Report External Penetration Test
Thinkwise Platform
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>27/02/2023</td>
<td>0.1</td>
<td>Initial draft for internal review</td>
</tr>
<tr>
<td>22/03/2023</td>
<td>1.0</td>
<td>Final version for client</td>
</tr>
<tr>
<td>03/05/2023</td>
<td>1.1</td>
<td>Updated version after retest</td>
</tr>
<tr>
<td>08/05/2023</td>
<td>1.2</td>
<td>Updated after customer feedback</td>
</tr>
</tbody>
</table>
# Table of Contents

1 – Management Summary ................................................. 5
2 – Scope and context ...................................................... 7
3 – Test approach and process evaluation ............................. 8
4 – Reconnaissance phase .................................................. 9
  4.1 – Public information sources ..................................... 9
  4.2 – Vulnerability scan and manual reconnaissance ............ 10
  4.3 – Network level scan ............................................... 11
5 – Penetration Test: application level ................................ 12
  5.1 – Authentication and Session management ...................... 12
    5.1.1 – Login mechanism ......................................... 12
    5.1.2 – Password change mechanism ............................ 16
    5.1.3 – Session management .................................... 23
  5.2 – Broken Access Control ........................................... 25
    5.2.1 - Access without authentication ............................ 25
    5.2.2 - Access to functions with higher privilege levels .... 28
  5.3 – Unrestricted File Upload ......................................... 32
  5.4 – Directory traversal and Remote file inclusion ............... 36
  5.5 – Cross-site scripting (XSS) ...................................... 37
  5.6 – SQL injection ..................................................... 40
  5.7 – SSL/TLS checks .................................................... 41
  5.8 – Error handling ..................................................... 42
    5.8.1 – Client errors (4xx) ......................................... 42
    5.8.2 – Server errors (5xx) ....................................... 43
  5.9 – Sensitive data exposure ......................................... 44
  5.10 – Security (mis)configuration .................................... 46
    5.10.1 – Stack information in HTTP response messages .... 46
    5.10.2 – HTTP Security Headers ................................ 47
    5.10.3 – Cookie settings .......................................... 48
6 – Audit results ........................................................... 49
  6.1 – General comments ............................................... 49
  6.2 – (T)SQL for business logic ....................................... 50
  6.3 – Implementation database connection (ADO.NET) ........ 51
  6.4 – Preview components ............................................ 52
7 – Conclusion and recommendations 53
Appendix A – Results DNS reconnaissance 55
1 – Management Summary

Thinkwise B.V. (from hereon: Thinkwise) is creator and owner of the Thinkwise Low Code software platform, which can be used to build robust software applications for clients in a short period of time, making use of the functionality and technology that is part of the platform.

To be able to prove to customers that the Thinkwise Low Code platform is sufficiently secure, Thinkwise has mandated nSEC/Resilience to perform a penetration test on the platform. This penetration test was expanded with a number of audit-like checks.

During the initial penetration test on the example application of the Thinkwise platform no findings of high or critical severity were done. Testers did not succeed in obtaining significant amounts of sensitive data nor take control of the server. This is a good result.

The audit also did not produce findings that required immediate actions.

After receiving feedback from Thinkwise on the initial findings, and retesting of fixed findings, only a small number of low severity findings remained. These findings of low severity can be seen as findings for which there is no direct urgency to address them, but implementing solutions for these findings will further raise the bar for potential attackers and establish a baseline that would give customers or external auditors increased confidence.
Chapter 6 of the report provides a complete overview of all the findings. Below is an overview giving the results, per category (as identified by OWASP).

<table>
<thead>
<tr>
<th>Topic/area</th>
<th>Test result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network level</td>
<td>Good – no vulnerabilities found</td>
</tr>
<tr>
<td>Auth &amp; session management</td>
<td>Findings – 2x low severity</td>
</tr>
<tr>
<td>Broken Access Control</td>
<td>Good – no vulnerabilities found</td>
</tr>
<tr>
<td>Unrestricted File Upload</td>
<td>Good – no vulnerabilities found</td>
</tr>
<tr>
<td>Directory traversal / File inclusion</td>
<td>Good – no vulnerabilities found</td>
</tr>
<tr>
<td>Cross-site scripting (XSS)</td>
<td>Good – no vulnerabilities found</td>
</tr>
<tr>
<td>SSL/TLS</td>
<td>Good – no vulnerabilities found</td>
</tr>
<tr>
<td>SQL injection</td>
<td>Good – no vulnerabilities found</td>
</tr>
<tr>
<td>Error handling</td>
<td>Good – no vulnerabilities found</td>
</tr>
<tr>
<td>Sensitive data exposure</td>
<td>Finding – 1x low severity</td>
</tr>
<tr>
<td>Security (mis)configuration</td>
<td>Good – no vulnerabilities found</td>
</tr>
<tr>
<td>Audit</td>
<td>Good – no vulnerabilities found</td>
</tr>
</tbody>
</table>
2 – Scope and context

Thinkwise B.V. (from hereon: Thinkwise) is creator and owner of the Thinkwise Low Code software platform, which can be used to build robust software applications for clients in a short period of time, making use of the functionality and technology that is part of the platform.

