Security assessments for software that matters.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Pentest-Report Passbolt SCIM Azure UI, Infra & Internals 01.-02.2026

Cure53, Dr.-Ing. M. Heiderich, Dr. N. Kobeissi, J. Ginesin

Index

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Introduction

“Passbolt is a hybrid credential platform. It is built-first for modern IT teams, yet simple enough for everyone. A sovereign, battle-tested solution that delivers for a team of 5, or an organisation of 5000.”

From https://www.passbolt.com/

This report details the results of a Cure53 penetration test and source code audit performed on the Passbolt SCIM plugin, Azure AD integration, and directory synchronization components.

For background information, the project was commissioned by Passbolt SA in November 2025. The core aims were to appraise the security proficiency of the aforementioned aspects and offer actionable guidance for identified vulnerabilities. The investigation undertakings were conducted by three experienced analysts across a two-week time frame in January and early February 2026 (CW04 and CW05). This review window comprised a total of fifteen evaluation days, which was deemed sufficient to achieve the desired coverage level.

Five distinct Work Packages (WPs) formed the basis of the scope and were defined as follows:

- **WP1:** White-box pen.-tests & code audits against SCIM Admin UI & Azure
- **WP2:** White-box pen.-tests & code audits against SCIM API & endpoints
- **WP3:** White-box pen.-tests & code audits against token auth & syncing
- **WP4:** White-box pen.-tests & code audits against plugin & user sync
- **WP5:** Cryptographic token entropy audit & authentication validation review

The pentesting methodology adhered to for this initiative was white-box. This comprehensive approach required access to all relevant system facets, including source code, test-supporting documentation, and other data points. These items were also referenced while completing the preparation phase shortly before the commencement of the audit (CW03 2026). The preliminary efforts effectively paved the way for productive and unhindered research into the targeted framework.

Cross-team discussions were handled on Slack. All personnel from both organizations that played an active part in this exercise were invited to join the dedicated channel. The collaborative process was fluid and the consulting team rarely needed to ask questions concerning the specifics of the underlying scope or testing requirements, as they were clearly understood from the outset.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Live reporting was offered and accepted to complement the ongoing studies. This was carried out on Slack to relay real-time insights regarding located findings.

Cure53 believes that ample explorative depth was achieved within the allotted evaluation period, yielding a total of twelve security deficiencies of varying impact and categorization. The vast majority of the tickets were assigned to the security vulnerabilities section (eleven), while the remaining issue pertains to a *Low*-risk hardening opportunity.

The SCIM plugin environment is characterized by structural integrity and clean implementation. While the yield of vulnerabilities is relatively high, the majority were filed with a *Medium* severity rating or lower. Moreover, the individual circumstances primarily constitute edge cases, rather than systemic failures.

Nonetheless, these discoveries should be rectified as soon as possible. All identified malpractices must be remediated in a timely manner, regardless of their perceived implications. Specifically, Cure53 advises prioritizing the resolution of race conditions and the implementation of general cryptographic enhancements, which will help to establish a robust security baseline. Addressing these specific technical areas will ensure that the platform's concurrency handling and data protection mechanisms meet industry standards.

A number of key chapters are outlined moving forward. Firstly, *Scope* provides all general setup information in bullet point form. The document then lists all security problems in chronological order of detection (rather than any other denominating factor, such as impact). Grouped into two subcategories, *Identified Vulnerabilities* and *Miscellaneous Issues*, each corresponding ticket offers a technical explanation, Proof-of-Concept (PoC) and/or steps to reproduce, code snippets of affected areas, and fix or preventative advice. Lastly, the *Conclusions* section summarizes Cure53's estimation of the targets based on the evidence collected, relaying next steps for the internal developer team to consider.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Scope

