aux-server/integration/admin.py*
- *App/enpass-aux-server/user/tasks.py*

Affected code:
```
response = requests.post(
            request_url,
            data = json.dumps(payload),
            headers=request_header
)
```

Generally speaking, optimal timeout configuration ensures that the application handles delays gracefully and avoids impermanent states.

To mitigate this issue, Cure53 advises incorporating a timeout to all *python-request* calls, which would effectively help to avoid indefinite program stalling.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

ENP-03-002 WP3: DoS via organization connector deletion *(Medium)*

Fix note: *This issue was mitigated during active testing and fix-verified by Cure53.*

To integrate the Enpass Hub into Enpass, an organization connector must be created within the Hub interface and subsequently specified during the setup process within the Admin console[1]. If this connector is deleted after the integration has been configured, functionality associated with the Enpass Hub integration - such as recovery - can be rendered useless. In the event an administrator attempts to access Enpass Hub functionalities post-connector deletion, the user will be logged out automatically and redirected to the login page.

Affected endpoint:
/admin/integration/organizationconnector/<id>/delete/

Steps to reproduce:
1. Create a new organization connector via *https://hub-pt.enpass.io/admin/integration/organizationconnector/*.
2. Perform the Enpass Hub integration in *https://console-pt.enpass.io/enpasshub/* using the previously generated access key.
3. Delete the connector created in Step 1.
4. Attempt to access an area such as the following: *https://console-pt.enpass.io/enpasshub/#/recovery*.

To mitigate this issue, Cure53 advises ensuring that the connector is no longer in use before deletion. Another effective countermeasure would be for the Admin Console application to automatically deactivate the integration as soon as access with the configured access key is no longer possible. The administrator should be informed about this weakness and the connector should only be removed after confirmation by the user.

[1] https://console-pt.enpass.io/

ENP-03-005 WP1: Vault key theft via recovery payload hijacking (*High*)

Fix note: *This issue was mitigated during active testing and fix-verified by Cure53.*

Note from Enpass*: Successful exploitation requires the following: The attacker is already a legitimate user of the organization and is able to obtain data by breaking the HTTPS connection itself.*

Whilst inspecting the cryptographic algorithms behind the recovery flow, Cure53 noted that recovery information is not bound to a specific user or recovery request. This potentially enables rogue users to recover another user's vault key.

Considering an attack model whereby a malicious user obtains access to another user's recovery data (submitted to the Hub endpoint */api/v1/recovery/setdata/*), they will subsequently be able to submit the recovery data as their own and instigate a password recovery for their own vault. The administrator user will remain oblivious to any wrongdoings here, since the recovery request appears to represent a legitimate request from the rogue user. Once the administrator approves the request, the attacker will be able to decrypt the victim user's vault key via the recovery response. In the event the adversary can also retrieve access to the victim's Enpass vault, they can proceed to decrypt any data stored within.

PoC:

1. A rogue user (e.g. *pentest5*) submits stolen recovery data from a victim (e.g. *pentest4*) as their own by using the following cURL command. Note that the rogue user can simply use their own authorization token to authenticate at the Hub.

```
curl -v \
       -H 'User-Agent: Mozilla/5.0 Enpass' \
       -H 'Accept: */*' \
       -H 'Authorization: Bearer [BEARER-TOKEN]' \
       -H 'Content-Type: application/json' \
       -X POST https://hub-pt.enpass.io/api/v1/recovery/setdata/ \
       -d '{"device":
{"country":"Germany","dm":"","dm_version":"","flavour":"windows","flavour
_version":"10","id":"83719783-d526-49cd-87af-
8c0ef0b3a075","lang":"de_DE","name":"DESKTOP-
38IUJ2O","os":"windows","os_version":"10","type":"desktop"},"pbkdf2_itera
tion_count":320000,"secret_changing_device":"DESKTOP-
38IUJ2O","secret_key":"dPl4HxOG/
OatksQCvp7Cx+6fDYud7lSVR+qw+uaYadeXFe9TnJelB8Gc3fnWyWREs8sNXZSUOEYA4csHMq
xHfrBveYWzGUJvIbVll04IcjymlaAMtCic9Q+TQken3A8ELSLP8zlykoeWTfROO5Y/
z2DdFt6DpDNy0ebQlLYGANZJNqqTFdJKD/9I80v+KpsRztQOio6cZVG50I5MX+T5/
zDVGYaqBFYJMYGFuQbFv6+izwB+WjD1MjN/
To2wTpdwLgHmR1uxPvIwXwUnwOZUlF1eegXFUGD3Wl1jlWKZ3StWLwfrM78ItVri8ijTtpAyj
5kk1l2YaHWNqWaxiXxNGC7YfZfs34t748fewv4sG1YlZ47JCldxaURyuZwrfxtOJyrzT26pxK
nhWneo1Z6KNameZe27JB8dOyWsJmmmcFZsfCkE8ANkfpaaBt+PwJ0dCOYJDfdbhKFM7QuS+4C
/
MuYINOIAg0JisOTJ0zM1rHSLoOW7zN2oY2OMO8lOAYrs","secret_last_changed_time":
1684584904,"team_slug":"a2552864-abd8-4fa4-9917-
ee05b8955bde","uid":"recovery","vault":
{"cloud":"microsoft_365","creator_email":"pentest5@acmebizness.com","loca
tion":
{"drive_id":"","folder_id":"","path":"","url":"","vault_space":"one_drive
"},"name":"Primary","pbkdf2_iteration_count":320000,"private":true,"team_
id":"pentest5@acmebizness.com","vault_id":"team-primary"}}'
```