To be able to prove to customers that the Thinkwise Low Code platform is sufficiently secure, Thinkwise has mandated nSEC/Resilience to perform a penetration test on an example application built using the platform.

Due to the nature of the application / platform, nSEC/Resilience has advised to also consider adding audit elements to the security evaluation on the platform. These audit elements will expand the coverage of the test activities to include more elements that are difficult to cover from an external dynamic penetration test. The results of this white box audit are also included in this report.

The attack surface (areas of the information system that an attacker or security evaluator can choose to initiate an attack) for the penetration test was defined as, and limited to:

- https://nsecresilience.thinklab.cloud/ (General)
- https://nsecresilience-prod.thinklab.cloud/universal/ (User interface)
- https://nsecresilience-prod.thinklab.cloud/indicium/iam/insights (Application tier)

It was explicitly allowed as part of the penetration test to investigate and exploit vulnerabilities in the assets in scope, as long as direct attack surface was limited to the definition above.

During the penetration test forensic research, code reviews and exhaustive DDOS testing were out of scope.
3 – Test approach and process evaluation

For the penetration test, 40 hours of testing was allocated. The testers received three test accounts for the Insights application, each with different access rights, so that proper tests for access control could take place. As such, the penetration test was executed grey-box.

Methodologically, the penetration test was performed in line with the PTES (infrastructure level) and OWASP WSTG (application level).

Reconnaissance for the penetration test was performed with industry-standard tooling (scanners and scripts) and by manually searching through public available sources. At network level also open ports and active services were investigated.

During the execution- and exploitation phase various tools were used. However, majority of the checks were performed manually, where internet traffic was investigated and manipulated with proxy tooling.

For the audit elements two resource days were allocated. The activities were for the largest part performed on location in Apeldoorn by two resources working in parallel.

No issues occurred during test execution.

After initial testing, Thinkwise gave feedback on the communicated findings. Some findings were not regarded as risk (or regarded as working as designed) and one finding was fixed and retested. This report describes the results after retest.
4 – Reconnaissance phase

Goal of the reconnaissance phase is to collect data from public sources and non-intrusive scans. Results from the reconnaissance phase give valuable information to be used in the execution phase.

4.1 – Public information sources

The following results are gathered from the open source intelligence reconnaissance phase (only the most relevant results are mentioned):

- See the full list of subdomain enumeration in Appendix A – DNS reconnaissance
- The identified domains have been checked using a tool that automatically makes screenprints based on a list of domains. No relevant results followed from these checks
- The IP addresses that corresponded to the identified subdomains were added to the scope for the external network level checks
- Search engine reconnaissance showed that job vacancy information contains technology information:
• Search for externally exposed software repositories did not yield results
• No other results were obtained from open source reconnaissance

4.2 – Vulnerability scan and manual reconnaissance

In addition to the collection of data from public sources, several vulnerability scans were performed with various tools, including the industry standard tool NetSparker and OWASP ZAP. Additionally, manual reconnaissance was performed to identify the application stack and attack vectors for the execution phase. The most relevant results:

• Manual reconnaissance reports that the target application is built on a Microsoft stack (IIS/10.0) with ASP.net
• Brute force directory and file scanners produced only a few results, none of which were relevant for the further investigation
• Exploration of the workflow shows various functionality that can be explored further, in particular upload functionality and preview functionality
• Many of the notifications from the scans were also found with manual checks/tests and are described in the paragraphs of Chapter 5
4.3 – Network level scan

In addition to the analysis of public sources, various scans have been performed on network level. These scans were performed using at least the industry standard tools Nessus and Nmap.

The main IP address (20.170.5.72) was scanned on network level, with the following results:

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Status</th>
<th>Service</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>TCP</td>
<td>Open</td>
<td>HTTP</td>
<td>Azure application gateway</td>
</tr>
<tr>
<td>443</td>
<td>TCP</td>
<td>Open</td>
<td>HTTPS</td>
<td>Azure application gateway</td>
</tr>
</tbody>
</table>

Next to the regular TCP network level scans, various firewall evasion techniques were deployed and UDP scanning was executed. From these checks, no findings were done on network level for the main IP address in scope.
5 – Penetration Test: application level

Following the outcomes of the reconnaissance phase, a number of aspects are analyzed in more detail. The executed actions and analyses are discussed in the following paragraphs, classified according the OWASP top 10.