- **Pen.-tests & code audits against Passbolt SCIM Azure UI, infra & internals**
  - **WP1:** White-box pen.-tests & code audits against SCIM Admin UI & Azure
    - **Focus areas:**
      - Administration interface for SCIM configuration
      - Azure AD (Microsoft Entra ID) integration components
      - SCIM settings management and validation
    - **Source code:**
      - URL:
        - https://githlab.com/passbolt/passbolt-pro-api/-/tree/develop/plugins/PassboltEe/Scim
      - Branch:
        - develop
  - **WP2:** White-box pen.-tests & code audits against SCIM API & endpoints
    - **Focus areas:**
      - User and group provisioning operations (create, read, update, delete)
      - Request validation and error handling
    - **Source code:**
      - URL:
        - https://githlab.com/passbolt/passbolt-pro-api/-/tree/develop
      - Branch:
        - develop
  - **WP3:** White-box pen.-tests & code audits against token auth & syncing
    - **Focus areas:**
      - Bearer token generation and storage
      - Token-based authentication flow between IdP and SP
      - SCIM authentication middleware
    - **Source code:**
      - URL:
        - https://githlab.com/passbolt/passbolt-pro-api/-/tree/develop/plugins/PassboltEe/Scim
      - Branch:
        - develop
  - **WP4:** White-box pen.-tests & code audits against plugin & user sync
    - **Focus areas:**
      - DirectorySync plugin components
      - User and group synchronization logic
      - Database transaction handling and state management
    - **Source code:**
      - URL:
        - https://githlab.com/passbolt/passbolt-pro-api/
    - Branch:
      - develop

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

- **WP5:** Cryptographic token entropy audit & authentication validation review
  - **Focus areas:**
    - Token entropy and randomness analysis
    - Cryptographic primitive review (hashing, encryption)
  - **Source code:**
    - URL:
      - https://githlab.com/passbolt/passbolt-pro-api/-/tree/develop/plugins/PassboltEe/Scim
    - Branch:
      - develop
  - **Official SCIM documentation:**
    - https://www.passbolt.com/docs/admin/user-provisioning/scim/
- **Test-supporting material was shared with Cure53**
- **All relevant sources were shared with Cure53**

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

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., *PBL-15-001*) to facilitate any future follow-up correspondence.

PBL-15-001 WP3: Timing leak on secret token comparison *(Medium)*

Cure53 noted that the Passbolt SCIM plugin's *ScimResolver* handles the authentication logic for SCIM requests. Within this implementation, the *find* function verifies the bearer "secret" token sent by the Identity Provider (e.g., Azure AD) against the internal Passbolt user authorized to perform SCIM operations.

However, the aforementioned function utilizes a non-constant time string comparison operation "===" between the bearer token sent by the SCIM and the internal bearer token in the configuration. As a result, a malicious IdP without knowledge of the internally configured bearer token may guess the correct token by observing timing variations across failed authentication attempts.

Affected file:
Identifier/Resolver/ScimResolver.php

Affected code:

```
public function find(array $conditions, string $type = self::TYPE_AND):
ArrayAccess|array|null
{
    [...]
    if (
        $conditions['secret_token'] === $scimConfig['secret_token'] &&
        !empty($scimConfig['scim_user_id'])
    ) {
        /** @var \App\Model\Table\UsersTable $Users */
        $Users = $this->getTableLocator()->get('Users');

        return $Users->findIndex(Role::GUEST)->where([
            $Users->aliasField('id') => $scimConfig['scim_user_id'],
        ])->first();
    }
}
```

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

To mitigate this issue, Cure53 recommends employing a constant-time string comparison operation to check the validity of the untrusted bearer token value provided by the IdP, such as PHP's *hash_equals*[1] function. Hence, IdPs without access to the requisite bearer token will be restricted from guessing it via a timing side-channel.

PBL-15-002 WP3: Suboptimal token generation randomness *(Low)*

Cure53 confirmed that the SCIM bearer token generation routine in Passbolt combines cryptographically secure randomness with the Mersenne Twister pseudorandom number generator, which undermines the entropy of the generated token. While the implementation correctly applies *random_bytes()* to generate 3600 bytes of cryptographically secure random data, it subsequently selects a 43-character substring using *mt_rand()* for offset selection. The Mersenne Twister PRNG is deterministic and unsuitable for cryptographic purposes, as its internal state can be reconstructed from as few as 650 observed inputs.

