NInet Security Evaluation Report

Manyfold

V 1.0
Amsterdam, December 7th, 2025
Public

Document Properties

Client	Manyfold
Title	NLnet Security Evaluation Report
Target	• Manyfold (https://github.com/manyfold3d/manyfold)
Version	1.0
Pentester	Stefan Vink
Authors	Stefan Vink, Marcus Bointon
Reviewed by	Marcus Bointon
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	November 20th, 2025	Stefan Vink	Initial draft
0.2	November 26th, 2025	Stefan Vink	Ready-for-Review
1.0	December 7th, 2025	Marcus Bointon	1.0

Contact

For more information about this document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Science Park 608 1098 XH Amsterdam The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

Between November 17, 2025 and November 20, 2025, Radically Open Security B.V. carried out a penetration test for Manyfold and NLnet NGI Zero Entrust.

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

1.2 Scope of work

The scope of the penetration test was limited to the following target:

- Manyfold (https://github.com/manyfold3d/manyfold)

The scoped services are broken down as follows:

- Pentesting: 4.25 days
- Reporting: 0.75 days
- **Total effort: 5 days**

1.3 Project objectives

ROS will perform a penetration test of the Manyfold web application with its developers in order to assess its security posture. To do so, ROS will investigate a hosted instance of the application and guide Manyfold in attempting to find vulnerabilities in it, exploiting any such found to try and gain further access and elevated privileges.

1.4 Timeline

The security audit took place between November 17, 2025 and November 20, 2025.

1.5 Results In A Nutshell

During this crystal-box penetration test of the Manyfold application we found 1 High, 2 Elevated, 1 Moderate and 5 Low-severity issues.

The most critical issues relate to access control and identity management. In particular, weaknesses in model sharing MAF-008 (page 18) and zip download functionality MAF-009 (page 12) can enable unauthorized access to private

model sets, while an OIDC auto-linking flaw MAF-004 (page 19) and the first-user auto-admin flow MAF-005 (page 26) increase the risk of account takeover or unintended administrator access.

Further issues include insufficient hardening of authentication and federation endpoints including missing rate limiting on OAuth/OIDC flows MAF-001 (page 20) and inadequate sanitization of data retrieved from Fediverse servers MAF-002 (page 21). Some sensitive information is exposed around client secrets and passwords in MAF-003 (page 25) and MAF-007 (page 27). Unsafe handling of uploaded zip files could lead to path traversal or denial-of-service MAF-010 (page 15).

Due to the time-boxed nature of this assessment we did not perform an in-depth retest of the issues reported in the previous penetration test. However, based on the GitLab issue tracker and incidental observations during testing, we note that many of those issues have been addressed.

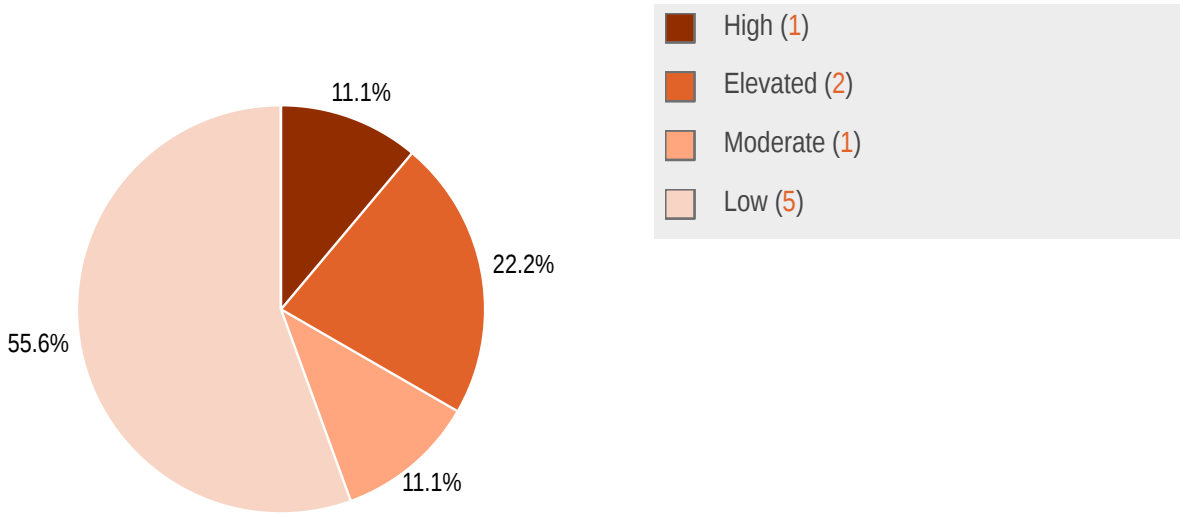
We also did not include any of the existing security-labelled issues in the public issue tracker as findings in this report, since they are already known and being tracked by the project.

1.6 Summary of Findings

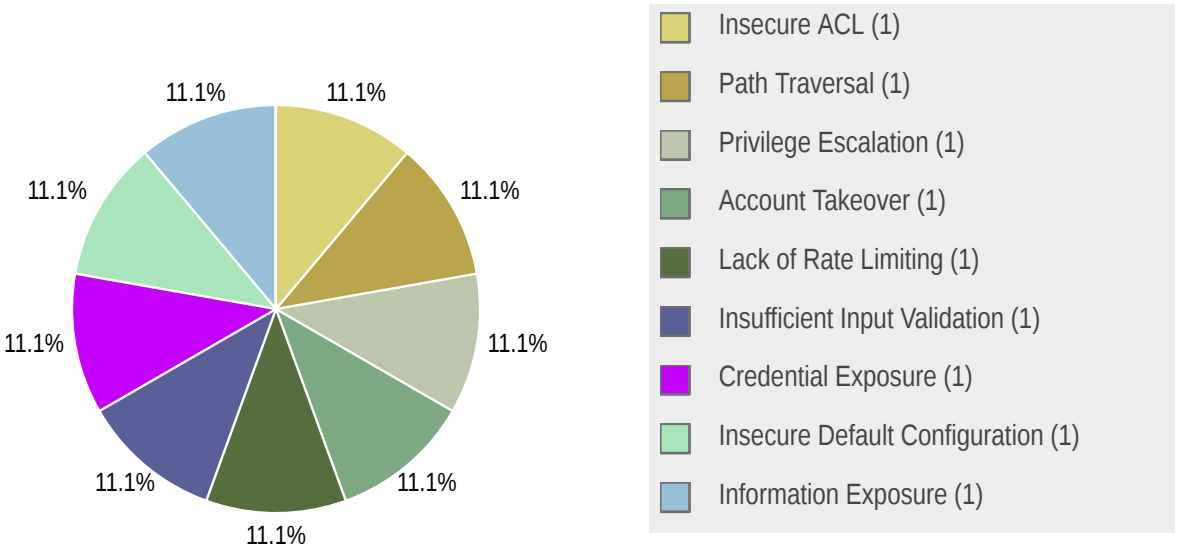
Info	Description
MAF-009 **High** **Type:** Insecure ACL **Status:** none	The application's zip file functionality does not properly enforce access controls, allowing users to download modelsets even when sharing is set to private.
MAF-010 **Elevated** **Type:** Path Traversal **Status:** none	A path traversal flaw exists in how uploaded zip files are processed.
MAF-008 **Elevated** **Type:** Privilege Escalation **Status:** none	Users with edit permissions on models can grant permissions to other users and even assign owner permissions.
MAF-004 **Moderate** **Type:** Account Takeover **Status:** none	The application links OIDC accounts to existing password-based accounts by email without verifying ownership of the address.
MAF-001 **Low** **Type:** Lack of Rate Limiting **Status:** none	The exposed OAuth client-credentials token and the optional OpenID Connect login endpoints do not have rate-limiting controls.

MAF-002 **Low** **Type:** Insufficient Input Validation **Status:** none	The Fediverse feature retrieves data from connected servers, and it is not sanitized properly before being displayed or executed.
MAF-003 **Low** **Type:** Credential Exposure **Status:** none	Client secrets for OAuth applications are exposed in plain text within the application's user interface.
MAF-005 **Low** **Type:** Insecure Default Configuration **Status:** none	The first user to authenticate automatically receives administrator privileges.
MAF-007 **Low** **Type:** Information Exposure **Status:** none	When administrators create new user accounts and set passwords, the passwords are displayed in plaintext instead of being obscured.

