Penetration Test Report

Open Tech Fund

V 1.0
Diemen, December 13th, 2019
Confidential

Document Properties

Client	Open Tech Fund
Title	Penetration Test Report
Targets	Source Code Audit and Pentest of https://github.com/globaleaks/GlobaLeaks/ Encryption System https://docs.google.com/document/ d/1Yn4OM5XO5G0PXDSlHYaEa5BsAucBmxhmBfRqhrAg_Jc Multi-Tenancy feature
Version	1.0
Pentesters	Pierre Pronchery, Stefan Vink
Authors	Stefan Vink, Pierre Pronchery, Marcus Bointon
Reviewed by	Marcus Bointon
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	November 5th, 2019	Stefan Vink	Initial draft
0.2	November 28th, 2019	Pierre Pronchery	Incorporated the findings from the source code audit
0.3	November 30th, 2019	Stefan Vink	Completed the report
0.4	December 2nd, 2019	Marcus Bointon	Review
0.5	December 12th, 2019	Stefan Vink	Added vendor feedback and retesting info.
1.0	December 13th, 2019	Pierre Pronchery	Updated the status on the issues retested.

Contact

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

Name	John Sinteur
Address	Overdiemerweg 28 1111 PP Diemen The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

Between October 28, 2019 and November 28, 2019, Radically Open Security B.V. carried out a penetration test for Open Tech Fund - GlobaLeaks

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 targets:

- Source Code Audit and Pentest of https://github.com/globaleaks/GlobaLeaks/
- Encryption System https://docs.google.com/document/d/1Yn4OM5XO5G0PXDSlHYaEa5BsAucBmxhmBfRqhrAg_Jc
- Multi-Tenancy feature

A breakdown of the scoped services can be found below:

- Scoping effort: 1 days
- Pentest GlobaLeaks frontend (incl. reporting): 5.5 days
- Source code audit GlobaLeaks server (incl. reporting): 5.5-16 days
- Review of the application encryption and design (incl. reporting): 1 days
- Retest and fix verification: 2 days
- **Total effort: 15 - 25.5 days**

1.3 Project objectives

The test is intended to gain insight into the security of the GlobaLeaks Whistleblower web application.

1.4 Timeline

The Security Audit took place between October 28, 2019 and November 28, 2019.

1.5 Results In A Nutshell

During the penetration test 2 High, 4 Elevated, 8 Moderate and 10 Low level severity issues were discovered.

The pentest revealed a severe access control vulnerability OTF-001 (page 16) in the multi-tenant admin authorization functionality that allowed a tenant admin to issue requests as the root admin and other tenant admin users which could give an attacker full control over the application and access to highly sensitive information. The implementation of the encryption functionality could expose sensitive data. One field is not encrypted when stored in the database OTF-004 (page 24). Admin is able to set a new password for a recipient OTF-003 (page 23) without the need for a recovery key. The Moderate and Low issues are mainly related to insecure communication, password policy, file upload, security misconfiguration, and broken access controls.

The review of the application encryption revealed two elevated OTF-003 (page 23) OTF-004 (page 24) and one moderate OTF-012 (page 42) finding. When encryption is enabled a recipient is only able to reset their password by using their recovery key. However, when an admin sets a new password for a recipient this is not required. Allowing administrators to reset user passwords and gain access to encrypted user data is a risk. For usability this is good but it reduces the overall security of the encrypted data. Based on the purpose of the application ROS suggests to choose security above useability and always require the recovery key to decrypt recipients data even when an admin sets the new password.

Auditing the database revealed that not all whistleblower sensitive data gets encrypted. The shorttitle field in the internal tip database table does not get encrypted when encryption has been enabled. In addition to this issue, no similar issues were found

Encrypting Clients Data is not enabled by default, existing tips and proof remain unencrypted and requires additional steps to work for new tips. Note that the developers mentioned that the encryption functionality is disabled by default, still an experimental feature and will be mandatory once it is not experimental anymore. Making the encryption mandatory will be a good solution and would solve the issues mentioned.

Beside the three findings no vulnerabilities have been identified on the protocol design which is following best practices. Users keys are curve25519 keys and are generated serverside at first user login. Private users keys are stored on the filesystem and encrypted using symmetric encryption. The symmetric key used for encrypting users keys is derived from the user password using the KDF function Argon2ID. Symmetric encryption is performed using XSalsa+Poly1305 and asymmetric encryption is performed using curve2519+XSalsa+Poly1305.

The source code audit revealed a behaviour possibly exposing server instances of GlobaLeaks in OTF-005 (page 29). The continuous integration tests were vulnerable to remote code execution in OTF-006 (page 30). Local privilege escalation was possible during the installation process in OTF-022 (page 69) or while making backups in OTF-024 (page 72). The source code also explicitly attempts to securely erase files in OTF-014 (page 45), although in practice this may have little to no effect on most combinations of modern operating systems, hardware (e.g. storage), and virtualization platforms. Besides a potential conflict when storing key material in OTF-023 (page 71), SSL/TLS connections were not validating certificates in OTF-002 (page 21), which could lead to information leaks in outgoing HTTPS and SMTP connections.

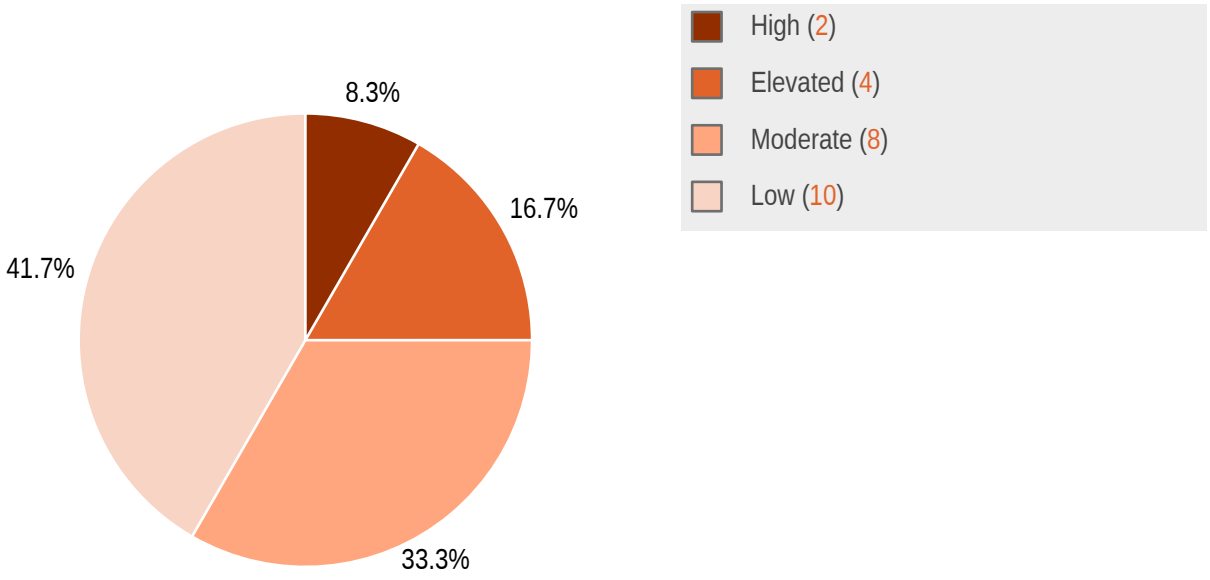
Note that after the pentest many of the issues mentioned in this report have been fixed (in GlobaLeaks version 3.11.50) or are to be fixed by the vendor. See the feedback section in each finding for more information.

1.6 Summary of Findings

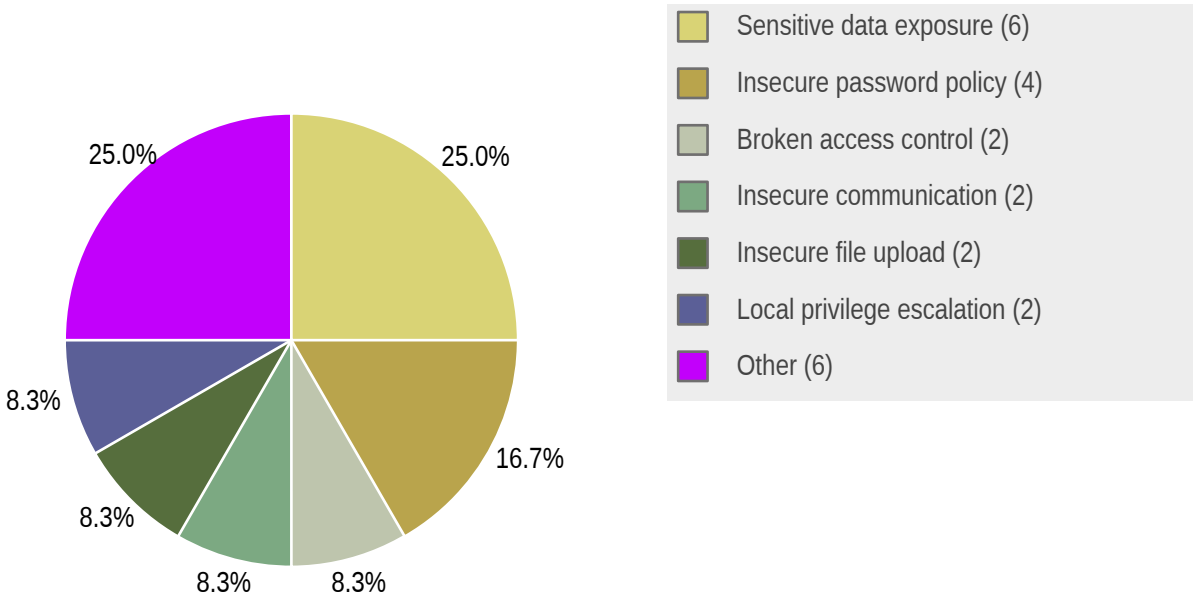
ID	Type	Description	Threat level
OTF-001	Broken Access Control	While logged in as a tenant admin, it is possible to issue requests as the root admin and other tenant admin users.	High
OTF-002	Insecure Communication	GlobaLeaks uses the OpenSSL library wrapper for Python for SSL/TLS connections, both as client or server. It does not seem to enforce any verification of remote certificates.	High
OTF-003	Sensitive Data Exposure	When encryption is enabled a recipient is only able to reset their password by using their recovery key. However, when an admin sets a new password for a recipient this is not required.	Elevated
OTF-004	Sensitive Data Exposure	The shorttitle field is not encrypted when encryption is enabled.	Elevated
OTF-005	Sensitive Data Exposure	An installation script was found to report information to a server controlled by GlobaLeaks when running.	Elevated
OTF-006	Remote Code Execution	A script meant to set up the continuous integration system will download and execute code directly from a remote server.	Elevated
OTF-007	Insecure Communication	After the setup wizard has been completed the platform is accessible over an insecure HTTP connection.	Moderate
OTF-008	Insecure Password Policy	A default password is created for the recipient user during the application setup.	Moderate
OTF-009	Insecure Password Policy	During the installation process the strength of the new admin password is not checked.	Moderate
OTF-010	Insecure File Upload	As a whistleblower it is possible to upload malicious files.	Moderate
OTF-011	Insecure Password Policy	The application does not have any account lockout mechanisms.	Moderate
OTF-012	Sensitive Data Exposure	During testing several data encryption issues were found.	Moderate
OTF-013	Unprotected Passwords	The external Backup and SMTP server passwords are saved in plain text in the database.	Moderate
OTF-014	Sensitive Data Exposure	There is a function intended to erase files securely by overwriting, but it is unlikely to work as intended.	Moderate
OTF-015	Insecure Password Policy	It is possible to set weak passwords by disabling the client-side password strength checks.	Low
OTF-016	Input Validation	The application does not validate or incorrectly validates input.	Low

OTF-017	Sensitive Data Exposure	A whistleblower's tip was posted using an insecure connection, but the submission status indicates that it was posted using a secure HTTPS connection.	Low
OTF-018	Broken Access Control	Submissions can be disabled, however, the way it's disabled doesn't actually prevent new submissions being submitted.	Low
OTF-019	Input validation	When logged in as an admin it is possible to set the option "Allow the following websites to embed the platform inside iframes". This option does not validate whether the user input is a URL, resulting in inserting other tags in the CSP header.	Low
OTF-020	Insecure File Upload	Only PNG image files should be allowed as logo uploads. However this check is only performed on the client side which can be disabled, resulting in allowing upload of files of any type.	Low
OTF-021	Security Misconfiguration	On a default install the SSH port of the server is publicly accessible.	Low
OTF-022	Local Privilege Escalation	An installation script writes to a file path relative to the current directory, without knowledge of the user running the script. This can be abused by local users in some conditions. The script is expected to be executed with the privileges of the root user.	Low
OTF-023	Data Corruption	GlobaLeaks generates a private key per tip received. This key is identified through a random string of 16 alphanumeric characters. Should the random number generator generate the same sequence as an existing key, the new key will be appended to the existing one.	Low
OTF-024	Local Privilege Escalation	Backups use the system /tmp directory in a way that may be exploitable.	Low

1.6.1 Findings by Threat Level



1.6.2 Findings by Type



1.7 Summary of Recommendations

ID	Type	Recommendation
OTF-001	Broken Access Control	• Correct the code to deny access to unauthorized users.
OTF-002	Insecure Communication	• Enforce certificate validation for SSL/TLS connections.
OTF-003	Sensitive Data Exposure	• Always require the recovery key to decrypt recipients data even when an admin sets the new password.
OTF-004	Sensitive Data Exposure	• Encrypt the `shorttitle` field in the database.
OTF-005	Sensitive Data Exposure	• Do not contact remote servers automatically with information (or metadata) that might compromise users or instances.
OTF-006	Remote Code Execution	• Do not trust code obtained directly from remote servers.

OTF-007	Insecure Communication	• Configure HTTPS automatically by using Letsencrypt during installation.
OTF-008	Insecure Password Policy	• Show a prominent warning in the admin interface or block access if the passwords for admin and recipient users have not been changed from defaults. • Show a warning in the admin interface when 2FA is not enabled. • Optionally generate random usernames during setup for both users.
OTF-009	Insecure Password Policy	• Validate new passwords against a strong password policy.
OTF-010	Insecure File Upload	• Scan uploaded files for malware. • Only open uploaded files in sandboxed environments.
OTF-011	Insecure Password Policy	• Lock out accounts temporarily after a number of failed login attempts.
OTF-012	Sensitive Data Exposure	• Enable encryption by default. • Ensure the current encryption status is displayed clearly on admin pages.
OTF-013	Unprotected Passwords	• Encrypt sensitive data such as passwords to prevent an attacker with read access from using them. • For passwords that only need to be verified a hashing algorithm such as Argon2 can be used. • For passwords that need to be used for services (and need to be known to the application), use symmetric encryption with integrity checking, such as libsodium provides.
OTF-014	Sensitive Data Exposure	• Do not assume "secure deletion" is actually effective. • Use filesystems backed only by volatile memory (e.g. RAM disks) to store temporary data in the clear.
OTF-015	Insecure Password Policy	• Implement and enforce a strong, consistent password policy on the server side as well as the client side. • Enable 2FA by default.
OTF-016	Input Validation	• Sanitize and validate all input data. • Escape all output appropriately.
OTF-017	Sensitive Data Exposure	• Modify the submission response to display the correct status.

OTF-018	Broken Access Control	• Verify the submission setting on the server side, and reject submissions when they are disabled.
OTF-019	Input validation	• Validate URLs before using them in CSP configuration.
OTF-020	Insecure File Upload	• Validate file and content type on the server side before accepting and storing uploaded files on the server.
OTF-021	Security Misconfiguration	• Restrict access by installing a firewall and restricting SSH access to specific source IPs.
OTF-022	Local Privilege Escalation	• Create this log file in a location inaccessible to local users.
OTF-023	Data Corruption	• Make sure collisions are detected and prevented sufficiently early in the submission process.
OTF-024	Local Privilege Escalation	• Generate files securely when writing to shared folders.

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, etc., afforded to the network. This usually involves trying to discover publicly available information by utilizing a web browser, visiting newsgroups, etc. An active form would be more intrusive and may show up in audit logs and may 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.

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

Through automated scans we were able to gain the following information about the software and infrastructure. Detailed scan output can be found in the sections below.

3.1 Automated Scans

As part of our active reconnaissance we used the following automated scans:

- nmap – https://nmap.org
- testssl.sh – https://testssl.sh

4 Findings

We have identified the following issues:

4.1 OTF-001 — Insecure Multi Tenant Isolation Leads to Full Access Other Tenants

Vulnerability ID: OTF-001 **Retest status:** Resolved

Vulnerability type: Broken Access Control

Threat level: High

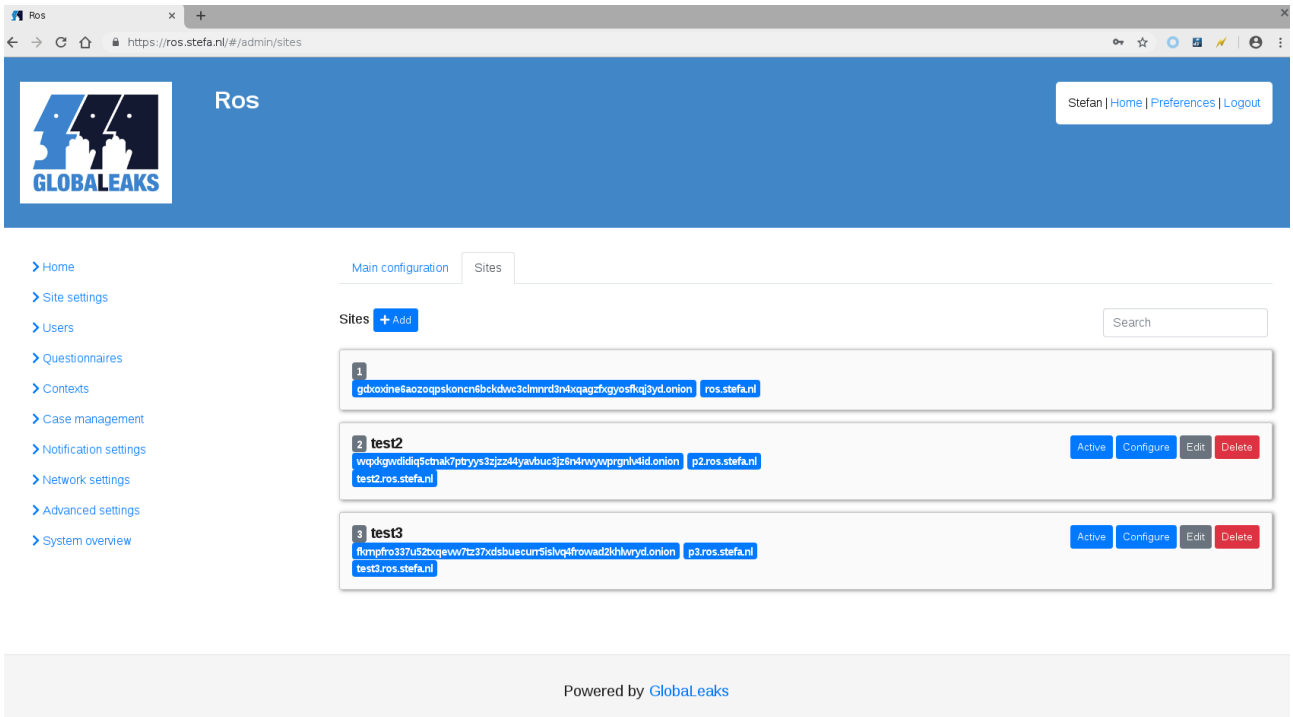
Description:

While logged in as a tenant admin, it is possible to issue requests as the root admin and other tenant admin users.

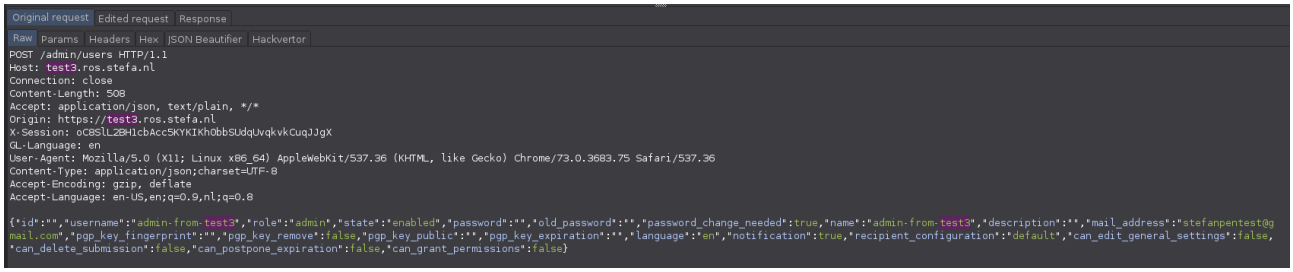
Technical description:

When logged in as a multi-tenant admin, it is possible to gain full control over the application and submissions by modifying the host header.