2. The rogue user submits a recovery request for their own vault:

```
curl -v \
       -H 'User-Agent: Mozilla/5.0 Enpass' \
       -H 'Accept: */*' \
       -H 'Authorization: Bearer [BEARER-TOKEN]' \
       -H 'Content-Type: application/json' -X POST https://hub-
pt.enpass.io/api/v1/recovery/request/ \
-d '{"device":
{"country":"Germany","dm":"","dm_version":"","flavour":"windows","flavour
_version":"10","id":"83719783-d526-49cd-87af-
8c0ef0b3a075","lang":"de_DE","name":"DESKTOP-
38IUJ2O","os":"windows","os_version":"10","type":"desktop"},"force_reques
t":true,"message":"","secret_changing_device":"DESKTOP-
8QHKLGM","secret_last_changed_time":1684161587,"team_slug":"a2552864-
abd8-4fa4-9917-ee05b8955bde","vault":{"vault_id":"team-primary"}}'
```

3. An administrator approves the recovery request.
4. The rogue user receives the recovery link, derives the response UUID, and fetches the recovery response from the Hub:

```
curl -H 'User-Agent: Mozilla/5.0 Enpass' \
       -H 'Accept: */*' \
       -H 'Authorization: Bearer [BEARER-TOKEN]' \
       -H 'Content-Type: application/json' \
       -X POST https://hub-pt.enpass.io/api/v1/sharing/fetch/ \
       -d '{"device":
{"country":"Germany","dm":"","dm_version":"","flavour":"windows","flavour
_version":"10","id":"83719783-d526-49cd-87af-
8c0ef0b3a075","lang":"de_DE","name":"DESKTOP-
38IUJ2O","os":"windows","os_version":"10","type":"desktop"},"team_slug":"
a2552864-abd8-4fa4-9917-ee05b8955bde","uuid":"add35a08-d8e7-467a-ac26-
9298d9e0a64a"}'
```

To mitigate this issue, Cure53 recommends cryptographically binding the recovery data for each vault to the owning user and vault. This information must be verified to match the requesting user's information by the Administrator's Enpass instance when approving a recovery request.

Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit but may assist an attacker in successfully achieving malicious objectives in the future. Most of these results are vulnerable code snippets that did not provide an easy method by which to be called. Conclusively, whilst a vulnerability is present, an exploit may not always be possible.

ENP-03-003 WP3: Lack of access key password policy *(Low)*

Fix note: *This issue was mitigated during active testing and fix-verified by Cure53.*

To enable the Enpass Hub integration, an organization connector must initially be created via *https://hub-pt.enpass.io/admin/integration/organizationconnector/*. Here, Cure53 validated that any character string is permissible as an access key, such as simply *1*. Henceforth, the capability to configure an insecure access key is facilitated, thereby lowering the difficulty with which an attacker could gain access to this connection and subsequently retrieve sensitive information or manipulate data under certain circumstances. However, successful exploitation of this behavior would require the attacker to gain access to the connection between the Admin Console and Enpass Hub.