5. 1 – Authentication and Session management

5.1.1 – Login mechanism

Upon login to the insights application (user interface) via https://nsecresilience-prod.thinklab.cloud/universal/, by default a username and password was required before any other interaction with the application was possible:
When either the username or password is incorrect, a neutral error message is returned. This prevents the login mechanism from being used for user enumeration which is good. On the login screen, a meta server URL and an application and platform could be configured.

It was attempted to change the meta server URL to a server in control of the testers, to see if credentials could be intercepted in this way. Doing so it was noticed that one request was intercepted by the server in our control:

The source IP address however was an attacker IP address and not an IP address controlled by Thinkwise (meaning that no SSRF attacks were possible).

When submitting the login credentials a POST request is made to /indicium/account/apio/login with the username and password in the request body:
A 204 “No Content” response is then returned together with an 
.AspNetCore.Identity.Application cookie, which acts as session cookie for the application. When successfully authenticated, this cookie is used for authorizing further requests made in the application.

An alternative login function for indicium local login was found via https://nsecresilience-prod.thinklab.cloud/indicium/account/ui/localLogin:

![Login Form](image)

Similar to the previous login, upon successful authentication a 

![Response](image)
This cookie holds the same authorization as the cookie that is returned by the user interface login however upon successful login the user is not immediately redirected to the user interface.

Using a higher privileged admin account with access to IAM, further login options could be configured, directly affecting the login mechanism.

In the login mechanisms as configured, no vulnerabilities were identified.
5.1.2 – Password change mechanism

For the Insight application there are possibilities to change user passwords:

- By regular users, for their own accounts (if enabled in the IAM)
- By a higher privileged user

The highly privileged user is responsible for setting appropriate security configurations such as password strength and expiration policy. Settings such as password expiry are not set by default.
Password change for IAM user (high privilege):

The IAM user (high privilege) is able to change passwords on behalf of users via the “update password” function. The password policy can be configured via IAM.
Password change for Insights user (low privileged):

If enabled by the IAM user, a lower privileged insights application user is able to change their own password. In this case the current password is required, which is good:
In the backend this current password is also passed on and a valid session cookie is required for the request to be successful:

It is in general recommended to enforce password policies by default instead of making it optional for the high privileged user to configure. This will help push the use of stronger passwords for applications and users making use of the Thinkwise software. However because the password policy can be defined by any organization through IAM no finding is included.
Password reset (unauthenticated)

The Insights application also contained a password reset mechanism in the form of a “password forgot” function for unauthenticated users (if password recovery is allowed by the administrator).

In this case a username must be entered, after which an email with a unique token is sent via e-mail that can be used to reset the password.

This function is not sensitive to username enumeration; the same response is given regardless of whether the input was correct or not.
After an existing username is entered, the user should receive an email with a validation token that has to be entered together with a new password:

When an invalid or no validation token is entered, an error is returned indicating that the token is incorrect:
When checking the received e-mail it is observed that the validation token is a 6-digit code, for example:

![Image of email excerpt]

This code is tested to have an expiration time of under 30 minutes, which makes brute force attacks on this code less likely. However there is no brute force attack detection present on the MFA function. An attacker seems to be able to try many combinations. It is advised to add more protection on the backend by for example temporarily blocking a user when more than 5 invalid MFA codes are submitted in a short timeframe. A finding of severity low was added for this.
5.1.3 – Session management

Checks were done for session management such as checks around validity, lifetime and data storage.

It was found that sessions do not end within a specific period of inactivity or after browser close. In some cases sessions remain active for longer periods due to specific requests generated from the GUI keeping the session alive. The session is ended on browser close (unless the user has selected the “remember me” option).

Generally it is desired from the perspective of security to by default have a mechanism for automatic session timeout after a short period of inactivity (a few hours maximum) that works in all situations, so that the user is forced to reauthenticate after not using an application for some time.

A finding of severity low was added for this observation, with the recommendation to enforce that a session timeout is implemented by default and is active in all scenario’s.

It was also found that after a user explicitly logs out via the logout function, requests made with the old session cookie (.AspNetCore.Identity.Application) would still return a valid response, some time after the user had logged out.
This means in the backend the session cookie is not immediately invalidated after explicit logout and can still return data that requires authentication. Because the sessions are short-lived and implementation of a fix would be very difficult (and potentially in conflict with leading RFCs) no finding will be included.