Affected file:
plugins/PassboltEe/Scim/src/Service/ScimSetSettingsService.php

Affected code:
```
public static function generateToken(): string
{
    $prefix = self::SCIM\_SECRET\_TOKEN\_PREFIX;
    $token = preg\_replace('/\[^a-zA-Z0-9\]/', '', base64\_encode(random\
_bytes(3600)));

    return $prefix . substr($token, mt\_rand(0, strlen($token) - 43), 43);
}
```

The entropy reduction can be quantified as follows: the original 3600 bytes from *random_bytes()* provides approximately 28,800 bits of entropy. After base64 encoding and filtering, approximately 4,800 alphanumeric characters remain. *mt_rand()* selects an offset within this string, although Mersenne Twister only provides approximately 31 bits of effective randomness. Consequently, the final token's entropy is bounded by this weakest link in the chain.

This weakness is challenging to directly exploit in isolation. However, attackers could combine this situation with other flaws, such as the timing leak outlined in PBL-15-001 and the unsalted hashing vector in PBL-15-004. The cumulative effect of these approaches in tandem significantly reduces the computational effort required to recover a valid token through brute force or statistical analysis.

To mitigate this issue, Cure53 recommends replacing *mt_rand()* with *random_int()*, which provides cryptographically secure integer generation.

[1] https://www.php.net/manual/en/function.hash-equals.php

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

PBL-15-003 WP3: Lack of bearer token expiry & revocation schemes *(Medium)*

To authenticate an external Identity Provider such as Azure AD for SCIM operations, the Passbolt Service Provider issues a single long-lived bearer token to the IdP. However, testing verified the absence of expiration, rotation, and revocation mechanisms for bearer tokens issued to IdPs, as tokens are statically generated. Tokens remain valid upon issuance until manually modified, which is not ideal for a production-grade system. If the IdP or Service Provider is compromised, the TLS connection between the IdP and SP is breached, or the token value is leaked via a side channel, manual intervention to replace the token is required.

To mitigate this issue, Cure53 recommends implementing OAuth 2.0 as the authentication framework between the selected IdP and Passbolt Service Provider, as recommended by the SCIM Protocol RFC.[2] While Azure AD does not support OAuth 2.0 without applying for a listing on Microsoft's dedicated gallery,[3] other IdPs support OAuth 2.0 by default, such as Okta.[4] Adopting OAuth 2.0 will provide several guarantees over long-lived static bearer tokens, including short-lived authentication windows; automated credential rotation; scoped accesses; and credential separation between the IdP and SP. Furthermore, OAuth 2.0 is a well-understood and supported standard, for which token requests create a discrete, loggable, and auditable trail of events.

PBL-15-004 WP1: Unsalted SHA256 hashing of bearer tokens *(Medium)*

Cure53 determined that the SCIM bearer token utilized by external IdPs to authenticate directory synchronization requests is stored using unsalted SHA256 hashing. This credential grants full SCIM API access, which in turn enables user creation, modification, and deletion within the Passbolt organization. The absence of a per-token salt and the enforcement of a fast hash function (rather than a password-specific algorithm) allow attackers in possession of the stored hash to feasibly recover the plaintext token via precomputation or brute force attacks.

Affected file:
plugins/PassboltEe/Scim/src/Form/Settings/ScimSettingsForm.php

Affected code:

```
protected function \_execute(array $data): bool
{
    if (isset($data\['secret\_token'\])) {
       $this->\_data\['secret\_token'\] = Security::hash($data\['secret\
_token'\], 'sha256');
    }

    return true; }
```

[2] https://datatracker.ietf.org/doc/html/rfc7644#section-2
[3] https://github.com/MicrosoftDocs/entra-docs/blob/main/docs/identity/enterprise-apps/v2-[...]-listing.md
[4] https://developer.okta.com/docs/guides/scim-provisioning-integration-prepare/main/

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

The overall risk is compounded by the persistence of the fault outlined in PBL-15-002, concerning weak entropy in token generation due to use of *mt_rand()*. When leveraged in tandem, the effective token search space is reduced from 2^{256} to approximately 2^{31} possibilities. A modern GPU can compute approximately 10 billion SHA256 hashes per second; given this reduced search space, exhaustive hashing completes in under 1 second.

An adversary in possession of the GPG-encrypted settings blob (via a database breach, backup exposure, or insider access) may be able to compromise the server's GPG private key and decrypt the settings to retrieve the SHA256 hash. Subsequently, a rainbow table lookup or brute-force attack can be mounted to recover the plaintext bearer token, which furnishes full SCIM API access for user creation, modification, and deletion.

