Intelsentrix

Example Company Ltd.

Penetration Testing Report

Web, API, mobile application, code and network security assessment

PREPARED BY
Mantas Sabeckis, MB "Otterly"

ASSESSMENT PERIOD
2026-06-01 - 2026-06-08

REPORT DATE
2026-06-08

Table of Contents

Notice and contacts

Notice

Information in this document should be treated as confidential when used in a real project, because it may include system architecture details, vulnerabilities and reproduction steps.

A security assessment is intended for a specific point in time. Findings and recommendations reflect the information collected during the assessment, the agreed scope and the access available at the time.

All testing activities in a real project must be performed only after explicit client approval, agreed testing windows and confirmed scope.

Contacts

Name	Role	Contact
Mantas Sabeckis	Security testing lead	info@otterly.lt
John Smith	Chief Executive Officer	john.smith@example.com
Jane Smith	Technical Lead	jane.smith@example.com

Results summary

Overview

Example Company Ltd. engaged Mantas Sabeckis, MB "Otterly", to perform a security assessment of web applications, APIs, mobile applications, source code and network assets. The goal was to test the systems in scope from a realistic attacker perspective and determine whether unauthorized access to systems, data or sensitive functionality could be obtained.

The assessment used a mixed approach: limited-knowledge, target-based testing for web, API, mobile and network areas, and whitebox review for code with agreed repository or source file access.

The results show how systems in the agreed scope could be affected by security weaknesses. Findings are classified by technical exploitability, business impact and remediation priority.

Scope of work

Area	Scope	Notes
Web	1 customer portal, 1 CRM system, 1 administration panel	Blackbox and authenticated testing
API	1 REST API, 42 endpoints, 4 user roles	OWASP API Security Top 10 scenarios
Mobile	1 iOS and 1 Android application	Static and dynamic application analysis
Code	3 code repositories: CRM, Dashboard, admin	Whitebox review within agreed code scope
Network	10 external IP addresses, 23 internal IP addresses	External and internal network node testing

Tools used

Category	Tools
Web / API	Burp Suite Professional, Acunetix, ffuf, sqlmap, custom scripts
Network	Nmap, nuclei and Nessus
Mobile	MobSF, Frida, apktool, proxy tools
Code	Semgrep, Gitleaks, Trufflehog

Result assessment methodology

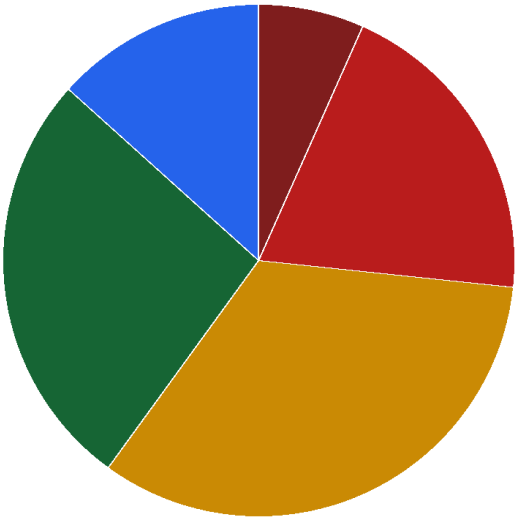
Findings are assessed and prioritized based on context: technical exploitability, business impact, data sensitivity, reachability, available access and the likelihood of a realistic attack scenario. CVSS is used as a starting point, while final priority is adjusted based on business context and practical exploitability.

Risk	Description
Critical	The finding represents direct and immediate risk to critical systems, sensitive data, infrastructure, code or business operations. Exploitation may result in complete compromise, significant data loss, business disruption or reputational damage.
High	The finding may result in unauthorized access to systems, applications, APIs, network nodes, code artifacts or confidential data. These findings should be addressed with high priority.
Medium	The finding may not provide full compromise on its own, but can be combined with other weaknesses or misconfigurations to increase risk. These findings should be fixed in a timely manner.
Low	The finding has limited direct impact, but can expose technical, configuration or process weaknesses that should be improved after higher-priority findings.
Informational	The finding is primarily a recommendation to improve system, code, network or process security maturity. It is usually not a standalone vulnerability.