Local browser storage was found to be mostly empty, not containing any sensitive or interesting data:

No further findings were done with regards to session management.
5. 2 – Broken Access Control

With broken access control, an attacker exploits references to objects or functions to access resources that the attacker is not authorized for. Often this only can be exploited if the authentication management/session management is inadequate. Examples can be references to files with predictable file names, or manipulating function parameters such as an organization ID.

For the Insights example application access control was checked from two perspectives: an attacker without any access to the application, and an attacker with a form of access trying to access functions/data corresponding to a higher access level.

5.2.1 - Access without authentication

Necessary checks were done to understand if an attacker can obtain unauthorized access to functions and/or data that they should not have access to.

The platform is set up to require a valid `AspNetCore.Identity.Application` cookie in order to retrieve information.

When this cookie is not present or invalid in a request, a “401 Unauthorized” error is returned as seen on the screenshot below:
This behavior was found to be consistently present; no exceptions were found.

Since the platform makes use of Microsoft’s OData standard for the API layer it was also tested if the OData metadata file could be retrieved without authentication. This is relevant because a publicly retrievable metadata file can expose application structure and parameters that could help an attacker find vulnerabilities.

The OData metadata file could however not be retrieved without authentication. Attempts to do so returned a “401 Unauthorized” response, which is good:

The same result followed for the OpenAPI documentation which can be used to view available API calls.
No findings were done for access without authentication.
5.2.2 - Access to functions with higher privilege levels

In this context, the vertical privilege escalation scenarios were investigated. Checks were performed to ensure that access control between different privileges is implemented correctly and consistently.

Various functions/endpoints for the application in-scope were tested across the different roles provided for the applications. As one user has access to the IAM application while the other has not, these could be tested for vertical privilege escalation.

As one of the tests, it was tried to edit the details of another user as low privileged user, by intercepting the request from the IAM to do so and replacing the session cookie:

As seen in the screenshot above, the other user does not seem to exist in the context of the low privileged user and a 404 not found error is returned. A 404 error is also returned when a GET request is made to retrieve another user.

Administrators in the IAM application have the ability to change passwords on behalf of a user. It was attempted to intercept the request to change a user’s password and change the session cookie to that of a low privileged user (that does not have access to this function) to see if the password of another user could be updated this way. As seen on the screenshot below, updating the password via this function does not require the current password:
After replacing the session cookie and sending the request however, a 404 “not found” error is returned, indicating this vertical privilege escalation attack did not succeed which is good.
Trying to access database event logs as low privileged user, a “401 Unauthorized” error is returned and the user is redirected to login screen:

Using the OData metadata file, further checks were done to see if the low privileged user (demo_nl) could access certain service document files that may reveal sensitive information.

For example the IAM admin user can access a service document with sensitive data such as password hashes via:

- nsecresilience-prod.thinklab.cloud/indicium/iam/iam/usr
When trying to access this as the low privileged user, a 404 “not found” error is returned. The low privileged user could only retrieve his/her own data, not including any password hashes:

No findings were done with regards to vertical privilege escalation.
5.3 – Unrestricted File Upload

With unrestricted file upload, an attacker uses the functionality to upload files. Instead of valid files other vulnerable files are uploaded. For example, a remote shell can be uploaded so that control can be taken on the server.

For file upload checks focus was placed on the INSIGHTS application because the upload functions that were present there are also available to normal end users (unlike the upload functions in the IAM application, which only administrators can access).

First, testers identified the unique upload features present in the INSIGHTS application.

In the Insight application photos can be uploaded of an individual. Several tests have been performed on this upload function. The first test was to upload files with potentially dangerous file extensions, such as .exe, .html, etc.

When a file with a disallowed file extension was selected in the UI, the following error was displayed:

![Unknown Image Type]

It was not possible from the frontend/UI to select a file with an extension other than image file extensions, like jpeg, jpg, png, etc.
However, it was possible to intercept the PATCH request (that makes up the first step in the upload process) and change the file extension from (for example) .png to .html:

After this, the second upload request (a POST message) was also intercepted. The Content-Type was changed to text/html and the request was sent:
After these changes were applied, the request was sent to the backend server and the .html file was successfully uploaded. This file was then temporarily available through a URL that could be accessed, for example:

- https://nsecresilience-prod.thinklab.cloud/indicium/iam/40/staged_employee(d7a460f3-301c-43e5-8253-6f350cc6c4f1)/INSIGHTS.download_photo(file_id=null)?t=1677244462911

Once upload has been completed and the uploaded file is stored on the backend server it cannot be opened in the context of the web application itself for example as part of a preview function. The file is immediately downloaded locally and can then be opened in a browser of choice. This mitigates most of the direct risk; if files with dangerous extensions could be made to be rendered directly by the webserver, this could lead to code execution.