1.6.1 Findings by Threat Level



1.6.2 Findings by Type



1.7 Summary of Recommendations

Info	Recommendation
MAF-009 **High** **Type:** Insecure ACL **Status:** none	• Implement proper access controls and authorization checks on the zip file download functionality to ensure it respects sharing settings.
MAF-010 **Elevated** **Type:** Path Traversal **Status:** none	• Implement strict input validation on the `Filename` parameter to prevent path traversal sequences. • Use a dedicated temporary directory for handling uploaded files and sanitize all user-supplied filenames. • Set appropriate rate limiting on the zip download functionality to mitigate DoS risks.
MAF-008 **Elevated** **Type:** Privilege Escalation **Status:** none	• Implement strict authorization checks before processing any permission change requests, ensuring users cannot grant privileges higher than those they possess. • Introduce another role that allows managing user permissions.
MAF-004 **Moderate** **Type:** Account Takeover **Status:** none	• Verify the `email_verified` claim from the OIDC provider. • Require users to be logged in before linking accounts. • Implement additional verification steps for email ownership.
MAF-001 **Low** **Type:** Lack of Rate Limiting **Status:** none	• Add per-IP throttles for `/oauth/token` and the OpenID Connect entrypoint/callback to cap token-issuance and OIDC login attempts. • Log and alert on repeated failures or throttled requests.

MAF-002 **Low** **Type:** Insufficient Input Validation **Status:** none	• Implement strict sanitization and input validation for all fields retrieved from external Fediverse servers before displaying or executing them. • Only allow HTML tags and attributes that are absolutely necessary for the application's functionality and user experience. • Use a mature library to safely sanitize any content received from external sources.
MAF-003 **Low** **Type:** Credential Exposure **Status:** none	• Only show the full secret value once when first generated, then mask it going forward. • Regularly rotate and expire client secrets according to security best practices.
MAF-005 **Low** **Type:** Insecure Default Configuration **Status:** none	• Design a safer procedure for setting up the initial admin account.
MAF-007 **Low** **Type:** Information Exposure **Status:** none	• Implement input masking on all password fields, obscuring characters as they are typed using asterisks or dots. Standard HTML `password` input tags do this.

2 Methodology

2.1 Planning

Our general approach during penetration tests is as follows:

1. **Reconnaissance**
   We attempt to gather as much information as possible about the target. Reconnaissance can take two forms: active and passive. A passive attack is always the best starting point as this would normally defeat intrusion detection systems and other forms of protection afforded to the app or network. This usually involves trying to discover publicly available information by visiting websites, newsgroups, etc. An active form would be more intrusive, could possibly show up in audit logs and might take the form of a social engineering type of attack.
2. **Enumeration**
   We use various fingerprinting tools to determine what hosts are visible on the target network and, more importantly, try to ascertain what services and operating systems they are running. Visible services are researched further to tailor subsequent tests to match.
3. **Scanning**
   Vulnerability scanners are used to scan all discovered hosts for known vulnerabilities or weaknesses. The results are analyzed to determine if there are any vulnerabilities that could be exploited to gain access or enhance privileges to target hosts.
4. **Obtaining Access**
   We use the results of the scans to assist in attempting to obtain access to target systems and services, or to escalate privileges where access has been obtained (either legitimately though provided credentials, or via vulnerabilities). This may be done surreptitiously (for example to try to evade intrusion detection systems or rate limits) or by more aggressive brute-force methods. This step also consist of manually testing the application against the latest (2021) list of OWASP Top 10 risks. The discovered vulnerabilities from scanning and manual testing are moreover used to further elevate access on the application.

2.2 Risk Classification

Throughout the report, vulnerabilities or risks are labeled and categorized according to the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting

These categories are:

- **Extreme**
  Extreme risk of security controls being compromised with the possibility of catastrophic financial/reputational losses occurring as a result.

- **High**

  High risk of security controls being compromised with the potential for significant financial/reputational losses occurring as a result.
- **Elevated**

  Elevated risk of security controls being compromised with the potential for material financial/reputational losses occurring as a result.
- **Moderate**

  Moderate risk of security controls being compromised with the potential for limited financial/reputational losses occurring as a result.
- **Low**

  Low risk of security controls being compromised with measurable negative impacts as a result.

3 Reconnaissance and Fingerprinting

We were able to gain information about the software and infrastructure through the following automated scans. Any relevant scan output will be referred to in the findings.

- Nmap – https://nmap.org
- Burp Suite Professional – https://portswigger.net/burp/pro
- Semgrep – https://semgrep.dev
- Trufflehog – https://github.com/trufflesecurity/trufflehog
- Testssl.sh – https://github.com/drwetter/testssl.sh

4 Findings

We have identified the following issues:

4.1 MAF-009 — Zip file functionality allows unauthorized modelset downloads

Vulnerability ID: MAF-009

Vulnerability type: Insecure ACL

Threat level: High

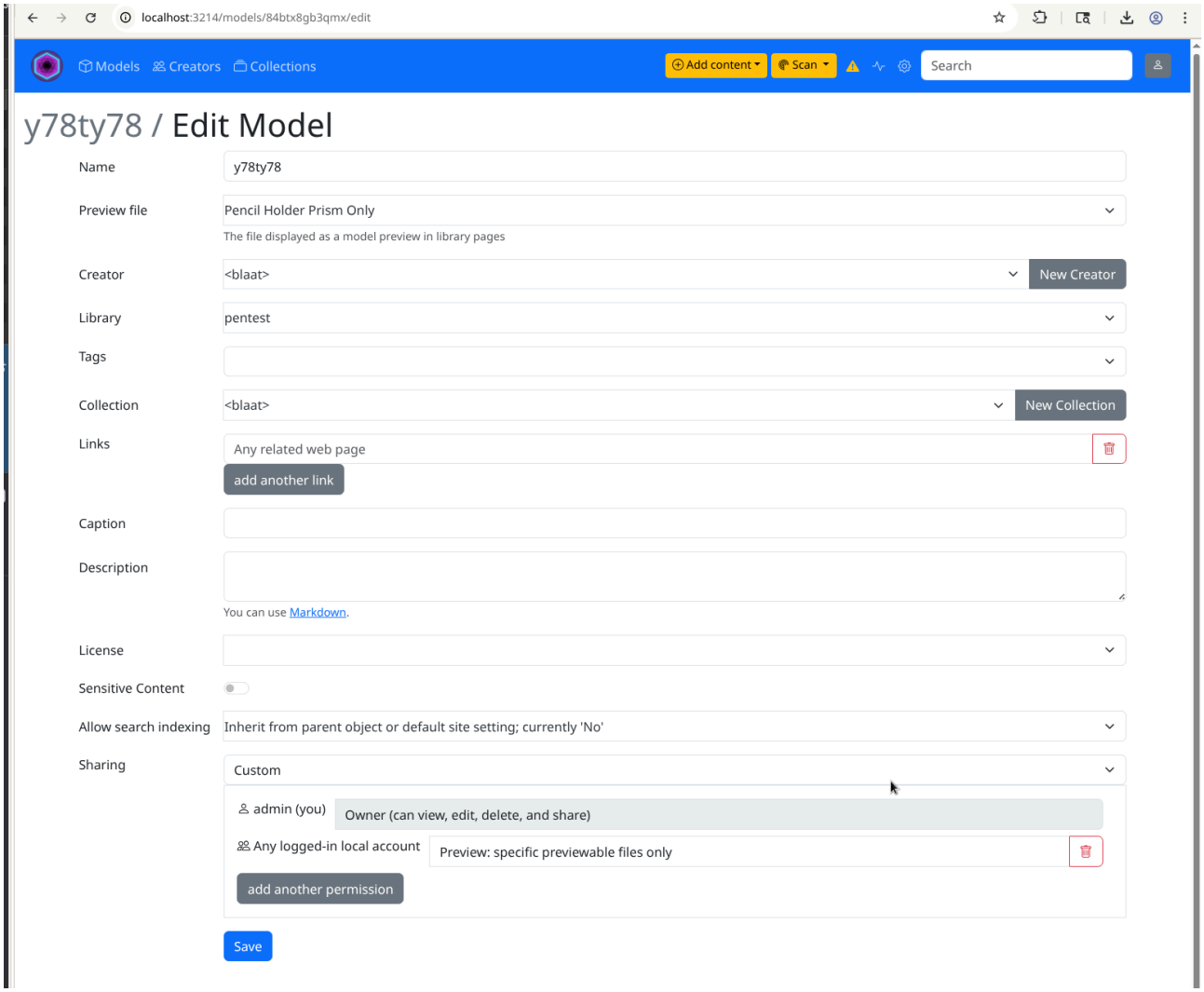
Description:

The application's zip file functionality does not properly enforce access controls, allowing users to download modelsets even when sharing is set to private.