Findings summary

By severity

Total: 15



Critical	1 (7%)
High	3 (20%)
Medium	5 (33%)
Low	4 (27%)
Informational	2 (13%)

By area

Total: 15

API	3 (20%)
WEB	3 (20%)
MOBILE	3 (20%)
CODE	3 (20%)
NETWORK	3 (20%)

Security risks

- Unauthorized access to other users' or organizations' data due to access control weaknesses.
- Possible sensitive data access or database service disruption through injection vulnerabilities.
- Increased initial access risk through vulnerable VPN services or broadly reachable administration interfaces.
- Session, token or login exposure due to weak session management and mobile storage weaknesses.
- Code and configuration issues may expose secret keys or sensitive functionality.

High-level recommendations

- Strengthen server-side authorization and object ownership checks.
- Fix injection risks by validating input and using safe query construction practices.
- Reduce the external attack surface by limiting administrative service exposure and updating vulnerable services.
- Strengthen session, token and login protection using secure storage, rotation and server-side invalidation.
- Improve code and configuration management by removing secrets from code and using secure secret storage.

Conclusion

The overall assessment risk is critical because the SQL injection finding may lead to sensitive data access or database service disruption, while other high and medium findings increase the risk of unauthorized access, sensitive function abuse and further attack scenarios. The highest-priority findings should be fixed first and retested after remediation.

Detailed findings

F-01 [API] Time-based SQL injection in API search

Affected area: Dashboard API, /api/v1/search

Risk: Critical

CVSS score: 8.8

Summary

The API accepts a user-controlled search parameter and passes it into a database query without sufficient parameterization. During testing, a time-based SQL injection was confirmed using a safe test payload.

Impact

An attacker may attempt to read sensitive data, enumerate database structure or create additional application load. The real impact depends on database user privileges and additional control mechanisms.

Reproduction steps

1. Log in to the test API environment with a regular user account.
2. Send a POST request to the search endpoint with a specially crafted parameter.

```
POST /api/v1/search
{
  "query": "test' AND 1=(SELECT 1 FROM pg_sleep(5))--",
  "page": 1
}
```

3. Compare a normal request with a delayed response request.
4. Confirm that the response time increases according to the injected delay, indicating SQL query execution.

Recommendation

Rewrite all database queries using parameterized queries or safe ORM parameters. Also restrict input format, review similar endpoints and add regression tests.

F-02 [WEB] Authorization bypass in an administrative function

Affected area: Administration panel

Risk: High

CVSS score: 8.8

Summary

The administrative function verifies that the user is authenticated, but does not sufficiently verify the required role or permission. As a result, a lower-privileged user can access actions that should be limited to administrators.

Affected locations

- PATCH /admin/users/{user_id}/role
- POST /admin/users/{user_id}/disable
- DELETE /admin/users/{user_id}
- POST /admin/organizations/{organization_id}/settings
- GET /admin/audit-log/export

Impact

Data modification, user management, configuration changes or other business-sensitive actions may be possible without appropriate authorization.

Reproduction steps

1. Log in as a user with administrator privileges.
2. Capture an HTTP request for an administrative action.

```
PATCH /admin/users/{user_id}/role
Authorization: Bearer <low_privileged_user_token>

{
  "role": "admin"
}
```

3. Replay the request using a lower-privileged user session.
4. Confirm that the system allows the action despite the missing role.

Recommendation

Verify both authentication and specific server-side permissions for every sensitive action. Introduce a centralized authorization layer and role-based negative tests.

F-03 [NETWORK] Publicly exposed VPN service with a known vulnerability

Affected area: VPN Gateway, vpn.example.lt:443

Risk: High

CVSS score: 8.8

Summary

An externally exposed VPN service was identified with a version associated with a publicly known high-risk vulnerability. The service is reachable from the internet and does not have additional IP restrictions.

Impact

If successfully exploited, the vulnerability could provide initial access or create conditions for lateral movement inside the network. The risk depends on the exact exploit conditions and additional safeguards.

Reproduction steps

1. Scan the external perimeter for open ports and services.
2. Identify the VPN product and version.

```
vpn.example.lt:443
Service: ExampleVPN Gateway
Detected version: ExampleVPN Gateway 8.2.1
```

3. Compare the detected version against public CVE databases and vendor security advisories.
4. Verify whether the service is reachable without an IP allowlist or additional protection.

Recommendation

Update the VPN component, restrict access by IP, enable MFA, review recent logins and check for signs of compromise.

F-04 [API] Broken access control allows access to another user's records

Affected area: User Documents API, /api/v1/users/{id}/documents

Risk: High

CVSS score: 8.1

Summary

API endpoints allow object identifiers to be changed and do not always verify whether the object belongs to the authenticated user. This matches an IDOR / broken access control scenario.