Logged in as Root Admin - Overview of root and tenant sites:

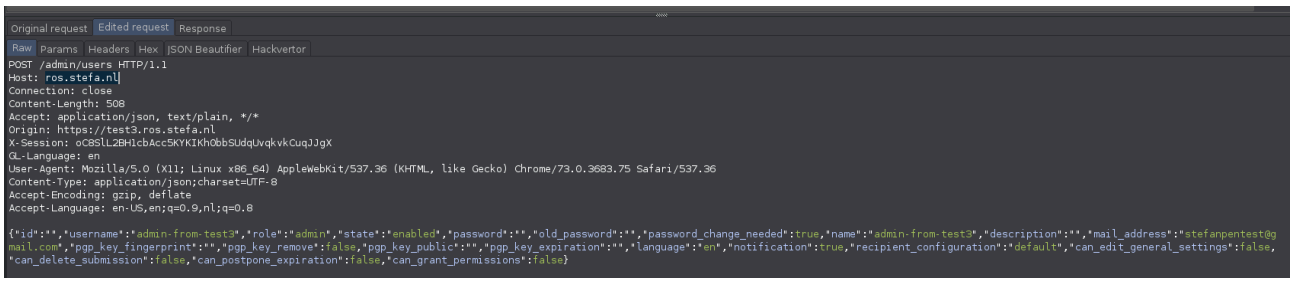


Logged in as admin of the test3 site and intercepting the add a new user request:



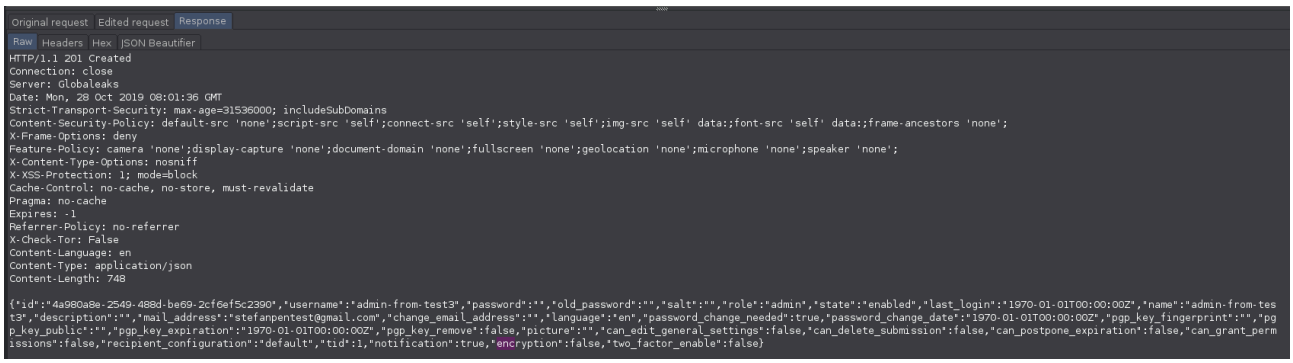
```
Original request | Edited request | Response

Raw | Params | Headers | Hex | JSON Beautifier | Hackvertor

POST /admin/users HTTP/1.1
Host: test3.ros.stefa.nl
Connection: close
Content-Length: 508
Accept: application/json, text/plain, */*
Origin: https://test3.ros.stefa.nl
X-Session: oC8SlL2BH1cbAcc5KYKIKhObbSUdqUvqkvkCuqJJgX
GL-Language: en
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.75 Safari/537.36
Content-Type: application/json;charset=UTF-8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,nl;q=0.8

{"id":"","username":"admin-from-test3","role":"admin","state":"enabled","password":"","old_password":"","password_change_needed":true,"name":"admin-from-test3","description":"","mail_address":"stefanpentest@gmail.com","pgp_key_fingerprint":"","pgp_key_remove":false,"pgp_key_public":"","pgp_key_expiration":"","language":"en","notification":true,"recipient_configuration":"default","can_edit_general_settings":false,"can_delete_submission":false,"can_postpone_expiration":false,"can_grant_permissions":false}
```

Changing the host header to that of the root site and add a new user called "admin-from-test3":



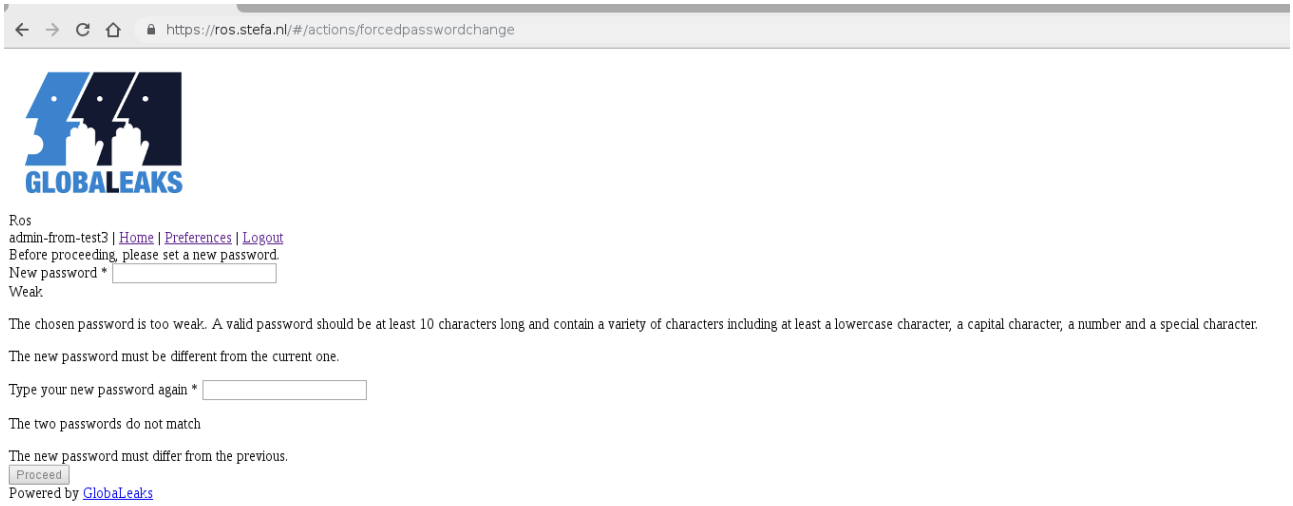
```
Original request | Edited request | Response

Raw | Params | Headers | Hex | JSON Beautifier | Hackvertor

POST /admin/users HTTP/1.1
Host: ros.stefa.nl
Connection: close
Content-Length: 508
Accept: application/json, text/plain, */*
Origin: https://test3.ros.stefa.nl
X-Session: oC8SlL2BH1cbAcc5KYKIKhObbSUdqUvqkvkCuqJJgX
GL-Language: en
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.75 Safari/537.36
Content-Type: application/json;charset=UTF-8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,nl;q=0.8

{"id":"","username":"admin-from-test3","role":"admin","state":"enabled","password":"","old_password":"","password_change_needed":true,"name":"admin-from-test3","description":"","mail_address":"stefanpentest@gmail.com","pgp_key_fingerprint":"","pgp_key_remove":false,"pgp_key_public":"","pgp_key_expiration":"","language":"en","notification":true,"recipient_configuration":"default","can_edit_general_settings":false,"can_delete_submission":false,"can_postpone_expiration":false,"can_grant_permissions":false}
```

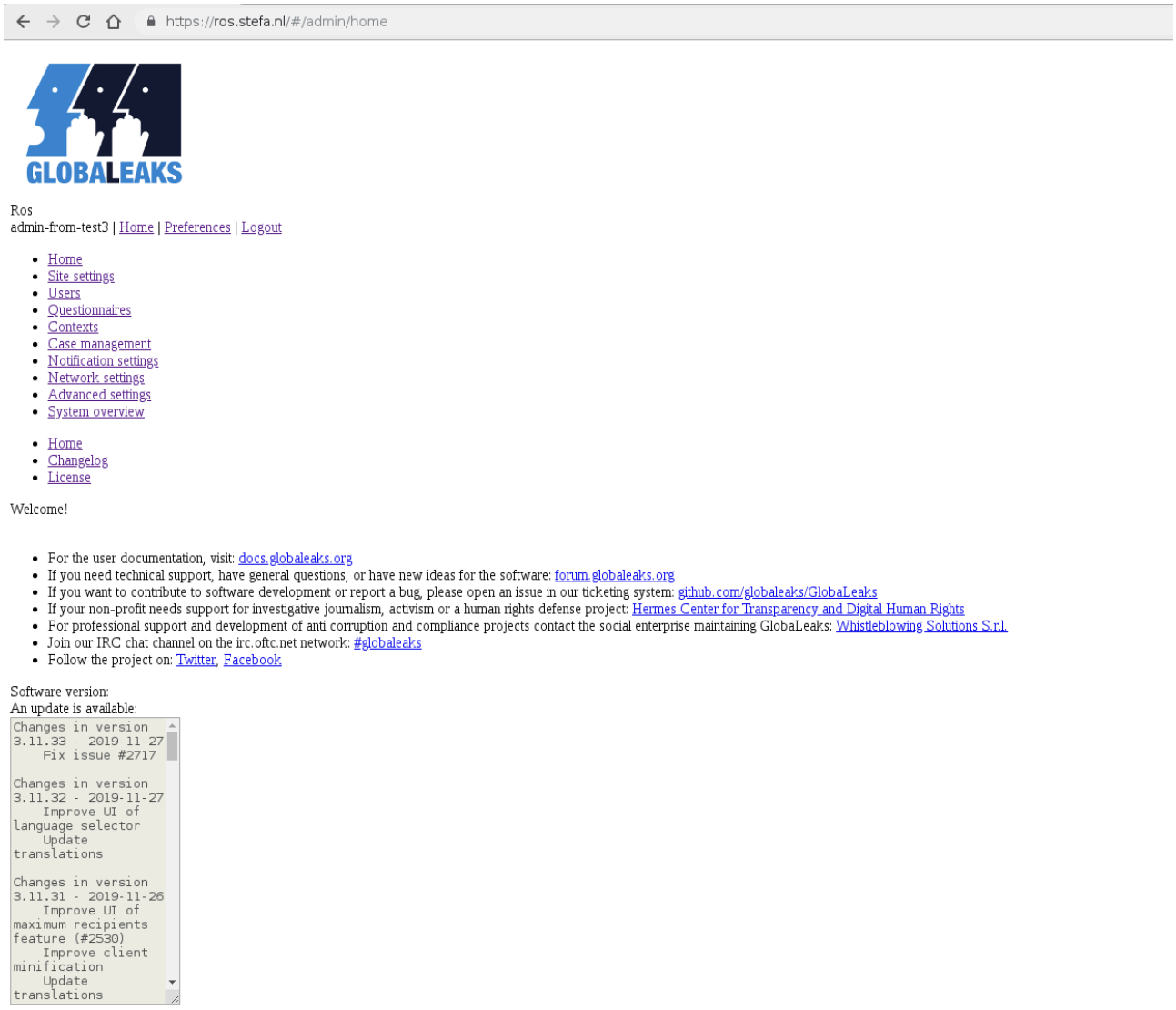
Response successful:



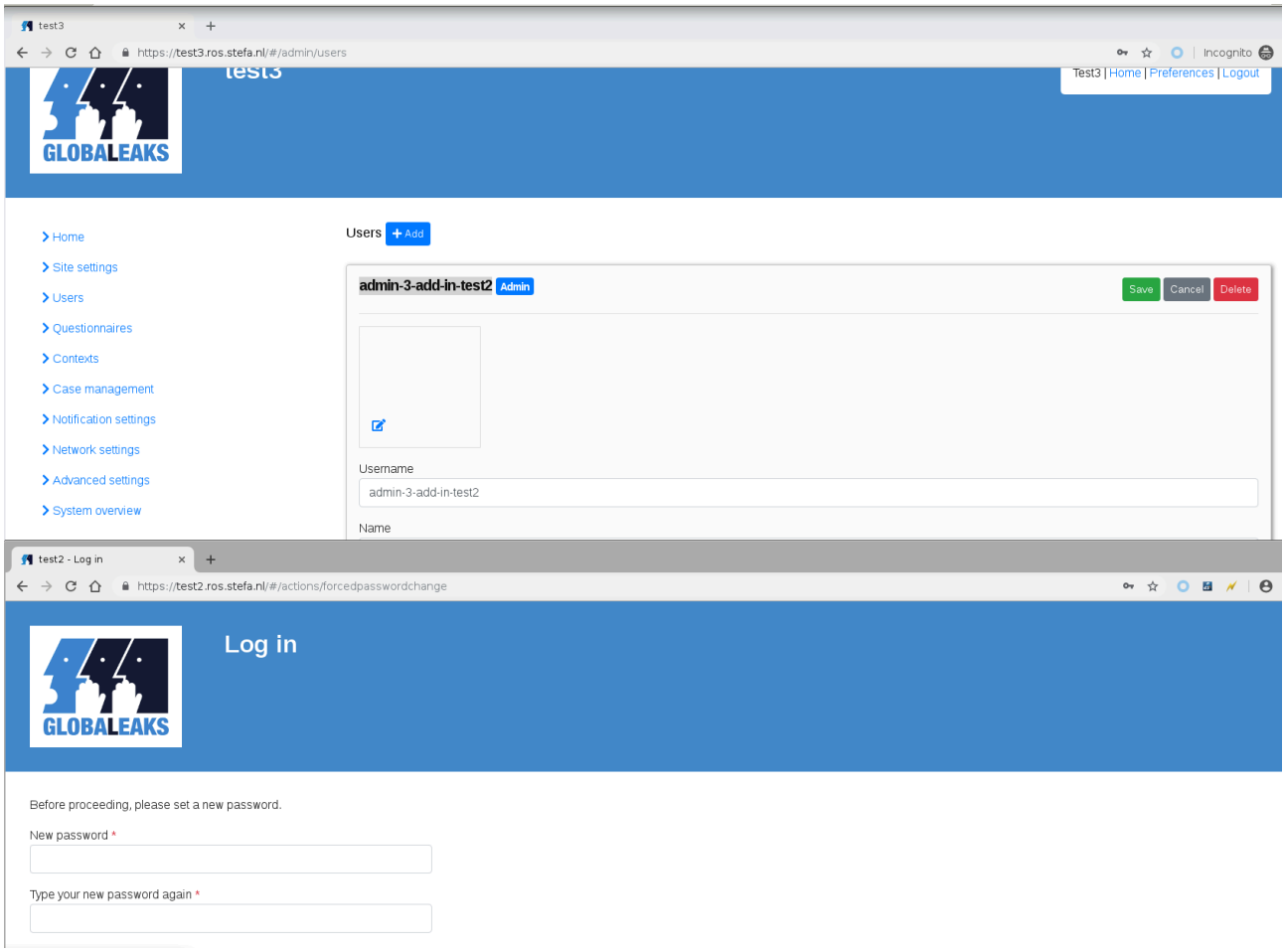
```
Original request | Edited request | Response

Raw | Headers | Hex | JSON Beautifier

HTTP/1.1 201 Created
Connection: close
Server: GlobaLeaks
Date: Mon, 28 Oct 2019 08:01:36 GMT
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Security-Policy: default-src 'none';script-src 'self';connect-src 'self';style-src 'self';img-src 'self' data:;font-src 'self' data:;frame-ancestors 'none';
X-Frame-Options: deny
Feature-Policy: camera 'none';display-capture 'none';document-domain 'none';fullscreen 'none';geolocation 'none';microphone 'none';speaker 'none';
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: -1
Referrer-Policy: no-referrer
X-Check-Tor: False
Content-Language: en
Content-Type: application/json
Content-Length: 748

{"id":"4a980a8e-2549-488d-be69-2cf6ef5c2390","username":"admin-from-test3","password":"","old_password":"","salt":"","role":"admin","state":"enabled","last_login":"1970-01-01T00:00:00Z","name":"admin-from-test3","description":"","mail_address":"stefanpentest@gmail.com","change_email_address":"","language":"en","password_change_needed":true,"password_change_date":"1970-01-01T00:00:00Z","pgp_key_fingerprint":"","pgp_key_public":"","pgp_key_expiration":"1970-01-01T00:00:00Z","pgp_key_remove":false,"picture":"","can_edit_general_settings":false,"can_delete_submission":false,"can_postpone_expiration":false,"can_grant_permissions":false,"recipient_configuration":"default","tid":1,"notification":true,"encryption":false,"two_factor_enable":false}
```

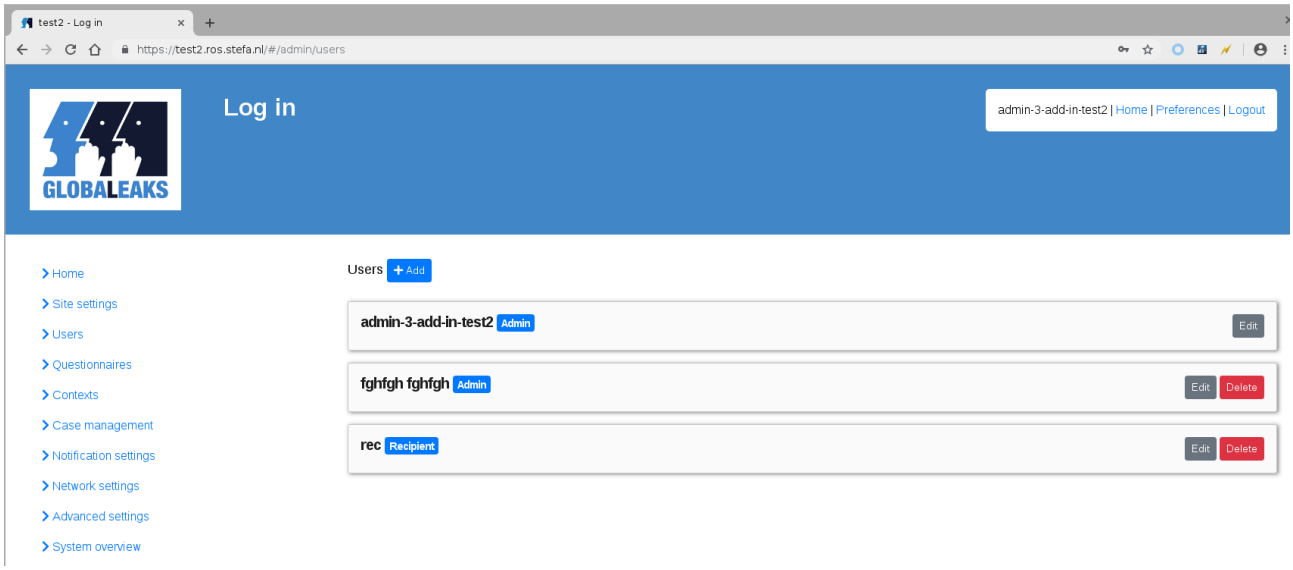
Login to the root website with the new created "admin-from-test3" account:


https://ros.stefa.nl/#/actions/forcedpasswordchange
GLOBALEAKS
Ros
admin-from-test3 | Home | Preferences | Logout
Before proceeding, please set a new password.
New password *
Weak
The chosen password is too weak. A valid password should be at least 10 characters long and contain a variety of characters including at least a lowercase character, a capital character, a number and a special character.
The new password must be different from the current one.
Type your new password again *
The two passwords do not match
The new password must differ from the previous.
Proceed
Powered by GlobaLeaks



Adding from the test3 site a new user to the test2 site:





During this audit this bug was verified, patched, retested and confirmed as fixed in this commit: https://github.com/globaleaks/GlobaLeaks/commit/da021b64d83af0b242bb2fae9a20d25de2ad5cba

Note that the multi-tenant and "Allow users to sign up" features are not enabled by default, resulting a "High" threat level instead of "Critical".

Feedback from GlobaLeaks Team:

Status: Fixed

Commit of the fix: https://github.com/globaleaks/GlobaLeaks/commit/da021b64d83af0b242bb2fae9a20d25de2ad5cba

Feedback from Pentest Team:

The commmit fixed the issue.

Impact:

Gaining unauthorized administrator access to the root and tenant sites could give an attacker full control over the application and access to highly sensitive information.

Recommendation:

- Correct the code to deny access to unauthorized users.

4.2 OTF-002 — SSL/TLS Connections Do Not Verify Certificates

Vulnerability ID: OTF-002	**Retest status:** Resolved
Vulnerability type: Insecure Communication	
Threat level: High	

Description:

GlobaLeaks uses the OpenSSL library wrapper for Python for SSL/TLS connections, both as client or server. It does not seem to enforce any verification of remote certificates.

Technical description:

In file `backend/globaleaks/utils/tls.py`, functions `TLSClientContextFactory()` and `new_tls_client_context()`:

```
194 class TLSClientContextFactory(ssl.ClientContextFactory):
195     def getContext(self):
196         return new_tls_client_context()
```

```
145 def new_tls_client_context():
146     ctx = SSL.Context(SSL.SSLv23_METHOD)
147
```

```
148     ctx.set_options(SSL.OP_NO_SSLv2 |
149                     SSL.OP_NO_SSLv3 |
150                     SSL.OP_SINGLE_ECDH_USE |
151                     SSL.OP_NO_COMPRESSION |
152                     SSL.OP_NO_TICKET |
153                     SSL.OP_NO_RENEGOTIATION)
154
155     ctx.set_mode(SSL.MODE_RELEASE_BUFFERS)
156     ctx.set_session_cache_mode(SSL.SESS_CACHE_OFF)
157
158     return ctx
```

The code initializing SSL connections does not enforce certificate validation.

Feedback from GlobaLeaks Team:

Status: Fixed

Commit of the fix: https://github.com/globaleaks/GlobaLeaks/commit/c569a65a4ef732c5bde8e582e86e68f188c5a319.

Feedback from Pentest Team:

The fix looks correct to us. Still, we would like to mention a potential caveat. While it is definitely a must for HTTPS traffic, it is possible to run into issues with SMTP TLS/STARTTLS. Many servers mey still use self-signed certificates unfortunately. It is possible to enforce validation for SMTP and inform about failures, or let the SMTP traffic go through and inform about validation issues (which is obviously less safe). In any case when sending e-mail through official SMTP servers (e.g. with the submission port, 587) the server certificate should really be valid, so if this is the expected scenario, then definitely enforce validation for SMTP as well.

This decision ultimately depends on the exact conditions of the deployment of GlobaLeaks, which can be different for each situation.

Impact:

Attackers may be able to conduct Man-in-the-Middle attacks (MitM) even though SSL/TLS is in use.

Recommendation:

- Enforce certificate validation for SSL/TLS connections.

4.3 OTF-003 — Admin Password Change Does Not Require Recipient Recovery Key

Vulnerability ID: OTF-003	**Retest status:** Not Retested
Vulnerability type: Sensitive Data Exposure	
Threat level: Elevated	

Description:

When encryption is enabled a recipient is only able to reset their password by using their recovery key. However, when an admin sets a new password for a recipient this is not required.

Technical description:

A recovery key is created when data encryption is enabled. The key is required when a recipient performs a password reset. Without this token the recipient is unable to login. However it was noticed that when an admin changes a recipient's password and the recipient logs in with the new password, the reset token is not required to gain access to the account.