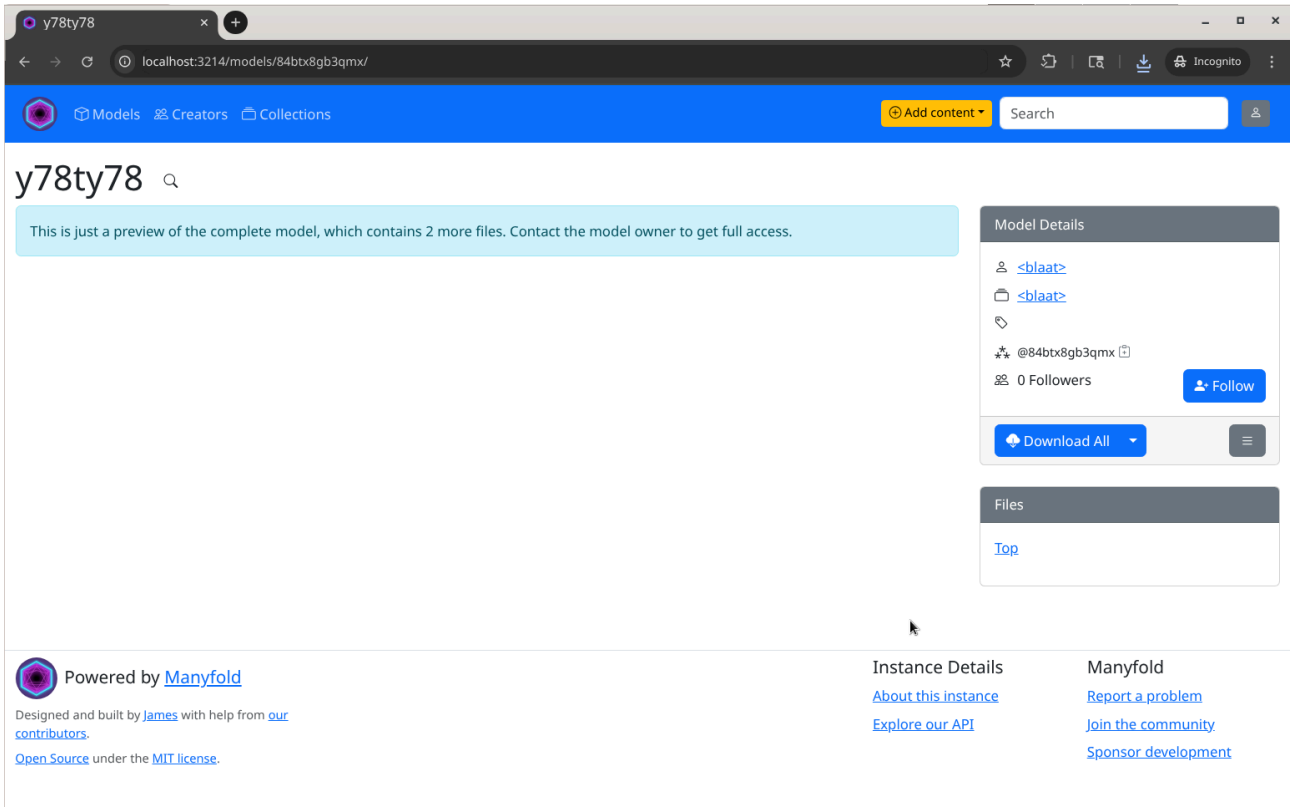
Technical description:

When a user sets the sharing settings for their modelset to custom (preview specific files only), the intended behavior is that other users should only see the preview but not be able to download the full modelset. However, we discovered that even with these restrictions in place, authenticated users can still use the zip file functionality to download the entire modelset contents by leveraging a preview mode loophole.

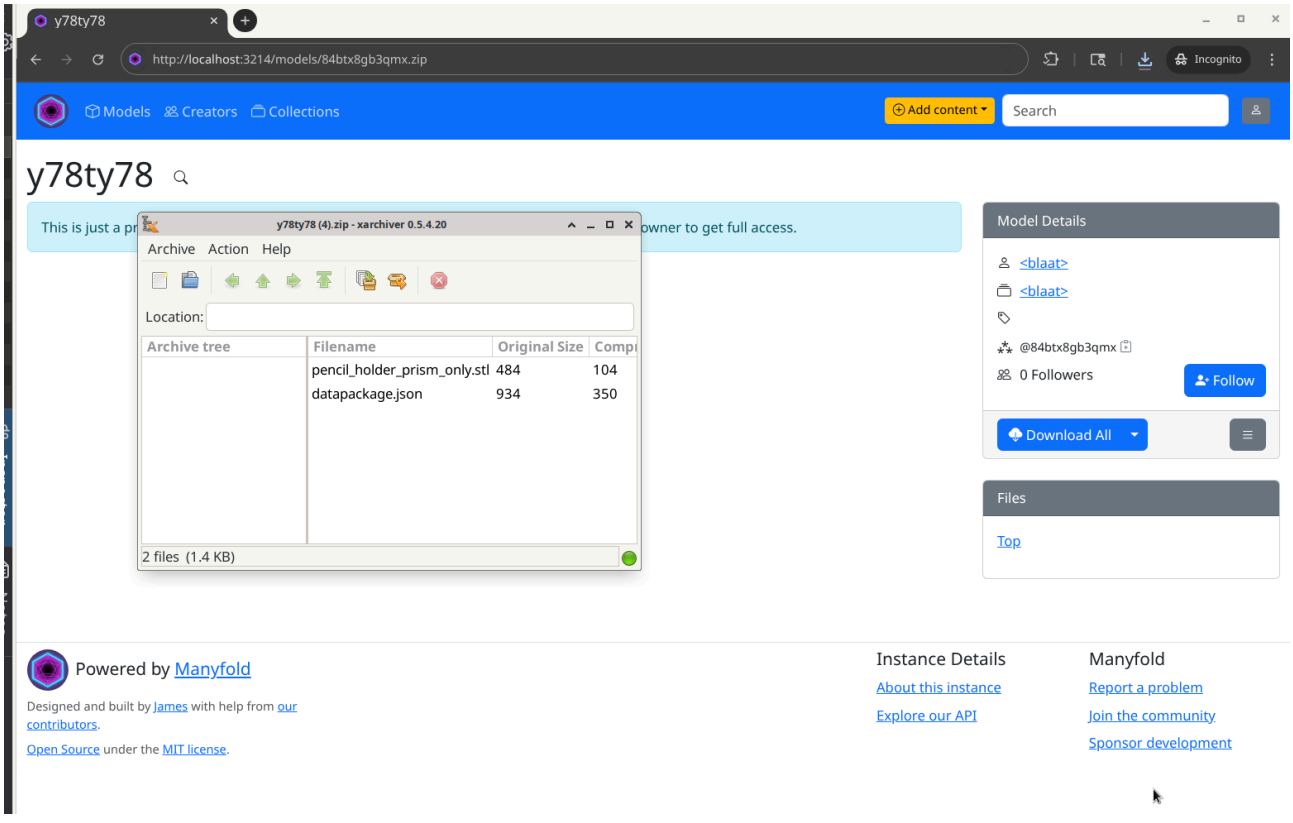
All logged-in users are only allowed to see preview-specific files:



When they open the model, they see only the preview:



However, when downloading all model files using the zip functionality, all the files appear in the zip archive:



Impact:

- Unauthorized access to potentially sensitive or confidential data within private modelsets.

Recommendation:

- Implement proper access controls and authorization checks on the zip file download functionality to ensure it respects sharing settings.