To mitigate this issue, Cure53 recommends replacing the unsalted SHA256 hash with Argon2id, a memory-hard password hashing algorithm that incorporates per-token salting and configurable computational cost.

PBL-15-005 WP2: Race condition in user creation *(High)*

Cure53 ascertained that the SCIM user creation endpoint contains a Time-of-Check to Time-of-Use (TOCTOU) race condition that permits duplicate user creation. The *create()* method performs a uniqueness check by querying for an existing user with the same email address, then proceeds to create the user in a separate database operation. A concurrent request can pass the same uniqueness check between these two operations, resulting in the creation of duplicate users with identical email addresses.

Affected file:
plugins/PassboltEe/Scim/src/Utility/Resource/UserScimResource.php

Affected code:

```
public function create(): static
{
    [...]
    $user = $this->findExistingUserEntity();   // CHECK
    if (!empty($user->scim_entry)) {
        throw new ConflictException(
            sprintf('The %s resource with userName `%s` already exist with
id `%s`',
                $this->getType(), $this->userName, $user->id),
            scimType: ScimException::SCIM_TYPE_UNIQUENESS,
        );
    }
    [...]
    $user = $this->Users->register([...], $uac);   // USE
    [...]
}
```

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Steps to reproduce:

1. Send two concurrent POST requests to */scimv2/{settingId}/Users* with identical email addresses.
2. Note that both requests pass the uniqueness check before either completes user creation.
3. Observe that two users have been created with the same email address.

The race window originates from the non-atomic nature of the uniqueness check (line 301) and user creation (line 325). In a concurrent environment, the following sequence can occur:

- T1: Thread A checks *findExistingUserEntity() = null*
- T2: Thread B checks *findExistingUserEntity() = null*
- T3: Thread A creates User (ID: *uuid-1*)
- T4: Thread B creates User (ID: *uuid-2*), which is a duplicate.

Successful exploitation of this vulnerability can cause SCIM synchronization corruption, validation bypassing, and inconsistent state between the IdP and Passbolt.

To mitigate this issue, Cure53 recommends wrapping the entire user creation operation in a database transaction with a *SELECT...FOR UPDATE* lock on the email address, ensuring atomicity between the uniqueness check and user insertion. Alternatively, uniqueness should be enforced at the database level with a unique constraint on the email column that rejects duplicate insertions.

PBL-15-006 WP2: User enumeration via error messages *(Low)*

Cure53 observed that the SCIM user creation endpoint discloses whether a user's email address exists in the system, as well as their internal UUID identifier. When attempting to create a user with an email address that already exists, the error response reveals both the existence of the user and their Passbolt internal identifier.

Affected file:
plugins/PassboltEe/Scim/src/Utility/Resource/UserScimResource.php

Affected code:

```
if (!empty($user->scim\_entry)) {
    throw new ConflictException(
        sprintf(
            'The %s resource with userName `%s` already exist with id `
                    %s`',
            $this->getType(),
            $this->userName,
            $user->id
        ),
```

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

```
            scimType: ScimException::SCIM_TYPE_UNIQUENESS,
        );
    }
```

An attacker with access to a valid SCIM bearer token can enumerate users by iterating through email addresses and observing any that return a conflict error. The disclosed information includes the user's system existence, internal UUID, and *userName* (SCIM attribute).

While a valid bearer token is required to perform this enumeration, the information disclosure enables reconnaissance that could facilitate further attacks, such as targeted phishing or social engineering against confirmed users.

To mitigate this issue, Cure53 recommends returning a generic conflict error message that obfuscates user internal identifiers.

PBL-15-007 WP5: Potential DoS via pre-authentication GPG decryption *(Low)*

Cure53 noticed a potential Denial-of-Service (DoS) vulnerability in the authentication logic of Passbolt's Service Provider SCIM plugin. The *find* function in *ScimResolver* executes a resource-intensive GPG decryption operation on the *ScimSettings* prior to verifying the validity of the authentication token provided by the IdP. Accordingly, an unauthenticated attacker can exhaust the Passbolt Service Provider's resources by flooding the SCIM endpoint with requests containing invalid tokens.