Feedback from GlobaLeaks Team:

Status: Fixed

Planning for the encryption feature to be released the team has removed the possibility for administrators to change the password. The team makes it notice that already planning in case encryption is enabled for testing the password reset results in the complete loss of access to the tips.

Commit of the fix: https://github.com/globaleaks/GlobaLeaks/commit/a068c9a7a5c6572636d4a0102c792ec344fb1752

After this penetration test completing the implementation that has been suggested, the new encryption feature audited in this penetration test will be is enabled the administrator will not be in the possibility to change the password but only to send a password reset token via email.

In case of password reset token the admin will be able only to re-issue a new user in place of the existing user with complete loss of the submissions access but will be available to issue a password reset with data loss where the user, with full data loss, will receive a password reset token in its email.

Feedback from Pentest Team:

Agree with the solution.

Impact:

An adversary that gains admin access is able to gain access to recipients data (including whistleblowers submissions).

Recommendation:

- Always require the recovery key to decrypt recipients data even when an admin sets the new password.

4.4 OTF-004 — The Shorttitle Field Is Not Encrypted

Vulnerability ID: OTF-004	**Retest status:** Resolved
Vulnerability type: Sensitive Data Exposure	
Threat level: Elevated	

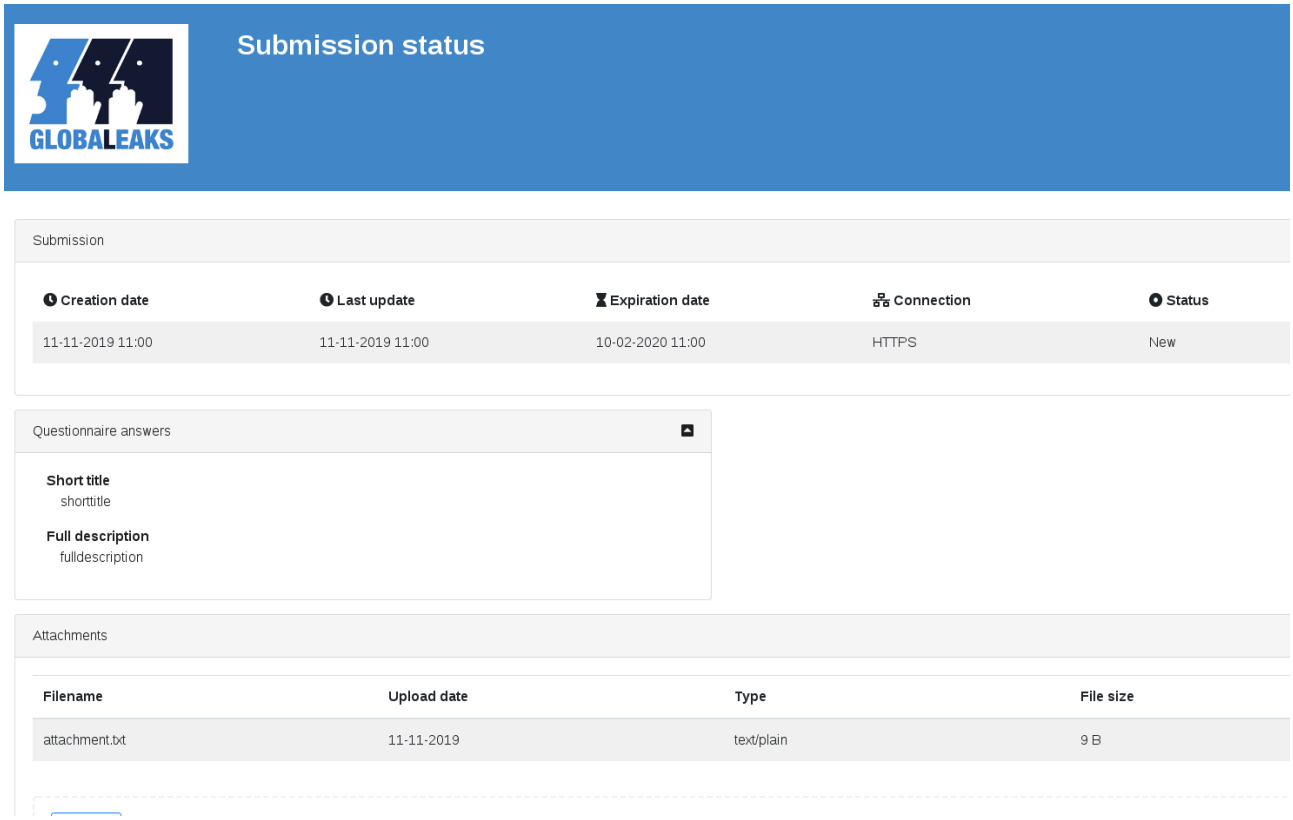
Description:

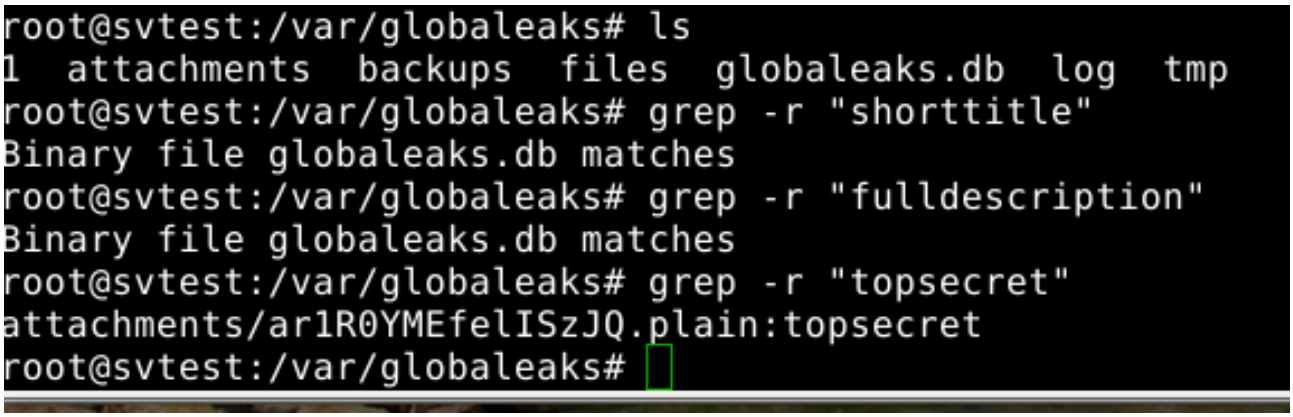
The `shorttitle` field is not encrypted when encryption is enabled.

Technical description:

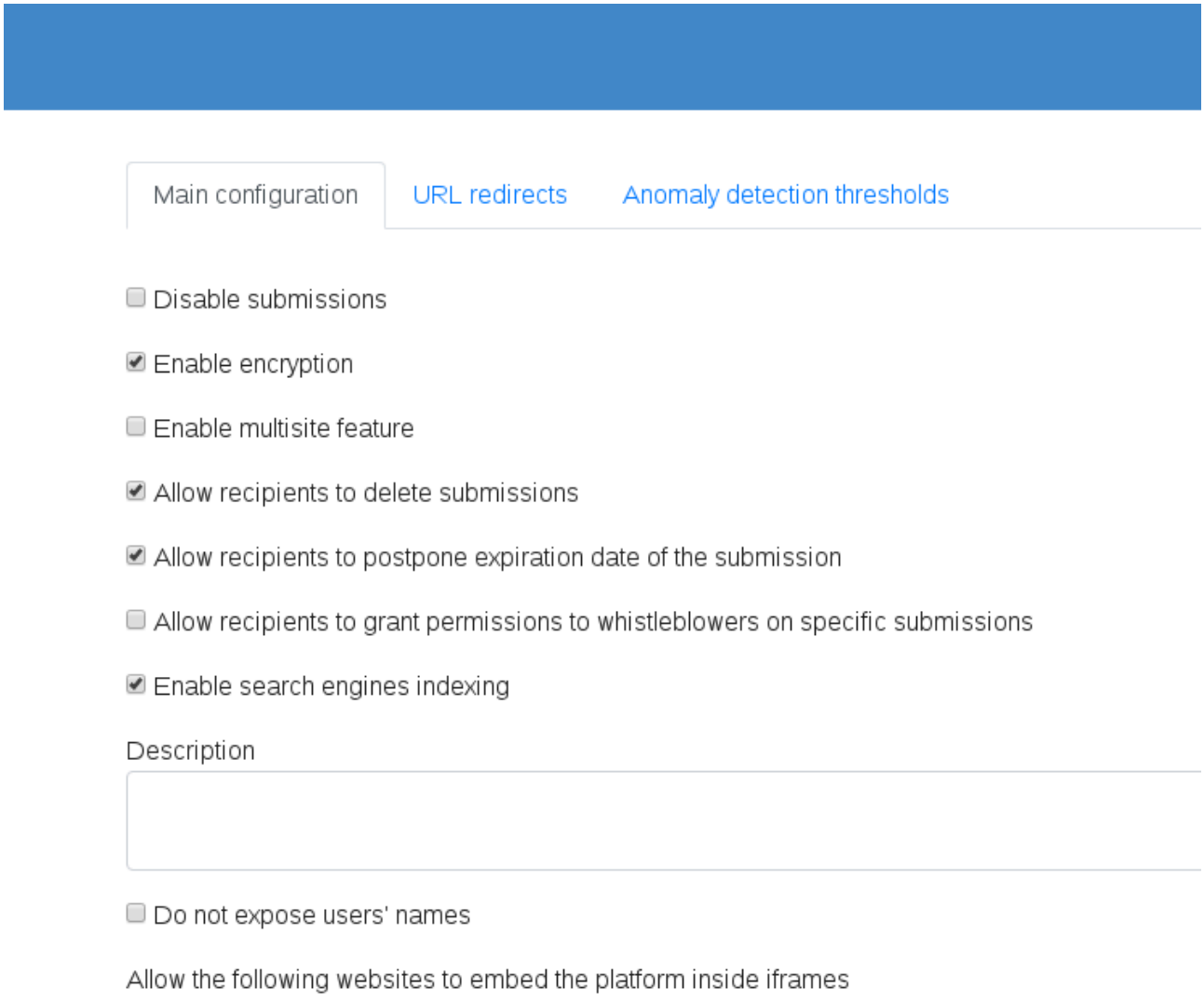
The `shorttitle` field in the internal tip database table does not get encrypted when encryption has been enabled.

Submission of an unencrypted tip by a whistleblower:

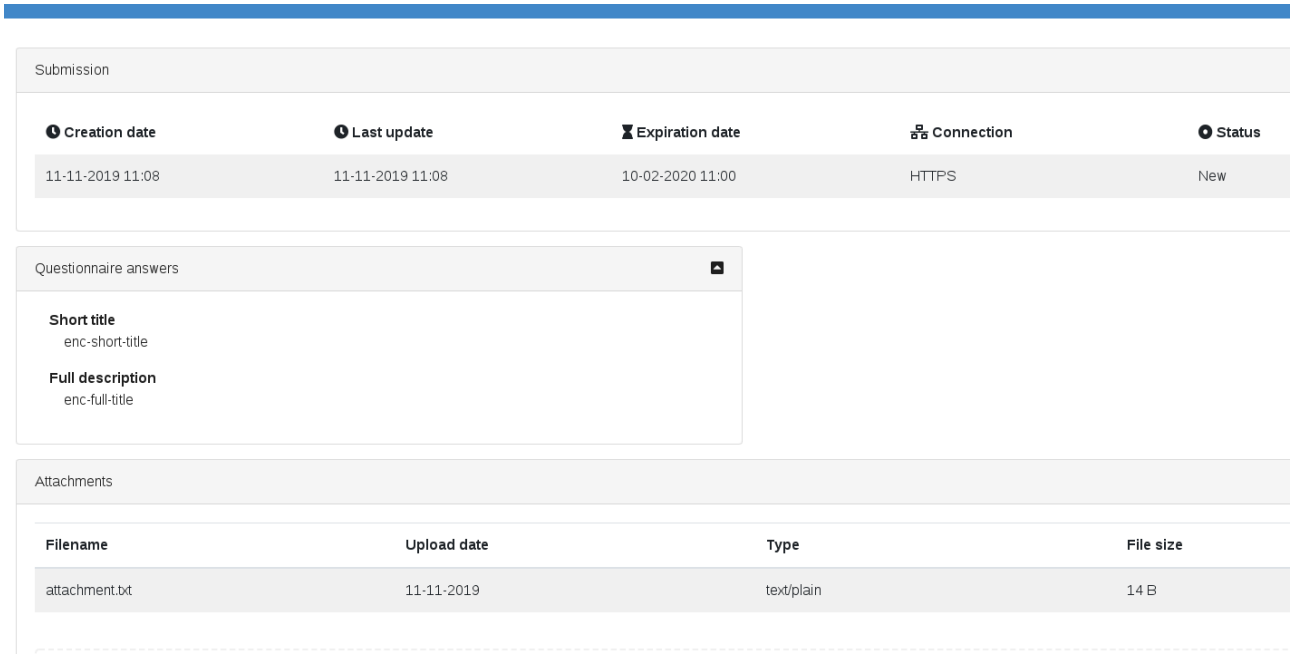




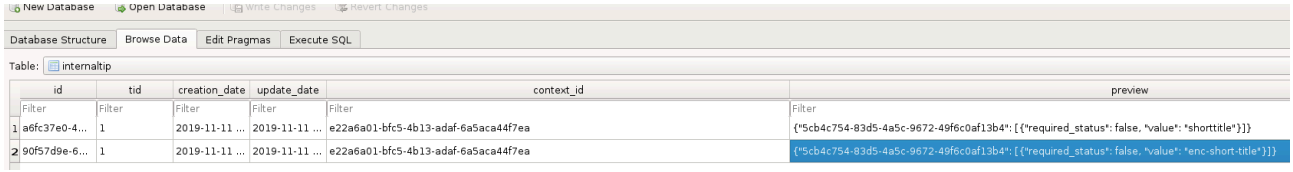
Admin enables encryption:



Submission of encrypted tip by a whistleblower:



The following example shows that `shorttitle` data is unencrypted:



Feedback from GlobaLeaks Team:

Status: Fixed

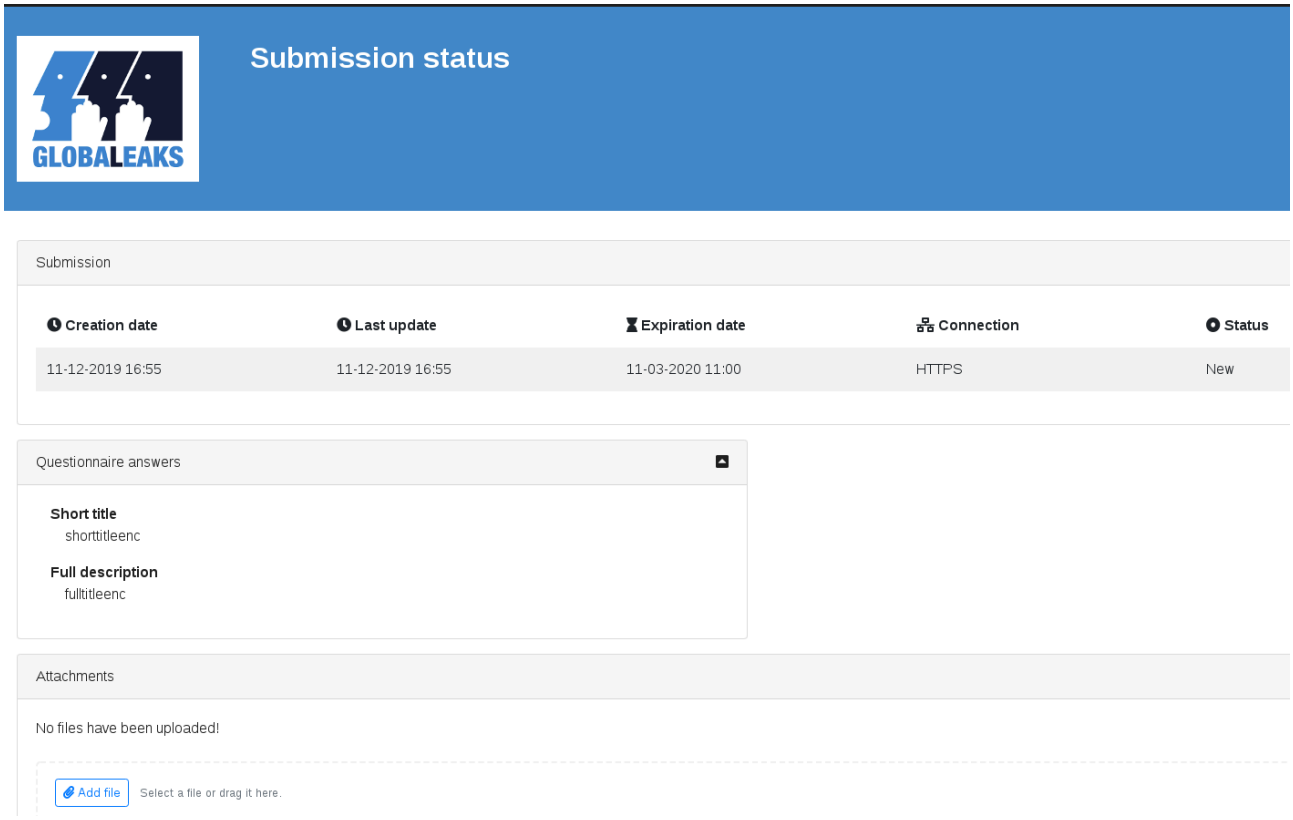
The issue has been further investigated and the development team identified to have to complete a part of the encryption in relation to the functionality of Tip Preview that enables to mark some of the Questions as subject to preview and these questions were currently not encrypted. This functionality has been now implemented and the issue is fixed.

Commit of the fix: https://github.com/globaleaks/GlobaLeaks/commit/656248b70e72a65fc87e6407c7338d650f6e3233

Feedback from Pentest Team:

Fixed

Submission of encrypted message:



Data is encrypted:

```
root@contrib-buster:/var/globaleaks# cat globaleaks.db | grep "shorttitleenc"
root@contrib-buster:/var/globaleaks# cat globaleaks.db | grep "fulltitleenc"
root@contrib-buster:/var/globaleaks#
```

Disabling encryption, submitting a new tip and retrieve the data out of the database:

```
root@contrib-buster:/var/globaleaks# cat globaleaks.db | grep "encryptionoff"
Binary file (standard input) matches
root@contrib-buster:/var/globaleaks# cat globaleaks.db | grep "encryptionoff2"
Binary file (standard input) matches
root@contrib-buster:/var/globaleaks# _
```

Impact:

Unencrypted data can be read by anybody that gains access to the database. This could lead to access of confidential information, possibly including whistleblower identities.

Recommendation:

- Encrypt the `shorttitle` field in the database.

4.5 OTF-005 — GlobaLeaks Installations Phone Home

Vulnerability ID: OTF-005	**Retest status:** Resolved
Vulnerability type: Sensitive Data Exposure	
Threat level: Elevated	

Description:

An installation script was found to report information to a server controlled by GlobaLeaks when running.

Technical description:

In file `scripts/install.sh`:

```
178   curl "https://deb.globaleaks.org/install-globaleaks.sh" \
179        -G -m 10 \
180        --data-urlencode "DISTRO=$REAL_DISTRO_CODENAME" \
181        --data-urlencode "LAST_COMMAND=$LAST_COMMAND" \
182        --data-urlencode "LAST_STATUS=$LAST_STATUS" \
183        >/dev/null 2>/dev/null
```

This command seems to be performed automatically each time this script is executed as the root user. It will send at least the following information to GlobaLeaks' remote server:

- Linux distribution in use
- Public IP address of the system installing GlobaLeaks
- Version information as sent by cURL

In some jurisdictions, the IP address is considered to be personal information, and this information might be considered sensitive anyway, given the purpose of this software.

cURL is usually very verbose, and will send very precise version information in its user agent string by default.

Feedback from GlobaLeaks Team:

Status: Fixed

The team considers that all the information reported to the installation server explicitly were already communicated to the server implicitly (e.g. with the download of the package from a precise repository (e.g. Debian/Buster) the server gets to know the date, ip, and type of the operating system of the server); Other bootstrapping activities of the software like the version check may disclose the system is active and thus that the system did not fail the install.

The implicit error reporting status has been temporarily completely removed asking users to report eventual installation errors via the ticketing system of the project. The team will consider eventually to reimplement the error reporting feature asking for explicit user consent.

Commit of the fix: https://github.com/globaleaks/GlobaLeaks/commit/b2db9638bc7f94f68ffcb508bbaa6ea694750af4.

Feedback from Pentest Team:

We think the operators of a whistleblowing platform will not expect the platform to be leaking information about itself back to the "mother ship". This, in particular when aiming to be running as a Tor service. It is perfectly possible to prepare a working system (e.g. as a Virtual Machine) without leaking its location or purpose (e.g. through a DVD install) and then install GlobaLeaks from source.

Then, some jurisdictions are particularly severe about "Personal Information", and that generally includes IP addresses. The mechanism put in place leaks IP addresses, and then more. cURL itself leaks security-relevant information in its default user-agent string (e.g. software versions), which can also endanger a newly-installed instance of GlobaLeaks.

Impact:

Privileged information might be leaked back to GlobaLeaks' own infrastructure, without the knowledge of the user installing its software. This may compromise the operational security of a new instance of this platform.

Recommendation:

- Do not contact remote servers automatically with information (or metadata) that might compromise users or instances.

4.6 OTF-006 — Remote Code Execution in the Travis Integration Script

Vulnerability ID: OTF-006	**Retest status:** Resolved
Vulnerability type: Remote Code Execution	
Threat level: Elevated	

Description:

A script meant to set up the continuous integration system will download and execute code directly from a remote server.