Checks in file upload functions require validations for file extensions on both the frontend and backend. On the Insights application this was only the case on the frontend. However for Thinkwise in general allowed file extensions can be configured in the Software Factory in the ‘extention whitelist’ setting that can be set for each ‘file storage location’. Because of this possibility no finding will be included.

In addition to tests on validations of file extensions, tests were also performed with the goal of establishing whether there is an active virus scanner active on the file upload functionalities. For this purpose a so-called EICAR file was used. The EICAR Standard Anti-Malware Test file is a special 'dummy' file which is used to test the correct operation of malware detection scanners. When an EICAR test file is placed on a file system, any virus scanner that is active on that file system will detect it exactly as if it were a malicious program. Alternatively, some virus scanners can check file contents as part of the upload function itself, and block files before they are placed on the operating system.
In the case of the Insights application the EICAR file was uploaded without any restrictions:

Since the EICAR file was not blocked by the webserver it did not seem that any virus scanner is active on the location where these files are saved. Since the file after saving the data, with the HR employees for example, is immediately downloaded locally into the client’s browser, this can bring a risk with it. In general the advice is to make sure that there is a virus scanner active on the file system to which files are uploaded. Because the implementation of security measures in this form are the responsibility of partner organizations or end users, no finding will be included for this.
5.4 – Directory traversal and Remote file inclusion

With directory traversal it is possible to change the URL of the web application in such a way that files in other directories (outside the location of the files of the web application) can be accessed. With remote file inclusion same thing happens but this is done through including remote files through file parameters in functions.

Moderate manual testing was done on regular directory traversal based on windows directories and files, respectively in both regular format as encoded format. As automated tool, next to NetSparker and OWASP ZAP also dotdotpwn was used. No vulnerabilities were identified. For file inclusion no attack vectors were identified.
5.5 – Cross-site scripting (XSS)

Cross-site scripting is a technique in which an attacker makes use of lack of sanitation of user input. The attacker tries to leverage such as vulnerability by identifying a place where any malicious input will be presented back to a user. The attacker then inputs malicious code, for example to steal cookies and send them to the attacker, and waits for another user to trigger the scripts. That way, the attacker can collect information or even control complete user browser sessions.

During the penetration test various forms and functions with user input were tested for cross-site scripting. Automated scans were used to test for reflected cross-site scripting extensively. Stored XSS was investigated mainly using manual testing.

It was found that generally html/javascript injection attempts are blocked by consistently by the application by applying entity encoding.

For example, using the admin IAM account, XSS payloads were injected inside the username of demo_nl user, as the name field is referenced often within the application.

The entity encoding was consistent across various reflected input fields:
However some exceptions were found, mainly by making use of the earlier found unrestricted file upload vulnerabilities across file upload functions.

For example in the insights application -> projects -> documents section, it was possible to upload any file, which could then be previewed inside the application. By uploading a .html file it was possible to get HTML code reflected inside the application:
However XSS was still prevented in this context by the usage of the content-security-policy security header which is good. Nevertheless, HTML injection can be used to alter the appearance of a page for other users that visit it. Therefore uploaded files can be used for the purpose of social engineering attacks. Although it is good to be aware of this, no finding was included because HTML extensions can be blocked in the Software Factory.

As part of XSS-like injection attacks a PDF generation function was tested under finance -> generate invoice -> print invoice. The underlying reason is that server-side PDF generators are sometimes vulnerable for HTML injection or SSRF attacks. By changing user input parameters to XSS and SSRF related payloads it was checked whether the PDF generator would handle these payloads correctly. From the tests performed there were no signs of reflected HTML/XSS or SSRF-related callbacks.

No findings were added.
5.6 – SQL injection

With SQL injection, like XSS, input provided by a user is not checked sufficiently for malicious content. With SQL injection this vulnerability is used to influence SQL statements used in the web application to extract or manipulate information from the web application database.

The application and API were explored through OWASP ZAP, which supports recognition of OData functions, and subsequently tested for SQL injection.