Affected file:
Identifier/Resolver/ScimResolver.php

Affected code:

```
public function find(array $conditions, string $type = self::TYPE_AND):
ArrayAccess|array|null
{
    [...]

    $scimConfig = (new ScimGetSettingsService())-
       >getSettingsDecryptedValue();
    if ($scimConfig['setting_id'] !== Configure::read('Scim.settingId')) {
        return null;
    }

    if (
        $conditions['secret_token'] === $scimConfig['secret_token'] &&
        !empty($scimConfig['scim_user_id'])
    ) {
        [...]
    }
```

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

```
    return null;
}
```

To mitigate this issue, Cure53 recommends integrating a caching mechanism for the decrypted *ScimSettings* schema. The *ScimSettings secret_token* field should be stored using a cryptographic hash such as Sha256. This revised approach will ensure that the application avoids expensive cryptographic operations for unauthenticated requests.

PBL-15-008 WP4: *ScimEntry* uniqueness race condition *(Medium)*

Testing confirmed that the *ScimEntriesTable* uniqueness check for SCIM names is susceptible to a TOCTOU race condition. Specifically, the *isUniqueScimName()* method queries the database to verify uniqueness, then returns control to the caller who subsequently performs the *INSERT* operation. This allows concurrent requests to pass the uniqueness check simultaneously before either completes insertion.

Affected file:
plugins/PassboltEe/Scim/src/Model/Table/ScimEntriesTable.php

Affected code:

```
public function isUniqueScimName(
    string $scimName,
    ?string $excludeId = null
): bool {
    $conditions = [$this->aliasField('scim_name') => $scimName];
    if ($excludeId) {
        $conditions[$this->aliasField('id') . ' !='] = $excludeId;
    }
    $exist = $this->find()
        ->where($conditions)
        ->whereNull($this->aliasField('deleted'))
        ->all()->count() > 0;

    return !$exist;
}
```

Steps to reproduce:

1. Send two concurrent POST requests to the SCIM Users endpoint with identical *userName* values.
2. Note that both requests invoke *isUniqueScimName()*, returning *true* each time, and proceed to *INSERT* their *ScimEntry* records.
3. Observe duplicate SCIM entries with the same *scim_name* value.

As such, attackers can bypass SCIM naming uniqueness constraints, facilitating synchronization corruption between the IdP and Passbolt.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Duplicate entries can ultimately lead to unpredictable behavior during subsequent sync operations. To mitigate this issue, Cure53 recommends incorporating a database-level *UNIQUE* constraint on the *scim_name* column and handling constraint violations gracefully in the application layer. Alternatively, *SELECT...FOR UPDATE* could be utilized within a transaction to serialize access.

PBL-15-009 WP4: Non-transactional group member operations *(Low)*

Cure53's assessment confirmed that group member addition operations in the *DirectorySync* plugin are not wrapped in database transactions. The *addGroupUsers()* method initially creates a *GroupUser* record via the groups service, then formulates the corresponding *DirectoryRelation* record. If the second operation fails after the first succeeds, the database is forced into an inconsistent state with orphaned group memberships. Hence, unintended privilege escalation could be enabled upon halfway-through-operation failure.

Affected file:
plugins/PassboltEe/DirectorySync/src/Actions/GroupSyncAction.php

Affected code:

```
private function addGroupUsers(Group $group, array $toAdd): void
{
    foreach ($toAdd as $userId => $role) {
        $data = $this->groupsUsersAddService->add($groupsUsersData);
        // ...
        $this→DirectoryRelations-
              >createFromGroupUser($data['group_user']);
        // No transaction wrapper
    }
}
```

Steps to reproduce:

1. Initiate a directory sync that adds users to a group.
2. Interrupt the process after *GroupUser* creation but before *DirectoryRelation* creation.
3. Observe that *GroupUser* records are orphaned without corresponding *DirectoryRelation* entries. Subsequent sync operations may fail to detect or correct the inconsistency.

In the aforementioned scenario, a user would retain group privileges (including admin access) even when the sync state indicates that they should be removed, since the cleanup logic relies on *DirectoryRelation* records to identify sync-managed memberships.

To mitigate this issue, Cure53 recommends wrapping the entire *addGroupUsers()* loop in a database transaction. Both *GroupUser* and *DirectoryRelation* creation must succeed atomically; otherwise, the entire operation should roll back.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

PBL-15-010 WP4: Directory entry foreign key race condition *(Low)*

Cure53 discovered that *DirectoryEntriesTable* implements a read-check-write pattern without sufficient locking, allowing concurrent updates to overwrite each other. The *updateOrCreate()* method reads an entry, checks if the *directory_name* has altered, then writes the update. Resultingly, a concurrent request performing the same sequence can lead to lost updates.