Technical description:

In file `travis/build_and_install.sh`:

```
 1 #!/bin/bash -x
 2 set -e
 3
 4 echo "Running Build & Install"
 5 distro="$(lsb_release -cs)"
 6 sudo apt-get -y update
 7
 8 sudo apt-get -y install curl git debhelper devscripts dh-apparmor dh-python python
 9
10 if [ $distro = "bionic" ] || [ "$distro" = "buster" ]; then
11   sudo apt-get -y install python3-pip python3-setuptools python3-sphinx
12 else
13   sudo apt-get -y install python-pip python-setuptools python-sphinx
14 fi
15
16 curl -sL https://deb.nodesource.com/setup_10.x | sudo bash -
```

This code downloads code from the remote server `deb.nodesource.com` and executes it directly as the root user.

Feedback from GlobaLeaks Team:

Status: Fixed

Commit of the fix: https://github.com/globaleaks/GlobaLeaks/commit/217b3b605ef667ead913e1b7b43f1417fd78c9a0.

Feedback from Pentest Team:

Relying on APT packages seems like the way to go. On one hand, it is possible to help Linux distributions such as Debian, Ubuntu, or Devuan keep packages up to date by reporting shortcomings or security vulnerabilities in the NPM packages affected. On another hand, it is also possible to prepare and host backports of these NPM packages as Debian packages, and use them in integration scripts.

Impact:

Continuous integration systems might be compromised by a remote server. This also applies to users running this script on their own systems.

Recommendation:

- Do not trust code obtained directly from remote servers.

4.7 OTF-007 — GlobaLeaks Does Not Use HTTPS by Default

Vulnerability ID: OTF-007 **Retest status:** Not Retested

Vulnerability type: Insecure Communication

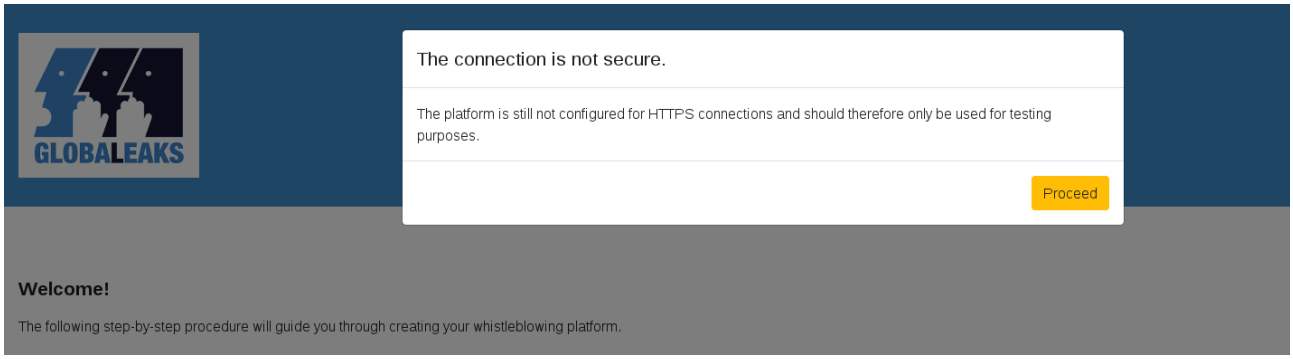
Threat level: Moderate

Description:

After the setup wizard has been completed the platform is accessible over an insecure HTTP connection.

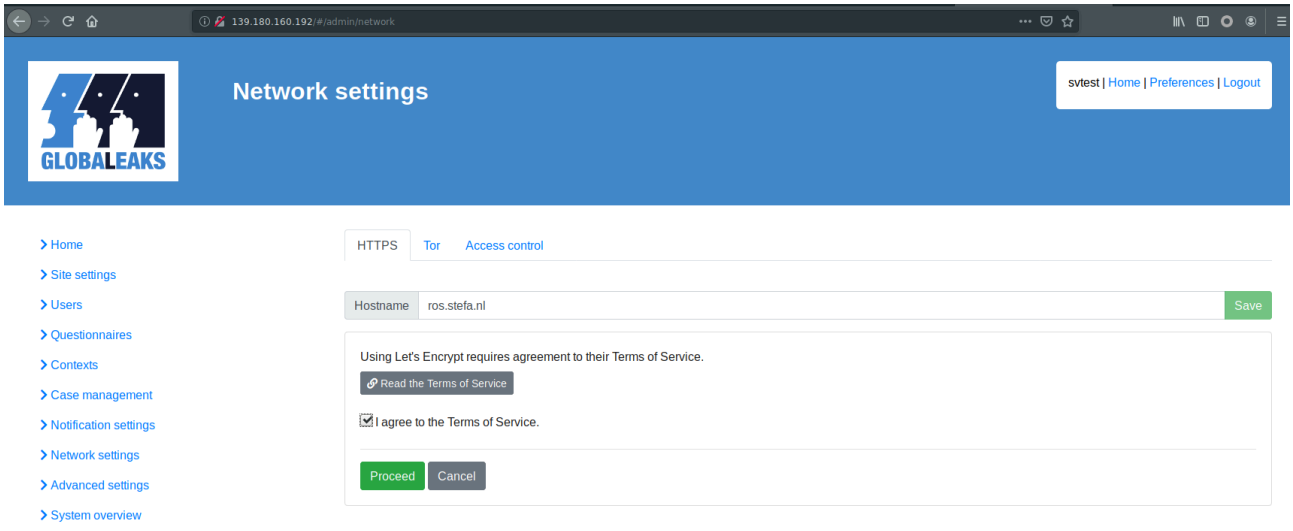
Technical description:

After setup configuration has been completed the platform is accessible over an insecure HTTP connection:



```
root@svtest:/tmp# ./install-globaleaks.sh
Checking preliminary packaging GlobaLeaks requirements
 + apt-key requirement met
 + apt-get requirement met
 + gpg requirement met
Detected OS: Debian - buster
Running: "apt-get -y update"... SUCCESS
 + curl requirement met
 + netstat requirement met
Running: "is_tcp_sock_free_check 0.0.0.0:80"... SUCCESS
Running: "is_tcp_sock_free_check 0.0.0.0:443"... SUCCESS
Running: "is_tcp_sock_free_check 127.0.0.1:8082"... SUCCESS
Running: "is_tcp_sock_free_check 127.0.0.1:8083"... SUCCESS
 + required TCP sockets open
Adding GlobaLeaks PGP key to trusted APT keys
Running: "apt-key add /tmp/tmp.pLncyHeFqx/globaleaks_key"... SUCCESS
Running: "rm /tmp/tmp.pLncyHeFqx/globaleaks_key"... SUCCESS
Installing software-properties-common
Running: "apt-get -y install software-properties-common"... SUCCESS
Updating GlobaLeaks apt source.list in /etc/apt/sources.list.d/globaleaks.list ...
Running: "apt-get update -y"... SUCCESS
Running: "apt-get install globaleaks -y"... SUCCESS
Install script completed.
GlobaLeaks should be reachable at:
+ http://127.0.0.1
+ http://139.180.160.192
For Professional Support requests please visit: https://www.globaleaks.org/contact/
Please report encountered issues to the Community Forum at https://forum.globaleaks.org
root@svtest:/tmp#
```

During the setup the user has to set administration and recipient passwords. Once logged in as the admin the connection can be changed to HTTPS:



Feedback from GlobaLeaks Team:

Status: To be fixed

The team recognizes this is a missing feature and informed that the current secure way to access the platform is to access it via Tor. In addition the team informed to be working already on the possibility to enable to optionally enroll with LetsEncrypt during the activation wizard and has already experimented implementing this; The development was suspended for UX/UI evaluations and will restart soon:

https://github.com/globaleaks/GlobaLeaks/issues/2019

Feedback from Pentest Team:

Providing this option is a good solution to make the application more secure.

Impact:

Traffic between the client and server can be intercepted by a man-in-the-middle. Should a client live in a country that does not care about human rights and performs mass surveillance, their IP address could be exposed and stored in mass surveillance logs.

Recommendation:

- Configure HTTPS automatically by using Letsencrypt during installation.

4.8 OTF-008 — Usage of Default Recipient Password

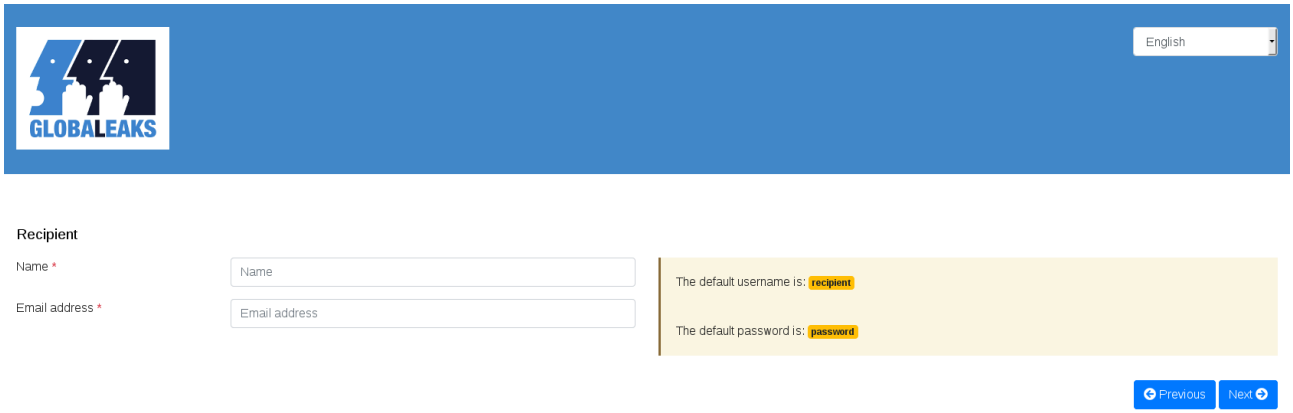
Vulnerability ID: OTF-008	**Retest status:** Not Retested
Vulnerability type: Insecure Password Policy	
Threat level: Moderate	

Description:

A default password is created for the recipient user during the application setup.

Technical description:

A default password is created for the recipient user during the application setup:



Feedback from GlobaLeaks Team:

Status: To be fixed

Current implementation privileged usability for the first setup wizard.The team has evaluated to apply the following changes:

Removal of the default password

Addition of an activation procedure sending a first activation link in the mailbox of the activated user.

Feedback from Pentest Team:

This is a good solution to fix this issue.

Impact:

Using default usernames and passwords increase the chances of a successful brute-force attack, especially if two-factor authentication is not enabled (it's disabled by default).

Recommendation:

- Show a prominent warning in the admin interface or block access if the passwords for admin and recipient users have not been changed from defaults.
- Show a warning in the admin interface when 2FA is not enabled.
- Optionally generate random usernames during setup for both users.

4.9 OTF-009 — The Setup Wizard Does Not Check the Password Strength

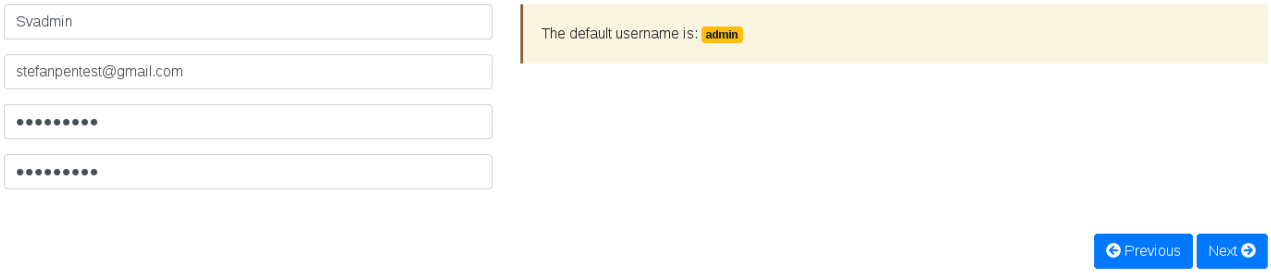
Vulnerability ID: OTF-009	**Retest status:** Resolved
Vulnerability type: Insecure Password Policy	
Threat level: Moderate	

Description:

During the installation process the strength of the new admin password is not checked.

Technical description:

During the installation wizard the strength of the new admin password is not checked.



Note that passwords in other parts of the application *do* get checked for their strength.

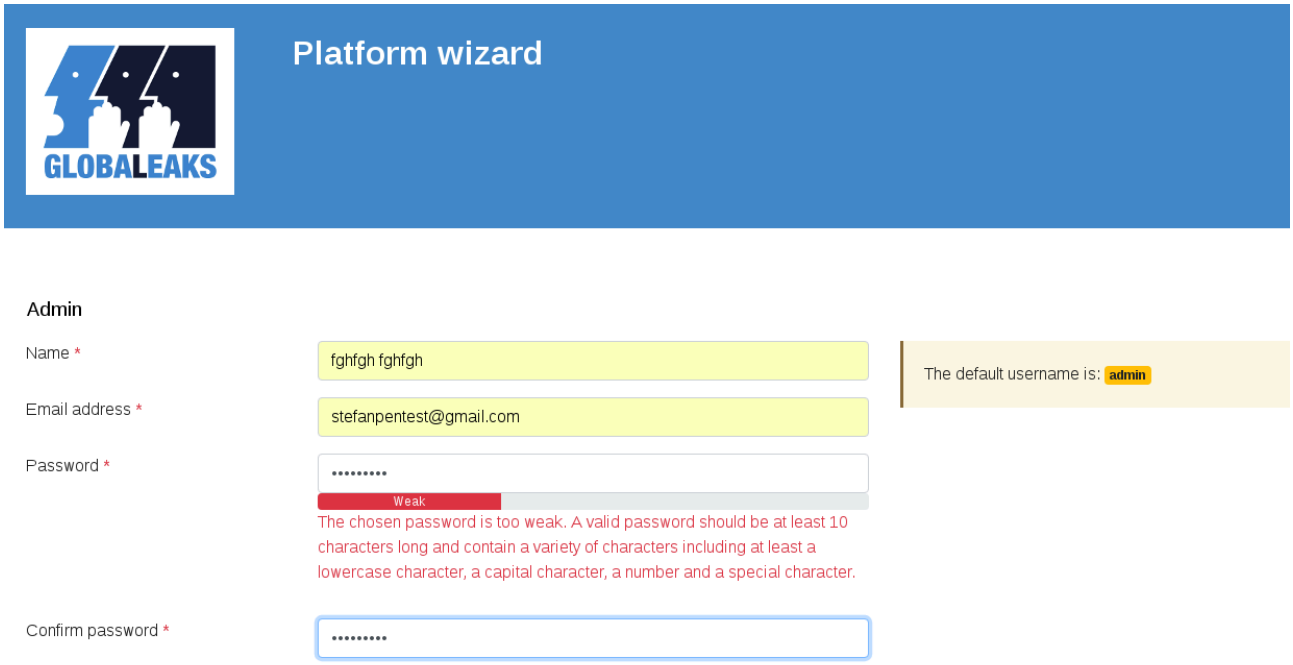
Feedback from GlobaLeaks Team:

Status: Fixed

The password check was actually in place but due a refactoring defect adding during the penetration test activities this appears to be a bug introduced Oct 14 with https://github.com/globaleaks/GlobaLeaks/commit/bf42ba760af81e8b91deb85883e0e0a497da03dd and removed on October 24 with https://github.com/globaleaks/GlobaLeaks/commit/0e761a57e239a37cc4adb4ca06c7c84d2d1594b8.

Feedback from Pentest Team:

Fixed



Impact:

Authentication can be easily bypassed due to weak passwords.

Recommendation:

- Validate new passwords against a strong password policy.

4.10 OTF-010 — No Restrictions in File Upload

Vulnerability ID: OTF-010 **Retest status:** Not Retested

Vulnerability type: Insecure File Upload

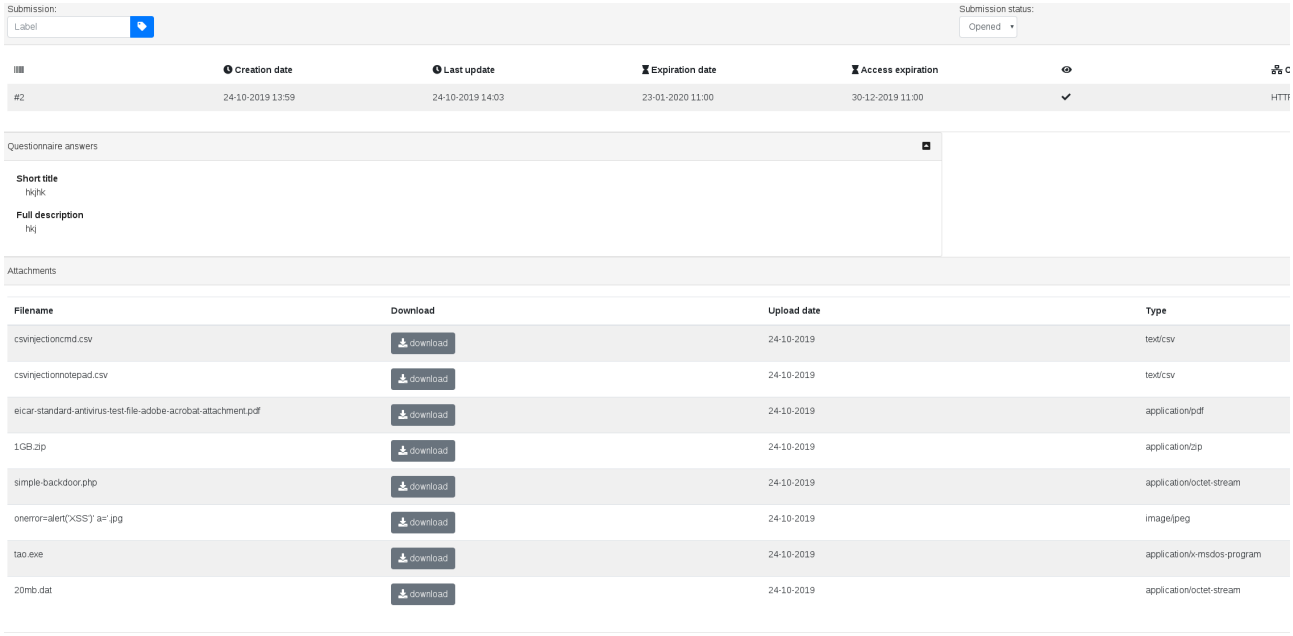
Threat level: Moderate

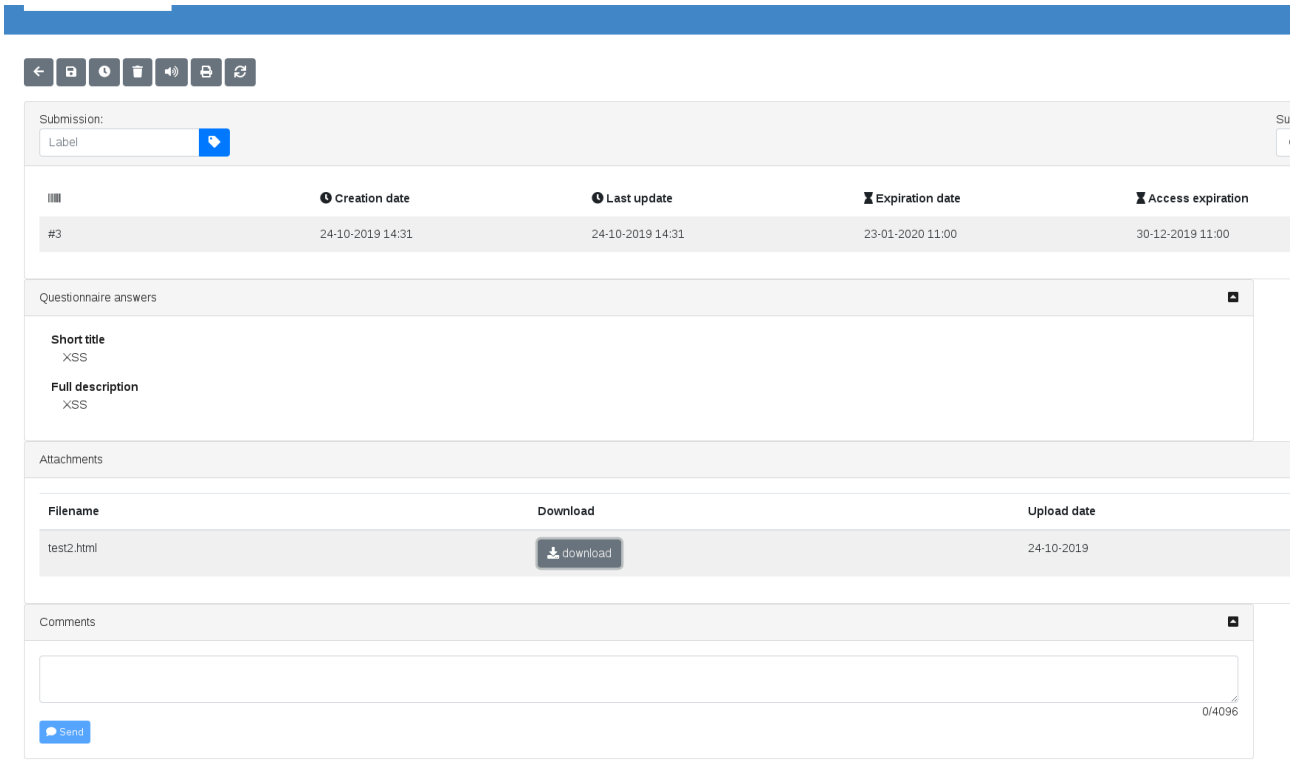
Description:

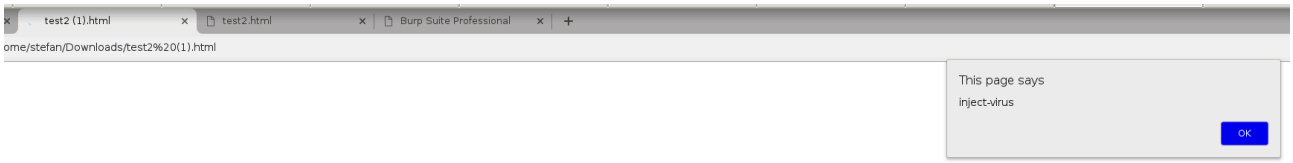
As a whistleblower it is possible to upload malicious files.