PoC:
The following request can be applied to create a new connector with a trivially weak access key. Pertinently, the CSRF token must be adapted or the request will fail.

```
POST /admin/integration/organizationconnector/add/ HTTP/1.1
Host: hub-pt.enpass.io
Cookie:
csrftoken=sIpmuIZOT0myN3lUmgOGmnxvzR4UGytvegJIZwDhFhF5HTrwIONdSiynt5c5cAc6;
sessionid=jfls8uy0gqs6pdklgbnznb3zshek8udh
Content-Length: 152
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/113.0.5672.93 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

csrfmiddlewaretoken=Eu6pQQaWjmPIc3PsjIkBlXchyOEjf6VJq2qLlEOp5D8f6TV4Fgj8RSd9s2Mu
L8Ek&name=Test+Weak+Secret+Key&secret_key=1&auth_secret_key=1&_save=Save
```

To mitigate this issue, Cure53 advises enforcing a password policy that guarantees only sufficiently secure passwords can be configured. A strong password is defined by the following characteristics:

- The password should comprise at least 10 characters.
- The password should consist of upper and lower case letters, numbers, and special characters.
- The password should not represent a frequently used genericism, such as a sequence of numbers, sequence of letters, dictionary entry, or similar.

Alternatively, another potential solution would be to generate the access key automatically. However, the developer team should utilize secure random number generators in this context to ensure only truly random access keys are generated.

ENP-03-004 WP3: Insecure PRNG usage in Hub Admin OTP generator *(Low)*

Fix note: *This issue was mitigated during active testing and fix-verified by Cure53.*

When authenticating against the Django admin interface (*/admin* endpoints), an OTP token is sent to the respective user's email address. This code is generated using the Python *random*[2] module, which is considered suboptimal for cryptographic purposes.

In light of this, an attacker may be able to gather enough pertinent information concerning past OTP codes to guess future tokens, rendering the second authentication factor via email less secure. Since access grants complete read and write access to the Hub database, the administrative interface constitutes a key area of sensitivity within the Hub's infrastructure and must be protected to the highest degree of efficacy.

Affected file:
enpass-aux-server/authorise/models.py

Affected code:
```
def gen_otp():
    '''
    Generate a random otp of predefined length
    '''
    return ''.join(random.choice(string.digits) for _ in range(OTP_LENGTH))
def gen_alphanum_otp():
    '''
    Generate a random otp of predefined length
    using characters defined
    '''
```