4.2 MAF-010 — Zip file path traversal and denial-of-service

Vulnerability ID: MAF-010

Vulnerability type: Path Traversal

Threat level: Elevated

Description:

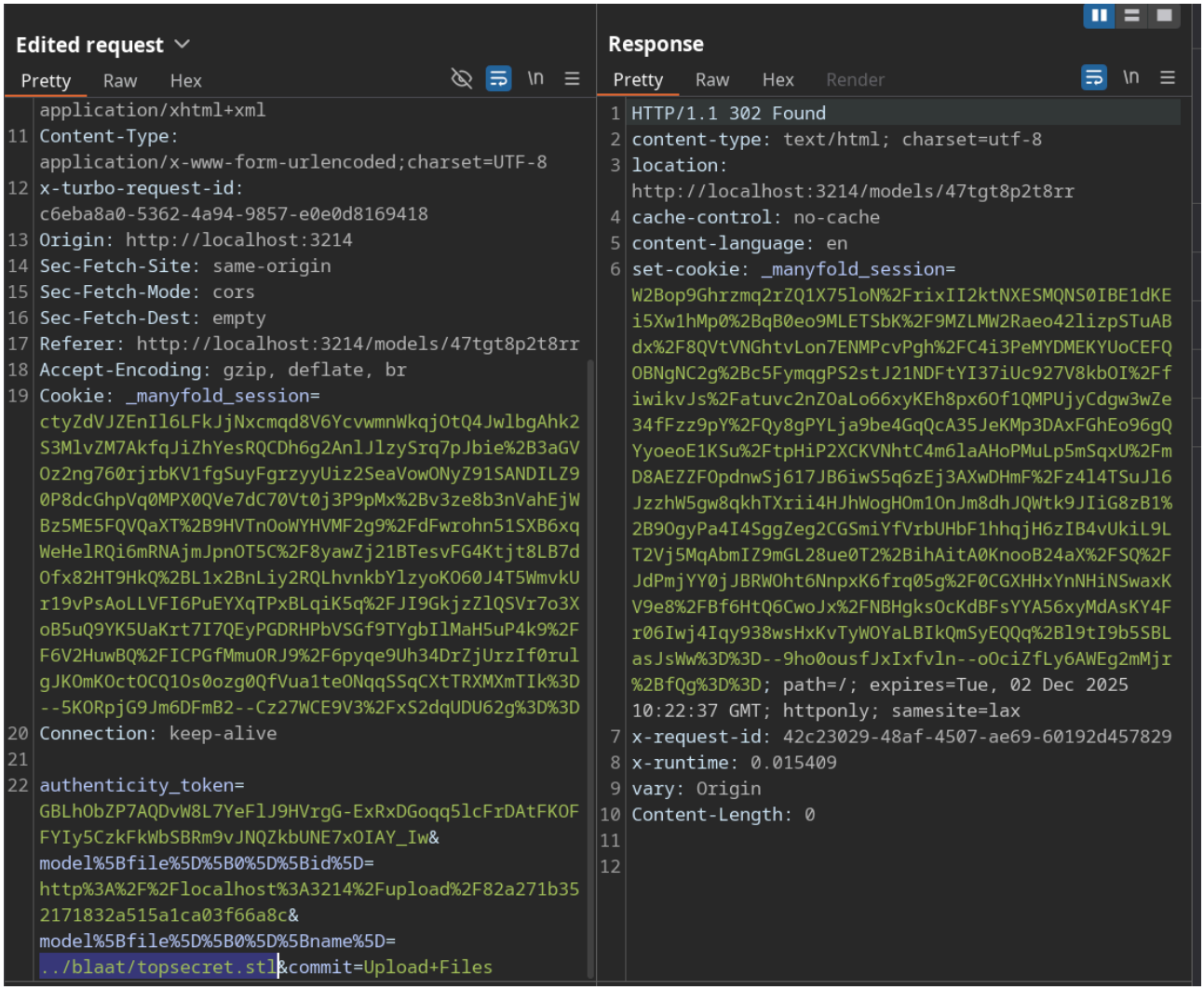
A path traversal flaw exists in how uploaded zip files are processed.

Technical description:

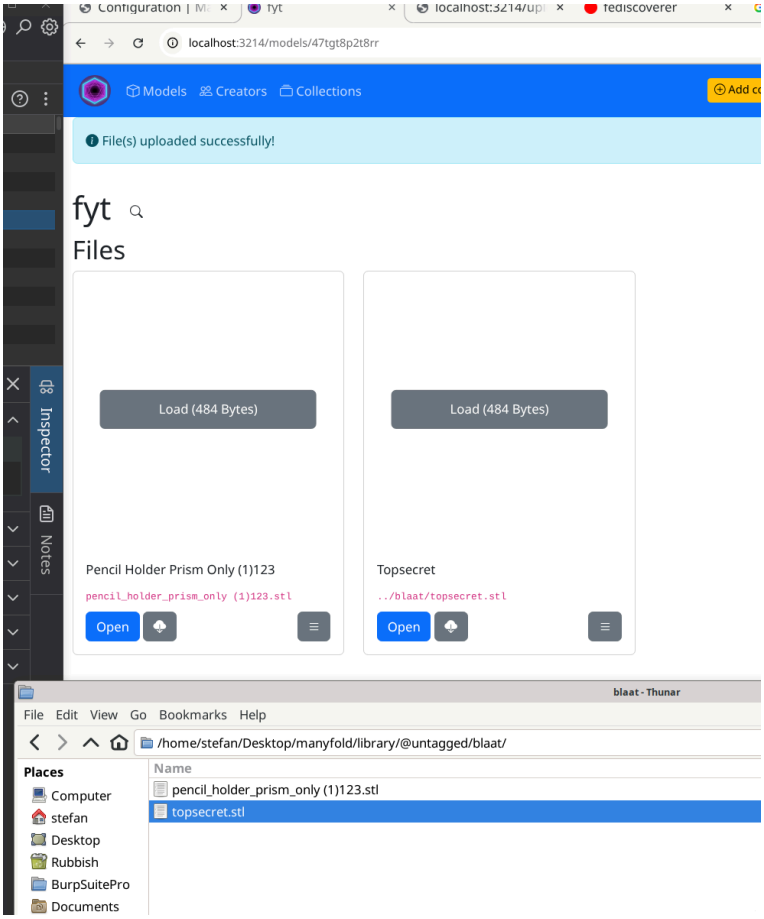
The vulnerability stems from improper validation of the `Filename` parameter during the processing of uploaded zip files. By supplying a maliciously crafted filename that includes relative path sequences (e.g., `../../coin_low1.fbx`), an attacker can traverse outside the intended upload directory and overwrite files located elsewhere on the server's filesystem.

Note that due to the limited accepted file types it was not possible to overwrite files beyond those that were allowed, limiting the attack vector.

Changing the filename adding path traversal characters:



Notice that the filename contains these characters and the file is placed in another directory.



Additionally, when dealing with these deeply nested paths within the uploaded zip file structure, the application entered a resource-intensive loop that consumed excessive CPU and memory resources, which could lead to a denial-of-service condition.

Impact:

- Arbitrary file upload/deletion outside the intended directory.
- Denial-of-service due to resource exhaustion from processing deep path structures.

Recommendation:

- Implement strict input validation on the `Filename` parameter to prevent path traversal sequences.
- Use a dedicated temporary directory for handling uploaded files and sanitize all user-supplied filenames.
- Set appropriate rate limiting on the zip download functionality to mitigate DoS risks.