No vulnerabilities were identified.
5.7 – SSL/TLS checks

The implementation of the encryption of internet traffic between the Insights demo application (nsecresilience.thinklab.cloud) and its users was analysed, with the following results:

- Certificate is delivered by Sectigo, and is assigned to *.thinklab.cloud
- The certificate is trusted and has a good validity (19th May, 2023)
- The webserver supports TLS 1.2, TLS 1.1 and TLS 1.0
- The cipher suites offered by the webserver to start session encryption are mostly adequate, but as a general rule it is advised to remove the TLS_RSA cipher suites. In general, an up to date advice can be found in paragraph 2.3 on:
- It is worth noting that the cipher suite names above are in the IANA format. These can be mapped to OpenSSL format through for example the table on https://testssl.sh/openssl-iana.mapping.html
- Other checks indicate that the other properties of SSL/TLS are secure. For example secure renegotiation is supported. Downgrade attack prevention (TLS_FALLBACK_SCSV) is not active, but although activating it is an improvement, there is not sufficient risk involved to warrant a finding to be included.

TLS 1.1 and TLS 1.0 are considered to be relatively weak protocols. The advice is to disable both. Because these findings are specific to the Insights application and not to the platform in general no finding will be included.
5.8 – Error handling

Analysis of error handling is important because in case exception- or errors are not processed correctly, sensitive information can be exposed to an attacker, for example a stack trace with directory names or filenames. The following conclusions as result of analysis:

5.8.1 – Client errors (4xx)

- **401 Unauthorized**: If a page is accessed or a request is made to which a user should not have access, a 401 Unauthorized error is consistently returned with in the response that the user does not have access to the requested resource. This also applies to unauthenticated requests:
404 Not found: When requesting a non-existing page a standard IIS 404 error is shown. The formatting of a standard IIS error page is missing, but the text is exactly the same. Therefore, it can still be concluded that an IIS server is running.

Because these error messages do not reveal sensitive information, like verbose error responses, the handling of 4xx errors is adequate.

5.8.2 – Server errors (5xx)

500 Internal Server Error: This message indicates that there is an error in your website's code. This is preventing the website from loading correctly. Using the Forgot Password feature, you have encountered a 500 internal server error:

Because these error messages do not reveal sensitive information, like verbose error responses, the handling of 5xx errors is adequate.
5.9 – Sensitive data exposure

With sensitive data exposure, information can be found which gives an attacker additional information with respect to the application or application landscape.

As per the OData standard, metadata was found, for example for the Insights application through the following URL:

- https://nsecresilience-prod.thinklab.cloud/indicium/iam/40/$metadata

It is advised to hide the OData service metadata. The metadata describes the structure of the entities exposed by the service. Attackers can use the metadata document to better understand the structure of the entities exposed by the OData service and create more targeted attacks based on this information.

Since the metadata for Thinkwise applications can be disabled via a setting in appsettings.json no finding will be included.

Next to checking content of the JavaScript files and other HTTP traffic, brute force directory scanning was performed.

In addition, some possible internal file paths had been found. This was available from the following URL:

- https://nsecresilience-prod.thinklab.cloud/indicium/errorLog

Further investigation revealed that this URL was only available to Administrators of the platform. Since this is only available to Administrators, no finding was raised for this.
Checks were also performed on Thinkwise's software factory application. These checks for example tried to validate whether the source code of this application could be inspected. This was indeed the case. Free .NET decompilers and code viewers such as DotPeek can be used to view the decompiled source without any limitations; no obfuscation is currently present within the source code:

It is advised to obfuscate the source code for the Software Factory. This can help protect the intellectual property of software companies or individual developers. A finding with low severity has been raised for this.
5.10 – Security (mis)configuration

For adequate security it is necessary that the correct configuration is chosen and implemented for all parts. This applies to the application but also to the webserver. Several checks were performed, whose main results are listed here.

5.10.1 – Stack information in HTTP response messages

An attacker will start a possible attack by making an overview of the application stack. The HTTP response message is an important instrument for gathering information about the application stack. It is considered best practice to remove as much information from the HTTP response messages as possible.

- Server: Microsoft-IIS/10.0
- X-Powered-By: ASP.NET

It is advised to remove these response headers if possible. This finding is specific for the Insights application and does not apply to the Thinkwise platform in general.
5.10.2 – HTTP Security Headers

Security headers are directives used by web applications to configure security defenses in web browsers. Based on these directives, browsers can make it harder to exploit client-side vulnerabilities such as Cross-Site Scripting or Clickjacking.

Headers that are present:

- Present headers:
  - X-XSS-Protection
  - CSP header
  - Referrer policy
  - X-Frame-Options
  - X-Content-Type-Options

Missing headers:

- Strict-Transport-Security

With the Strict-Transport-Security security header, you ensure that users can only access the Web application via the HTTPS protocol and that any future attempts to access it using HTTP should automatically be converted to HTTPS. It is advised to always add this header. However because this is usually an infrastructure level setting, this header will not be set from the Thinkwise platform and no finding will therefore be incorporated.
5.10.3 – Cookie settings

Cookies used by the applications to store important data on the user's side. It is important to protect this data. It can be done in part by providing the correct settings to the cookies from the server-side. On the one hand, it is good to give important cookies the so-called Secure flag, which forces these cookies only to be sent over secure connections (HTTPS). To prevent cookies from being vulnerable to cross-site scripting (XSS attacks), cookies can be provided with the flag HttpOnly, which prevents a cookie from being read by an attacker. Another best practice is the SameSite cookie attribute that helps in preventing CSRF.