Impact

An attacker may access or modify other users' records, orders, files, payment information or other sensitive objects.

Reproduction steps

1. Log in as user A and obtain the identifier of an owned object.
2. Log in as user B or use another known object identifier.

```
GET /api/v1/users/{other_user_id}/documents
Authorization: Bearer <user_a_token>
```

3. Change the object ID in the request and observe whether the API returns or modifies data belonging to another user.

Recommendation

Verify object ownership and permissions on the server side for every endpoint action. Separate global object IDs from user-accessible object lists and add negative authorization tests.

F-05 [WEB] Unsafe file upload allows dangerous files to be stored

Affected area: File upload functions

Risk: Medium

CVSS score: 6.1

Summary

The file upload function does not sufficiently validate file type, content and filename. Some upload endpoints allow dangerous extensions or path manipulation.

Affected locations

- POST /api/v1/files/upload
- POST /api/v1/profile/avatar
- POST /api/v1/messages/{message_id}/attachments
- PUT /api/v1/files/{file_id}
- GET /uploads/{filename}

Impact

The system may be used for unwanted file hosting, phishing scenarios, storage pollution or limited path manipulation if additional controls are missing.

Reproduction steps

1. Submit a file with a dangerous extension to the upload function.
2. Change the filename to a value containing special characters or path sequences.

```
POST /api/v1/files/upload
filename=../sample.html
content-type=text/html
```

3. Verify whether the server accepts the file and returns a publicly accessible link.

Recommendation

Use an allowlist for file extensions and MIME types, validate file content, generate safe filenames, store files outside the web root and restrict public access.

F-06 [WEB] Stored XSS in profile URL field

Affected area: Profile URL field and administration review page

Risk: Medium

CVSS score: 6.1

Summary

The server allows a profile URL value using the javascript scheme, and the administration review page renders this value as an active link without sufficient server-side validation. During testing, a stored XSS scenario was confirmed in the administration panel.

Impact

An attacker can save a malicious profile URL that executes when an administrator views the user profile. Depending on session controls and browser protections, this may allow actions in the administrator's browser context or exposure of sensitive page data.

Reproduction steps

1. Log in as a regular user and update the profile URL field through the API.
2. Submit a javascript-scheme value in the profile URL field.

```
PATCH /api/v1/profile
{
  "website_url": "javascript:alert(document.domain)"
}
```

3. Log in to the administration panel and open the user profile review page.
4. Confirm that the submitted JavaScript code executes in the browser.

Recommendation

Validate URLs on the server side and only allow explicitly defined schemes such as HTTPS. In the administration panel, render all user-provided values using context-appropriate escaping and never use dangerous schemes for active links.

F-07 [CODE] AWS access key found in code repository

Affected area: Backend repository

Risk: Medium

CVSS score: 5.9

Summary

An AWS Access Key ID starting with the AKIA prefix was found in the code repository. The key's activity and permissions were not validated during testing, but this type of value in code indicates a potential cloud access key leak.

Impact

If the associated secret key is active and has meaningful permissions, an attacker could access AWS resources, read or modify data, create additional resources or cause financial damage. The risk is assessed as medium because the key was found in a private repository, not in publicly accessible code. Real impact depends on key activity, IAM permissions and additional AWS controls.

Reproduction steps

1. Review the repository and git history with secrets detection tools.
2. Identify an AWS Access Key ID value starting with the AKIA prefix.

```
AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
```

3. Check which file and commit introduced the value.
4. Check whether the same value is used in CI/CD configuration or environment files.

Recommendation

Immediately check whether the key is active in AWS IAM, rotate or remove the affected key, review its CloudTrail usage history and remove the value from the repository and its history. Add pre-commit and CI secrets checks.

F-08 [MOBILE] Session data stored in external device storage

Affected area: Mobile application SD card storage

Risk: Medium

CVSS score: 5.5

Summary

The mobile application stores session data in external device storage, which may be more broadly accessible than private application storage. During testing, the session file was found in an SD card style storage path.

Impact

If the device is compromised, connected to an untrusted environment or another application has access to external storage, session data may be read and used for further account analysis or takeover attempts.

Reproduction steps

1. Log in to the mobile application on a test device.
2. Inspect external storage directories and files created by the application.

```
/sdcard/Android/data/com.example.app/files/session.json
{
  "session_token": "eyJhbGciOiJIUzI1NiIs..."
}
```

3. Identify session data stored in an SD card style storage location.

Recommendation

Store session data only in private application storage or the platform secure storage. Do not use external SD card style storage for sensitive data and clear session files on logout.