4.3 MAF-008 — Unauthorized permission escalation in model sharing

Vulnerability ID: MAF-008

Vulnerability type: Privilege Escalation

Threat level: Elevated

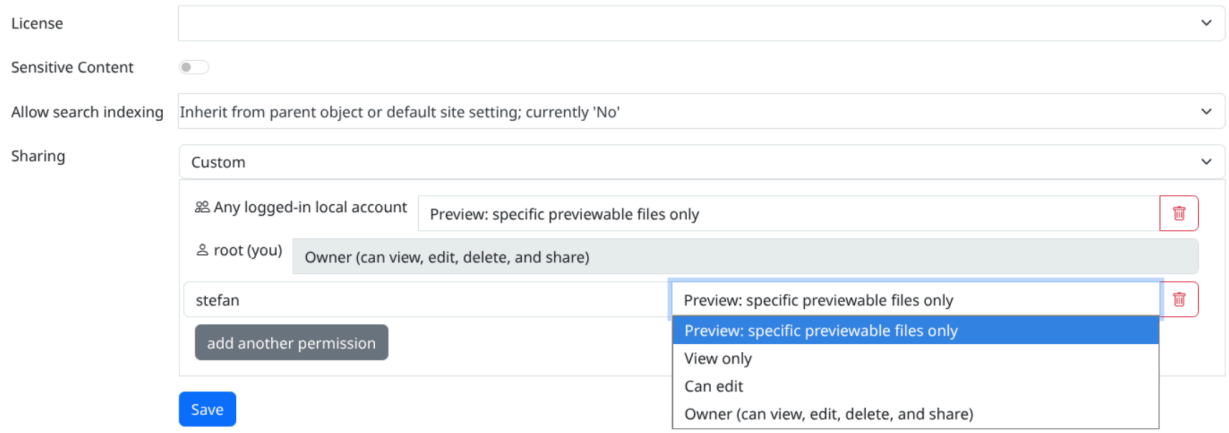
Description:

Users with edit permissions on models can grant permissions to other users and even assign owner permissions.

Technical description:

The application allows users assigned with edit permissions on a particular model to grant access to other users.

In the user interface, it is not clear what edit actually means. It suggests an ability to edit model information, not to assign permissions to other users; a separate role would be expected for permission management.



Additionally, while users cannot give themselves higher privileges, they can grant other users permissions that exceed their own. For instance, a user with edit rights can give another user owner rights to the model, which would allow deletion of the model.

Impact:

- Potential exposure of confidential information through unintended sharing with unauthorized parties.
- Violation of intended access control policies and segregation of duties.

Recommendation:

- Implement strict authorization checks before processing any permission change requests, ensuring users cannot grant privileges higher than those they possess.
- Introduce another role that allows managing user permissions.

4.4 MAF-004 — OIDC account auto-linking via unverified email

Vulnerability ID: MAF-004

Vulnerability type: Account Takeover

Threat level: Moderate

Description:

The application links OIDC accounts to existing password-based accounts by email without verifying ownership of the address.

Technical description:

The `User.from_omniauth(auth)` method matches users solely based on the email from OIDC, then updates their account with OIDC credentials. An attacker can register an OIDC account with a victim's email and take over their Manyfold password-based account if they use an OIDC provider that doesn't verify emails.

Impact:

- An attacker can hijack a victim's account without interaction.
- Bypasses the normal password reset flow.
- Provides access to the victim's data.

Rated as Moderate because the email account usually already exists and is under the user's control, but the risk remains if identity verification is weak.

Recommendation:

- Verify the `email_verified` claim from the OIDC provider.
- Require users to be logged in before linking accounts.

- Implement additional verification steps for email ownership.

4.5 MAF-001 — Lack of rate limiting on OAuth and OpenID endpoints

Vulnerability ID: MAF-001

Vulnerability type: Lack of Rate Limiting

Threat level: Low

Description:

The exposed OAuth client-credentials token and the optional OpenID Connect login endpoints do not have rate-limiting controls.

Technical description:

We observed that the `/oauth/token` and `/users/auth/openid_connect` API endpoints do not implement any form of rate limiting.

Impact:

- Attackers can attempt to brute-force OAuth client IDs/secrets or flood token-issuance attempts on `/oauth/token` without per-IP or per-client throttling.
- When OIDC is enabled, unauthenticated requests to `/users/auth/openid_connect` and its callback can be hammered without limits, bypassing the protections applied to native login and enabling credential-stuffing or login-spam attacks from a single source.

Recommendation:

- Add per-IP throttles for `/oauth/token` and the OpenID Connect entrypoint/callback to cap token-issuance and OIDC login attempts.
- Log and alert on repeated failures or throttled requests.

4.6 MAF-002 — Fediverse connections allow execution of potentially unsafe data from other servers

Vulnerability ID: MAF-002

Vulnerability type: Insufficient Input Validation

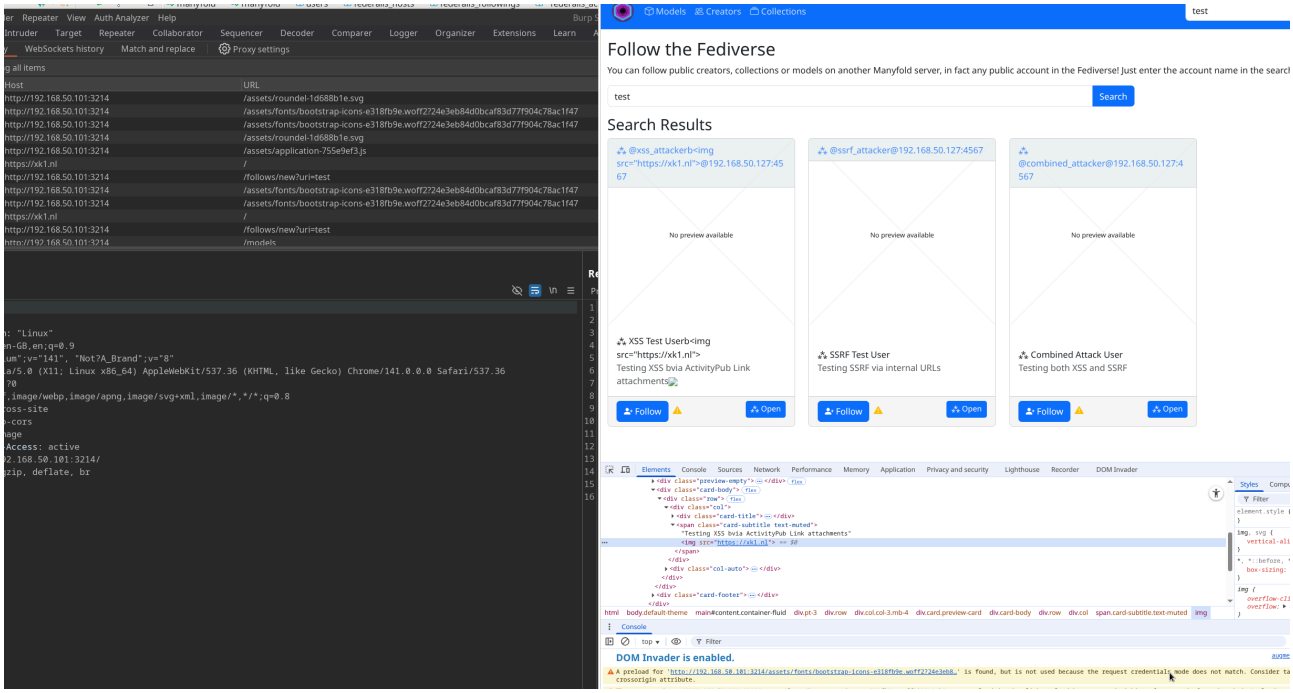
Threat level: Low

Description:

The Fediverse feature retrieves data from connected servers, and it is not sanitized properly before being displayed or executed.