[2] https://docs.python.org/3/library/random.html

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
    characters = '123456789ABCDEFGHJKMNPQRSTUVWXYZ'
    return ''.join(random.choice(characters) for _ in range(OTP_LENGTH))
```

To mitigate this issue, Cure53 recommends replacing the Python *random* module with the cryptographically secure equivalent *secrets*. This approach complies with the recommendations stipulated in the Python documentation to adequately protect the Hub's administration interface.[3][4]

ENP-03-006 WP3: Lack of rate limiting for Hub Admin authentication endpoint *(Info)*

***Note**: This issue is considered a false positive.*

Whilst verifying the problematic behaviors discussed in ticket ENP-03-004, Cure53 noted that rate limiting has not been established for the Hub Admin authentication endpoint, therefore permitting endless retries of OTP codes.

Whilst the OTP codes for this endpoint represent alphanumeric codes comprising six characters, the likelihood of guessing the correct code within its validity time is relatively minimal. However, this is still considered substandard and could benefit from improvement, particularly considering the correlatory flaw of utilizing a weak random number generator for generating OTP codes.

To mitigate this issue, Cure53 suggests enabling rate limiting for the Hub Admin panel's authentication endpoint.

[3] https://docs.python.org/3/library/random.html#module-random
[4] https://docs.python.org/3/library/secrets.html

Conclusions

In context, this assessment represented the second security evaluation performed by Cure53 in 2023 focussing on the Enpass Vault Password Recovery, Vault Sharing, and Hub features. This second iteration was deemed a requirement due to the additional code changes and comprehensive coverage requirement since the first assignment in development phase.

Cure53 engaged in constant discussions with the client via a dedicated Slack channel. The project maintainers provided prompt assistance to the test team when requested. Similarly, Cure53 conducted live reporting and provided thorough descriptions of each issue during the active engagement.

The fundamental objectives were to identify web and backend application related vulnerabilities and weaknesses in the recovery and vault sharing functionality, such as XSS. The audit team also strove to identify unauthenticated access, misuse of the backend APIs, injection attacks, or DoS attack vectors. Furthermore, Cure53 endeavored to ascertain whether any faults were persisted by old or vulnerable dependencies.

Concerning the overall impressions garnered following the finalization of this exercise, Cure53 would first like to draw attention to the praiseworthy aspects encountered whilst probing the web and API components:

- The solution was verified to be adequately robust against a plethora of traditional web application security attack vectors. For example, the test team was unable to identify any detrimental flaws connected with command injection, SQLi, XSS, CSRF, SSRF, or RCE throughout the course of this review.
- The application exhibited exemplary handling of user input, which serves to neutralize HTML injection and XSS risks.
- The session implementation proved resistant to all manipulation and cracking attempts instigated against it.

Conversely, one would now like to highlight some of the pressing concerns encountered during Cure53's stringent appraisals against each specific work package, starting with WP1 and WP2 as follows:

- Close attention was paid to the cryptography code behind the vault recovery and vault sharing processes. The code utilizes OpenSSL for low-level cryptographic primitives and secure random number generation. However, the discovery was made that the design does not account for the scenario whereby malicious actors hold legitimate user accounts. With this in mind, ticket ENP-03-005 pertains to the method by which a legitimate user, with the ability to steal the encrypted vault's recovery information, can trick the recovery admin to recover another user's vault secret.

- In addition, Cure53 acknowledged that the vault sharing and vault recovery code processes leverage the exact same cryptographic algorithm. Essentially, this means that the produced ciphertexts are interchangeable. However, since Enpass uses a dedicated set of encryption keys for each shared vault - in differentiation to the recovery key set - no attack vector could be found. Nevertheless, the developer team may wish to introduce a distinguisher within the cryptographic payloads to help prevent future attacks in this area.

Next, the testing team's efforts against the scope items within WP3 yielded a selection of negative impressions, as detailed below:

- Cure53 noted that there are some flaws in the integration of the Enpass Hub with the Admin Console, which could lead to an assumption of trust between these two systems. As a result, it can be assumed that a compromise of the Enpass Hub could have an impact on the Admin Console. With this in mind, the following vulnerabilities (both due to possible configuration weaknesses) have been identified in the integration.

- Due to modifications performed in the Enpass Hub web application, the functionality within the Admin Console may be rendered futile. Furthermore, Cure53 validated the absence of a mechanism to resolve this problem within the Admin Console. Supporting guidance on this weakness is offered in ticket ENP-03-002.

- The audit team confirmed the capability to configure a highly insecure access key by Organization Admin consisting of only one character. This could allow an attacker to compromise the connection between the Enpass Admin Console and Enpass Hub, henceforth retrieving access to sensitive information (see ENP-03-003).

Elsewhere, to complement the specific work package explorations, the testing team initiated advanced inspection techniques against other pertinent Enpass software ecosystem characteristics:

- The bearer tokens adopted for authentication were rigorously analyzed. Following this, Cure53 verified the usage of JWT, which prompted detailed auxiliary threat assessments. However, when attempting to manipulate the JWTs in order to escalate privileges in the application or influence transmitted data, the session was terminated immediately and restricted all further actions.

- The sound method by which authentication mechanisms are implemented serves to block the possibility of carrying out attacks such as CSRF.

- The available APIs were examined for authorization errors, which involved conducting both source and dynamic analyses in tandem. Here, Cure53 determined the impossibility of gaining access to foreign data. Likewise, administrative functions could not be invoked within the context of a low-privileged user. Consequently, one can presume that the verification of the existing ACLs had been implemented correctly.

In conclusion, Cure53 strongly encourages installing additional security controls to improve the system's security posture, as outlined in the *Miscellaneous Issues* segment. These deficiencies were mostly assigned a *Low* or merely *Info* severity rating.

In general, it is advisable to fix all the vulnerabilities listed in this report, including informational and low severity tickets, to the extent practicable. This will not only improve the security posture of the platform, but also minimize the accumulation of tickets in future security exercises. Note that at the time of writing, all fixes were already in place, highlighting the timely delivery of security patches by the Enpass team.

Moving forward, it is recommended to regularly test the application - at least once a year or after significant amendments - to ensure that new features do not introduce security vulnerabilities. This proven strategy consistently reduces the number of security issues and increases resilience to external online attacks.

Cure53 would like to thank Vinod Kumar, Harsh Valecha, Ankur Gupta, and Yogesh Kumar from the Enpass Technologies Inc. team for their excellent project coordination, support, and assistance, both before and during this assignment.