F-09 [CODE] Session is not invalidated server-side after logout

Affected area: AuthService logout function

Risk: Medium

CVSS score: 5.3

Summary

The logout function removes the client-side cookie or token, but the server-side session remains active until its configured expiry time.

Impact

If the session token is stolen, an attacker may continue acting as the authenticated user even after the user logs out.

Reproduction steps

1. Log in and save the session cookie or token value.
2. Log out using the application logout function.

```
GET /api/v1/me
Cookie: session=<old_session_after_logout>
```

3. Replay an authenticated API request using the old session value.
4. Confirm that the server still accepts the old session.

Recommendation

Invalidate the session server-side during logout, shorten session lifetime, implement refresh token rotation and provide a way to revoke compromised sessions.

F-10 [API] Insufficient request throttling on sensitive endpoints

Affected area: Auth API, /login and /verify-code

Risk: Low

CVSS score: 3.5

Summary

Several API endpoints accept many repeated requests from the same session or IP address without an effective rate limiting mechanism.

Impact

The finding does not provide unauthorized access by itself, but it can make automated attempts, one-time-code guessing, email abuse or additional infrastructure load easier.

Reproduction steps

1. Select a sensitive endpoint, for example login or code validation.
2. Send a series of requests in a short period of time.
3. Check whether the system starts throttling attempts or returns a protection response.

Recommendation

Implement IP, account and endpoint-level throttling, progressive cooldown, audit logs and anomaly alerts.

F-11 [CODE] Passwords stored in plaintext

Affected area: AuthService legacy import function

Risk: Low

CVSS score: 3.3

Summary

Code review identified that the legacy import function temporarily stores initial passwords in plaintext. The function is not directly exposed to the internet, but this practice increases risk in case of an incident, misconfiguration or backup leak.

Impact

If these values are stored in the production database, logs or backups, a limited number of users could be affected. The risk is assessed as low because of the limited scope and additional conditions required for exploitation.

Reproduction steps

1. Review the legacy user import function and its related data model.
2. Check whether the initial password is processed through a secure hashing mechanism before storage.

```
legacy_import.initial_password = "DemoPassword123"
```

3. Confirm that the password value remains readable in the import table.

Recommendation

Remove the plaintext password field, generate initial passwords only for one-time delivery, and process every stored password with bcrypt, Argon2id or another password-specific algorithm.

F-12 [NETWORK] Exposed administrative service without additional access control

Affected area: Administration portal, admin.example.lt

Risk: Low

CVSS score: 3.1

Summary

An administrative login page is reachable from the public internet. No authentication bypass or other direct exploitation scenario was confirmed during testing, but this exposure increases the external attack surface.

Impact

The finding does not provide system access on its own, but the administrative interface can become an additional target for password guessing, social engineering or future product vulnerabilities.

Reproduction steps

1. Scan subdomains and ports.
2. Identify the administration panel based on HTTP responses and page title.

```
https://admin.example.lt/login
```

3. Verify whether the service is reachable from the public internet without IP restrictions.

Recommendation

Move the administrative service behind VPN or IP allowlisting, enable MFA, limit login attempts and monitor anomalous logins.

F-13 [MOBILE] Debug features left in the production mobile application build

Affected area: Mobile application release build

Risk: Low

CVSS score: 3.1

Summary

Debug settings or diagnostic features that should not be active in the final build were found in the production application version.

Impact

This may make application analysis, information gathering or internal logic understanding easier, especially when combined with other mobile application weaknesses.

Reproduction steps

1. Obtain the production APK file for the application version in scope.
2. Unpack the APK and review the AndroidManifest.xml file.

```
$ apktool d app-release.apk -o app-release
$ grep -n 'android:debuggable' app-release/AndroidManifest.xml
<application android:debuggable="true" ...>
```

3. Confirm that the application element has android:debuggable set to true.

Recommendation

Disable debug settings in production builds, separate debug and release configurations, and verify build security parameters during CI/CD.

F-14 [MOBILE] Missing certificate pinning for sensitive API communication

Affected area: Mobile API communication, api.example.lt

Risk: Informational

Summary

The mobile application trusts the device certificate store and does not use certificate pinning for sensitive API requests. In the test environment, traffic could be intercepted and analyzed using a user-installed certificate.

Impact

This is a recommendation for additional protection of sensitive mobile API communication. The finding does not provide system access on its own and normally requires additional conditions, so it is assessed as informational.