Technical description:

The following example shows that malicious files such as .exe and XSS .html can be uploaded:







Given the purpose of the application, it is reasonable to expect whisteleblowers to upload evidence in any format. Howeverm this creates a risk for admins, recipients, and whistleblowers (if enabled) as uploaded files could be malicious.

Feedback from GlobaLeaks Team:

Status: to be fixed

The team will work on the possible fix trying to understand how to block contents as suggested. Current status of the research on the project prevented to make these restrictions due to the unavailability of the component that is able to perform checks in streaming during upload without storing the file in plaintext on the filesystem. Issue related to issue: OTF-020 (page 64).

Feedback from Pentest Team:

No feedback required.

Impact:

Malicious files uploaded and processed by the server may allow distribution of malware which can be unknowingly downloaded by users, or used in deliberate attack against the application and its users.

Recommendation:

- Add a default, configurable warning that informs users that files could be malicious. The warning should be given only the first time a file is downloaded during a session, otherwise it might become annoying and less effective. Optionally include a link to background information on how files could be securely opened, e.g. analyse and open the file by using a sandbox VM that has no internet access.
- The server could utilise anti-virus software to scan uploads prior to storage, transfer or processing, which will stop most known malware and viruses. Note that adding a virus scanner (such as the open source CLAMAV) does not fully protect the server and end user as it is reasonably feasible to bypass virus scanners.
- When using a virus scanner, ensure that any "online file analysis" functionality is disabled, as it could result in sending confidential files to the antivirus vendor.

4.11 OTF-011 — No Account Lockout May Facilitate Brute Force Password Attack

Vulnerability ID: OTF-011 **Retest status:** Not Retested

Vulnerability type: Insecure Password Policy

Threat level: Moderate

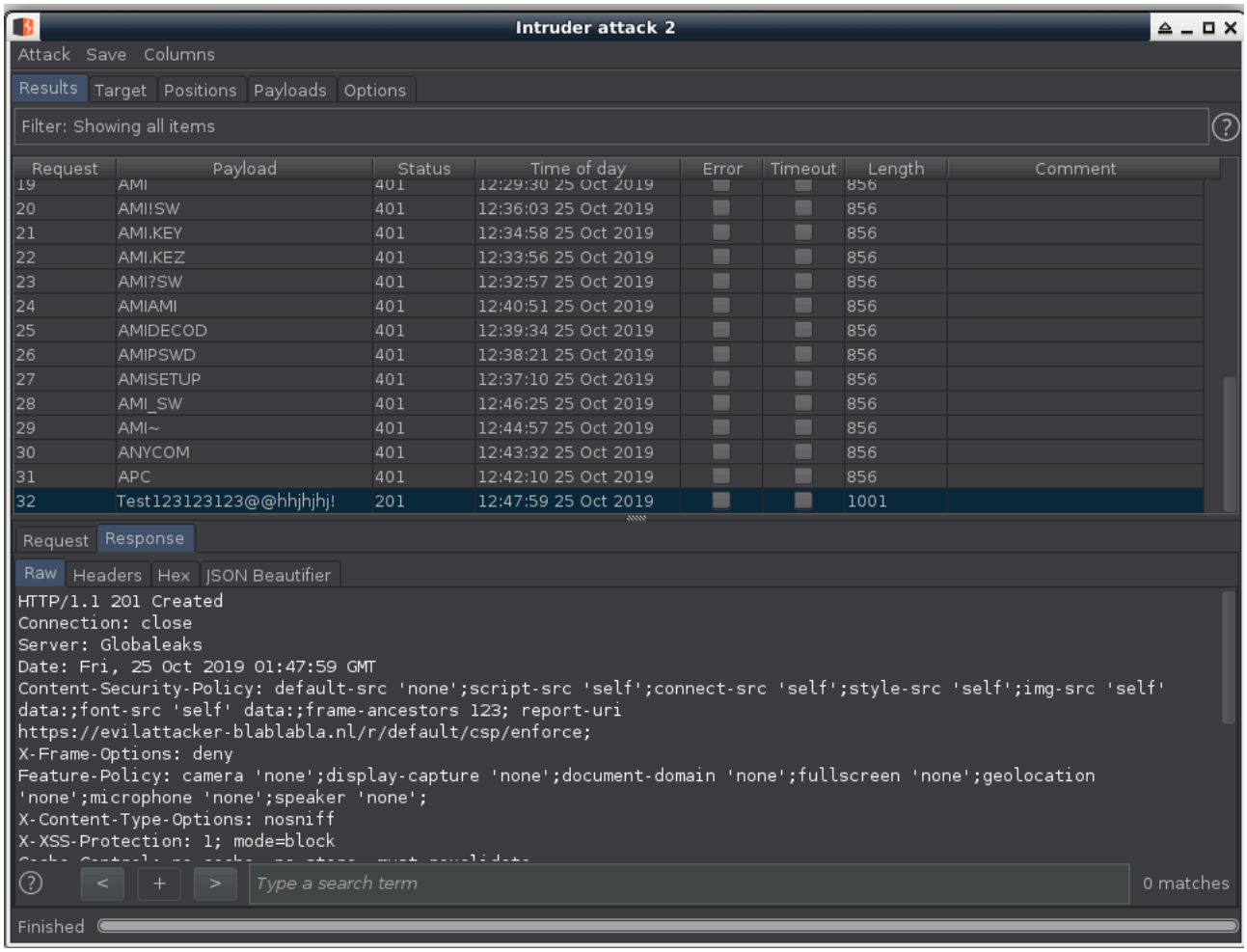
Description:

The application does not have any account lockout mechanisms.

Technical description:

We performed a significant number of failed login attempts against an existing application account, followed by successful authentication. This indicates that the account was not locked out because of the login failures.

There is some rate limiting in place but this still allows password brute-forcing by lowering the request rate. Given the other password and 2FA findings mentioned in this report, brute-forcing could be feasible.



Feedback from GlobaLeaks Team:

Status: to be fixed

The team is aware of this issue and is trying to solve the issue without impacting on the usability of the typical system. Current proposal of remediation at: https://github.com/globaleaks/GlobaLeaks/issues/48

Feedback from Pentest Team:

The proposal described in the link would assist to make bruteforce attacks less feasible. Also consider (temporary) blocking the IP-address of the attacker.

Impact:

An attacker can conduct brute force password attacks to establish valid credentials for the service. This would allow authenticated access to the service, potentially exposing sensitive information to an unauthorised user.

Recommendation:

- Lock out accounts after a number of unsuccessful attempts. A balance should be established to allow a reasonable number of failed attempts after which the accounts will be locked. For sensitive applications, we recommend allowing three to five failures. Ideally this lockout would be temporary, using an exponentially increasing lockout duration so as to prevent brute force attacks without introducing denial of service conditions for legitimate users.

4.12 OTF-012 — Data Encryption Issues

Vulnerability ID: OTF-012	**Retest status:** Not Retested
Vulnerability type: Sensitive Data Exposure	
Threat level: Moderate	

Description:

During testing several data encryption issues were found.

Technical description:

During testing the following issues were found (root and tenant sites):

- When enabling encryption the encryption does not immediately work for new tips. The client has to log in as the recipient user and is required to set a new password. After this sequence the encryption works. The recipient user should also receive a recovery key that is required in case they need to reset their password. However the recovery key was not shown to the user. It was possible to manually retrieve by manually issuing a REST request.
- Existing tips (including attachments) remain unencrypted.
- Encryption is not enabled by default.

Note that the encryption functionality is disabled by default and that this is still an experimental feature.

Feedback from GlobaLeaks Team:

Status: To be fixed with the official realease of the automatic encryption feature

Current documented behaviour of the application if to explicitly require a PGP configuration to encrypt files upload and notification emails. This audit has been explicitly requested to evaluate the new encryption implementation that will completely address the described behaviour passing from a status in which only files and emails are encrypted requiring users to configure manually a PGP key to a status in which the application will be able to generate automatically key

used by the system to encrypt automatically any exchange including not only the attached files and the notifications emails but also the questionnaire answers comments, messages, metadata of files, and other metadata. This new mode will be the only one implemented by the system and will be automatically enabled for every setup. By enabling the feature by default many of the issues will be automatically solved: keys will be created at first login, users and their keys will be enabled at first login, tips will be always encrypted. Previous setups will be able to upgrade to the new mode.

Feedback from Pentest Team:

Agree with clients feedback.

Impact:

Unencrypted data can be read by anybody that manages to gain access to the database. This could lead to a breach of confidential information and possibly reveal the identities of whistleblowers.

Recommendation:

- Enable encryption by default.
- Add an encryption status display to the whistleblower submission and recipient overview pages, and create a job that monitors its status.
- Add an administrator page that shows the encryption status for each submission.
- If encryption is not working, a warning message should be shown on the admin page.
- Enable encryption for already submitted tips.

4.13 OTF-013 — External Backup and SMTP Server Passwords Not Encrypted in the Database

Vulnerability ID: OTF-013	**Retest status:** Not Retested
Vulnerability type: Unprotected Passwords	
Threat level: Moderate	

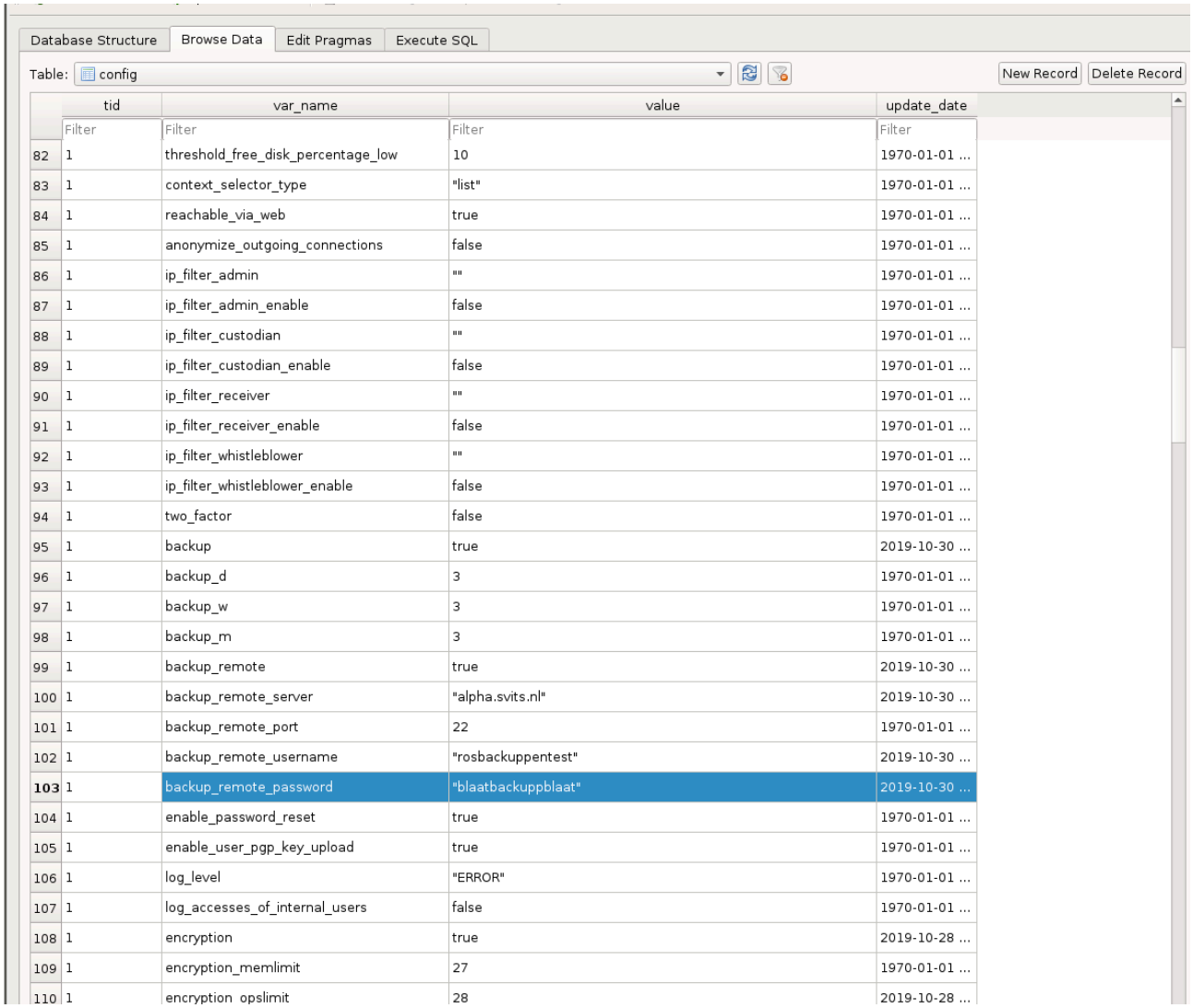
Description:

The external Backup and SMTP server passwords are saved in plain text in the database.

Technical description:

The external Backup and SMTP server passwords are saved in plaintext in the database.

19	1	smtp_server	"mail.globaleaks.org"	1970-01-01 ...
20	1	smtp_port	9267	1970-01-01 ...
21	1	smtp_username	"globaleaks"	1970-01-01 ...
22	1	smtp_password	"globaleaks"	1970-01-01 ...
23	1	smtp_source_email	"notification@mail.globaleaks.org"	1970-01-01 ...
24	1	smtp_security	"TLS"	1970-01-01 ...



	tid	var_name	value	update_date
	Filter	Filter	Filter	Filter
82	1	threshold_free_disk_percentage_low	10	1970-01-01 ...
83	1	context_selector_type	"list"	1970-01-01 ...
84	1	reachable_via_web	true	1970-01-01 ...
85	1	anonymize_outgoing_connections	false	1970-01-01 ...
86	1	ip_filter_admin	""	1970-01-01 ...
87	1	ip_filter_admin_enable	false	1970-01-01 ...
88	1	ip_filter_custodian	""	1970-01-01 ...
89	1	ip_filter_custodian_enable	false	1970-01-01 ...
90	1	ip_filter_receiver	""	1970-01-01 ...
91	1	ip_filter_receiver_enable	false	1970-01-01 ...
92	1	ip_filter_whistleblower	""	1970-01-01 ...
93	1	ip_filter_whistleblower_enable	false	1970-01-01 ...
94	1	two_factor	false	1970-01-01 ...
95	1	backup	true	2019-10-30 ...
96	1	backup_d	3	1970-01-01 ...
97	1	backup_w	3	1970-01-01 ...
98	1	backup_m	3	1970-01-01 ...
99	1	backup_remote	true	2019-10-30 ...
100	1	backup_remote_server	"alpha.svits.nl"	2019-10-30 ...
101	1	backup_remote_port	22	1970-01-01 ...
102	1	backup_remote_username	"rosbackuppentest"	2019-10-30 ...
103	1	backup_remote_password	"blaatbackuppblaat"	2019-10-30 ...
104	1	enable_password_reset	true	1970-01-01 ...
105	1	enable_user_pgp_key_upload	true	1970-01-01 ...
106	1	log_level	"ERROR"	1970-01-01 ...
107	1	log_accesses_of_internal_users	false	1970-01-01 ...
108	1	encryption	true	2019-10-28 ...
109	1	encryption_memlimit	27	1970-01-01 ...
110	1	encryption_opslimit	28	2019-10-28 ...

Note that the Remote Server Backup Functionality was not functional during testing. When the developers decide to introduce this functionality it would be recommended to encrypt the backups by default as well.

Feedback from GlobaLeaks Team:

Status: Partially fixable

Remediation to be identified. Considerations that limit current exposure:

The issue currently affects only the SMTP Servers passwords as External Backups are not implemented (it is just drafted a possible scheduler for backups). The password of the SMTP server is stored in plaintext as it requires to be used in plaintext during mail notifications.

Feedback from Pentest Team:

The general rule is to not store passwords in plaintext were possible.

Impact:

An attacker with read access to the database can obtain backup and SMTP server credentials that can be used to access and delete backups or used for spam purposes.

Recommendation:

- Encrypt sensitive data such as passwords to prevent an attacker with read access from using them.
- For passwords that only need to be verified a hashing algorithm such as Argon2 can be used.
- For passwords that need to be used for services (and need to be known to the application), use symmetric encryption with integrity checking, such as libsodium provides.

4.14 OTF-014 — Possibly Ineffective Secure File Erase Functionality

Vulnerability ID: OTF-014	**Retest status:** Not Retested
Vulnerability type: Sensitive Data Exposure	
Threat level: Moderate	

Description:

There is a function intended to erase files securely by overwriting, but it is unlikely to work as intended.

Technical description:

The function `overwrite_and_remove()` in file `backend/globaleaks/utils/security.py` is meant to securely erase a file. However, there are at least three caveats that will likely result in this failing to perform as intended.

In file `backend/globaleaks/utils/security.py`, function `overwrite_and_remove()`:

```
90 def overwrite_and_remove(absolutefpath, iterations_number=1):
```

```
 91     """
 92     Overwrite the file with all_zeros, all_ones, random patterns
 93
 94     Note: At each iteration the original size of the file is altered.
 95     """
 96     log.debug("Starting secure deletion of file %s", absolutefpath)
 97
 98     randomgen = random.SystemRandom()
 99
100     try:
101         # in the following loop, the file is open and closed on purpose, to trigger flush
 operations
102         all_zeros = "\0\0\0\0" * 1024               # 4kb of zeros
103
104         if sys.version_info[0] == 2:
105             all_ones = "FFFFFFFF".decode("hex") * 1024  # 4kb of ones
106         else:
107             all_ones = "\xFF" * 4096
108
109         for iteration in range(iterations_number):
110             OPTIMIZATION_RANDOM_BLOCK = randomgen.randint(4096, 4096 * 2)
111
112             random_pattern = ""
113             for _ in range(OPTIMIZATION_RANDOM_BLOCK):
114                 random_pattern += str(randomgen.randrange(256))
115
116             log.debug("Excecuting rewrite iteration (%d out of %d)",
117                       iteration, iterations_number)
118
119             _overwrite(absolutefpath, all_zeros)
120             _overwrite(absolutefpath, all_ones)
121             _overwrite(absolutefpath, random_pattern)
```

Where `_overwrite()` really does this:

```
 79 def _overwrite(absolutefpath, pattern):
 80     count = 0
 81     length = len(pattern)
 82
 83     with open(absolutefpath, 'w+') as f:
 84         f.seek(0)
 85         while count < length:
 86             f.write(pattern)
 87             count += len(pattern)
```

As can be seen here, the code assumes that the Operating System will effectively open a file and overwrite its data in-place on the disk. However, at least three caveats apply.

First, this code relies on APIs and standards defined by the python language. However, neither this language nor its underlying implementation (POSIX and ANSI C through C and assembly) make any promise that files opened with the built-in `open()` function call will effectively be opened in-place. This is actually up to each and every implementation, and the most common operating systems (Linux, Windows, macOS) do not make any such promises either.

To be clearer, some of these operating systems specifically do not implement this promise. This is typically the case for journaled filesystems (sometimes called "log-based"). Therefore, in practice, filesystems such as ext3, ext4, JFS,

ReiserFS, XFS, FFS (in log mode) will not overwrite files in-place. In most cases, they will even create partial copies to different sections of the volume, in order to prepare transactions for instance.

Morever, some of the technologies implementing the storage layer may also leak or make copies of data without any way to erase them. This is true of unencrypted network storage protocols (e.g. NFS), dynamic containers (e.g. LVM, ZFS), performance and recovery measures (e.g. swap, RAID, RAID caches, backups), and last but not least, the storage technology itself. (e.g. SSD, hard disks, flash drives...)

In practice, most tools implementing this functionality warn against some or all of these shortcomings - even when implemented at a lower level. This is the case of GNU shred for instance, as part of the GNU Coreutils (see https://www.gnu.org/software/coreutils/manual/html_node/shred-invocation.html).

The exact same code is also found in file `backend/globaleaks/utils/fs.py`.

Feedback from GlobaLeaks Team:

Status: Fixed

The team is aware of this and rely on the encryption performed by the system and has always considered this legacy additional measure as a possible additional security measure with its known limitations. For this reason the team has considered to add just a comment in the code clarifying it.

Feedback from Pentest Team:

Keeping this code in place will remain useful for filesystems where this technique actually works; it sounds like a good idea to us. However, the possible shortcomings of this code need to be clearly documented, and it does not seem possible to fix this issue without enforcing the corresponding requirements on the underlying platform.

Impact:

Sensitive data may remain reachable even after supposedly "secure" deletion.

Recommendation:

- Do not assume "secure deletion" is actually effective.
- Use filesystems backed only by volatile memory (e.g. RAM disks) to store temporary data in the clear.

4.15 OTF-015 — No Server Side Password Strength Check

Vulnerability ID: OTF-015 **Retest status:** Unresolved

Vulnerability type: Insecure Password Policy

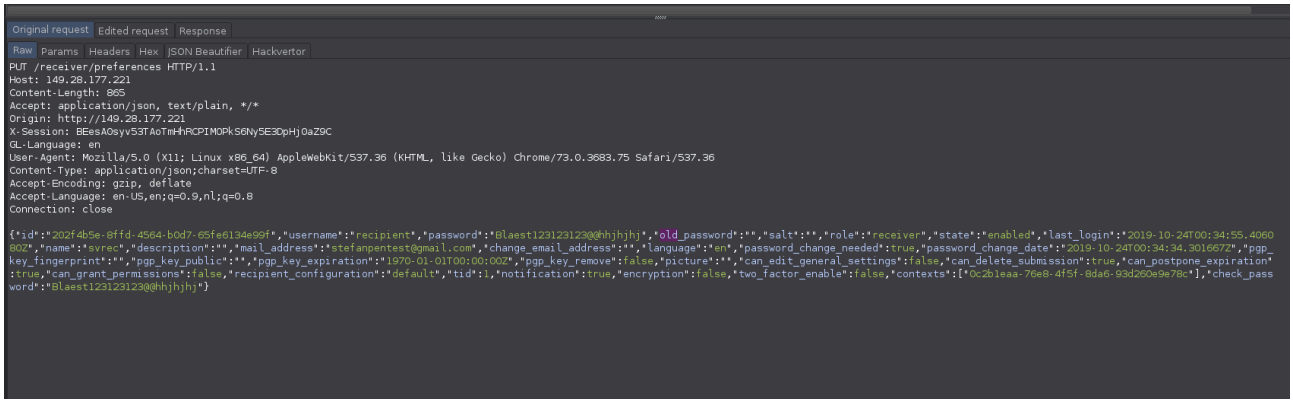
Threat level: Low

Description:

It is possible to set weak passwords by disabling the client-side password strength checks.