The .AspNetCore.Identity.Application is used for access control and typically security settings should be evaluated for that cookie.

For this specific cookie, the settings that are available for cookies that will avoid cookie values being intercepted through a man-in-the-middle attack (“secure”) and through XSS (“httponly”) are both active.

The “SameSite” setting however is currently set to “Lax” which means that a number of CSRF scenario’s are not blocked. It is advised to change this setting to “Strict” if possible. However because an antiforgery cookie is also being used, which should mitigate CSRF attacks as well, no findings are included.
6 – Audit results

Because software created using recent Microsoft technology is in obvious areas often secure by default, the quality and coverage of security testing can be raised by performing further audit like checks on the software platform.

These white box checks are proposed to be performed on location, where an nSEC/Resilience consultant will perform the checks together with Thinkwise resources. For the audit activities in performed in 2023 focus was placed on the most relevant attention points as identified during a previous audit in 2020. Backend code security was investigated using sample based code inspection with developers on processing of user input that is processed in the backend, for example in TSQL. In addition to this a security analysis was performed of the overall application architecture with the purpose of identifying potential security issues or weaknesses in interfaces or application entry/exit points.

6.1 – General comments

Newer versions of the Thinkwise platform components Indicium and Frontend have been made available recently. The older versions of these components can be considered to be effectively end of life and have been placed out of scope for the audit.

Thinkwise currently already performs a number of security related checks as part of their CI/CD pipeline:

- Static analysis based on Sonarcloud and Eslint for frontend/Universal
- Open source component checks frontend/Universal: Yarn

Open source component checks for other components are performed on ad hoc basis and are performed manually (NuGET repositories).
6.2 – (T)SQL for business logic

The functionality of an application made with the Thinkwise platform is generated based on what is defined in the Software Factory and the Indicium middle layer. This process also includes generating parameterized (T)SQL statements

- Default procedures that are used for example to work with input forms on application level
- Layout procedures that are meant to indicate which fields should be visible and which not
- Context procedures that process tasks or generate reports
- Database triggers
- Batch procedures
- Process procedures, through which workflows are defined

(T)SQL statements can only be edited by users that have access to the database or Software Factory. These types of access are typically not obtained by users of the end client but only by Thinkwise or partner organizations.

Because the (T)SQL statements are generated at runtime, they can’t be checked using static code security checkers such as sonarcube.

Based on what was discussed in relation to the (T)SQL statements, no risks were identified because all generated statements are parameterized and there is no opportunity to change these statements.
6.3 – Implementation database connection (ADO.NET)

In the application landscape of an active Thinkwise application at runtime, the Indicium middle layer has an active connection to the IAM component based on connection settings that are set in the platform by default.

The configuration for any connections that need to be made to other databases are stored in the IAM. The configurations are made in the Software Factory, which then synchronizes to the IAM.

The passwords for the connections were also stored without encryption for the Insights application. However for the Thinkwise platform in general it is possible to store these passwords encrypted.
Another observation was related to the way the initial credentials for the database pool are stored. Also due to the currently available documentation (docs.thinkwisesoftware.com/docs/deployment/indicium) these credentials are often stored hardcoded in appsettings.json. More secure alternatives are available; therefore no finding will be included.

6.4 – Preview components

Based on a discussion of the application landscape, a potential attack vector was found in the availability of file preview functionality in the platform. For example, HTML files can be viewed as PDF, and Excel files can be viewed to HTML files.

The preview components make use of the gembox library (gemboxsoftware.com). Thinkwise has obtained a license to make use of gembox as a part of the Thinkwise platform.

Based on the audit discussions, a number of additional tests have been defined for the penetration test. No findings have been done for the preview functionality itself. The gembox software does not seem to contain any known vulnerabilities or default configuration errors.

An observation that was done is the fact that the gembox licence key is included hardcoded in the Thinkwise platform files. However this key is not usable from other environments because it is digitally signed.
7 – Conclusion and recommendations

During the initial penetration test on the example application of the Thinkwise platform no findings of high or critical severity were done. Testers did not succeed in obtaining significant amounts of sensitive data nor take control of the server. This is a good result.

The audit also did not produce findings that required immediate actions.