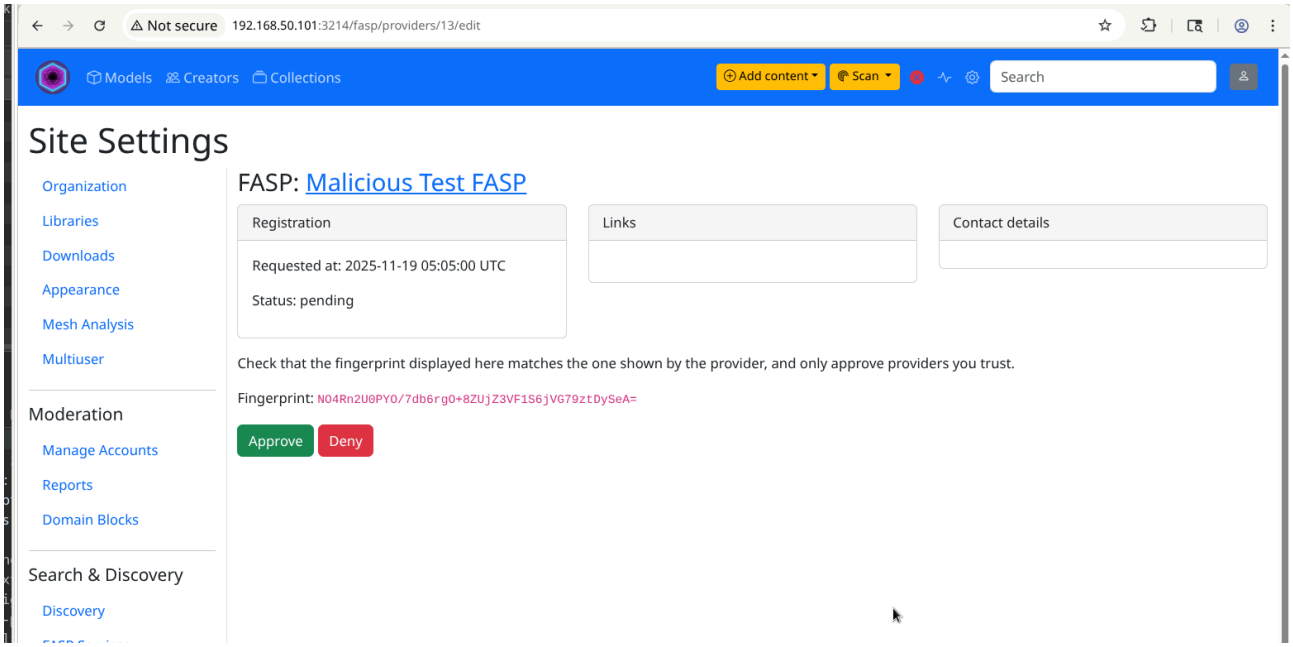
Technical description:

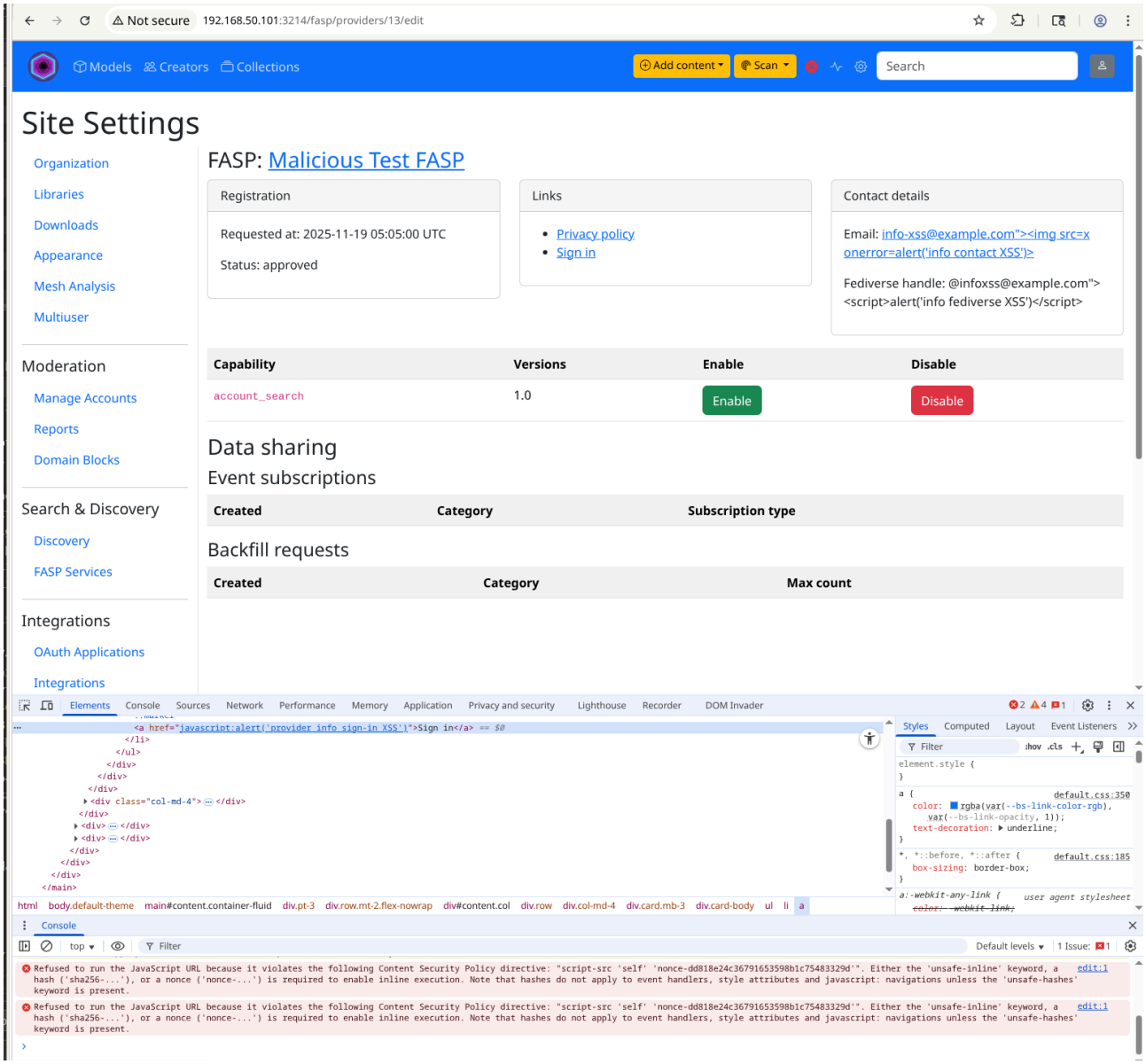
When a user conducts a search on a connected Fediverse server by entering a URI (e.g., `http://192.168.50.101:3214/follows/new?uri=test`), the application fetches metadata from that server, including fields like `preferredUsername`, `name`, and `summary`.