Technical description:

It is possible to set weak passwords by disabling the client-side password strength checks.



Feedback from GlobaLeaks Team:

Status: Fixed

The team in order to currently simplify the codebase did not consider this as a possible vulnerability considering the password strength evaluation to be a support system for lecit users and not considering an attacker wanting explicitly to tweak the client application for setting an unsecure password. Considering the audit report the team proposes to implement this check on server-side by using the same algorithm implemented client-side.

Feedback from Pentest Team:

Not Fixed, still possible to disable the client-side check. (Software version: 3.11.48)

Impact:

User accounts with weak passwords could be compromised via brute-force or dictionary attacks, for instance trying top 100 common passwords.

Recommendation:

Implement and enforce a strong, consistent password policy on the server side as well as the client side. Enable 2FA by default.
Basic recommendations for a strong password policy would be:

- Have at least 12 characters.
- Not contain words found in dictionaries or known public data breaches.

4.16 OTF-016 — Improper Input Validation

Vulnerability ID: OTF-016 **Retest status:** Not Retested

Vulnerability type: Input Validation

Threat level: Low

Description:

The application does not validate or incorrectly validates input.

Technical description:

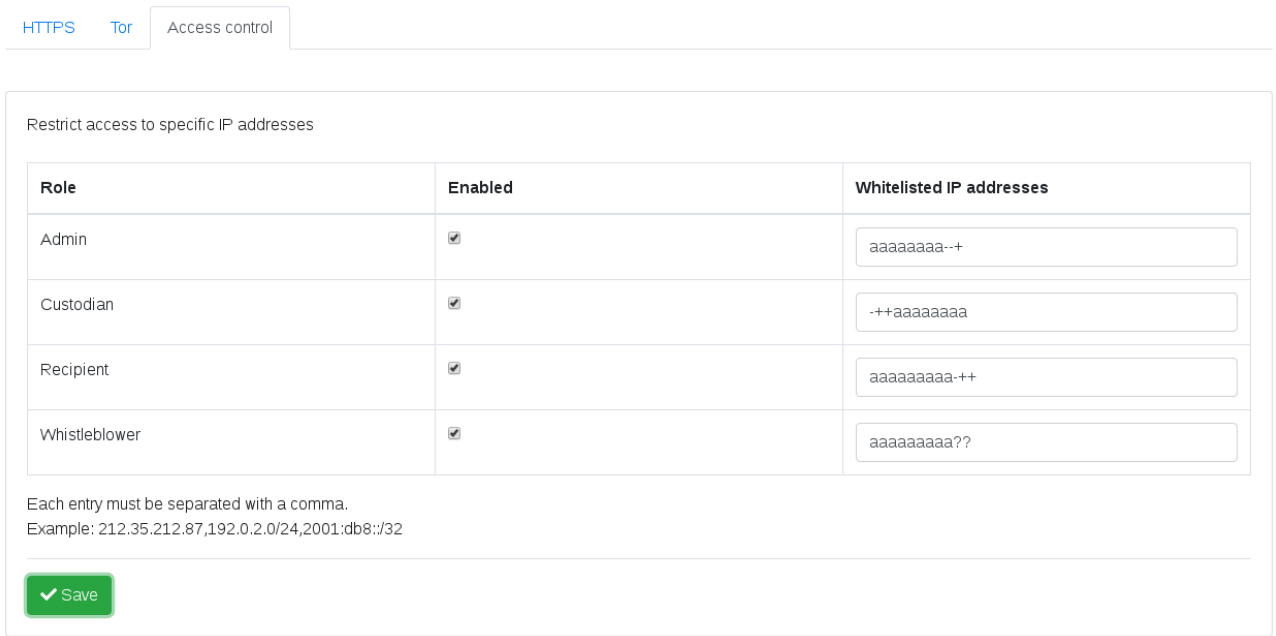
The following input validation issues were found:

1. It was found possible to inject Javascript into the whistleblowers comments section:

This demonstrates that it is possible to inject arbitrary scripts into the application's response (in JSON) which is against best practice. The response does not state that the content type is HTML and is only directly exploitable if a browser can be made to interpret the response as HTML, however no modern browser will interpret the response as HTML.

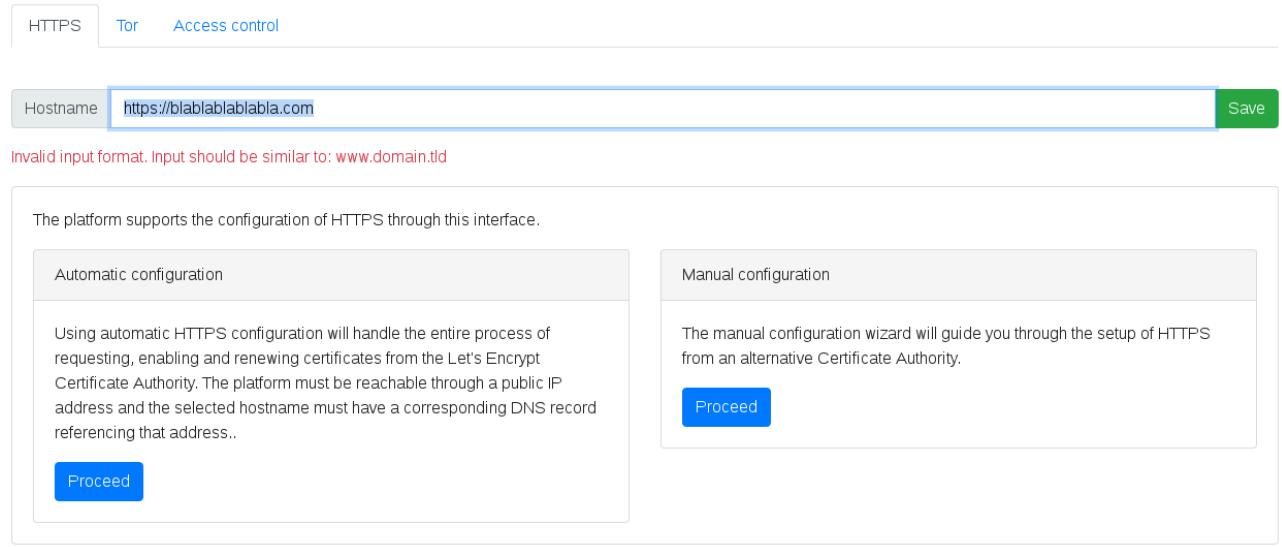
2.IP access control:

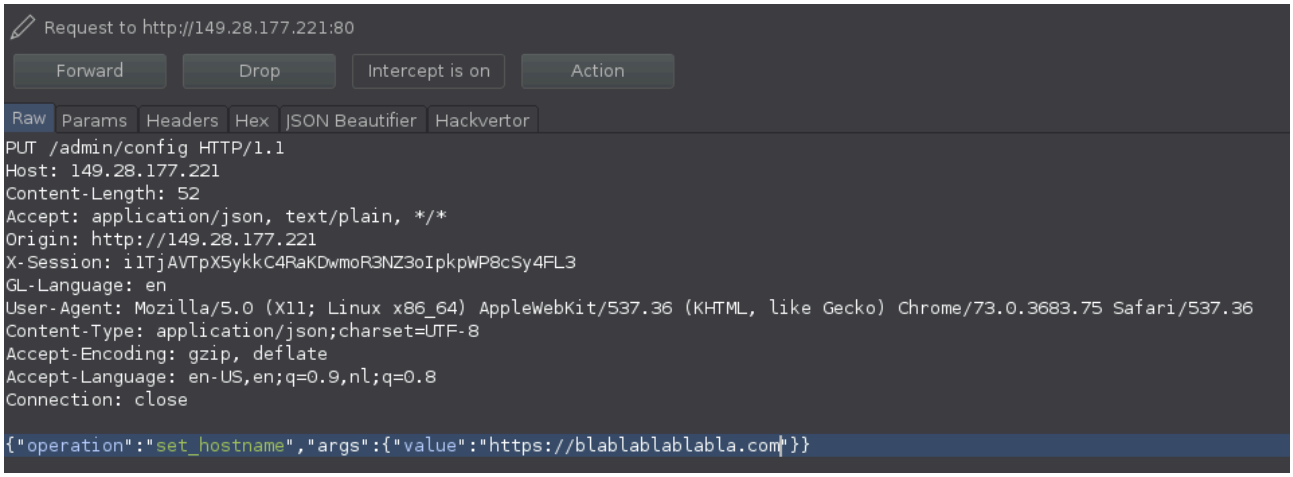
This input allows characters that are not valid in IP addresses:

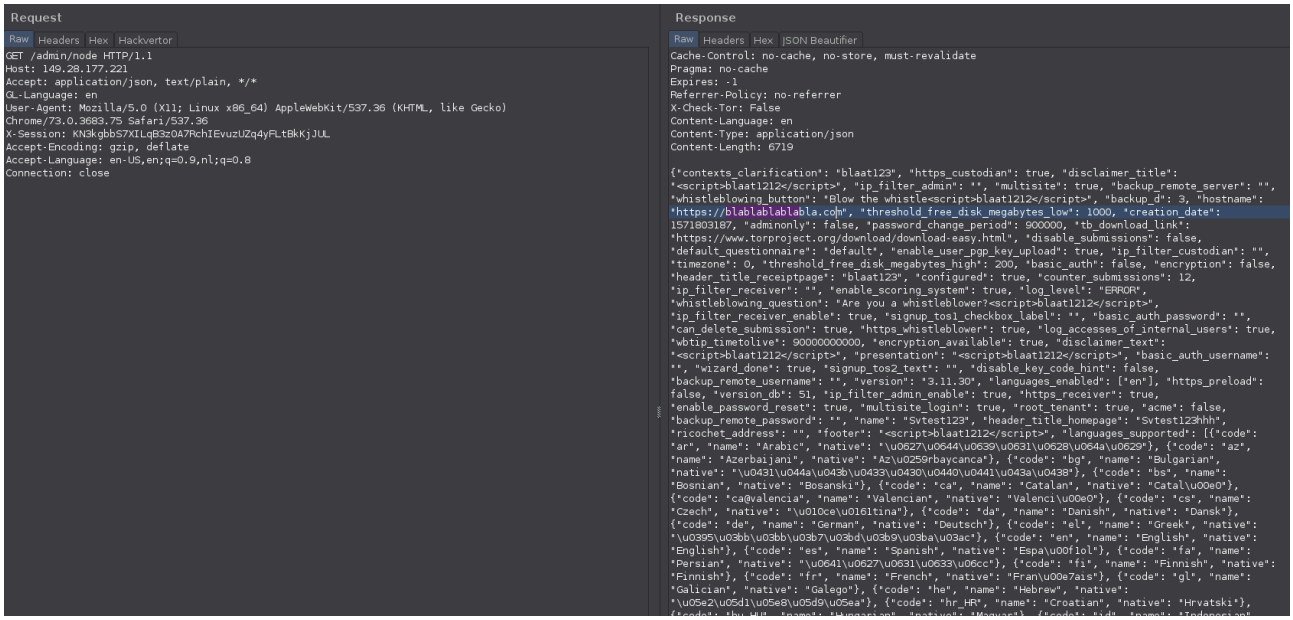


3. Set the hostname:

Bypassing client-side verification allows the use of potentially dangerous hostname characters, for instance `https://`:







Feedback from GlobaLeaks Team:

Status: To be fixed

The team has currently relied on the effectiveness of the AngularJS protections + server-side headers protection but considering the validity of the reported suggestions is evaluating to implement additional server-side escaping selecting a proper library like https://pypi.org/project/MarkupSafe/

Feedback from Pentest Team:

Using a Serverside escaping library will improve the input validation and will make the application more secure.

Impact:

An attacker could provide unexpected values and cause a program crash or excessive consumption of resources, such as memory and CPU.
An attacker could read confidential data if they are able to control resource references.
An attacker could use malicious input to modify data or possibly alter control flow in unexpected ways, including arbitrary command execution.

Recommendation:

- Assume all input is malicious. Use an 'accept known good' input validation strategy i.e. use a whitelist of acceptable inputs. Reject any input that does not strictly conform to specifications, or transform it into something that does (e.g. by stripping HTML tags).
- When performing input validation, consider all potentially relevant properties, including length, type of input, the range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules.

- Do not rely exclusively on looking for malicious or malformed inputs (i.e. do not rely on a blacklist). A blacklist is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However blacklists can be useful for detecting known or likely attacks or determining which inputs are so malformed that they should be rejected outright.
- For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then these modified values would be submitted to the server, so it's important that the server applies the same rules.
- Even though client-side checks provide minimal benefits with respect to server-side security, they are still useful. First, they can support intrusion detection. If the server receives input that should have been rejected by the client, then it may be an indication of an attack. Second, client-side error-checking can provide helpful feedback to the user about the expectations for valid input. Third, there may be a reduction in server-side processing time for accidental input errors, although this is typically a small saving.
- When your application combines data from multiple sources, perform the validation after the sources have been combined. The individual data elements may pass the validation step but violate the intended restrictions after they have been combined. Inputs should be decoded and canonicalised to the application's current internal representation before being validated.
- Make sure that your application does not inadvertently decode the same input twice. Such errors could be used to bypass whitelist schemes by introducing dangerous inputs after they have been checked.
- Consider performing repeated canonicalisation until your input does not change any more. This will avoid double-decoding and similar scenarios, but it might inadvertently modify inputs that are allowed to contain properly-encoded dangerous content.

4.17 OTF-017 — Submission Confidentiality Check Not Working Properly

Vulnerability ID: OTF-017 **Retest status:** Not Retested

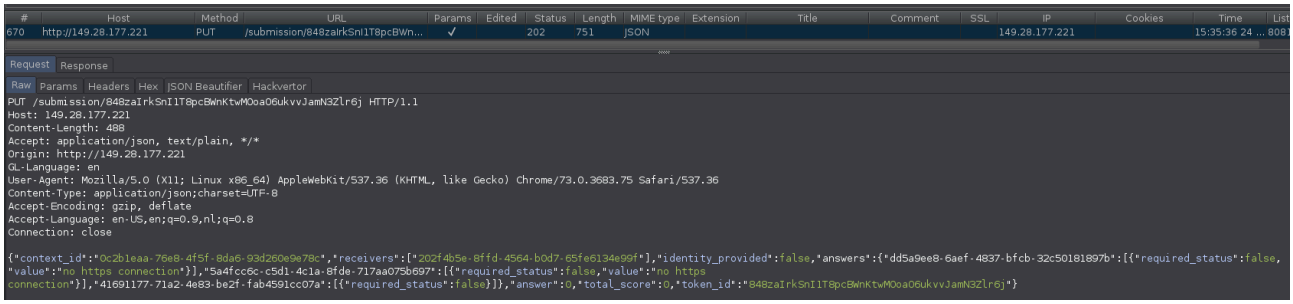
Vulnerability type: Sensitive Data Exposure

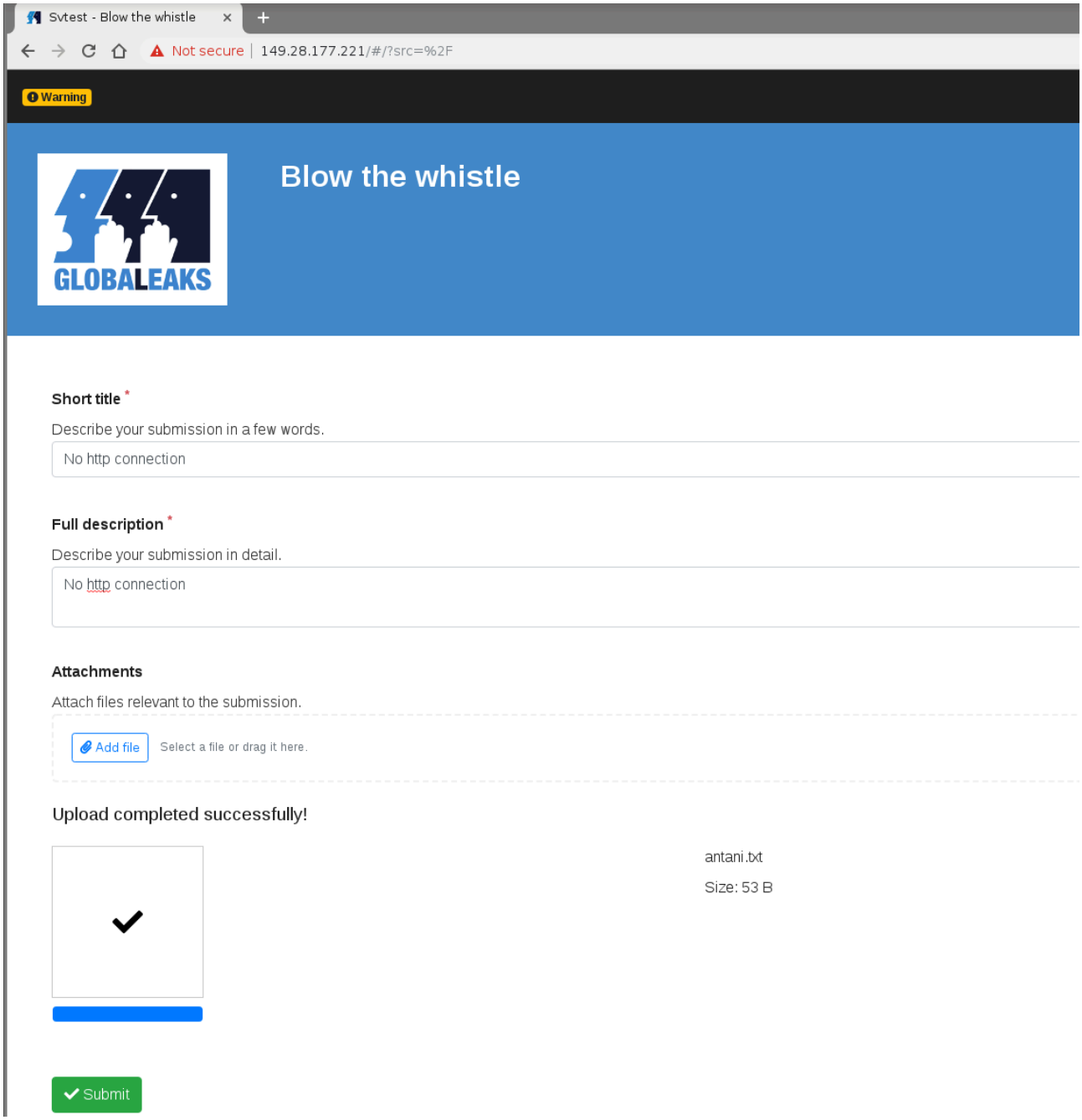
Threat level: Low

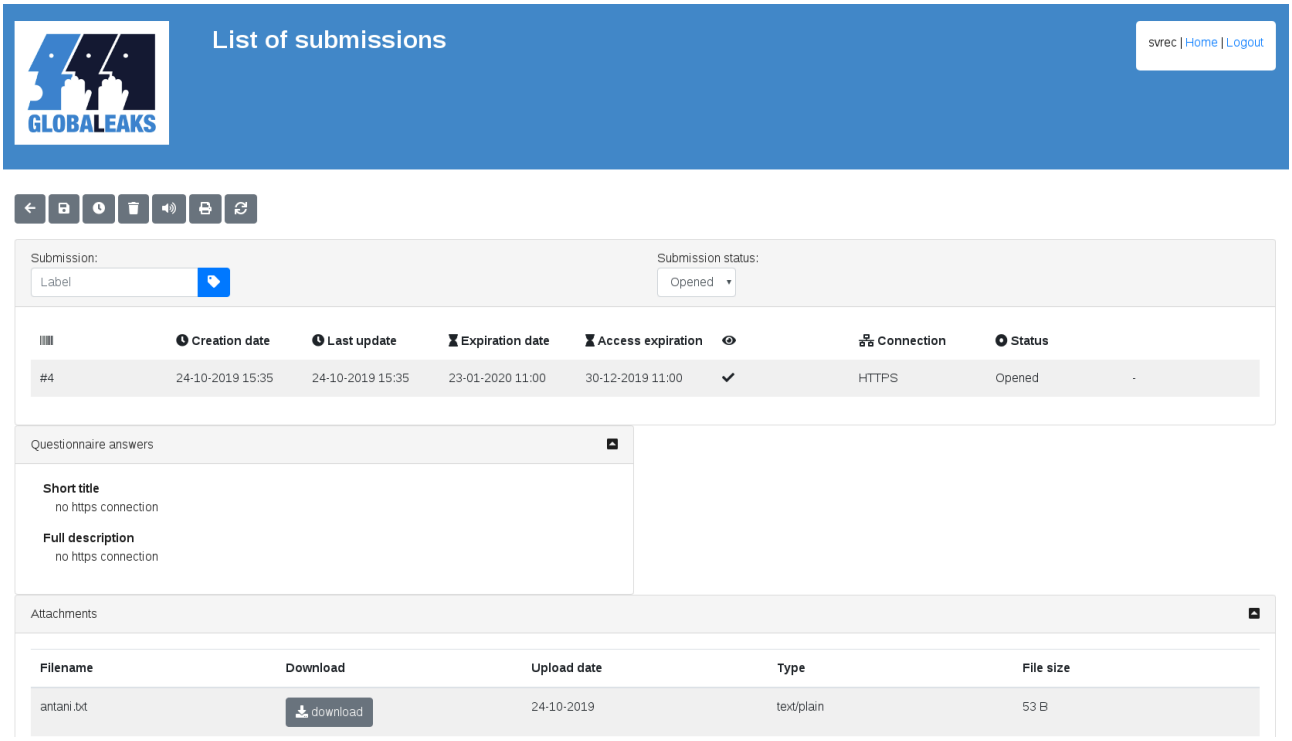
Description:

A whistleblower's tip was posted using an insecure connection, but the submission status indicates that it was posted using a secure HTTPS connection.