In essence, this situation causes directory synchronization state corruption. Due to lost updates, the Passbolt database will no longer accurately reflect the IdP's state, fostering incorrect user provisioning or deprovisioning decisions.

Affected file:
plugins/PassboltEe/DirectorySync/src/Model/Table/DirectoryEntriesTable.php

Affected code:

```
public function updateOrCreate(array $data, string $model): array|bool|
DirectoryEntry
{
    try {
        $entry = $this->get($data['id'], contain: [$model]);
        if (is_string($data['directory_name']) &&
            $entry->directory_name !== $data['directory_name']) {
            $entry->directory_name = $data['directory_name'];
            $this->save($entry);
        }
    // ...
}
```

Steps to reproduce:

1. Initiate two concurrent directory sync operations that update the same entry.
2. Thread A reads an entry with *directory_name = “dn1”.*
3. Thread B reads an entry with *directory_name = “dn1”.*
4. Thread A writes *directory_name = “dn2”.*
5. Thread B writes *directory_name = “dn3”*, overwriting Thread A’s update.
6. Observe that Thread A’s update is silently lost.

To mitigate this issue, Cure53 recommends leveraging *SELECT...FOR UPDATE* to lock the row during the read-check-write sequence. Alternatively, optimistic locking should be utilized with a version column that is checked during the update operation.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

PBL-15-011 WP4: Lack of transaction wrapper in production sync *(Low)*

Cure53 noticed that the DirectorySync plugin wraps sync operations in a database transaction for dry-run mode, but not for production mode. The *execute()* method in SyncAction explicitly initiates a transaction and rolls back for dry-run operations, yet executes production sync operations without any transaction wrapper. If the sync process fails midway, the database encounters a partially-synchronized state, which could facilitate user database corruption.

Affected file:
plugins/PassboltEe/DirectorySync/src/Actions/SyncAction.php

Affected code:

```
public function execute(): ActionReportCollection
{
    if ($this->isDryRun()) {
        $conn = $this->Users->getConnection();
        $conn->begin();
        $conn->transactional(function (): void {
            $this->_execute();
        });
        $conn->rollback();
    } else {
        $this->_execute();  // No transaction
    }
}

protected function _execute(): void
{
    $this->beforeExecute();
    $this->processEntriesToDelete();  // No transaction
    $this->processEntriesToCreate();  // No transaction
    $this->afterExecute();
}
```

Steps to reproduce:

1. Initiate a production directory sync that deletes multiple users.
2. Wait until several (but not all) deletions have completed, then interrupt the process.
3. Observe that the database is forced into a partially-synchronized state and all subsequent sync operations face an inconsistent starting state.

Pertinently, automatic rollback will not occur if 5 out of 10 users are deleted prior to failure. The system requires manual intervention to restore consistency, while subsequent sync operations may produce unpredictable results.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

To mitigate this issue, Cure53 recommends wrapping the *_execute()* method in a transaction for production mode (i.e., identical to the dry-run implementation but without the rollback). This will guarantee that atomic sync operations either comprehensively complete or roll back entirely.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

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, while a vulnerability is present, an exploit may not always be possible.

PBL-15-012 WP1: Potential admin lockout via malicious IdP request *(Low)*

The SCIM Identity Provider that the Passbolt Service Provider interfaces with may arbitrarily update the “active” status of any mapped user, which in turn is mapped onto the internal *disabled* field in the database. However, guard clauses have not been established to prevent the IdP from disabling critical users, such as the instance administrator or user account tied to the SCIM configuration. Owing to this premise, a malicious IdP or attacker that can successfully Man-in-the-Middle the TLS connection between the Passbolt SP and IdP can disable administrative and configuration users, enabling admin lockouts and DoS conditions. In this scenario, manual intervention would be required to restore service.