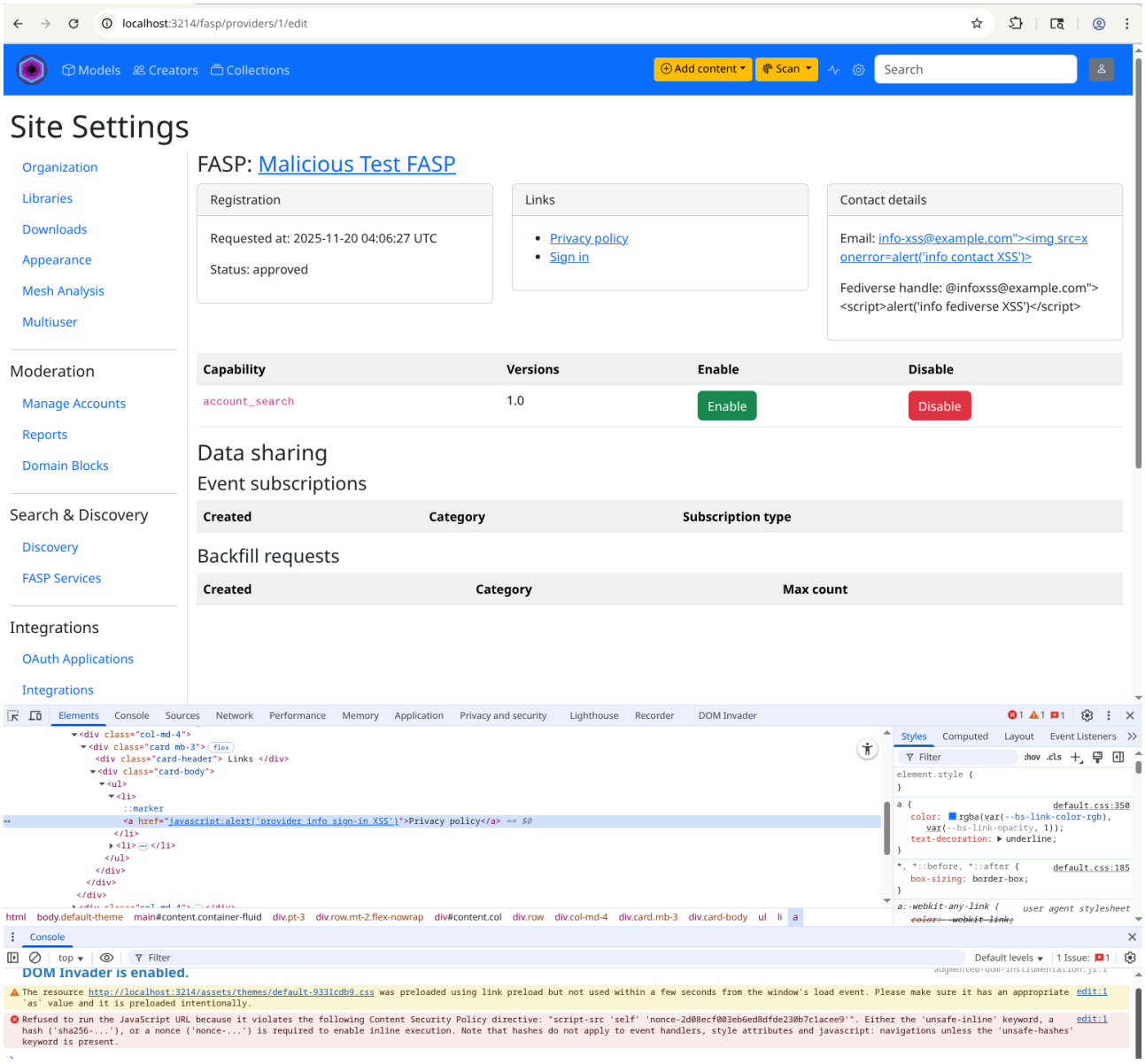


While some malicious input in these fields is stripped (such as XSS attempts), other potentially unsafe content, such as the externally loaded image shown in the screenshot above using the `img` tag, can still be executed in Manyfold.

Additionally, when a FASP server registers (which requires approval by a Manyfold admin), malicious data can be added:


Not secure
192.168.50.101:3214/fasp/providers/13/edit
Models
Creators
Collections
Add content
Scan
Search
Site Settings
Organization
Libraries
Downloads
Appearance
Mesh Analysis
Multiuser
Moderation
Manage Accounts
Reports
Domain Blocks
Search & Discovery
Discovery
FASP: Malicious Test FASP
Registration
Requested at: 2025-11-19 05:05:00 UTC
Status: pending
Links
Contact details
Check that the fingerprint displayed here matches the one shown by the provider, and only approve providers you trust.
Fingerprint: NO4Rn2U0PYO/7db6rgO+8ZUjZ3VF1S6jVG79ztDySeA=
Approve
Deny


Not secure 192.168.50.101:3214/fasp/providers/13/edit
Models Creators Collections
Add content Scan
Search
Site Settings
Organization
Libraries
Downloads
Appearance
Mesh Analysis
Multiuser
Moderation
Manage Accounts
Reports
Domain Blocks
Search & Discovery
Discovery
FASP Services
Integrations
OAuth Applications
Integrations
FASP: Malicious Test FASP
Registration
Requested at: 2025-11-19 05:05:00 UTC
Status: approved
Links
Privacy policy
Sign in
Contact details
Email: info-xss@example.com"><img src=x onerror=alert('info contact XSS')>
Fediverse handle: @infoxss@example.com"><script>alert('info fediverse XSS')</script>
Capability
Versions
Enable
Disable
account_search
1.0
Enable
Disable
Data sharing
Event subscriptions
Created
Category
Subscription type
Backfill requests
Created
Category
Max count
Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse Recorder DOM Invader
Sign in == $0
Styles Computed Layout Event Listeners
Refused to run the JavaScript URL because it violates the following Content Security Policy directive: "script-src 'self' 'nonce-dd818e24c36791653598b1c75483329d'". Either the 'unsafe-inline' keyword, a hash ('sha256-...'), or a nonce ('nonce-...') is required to enable inline execution. Note that hashes do not apply to event handlers, style attributes and javascript: navigations unless the 'unsafe-hashes' keyword is present.



The CSP blocks execution of malicious links containing JavaScript code, but it is better to block potentially malicious code entirely in case a CSP bypass is found.

Impact:

- Attackers could map user IP addresses visiting the application by abusing allowed tags such as `<img>`.

Recommendation:

- Implement strict sanitization and input validation for all fields retrieved from external Fediverse servers before displaying or executing them.

- Only allow HTML tags and attributes that are absolutely necessary for the application's functionality and user experience.
- Use a mature library to safely sanitize any content received from external sources.

4.7 MAF-003 — OAuth app client secrets shown in plain text

Vulnerability ID: MAF-003

Vulnerability type: Credential Exposure

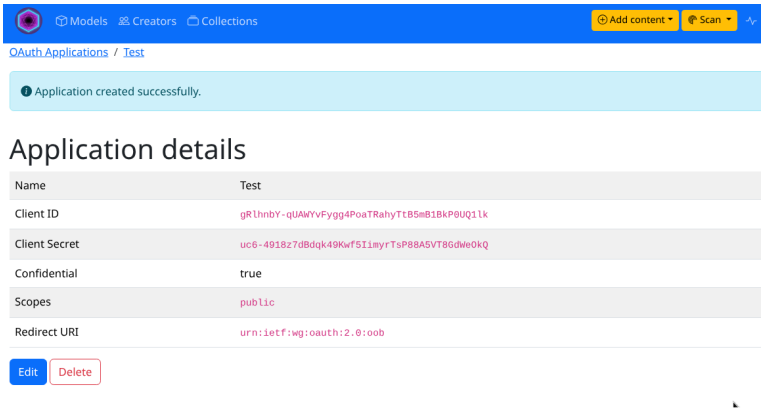
Threat level: Low

Description:

Client secrets for OAuth applications are exposed in plain text within the application's user interface.

Technical description:

Client secrets associated with OAuth integrations are displayed in plain text directly within the UI:



Impact:

- Potential leak of credentials if a bystander can view the screen, or it is inadvertently shown while screen sharing.

Recommendation:

- Only show the full secret value once when first generated, then mask it going forward.

- Regularly rotate and expire client secrets according to security best practices.