Technical description:







Feedback from GlobaLeaks Team:

Status: Accepted Risk

For simplicity of the implementation this check asses only the difference between traditional HTTP (called HTTPS) and Tor. Infact a production system will be always implemented only with HTTPS or Tor and to prevent users from using plain HTTP in production there is a big warning that clarify the user that says: “The connections is not secure The platform is still not configured for HTTPS connections and should therefore only be used for testing purposes.”

Feedback from Pentest Team:

Agree with feedback. This is the reason why the threatlevel is rated as Low.

Impact:

This would give users a false sense of security. Note that using an insecure HTTP connection would already trigger alarms on most modern browsers resulting in a Low instead of Moderate threat level.

Recommendation:

- Modify the submission response to display the correct status.

4.18 OTF-018 — Users Can Still Post When Submissions Are Disabled

Vulnerability ID: OTF-018 **Retest status:** Resolved

Vulnerability type: Broken Access Control

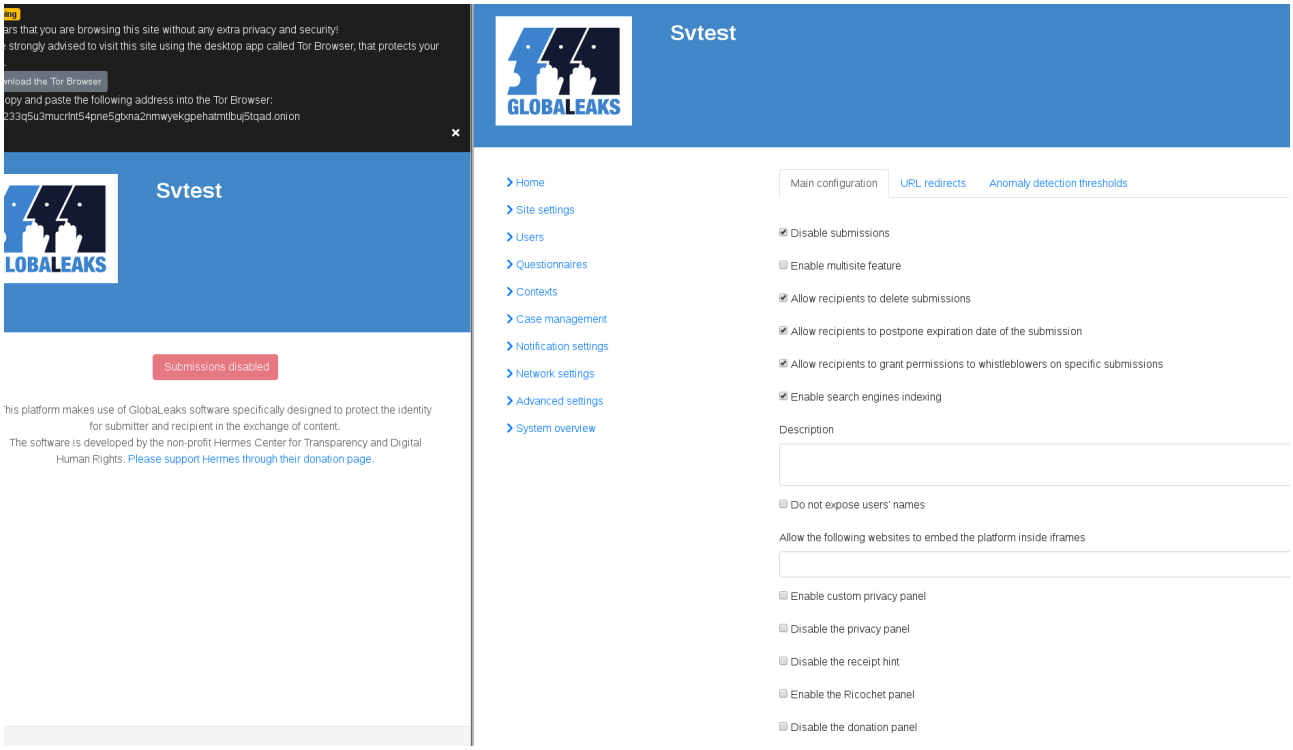
Threat level: Low

Description:

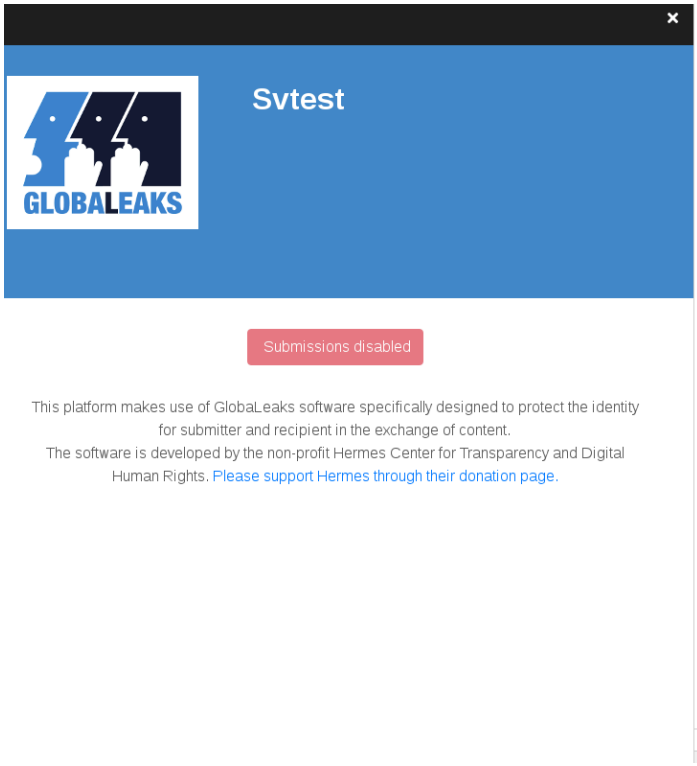
Submissions can be disabled, however, the way it's disabled doesn't actually prevent new submissions being submitted.

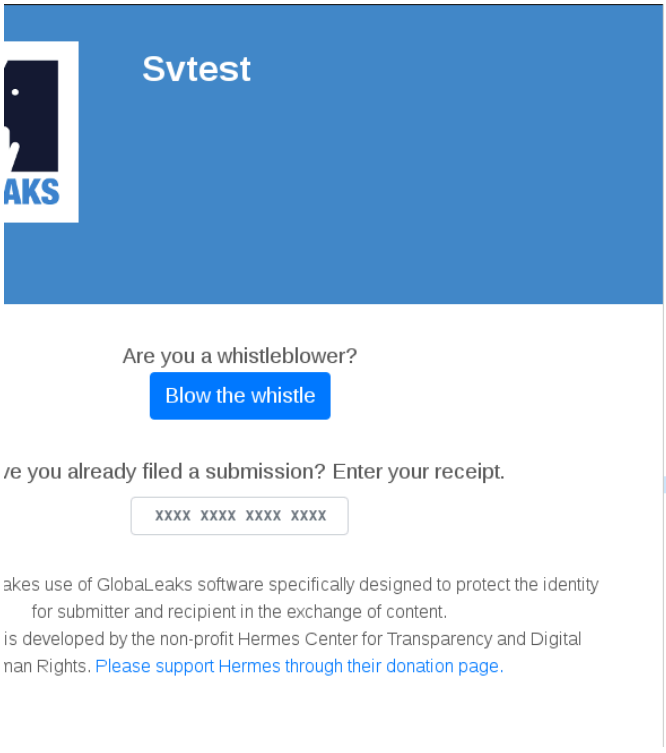
Technical description:

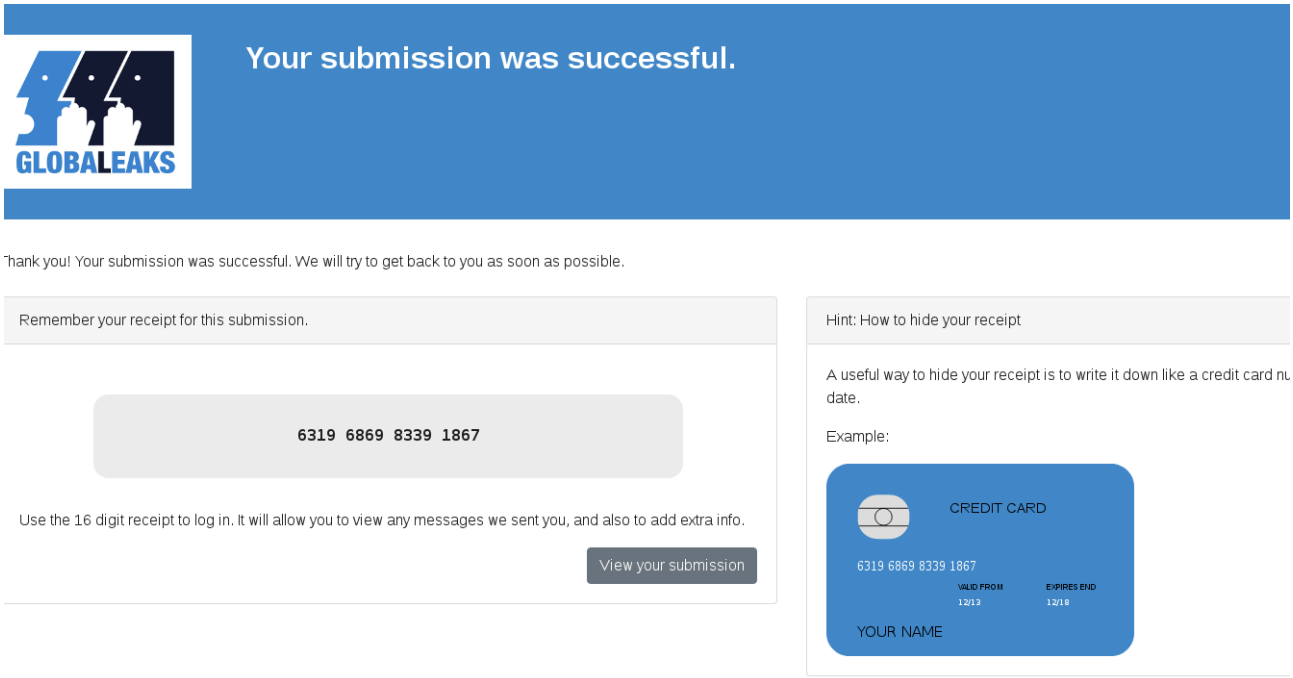
Submissions can be turned off in the admin portal resulting in disabling the submissions form on the website:



Changing the code on the client side allows the button to be enabled again and allows successful submission of tips:


Svtest
GLOBALEAKS
Submissions disabled
This platform makes use of GlobaLeaks software specifically designed to protect the identity for submitter and recipient in the exchange of content.
The software is developed by the non-profit Hermes Center for Transparency and Digital Human Rights. Please support Hermes through their donation page.


Svtest
Are you a whistleblower?
Blow the whistle
/e you already filed a submission? Enter your receipt.
XXXX XXXX XXXX XXXX
akes use of GlobaLeaks software specifically designed to protect the identity for submitter and recipient in the exchange of content.
is developed by the non-profit Hermes Center for Transparency and Digital
nan Rights. Please support Hermes through their donation page.



Feedback from GlobaLeaks Team:

Status: Fixed

Commit of the fix: https://github.com/globaleaks/GlobaLeaks/commit/a8f97e76b5e76ab0717b3f41dddd42210325e4c1#diff-4b2133b4ef298af6675f3adc8a65e502

Feedback from Pentest Team:

Fixed

Impact:

The application logic can be bypassed.

Recommendation:

- Verify the submission setting on the server side, and reject submissions when they are disabled.

4.19 OTF-019 — Header Injection

Vulnerability ID: OTF-019 **Retest status:** Resolved

Vulnerability type: Input validation

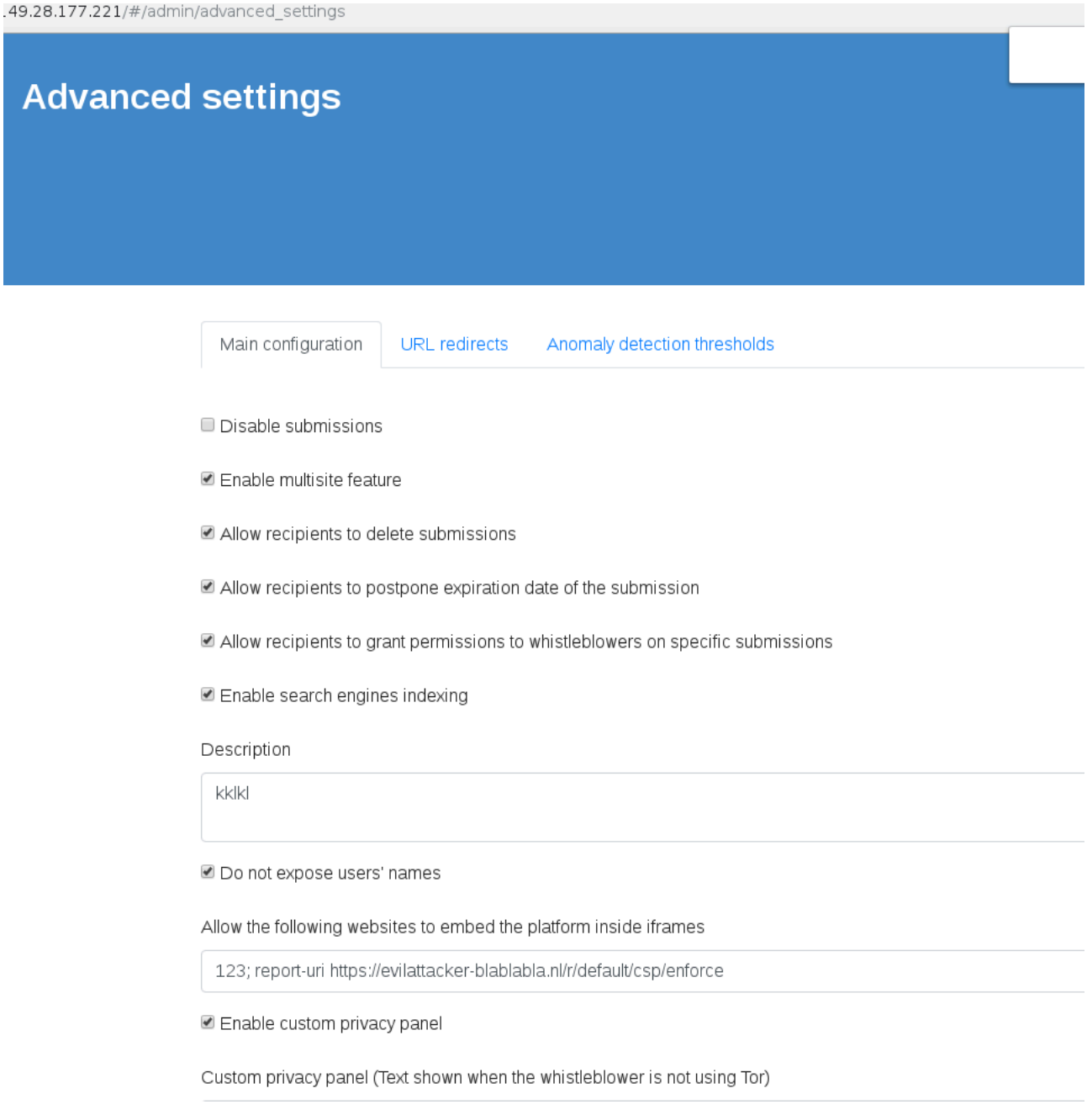
Threat level: Low

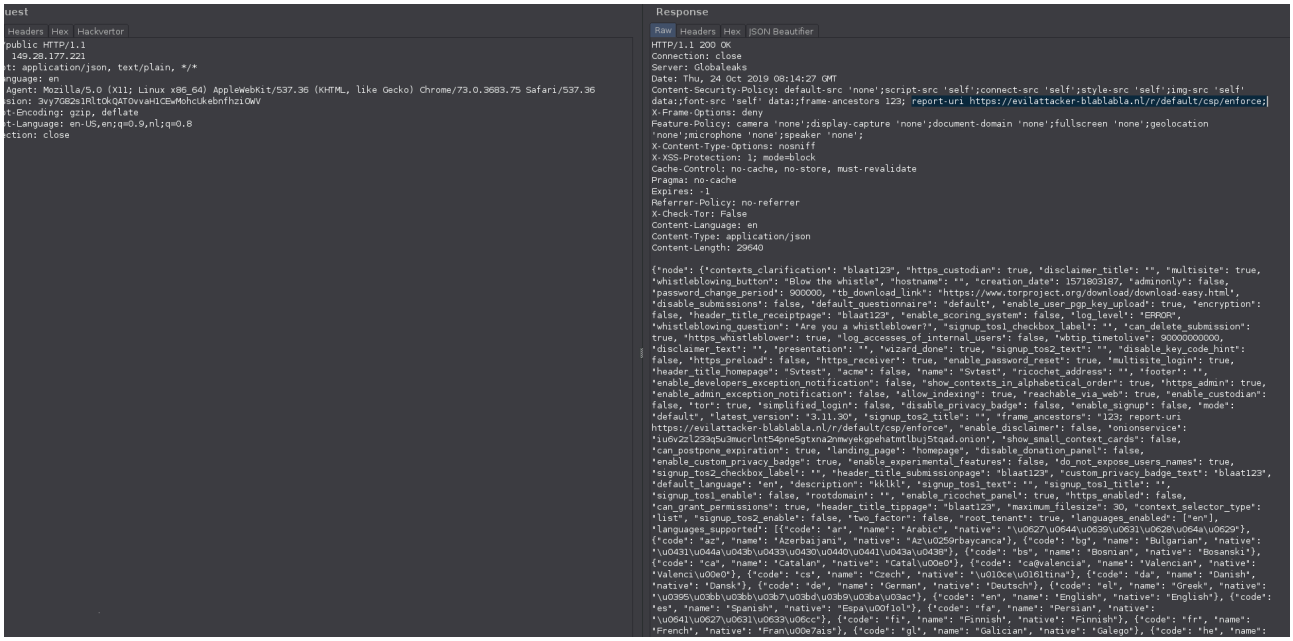
Description:

When logged in as an admin it is possible to set the option "Allow the following websites to embed the platform inside iframes". This option does not validate whether the user input is a URL, resulting in inserting other tags in the CSP header.

Technical description:

When logged in as an admin it is possible to set the option "Allow the following websites to embed the platform inside iframes". This option does not validate whether the user input is an URL resulting inserting other tags to the CSP header.


.49.28.177.221/#/admin/advanced_settings
Advanced settings
Main configuration
URL redirects
Anomaly detection thresholds
Disable submissions
Enable multisite feature
Allow recipients to delete submissions
Allow recipients to postpone expiration date of the submission
Allow recipients to grant permissions to whistleblowers on specific submissions
Enable search engines indexing
Description
kklkl
Do not expose users' names
Allow the following websites to embed the platform inside iframes
123; report-uri https://evilattacker-blablabla.nl/r/default/csp/enforce
Enable custom privacy panel
Custom privacy panel (Text shown when the whistleblower is not using Tor)



Feedback from GlobaLeaks Team:

Status: Fixed

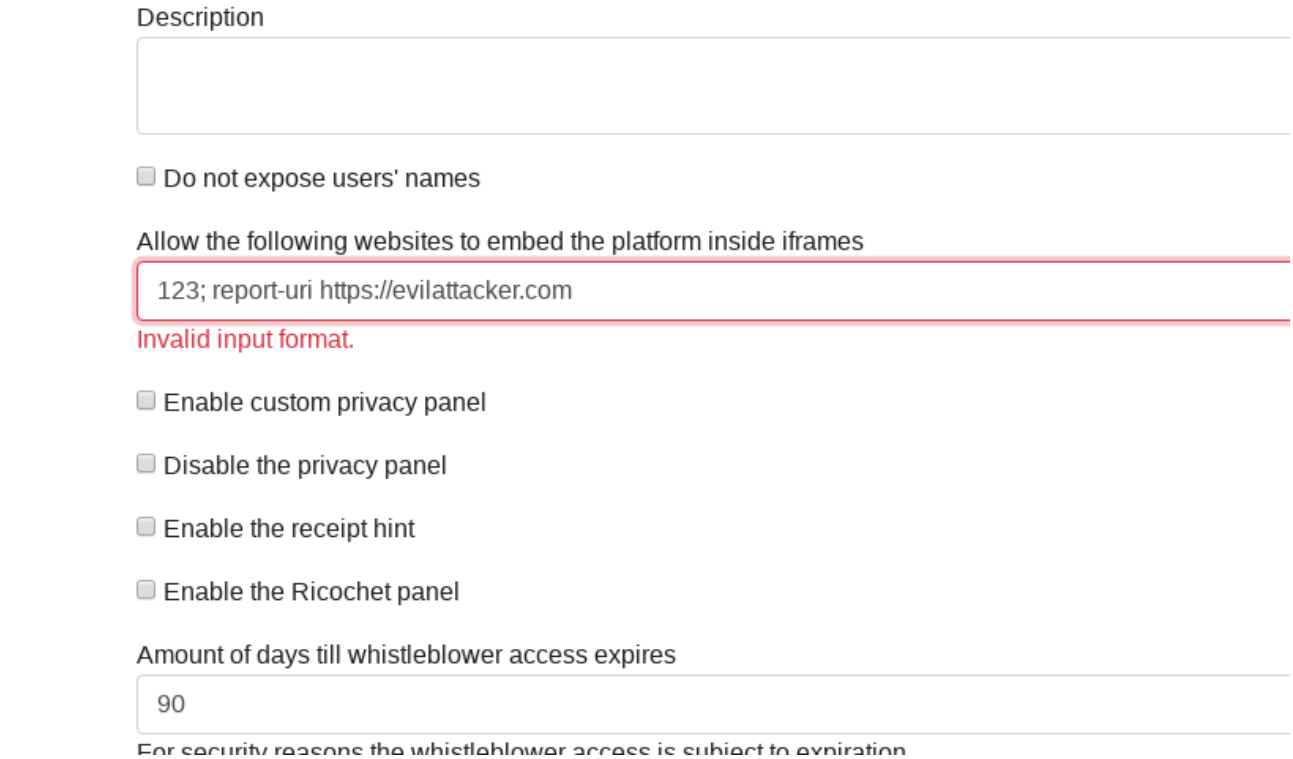
Fixed with the addition of a a check using a strict regexp check both on the backend and the frontend enabling only relevant charset.