Affected file:
plugins/PassboltEe/Scim/src/Utility/Resource/UserScimResource.php

Affected code:

```
public function setFromScim(array $data): static
{
    $this->validateScimUserData($data);

    $this->externalId = $data['externalId'] ?? null;
    $this->userName = $data['userName'] ?? null;
    $this->firstName = $data['name']['givenName'] ?? null;
    $this->lastName = $data['name']['familyName'] ?? null;
    $this->middleName = $data['name']['middleName'] ?? null;
    if (isset($data['active'])) {
        $this->active = (bool)$data['active'];
    }
    $emails = Hash::extract($data, 'emails.{n}[type=work].value');
    $this->email = $emails[0] ?? null;

    return $this;
}
```

To mitigate this issue, Cure53 recommends modifying the *updateDatabaseUser* function in the *UserScimResource* to reject certain *active: false* updates if either the previous administrator or user tied to the SCIM configuration (denoted by the database’s *scim_user_id* field) are disabled. As a result, any malicious requests originating from the IdP will be blocked, preventing critical user disabling and safeguarding system availability.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Conclusions

For this Jan-Feb 2026 engagement, Cure53 performed a white-box penetration test and source code audit focusing on Passbolt's SCIM plugin, Azure AD integration, and directory synchronization components. A team of three skill-matched assessors were designated fifteen to examine five dedicated WPs. Namely, the SCIM Admin UI, SCIM API endpoints, token authentication mechanisms, plugin synchronization logic, and cryptographic implementations were all extensively scrutinized. Twelve security flaws were detected, of which one received a *High* severity rating, four were tagged as *Medium*, and seven were filed as *Low*.

The most significant discovery entails a TOCTOU race condition in the SCIM user creation endpoint (PBL-15-005), which permits duplicate user creation via concurrent requests. This vector stems from the separation between the uniqueness check and subsequent database insertion, a pattern that appears consistently throughout the codebase. Similar race conditions were identified in the *ScimEntry* uniqueness validation (PBL-15-008), directory entry updates (PBL-15-010), and group membership operations (PBL-15-009).

In addition, the production directory sync operation lacks the transaction wrapper that is correctly implemented for dry-run mode (PBL-15-011). The in-house team should systematically analyze all database operations to ensure atomicity through proper transaction handling, *SELECT...FOR UPDATE* locking, and database-level uniqueness constraints.

The bearer token authentication scheme presents several interrelated faults. The token generation routine combines cryptographically secure randomness with the deterministic Mersenne Twister PRNG, thus reducing effective entropy (PBL-15-002). Moreover, tokens are stored using unsalted SHA256 hashing (PBL-15-004) and verification adopts non-constant-time string comparison (PBL-15-001). While the individual pitfalls pose limited risk, simultaneous exploitation significantly reduces the computational effort required to recover a valid token. Furthermore, the static nature of these tokens - in which expiration, rotation, and revocation mechanisms are absent (PBL-15-003) - means that compromised credentials will remain valid indefinitely.

In light of this, Cure53 recommends adopting OAuth 2.0 for IdP authentication where supported. For Azure AD integration specifically, optimal cryptographic practices should be conformed to, such as constant-time comparison via *hash_equals()*, memory-hard hashing via Argon2id, and exclusive use of *random_int()* for cryptographic operations.

The test team’s inspection of the SCIM Admin UI and Azure AD integration revealed that the Service Provider fails to validate requests from the IdP against critical user accounts. A malicious or compromised IdP can disable administrative or SCIM configuration users, inducing administrative lockout (PBL-15-012).

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Additionally, the SCIM configuration decryption occurs prior to token verification, exposing a potential DoS condition via repeated authentication attempts with invalid tokens (PBL-15-007). The user creation endpoint also discloses internal user identifiers in error messages (PBL-15-006), facilitating reconnaissance by authenticated attackers.

Despite these limitations, Cure53 is pleased to confirm that the encompassing security posture of the Passbolt SCIM plugin is performant. The codebase is astutely organized, idiomatic, and legible. The identified behaviors primarily relate to common pitfalls in concurrent systems and the constraints imposed by Azure AD's limited OAuth 2.0 support, rather than fundamental architectural errors. The development team should find ameliorating each situation straightforward to achieve. Cure53 recommends prioritizing the race condition fixes and cryptographic improvements, as these address the most impactful findings and will serve as a cornerstone for secure development patterns moving forward.

Cure53 would like to thank Remy Bertot from the Passbolt SA team for his excellent project coordination, support, and assistance, both before and during this assignment.