4.8 MAF-005 — First-user auto-admin flow poses security risks

Vulnerability ID: MAF-005

Vulnerability type: Insecure Default Configuration

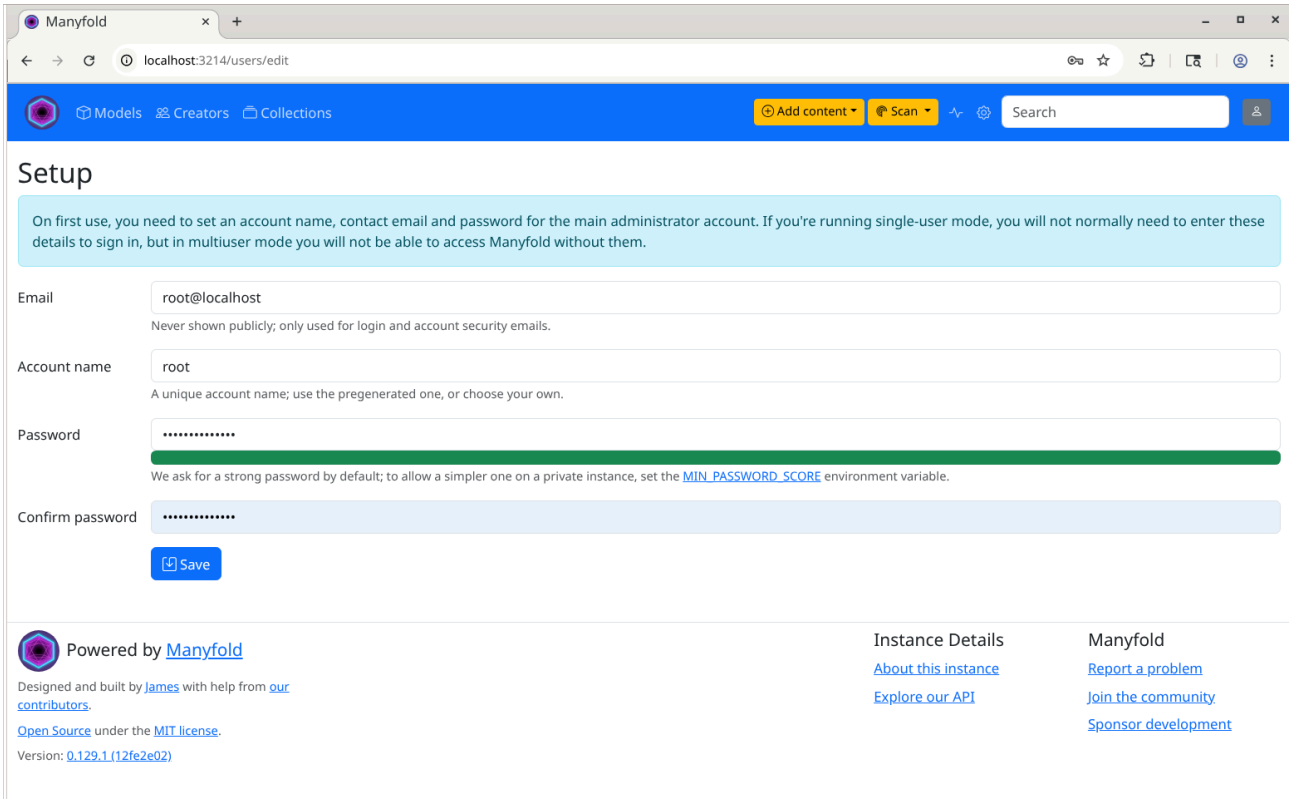
Threat level: Low

Description:

The first user to authenticate automatically receives administrator privileges.

Technical description:

Upon a fresh installation, the first user who successfully authenticates using normal login is automatically granted full administrative privileges.



While an attacker need to know that a server is being set up to be able to attack it, this is often revealed through certificate transparency logs, as obtaining certificates often happens just before app installations.

Impact:

- Unauthorized users could obtain administrator-level access to the system. The issue is rated low-severity because administrators are expected to read the documentation and review the application before running it, but this default still expands the attack surface unnecessarily.
- Sensitive data and functionality may be exposed to malicious actors.
- The integrity of the platform's security controls could be compromised.

Recommendation:

- Establish secure procedures for initializing administrative accounts during setup. For instance, generate a random administrative account and password during setup and show this in the console when running the application for the first time, after which the user can create their own admin account.
- Provide clear documentation and guidance on securely configuring the auto-admin flow.

4.9 MAF-007 — User passwords not obfuscated during creation

Vulnerability ID: MAF-007

Vulnerability type: Information Exposure

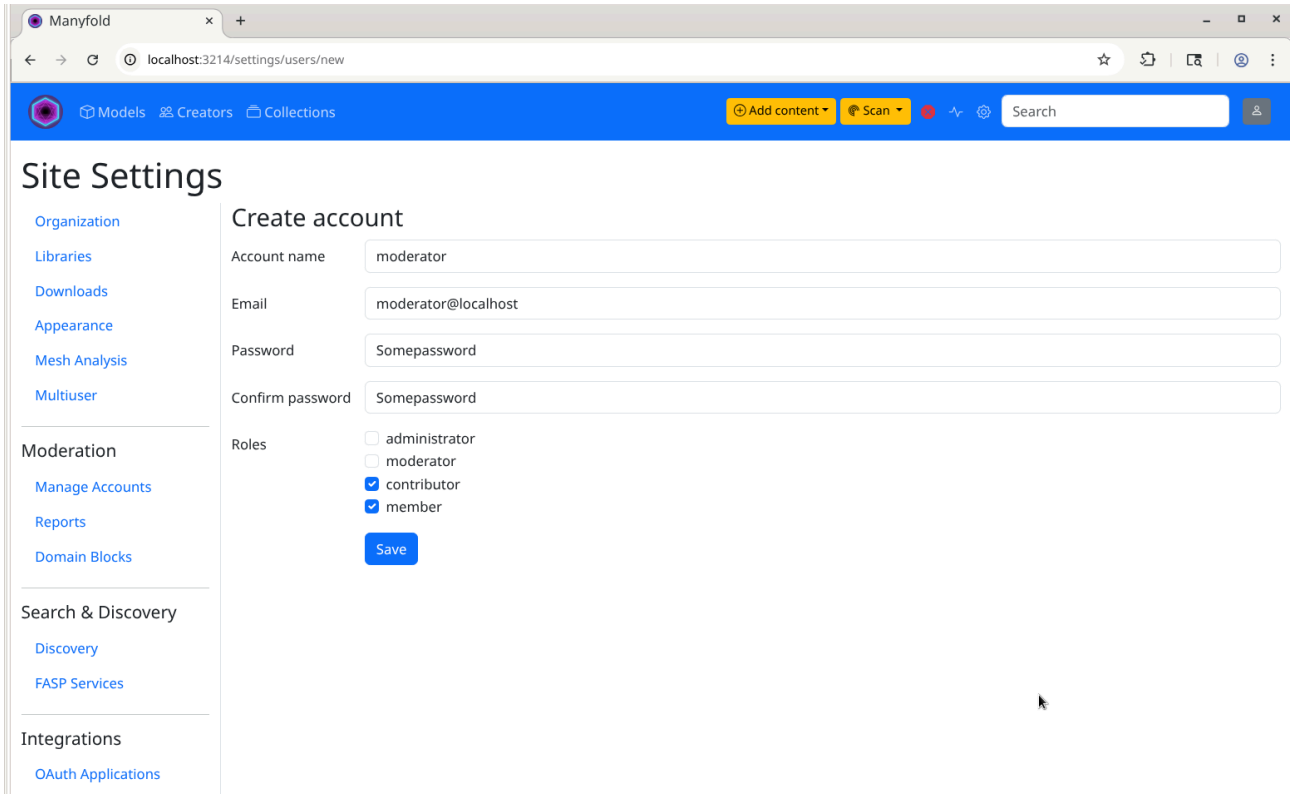
Threat level: Low

Description:

When administrators create new user accounts and set passwords, the passwords are displayed in plaintext instead of being obscured.

Technical description:

When an administrator creates a new user account through the web interface, the password entered by the admin is visible on-screen as they type. This lack of obfuscation occurs even though sensitive data such as passwords should be masked to prevent shoulder surfing and other casual observation attacks.



Impact:

- Increases risk of password compromise through shoulder-surfing, accidental screen sharing, or other observation attacks.

Recommendation:

- Implement input masking on all password fields, obscuring characters as they are typed using asterisks or dots. Standard HTML `password` input tags do this.

5 Future Work

- **Retest of findings**
  When mitigations for the vulnerabilities described in this report have been deployed, perform a repeat test to ensure that they are effective and have not introduced other security problems.
- **Regular security assessments**
  Security is a process that must be continuously evaluated and improved; this penetration test is just a single snapshot. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security.

6 Conclusion

During this crystal-box penetration test of the Manyfold application we found 1 High, 2 Elevated, 1 Moderate and 5 Low-severity issues.

Overall, the Manyfold application maintains a solid security posture with modern functionality and active maintenance practices. The crystal-box penetration test showed that several important improvements are still needed around access control, identity management, and the secure handling of user-supplied data.

The application maintains good practices in its security measures, and we did not identify major issues such as SQL injection or remote code execution during this engagement.

We were positively impressed by the team's responsiveness, transparency, and use of issue tracking to follow up on previous findings. We observed that a number of findings from our previous pentest have been resolved. It was a pleasure to work with the Manyfold team, and we are confident that, while maintaining focus on security, the project can continue to build on this strong foundation.

We recommend fixing all of the issues found and then performing a retest in order to ensure that mitigations are effective and that no new vulnerabilities have been introduced.

Finally, we want to emphasize that security is a process that must be continuously evaluated and improved – this penetration test is just a one-time snapshot. We hope that this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end.

Please don't hesitate to let us know if you have any further questions, or need further clarification on anything in this report.

Appendix 1 Testing team

Stefan Vink	With over 20 years in IT and a decade specializing in penetration testing, Stefan brings exceptional cybersecurity expertise to global organizations. His technical proficiency spans security automation, system monitoring, web development, AI and Windows/ Linux administration. Stefan holds prestigious certifications including MCITP, CCNA, LPIC, OSCP, and has completed the CISSP examination, validating his comprehensive security knowledge. Beyond his professional achievements, Stefan enjoys travel, hiking, tennis, and chess. His passion for AI and automation extends throughout both his professional and personal interests. He currently lives with his family in Melbourne, Australia.
Melanie Rieback	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.