Reproduction steps

1. Install a proxy certificate on the test device.
2. Route application traffic through the testing proxy.
3. Verify whether the application continues to execute API requests through the intermediary proxy.

Recommendation

Implement certificate pinning for sensitive API domains, define a safe pin rotation process and test that pinning cannot be disabled by debug-build logic in production.

F-15 [NETWORK] TLS configuration gaps in external services

Affected area: Public web and API services

Risk: Informational

Summary

Some external services support outdated TLS settings or do not consistently apply security headers.

Impact

This was not a directly exploitable finding in this assessment, but standardizing configuration would improve overall transport layer security and reduce risk in specific scenarios requiring additional conditions.

Reproduction steps

1. Scan TLS configuration and security headers for external domains.
2. Compare results against the organization's standard and public best practices.
3. Check whether gaps repeat across multiple services.

Recommendation

Standardize the TLS policy, enable HSTS, disable outdated protocols and cipher suites, and verify changes across all environments.

Remediation and retesting report

This section is used for remediation planning and retesting. It shows finding priority, remediation status and shared risks that can be fixed as common technical problems.

Remediation work summary

ID	Type	Risk	CVSS	Prior.	Status	Retest
F-01	API	**Critical**	8.8	**P1**	Open	Not started
F-02	Web	**High**	8.8	**P2**	Open	Not started
F-03	Network	**High**	8.8	**P2**	Open	Not started
F-04	API	**High**	8.1	**P2**	Open	Not started
F-05	Web	**Medium**	6.1	**P3**	Open	Not started
F-06	Web	**Medium**	6.1	**P3**	Open	Not started
F-07	Code	**Medium**	5.9	**P3**	Open	Not started
F-08	Mobile	**Medium**	5.5	**P3**	Open	Not started
F-09	Code	**Medium**	5.3	**P3**	Open	Not started
F-10	API	**Low**	3.5	**P4**	Open	Not started
F-11	Code	**Low**	3.3	**P4**	Open	Not started
F-12	Network	**Low**	3.1	**P4**	Open	Not started
F-13	Mobile	**Low**	3.1	**P4**	Open	Not started
F-14	Mobile	**Informational**	-	**P4**	Open	Not started
F-15	Network	**Informational**	-	**P4**	Open	Not started

Priority logic

Priority	Applies to	Recommended action
P1	**Critical risk**	Fix immediately and retest with first priority.
P2	**High risk**	Plan remediation in the nearest release or sprint cycle.
P3	**Medium risk**	Add to the security backlog and fix based on product risk.
P4	**Low / informational risk**	Handle as hardening, technical debt or process improvement work.

Detailed remediation plan

Prior.	Shared risk	How to fix
P1	SQL injection and database query safety Related: F-01	Rewrite vulnerable queries using parameterized queries or safe ORM parameters. Review similar search and filter endpoints, add regression tests and restrict database user privileges.
P2	Authorization, ACL and object ownership Related: F-02, F-04	Implement a centralized authorization layer, verify specific server-side permissions and validate object ownership in every endpoint action. Add negative tests for every role.
P2	External perimeter and administration exposure Related: F-03, F-12	Update the VPN component, restrict access by IP or VPN, enable MFA and place administrative interfaces behind additional access control. Review login logs and regularly check the public attack surface.
P3	User input, files and XSS Related: F-05, F-06	Apply an allowlist for file types, validate content, generate safe filenames and store files outside the web root. Validate URLs and other user input server-side, and render administration panel output with context-appropriate escaping.
P3	Sessions, passwords and sensitive data Related: F-08, F-09, F-11	Store session data only in secure platform or private application storage, invalidate sessions server-side on logout and remove plaintext password fields. Apply shorter session lifetime and safe token rotation.
P3	Cloud access key management Related: F-07	Check whether the AWS key is active, rotate or remove the affected key, review CloudTrail logs and remove the value from repository history. Add pre-commit and CI secrets checks and use a secrets manager.
P4	Automated request throttling Related: F-10	Implement IP, account and endpoint-level throttling, progressive cooldown, audit logs and anomaly alerts. Test throttling so legitimate user activity is not blocked.
P4	Mobile release hardening Related: F-13, F-14	Disable debug settings in production builds and verify release configuration in CI/CD. Treat certificate pinning as an additional protection for sensitive API communication.
P4	TLS and transport layer hardening Related: F-15	Standardize the TLS policy, enable HSTS, disable outdated protocols and cipher suites, and verify changes across all environments.