After receiving feedback from Thinkwise on the initial findings, and retesting of fixed findings, only a small number of low severity findings remained. These findings of low severity can be seen as findings for which there is no direct urgency to address them, but implementing solutions for these findings will further raise the bar for potential attackers and establish a baseline that would give customers or external auditors increased confidence.

A findings overview with the vulnerabilities ordered by severity can be found on the next page.
<table>
<thead>
<tr>
<th>Description</th>
<th>Category</th>
<th>Severity</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>In some cases sessions remain active after long periods of inactivity</td>
<td>Authentication &amp; Session Management</td>
<td>Low</td>
<td>Enforce that the mechanism to automatically end sessions based on user inactivity works in all situations</td>
</tr>
<tr>
<td>No brute force protection on the MFA token for the password reset function</td>
<td>Authentication &amp; Session Management</td>
<td>Low</td>
<td>Add functionality to block a user or a code after 5 incorrect attempts within a short timeframe</td>
</tr>
<tr>
<td>Runtime components of the Thinkwise platform can be decompiled into readable source code</td>
<td>Sensitive Data Exposure</td>
<td>Low</td>
<td>It is advised to obfuscate/hide the source code to protect intellectual property</td>
</tr>
</tbody>
</table>
## Appendix A – Results DNS reconnaissance

<table>
<thead>
<tr>
<th>Subdomain</th>
<th>IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>community.thinkwisesoftware.com</td>
<td>52.222.139.88</td>
</tr>
<tr>
<td>docs.thinkwisesoftware.com</td>
<td>144.178.66.249</td>
</tr>
<tr>
<td>office.thinkwisesoftware.com</td>
<td>144.178.66.249</td>
</tr>
<tr>
<td>updates.thinkwisesoftware.com</td>
<td>144.178.66.249</td>
</tr>
<tr>
<td>ssh.thinkwisesoftware.com</td>
<td>185.162.30.162</td>
</tr>
<tr>
<td>universal.thinkwisesoftware.com</td>
<td>144.178.66.244</td>
</tr>
<tr>
<td>msoid.thinkwisesoftware.com</td>
<td>40.126.32.68</td>
</tr>
<tr>
<td>lyncdiscover.thinkwisesoftware.com</td>
<td>52.112.196.12</td>
</tr>
<tr>
<td><a href="http://www.thinkwisesoftware.com">www.thinkwisesoftware.com</a></td>
<td>199.60.103.28</td>
</tr>
<tr>
<td>metrics.thinkwisesoftware.com</td>
<td>144.178.66.249</td>
</tr>
<tr>
<td>staging.thinkwisesoftware.com</td>
<td>199.60.103.228</td>
</tr>
<tr>
<td>mail.thinkwisesoftware.com</td>
<td>144.178.66.243</td>
</tr>
<tr>
<td>offers.thinkwisesoftware.com</td>
<td>199.60.103.28</td>
</tr>
<tr>
<td>enterpriseregistration.thinkwisesoftware.com</td>
<td>20.190.137.40</td>
</tr>
<tr>
<td>registry.thinkwisesoftware.com</td>
<td>195.154.68.114</td>
</tr>
<tr>
<td>autodiscover.thinkwisesoftware.com</td>
<td>144.178.66.243</td>
</tr>
<tr>
<td>kibo.thinkwisesoftware.com</td>
<td>144.178.66.243</td>
</tr>
<tr>
<td>vpn.thinkwisesoftware.com</td>
<td>144.178.66.242</td>
</tr>
<tr>
<td>blog.thinkwisesoftware.com</td>
<td>199.60.103.28</td>
</tr>
<tr>
<td>sip.thinkwisesoftware.com</td>
<td>52.112.192.11</td>
</tr>
<tr>
<td>enterpriseenrollment.thinkwisesoftware.com</td>
<td>20.91.147.72</td>
</tr>
<tr>
<td>insights.thinkwisesoftware.com</td>
<td>144.178.66.244</td>
</tr>
<tr>
<td>filecap.thinkwisesoftware.com</td>
<td>144.178.66.245</td>
</tr>
<tr>
<td>thinklab.thinkwisesoftware.com</td>
<td>185.149.37.40</td>
</tr>
<tr>
<td>prtg.thinkwisesoftware.com</td>
<td>144.178.66.244</td>
</tr>
<tr>
<td>tsf-quarantine.thinkwisesoftware.com</td>
<td>144.178.66.242</td>
</tr>
<tr>
<td>webmail.thinkwisesoftware.com</td>
<td>95.97.179.163</td>
</tr>
<tr>
<td>licensing.thinkwisesoftware.com</td>
<td>144.178.66.249</td>
</tr>
</tbody>
</table>