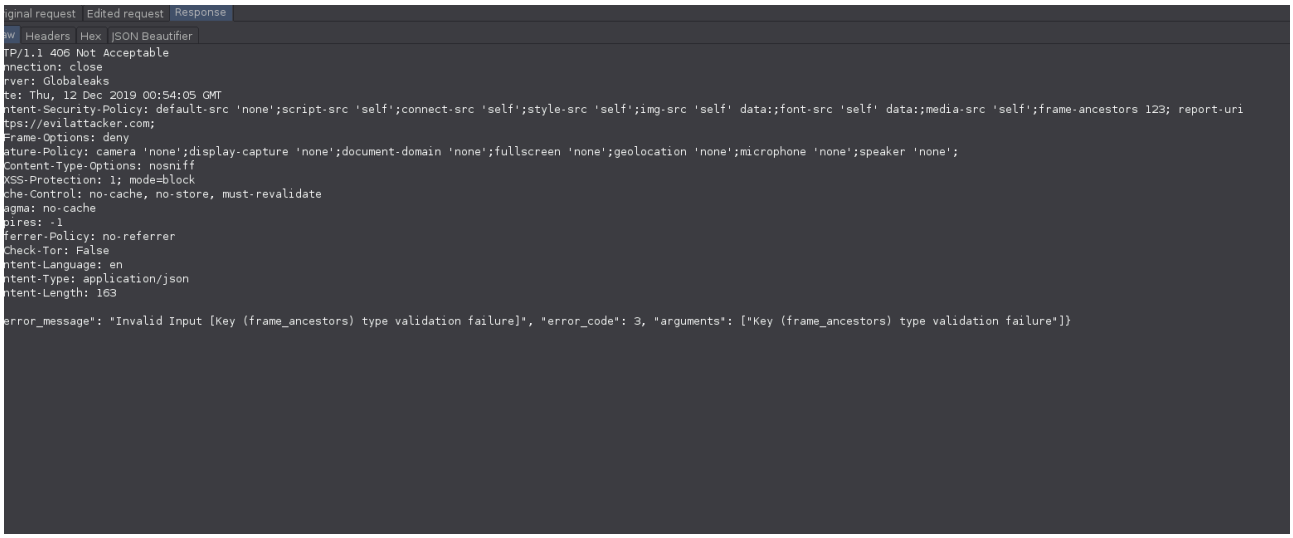
```
Regexp: ^((([0-9]+,)*[0-9]+)|)$
```

Commit of the fix: https://github.com/globaleaks/GlobaLeaks/commit/9be7e5abac03ac775d5f72c3fffd06a0c56f4cb7

Feedback from Pentest Team:

Fixed





Impact:

Modification of the content security policy may permit resource origins that should be blocked.

Recommendation:

- Validate URLs before using them in CSP configuration.

4.20 OTF-020 — Improper File Extension Validation in the Logo Upload Functionality.

Vulnerability ID: OTF-020 **Retest status:** Not Retested

Vulnerability type: Insecure File Upload

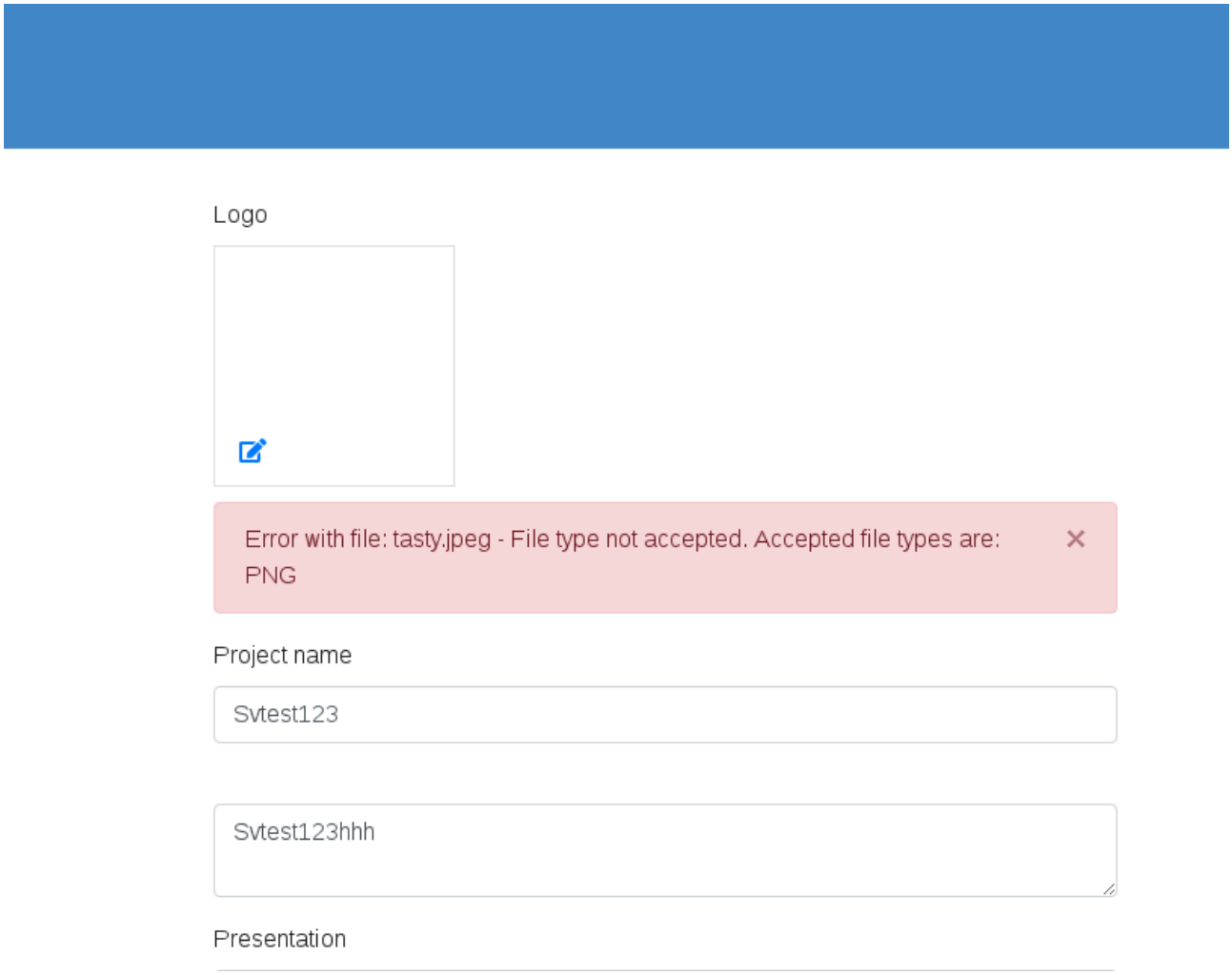
Threat level: Low

Description:

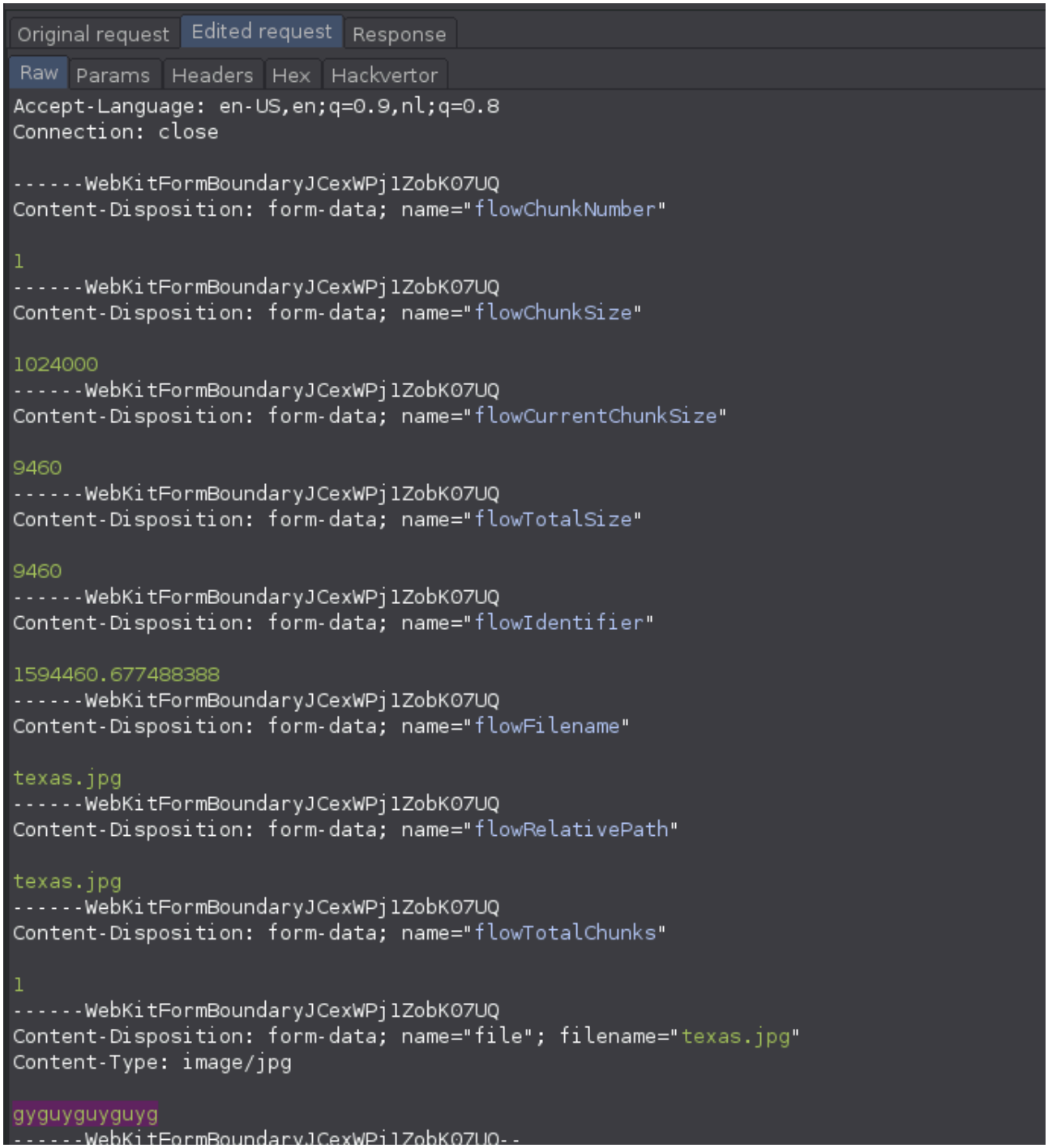
Only PNG image files should be allowed as logo uploads. However this check is only performed on the client side which can be disabled, resulting in allowing upload of files of any type.

Technical description:

To upload a new (site or context) logo as an admin or as a recipient (by default the recipient does not have access) only PNG files are allowed:



This is only validated on the client-side:


Original request
Edited request
Response
Raw
Params
Headers
Hex
Hackvertor
Accept-Language: en-US,en;q=0.9,nl;q=0.8
Connection: close
------WebKitFormBoundaryJCexWPj1ZobK07UQ
Content-Disposition: form-data; name="flowChunkNumber"
1
------WebKitFormBoundaryJCexWPj1ZobK07UQ
Content-Disposition: form-data; name="flowChunkSize"
1024000
------WebKitFormBoundaryJCexWPj1ZobK07UQ
Content-Disposition: form-data; name="flowCurrentChunkSize"
9460
------WebKitFormBoundaryJCexWPj1ZobK07UQ
Content-Disposition: form-data; name="flowTotalSize"
9460
------WebKitFormBoundaryJCexWPj1ZobK07UQ
Content-Disposition: form-data; name="flowIdentifier"
1594460.677488388
------WebKitFormBoundaryJCexWPj1ZobK07UQ
Content-Disposition: form-data; name="flowFilename"
texas.jpg
------WebKitFormBoundaryJCexWPj1ZobK07UQ
Content-Disposition: form-data; name="flowRelativePath"
texas.jpg
------WebKitFormBoundaryJCexWPj1ZobK07UQ
Content-Disposition: form-data; name="flowTotalChunks"
1
------WebKitFormBoundaryJCexWPj1ZobK07UQ
Content-Disposition: form-data; name="file"; filename="texas.jpg"
Content-Type: image/jpg
gyguyguyguyg
------WebKitFormBoundaryJCexWPj1ZobK07UQ--


Raw
Headers
Hex
HTTP/1.1 201 Created
Connection: close
Server: Globaleaks
Date: Thu, 24 Oct 2019 21:11:40 GMT
Content-Security-Policy: default-src 'none';script-src 'self';connect-src 'self';style-src 'self';img-src 'self' data:;font-src 'self' data:;frame-ancestors 123; report-uri
https://evilattacker-blablabla.nl/r/default/csp/enforce;
X-Frame-Options: deny
Feature-Policy: camera 'none';display-capture 'none';document-domain 'none';fullscreen 'none';geolocation 'none';microphone 'none';speaker 'none';
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: -1
Referrer-Policy: no-referrer
X-Check-Tor: False
Content-Language: en
Content-Type: text/html
Content-Length: 0

```
POST /admin/files/logo HTTP/1.1
Host: 149.28.177.221
Content-Length: 1091
Origin: http://149.28.177.221
GL-Language: en
X-Session: BJSig83eNyOUcJEORiAQy70kw7VNRfFCRL2Vc4GvWr
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecl
Chrome/73.0.3683.75 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryaAumDgliyLbzB
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,nl;q=0.8
Connection: close

------WebKitFormBoundaryaAumDgliyLbzBBa3
Content-Disposition: form-data; name="flowChunkNumber"

1
------WebKitFormBoundaryaAumDgliyLbzBBa3
Content-Disposition: form-data; name="flowChunkSize"

1024000
------WebKitFormBoundaryaAumDgliyLbzBBa3
Content-Disposition: form-data; name="flowCurrentChunkSize"

1141
------WebKitFormBoundaryaAumDgliyLbzBBa3
Content-Disposition: form-data; name="flowTotalSize"

1141
------WebKitFormBoundaryaAumDgliyLbzBBa3
Content-Disposition: form-data; name="flowIdentifier"

1693021.4421024292
------WebKitFormBoundaryaAumDgliyLbzBBa3
Content-Disposition: form-data; name="flowFilename"

image.php.png
------WebKitFormBoundaryaAumDgliyLbzBBa3
Content-Disposition: form-data; name="flowRelativePath"

image.php.png
------WebKitFormBoundaryaAumDgliyLbzBBa3
Content-Disposition: form-data; name="flowTotalChunks"

1
------WebKitFormBoundaryaAumDgliyLbzBBa3
Content-Disposition: form-data; name="file"; filename="image.png"
Content-Type: image/png

<script>alert("XSS");</script>
------WebKitFormBoundaryaAumDgliyLbzBBa3--
```

Base64 image shown on the homepage:

```
<img data-ng-click="setHomepage()" class="img-fluid" alt="project logo" data-ng-src="" role="button" tabindex="0" src="">
```

Base64 decoded:

```
<img data-ng-click="setHomepage()" class="img-fluid" alt="project logo" data-ng-src="<script>alert("XSS");</script>" role="button" tabindex="0" src="<script>alert("XSS");</script>">
```

Feedback from GlobaLeaks Team:

Status: To be fixed

Remediation to be identified. Issue related to issue: OTF-010 (page 37)

Feedback from Pentest Team:

No feedback required.

Impact:

The risk is low as it was not possible to create a working XSS because the content-type cannot be controlled by the user; it is hard-coded in the application code as `data:image/png;base64`.

Recommendation:

- Validate file and content type on the server side before accepting and storing uploaded files on the server.

4.21 OTF-021 — SSH Port Publicly Exposed on Default Installation

Vulnerability ID: OTF-021 **Retest status:** Not Retested

Vulnerability type: Security Misconfiguration

Threat level: Low

Description:

On a default install the SSH port of the server is publicly accessible.

Technical description:

During this audit a standard Debian 10 image was used, which opens the SSH port by default. Having this port publicly exposed increases the attack surface. Ports 80 and 443 are also open, but are required for the GlobaLeaks application to work.

Note that hardening the OS is the responsibility of the sysadmin, though reminding them and providing ways to reduce the attack surface is something that application vendors could assist in.

GlobaLeaks mentioned during testing feedback that they are currently working on sandboxing the GlobaLeaks application, leaving the user to do the rest, but acknowledges that this can be improved in the documentation too.

Feedback from GlobaLeaks Team:

Status: Partially fixable

The team considers this topic of server securitization as external to the software control. The software does currently implement some firewall sandboxing but still just in strict relation to the software that is sandboxed in its operativity. The team understands the suggestions and will discuss the topic in the documentation of the software providing some general instructions on where to read about how to generally security the server on which GlobaLeaks is set up.

Feedback from Pentest Team:

Agree with the solution.

Impact:

Publicly open ports increase the attack surface of the server unnecessarily.

Recommendation:

- Restrict access by installing a firewall and restricting SSH access to specific source IPs.

4.22 OTF-022 — Local Privilege Escalation or Data Corruption in the Installation Script

Vulnerability ID: OTF-022 **Retest status:** Not Retested

Vulnerability type: Local Privilege Escalation

Threat level: Low

Description:

An installation script writes to a file path relative to the current directory, without knowledge of the user running the script. This can be abused by local users in some conditions. The script is expected to be executed with the privileges of the root user.

Technical description:

In file `scripts/install.sh`:

```
27 function DO () {
28   CMD="$1"
29
30   if [ -z "$2" ]; then
31     EXPECTED_RET=0
32   else
33     EXPECTED_RET=$2
34   fi
35
36   echo -n "Running: \"$CMD\"... "
37   eval $CMD &>${LOGFILE}
38
39   STATUS=$?
40
41   last_command $CMD
42   last_status $STATUS
43
44   if [ "$STATUS" -eq "$EXPECTED_RET" ]; then
45     echo "SUCCESS"
46   else
47     echo "FAIL"
48     echo "Ouch! The installation failed."
49     echo "COMBINED STDOUT/STDERR OUTPUT OF FAILED COMMAND:"
50     cat ${LOGFILE}
51     exit 1
52   fi
53 }
54
55 LOGFILE="./install.log"
```

In this code, the output of every command executed is output to the file `install.log`. When executing this script in a folder such as `/tmp`, where every user can create arbitrary files, a local user can create a symbolic link with this name, pointing to other files in the system, and possibly altering information or gaining privileges of the root user.

Feedback from GlobaLeaks Team:

Status: To be fixed

The team will evaluate and address the issue by using the standard mktemp approach by relying on utilities of this family for the related script language (e.g. Bash / Python) Issue related to issue OTF-014 (page 45); the two issues will be managed with the same solution.

Feedback from Pentest Team:

Indeed using mktemp is generally the way to go, depending on the language used; be careful to use `mkstemp()` or `mkdtemp()` as opposed to `mktemp()` when using C (or a C binding), since the latter is considered insecure by today's standards.

Impact:

Local users may be able to elevate privileges to that of the root user.

Recommendation:

- Create this log file in a location inaccessible to local users.

4.23 OTF-023 — Potential Conflict Between Keys

Vulnerability ID: OTF-023 **Retest status:** Not Retested

Vulnerability type: Data Corruption

Threat level: Low

Description:

GlobaLeaks generates a private key per tip received. This key is identified through a random string of 16 alphanumeric characters. Should the random number generator generate the same sequence as an existing key, the new key will be appended to the existing one.

Technical description:

According to the developers of GlobaLeaks, this was considered and a UNIQUE constraint prevents the collision from happening. Colliding submissions should just fail, after being rejected by the database.

In any case, the database constraint really has to be met and handled before anything is written to disk to prevent a collision from harming existing data.

Feedback from GlobaLeaks Team:

Status: Should be already fixed; Under review

To address this issue the system already implements a database constraint on transactions. In case of failures the operations performed on the database are rolled-back and submission resulted in a conflict generates an error. This in general happens in the system for any other conflict like for example in UUID4 generation of new database objects.

Feedback from Pentest Team:

Make sure so that such database constraints are really met and handled before anything is written to disk; this might not be the case when using transactions for instance.

Impact:

Key material may be corrupted.

Recommendation:

- Make sure collisions are detected and prevented sufficiently early in the submission process.

4.24 OTF-024 — Potential Local Privilege Escalation Through Backups

Vulnerability ID: OTF-024 **Retest status:** Not Retested

Vulnerability type: Local Privilege Escalation

Threat level: Low

Description:

Backups use the system `/tmp` directory in a way that may be exploitable.

Technical description:

In file `backend/bin/gl-admin`, function `default_backup_path()`:

```
45 def default_backup_path():
46     t = datetime.now().strftime("%y_%m_%d")
47     name = "globaleaks_backup_{}.tar.gz".format(t)
48     return os.path.join("/tmp", name)
```

This code defaults to writing to a file with a predictable name in the `/tmp` directory when creating backups. This could be abused by a local attacker through a symlink attack, where the attacker would prepare symlinks pointing to a directory under its own control (and possibly gain access to the entire data) or trick a GlobaLeaks administrator into creating or erasing one of its own files.

Feedback from GlobaLeaks Team:

Status: To be fixed

Issue related to issue OTF-022 (page 69); the two issues will be managed with the same solution.

Feedback from Pentest Team:

The same feedback applies.

Impact:

Local users may access the database or escalate privileges when backups are created.

Recommendation:

- Generate files securely when writing to shared folders.

5 Non-Findings

In this section we list some of the things that were tried but turned out to be dead ends.

5.1 GlobaLeaks Defaults to Generating HTTP Links for Tor Services

In file `backend/globaleaks/utils/templating.py`, methods `NodeKeyword::TorSite()`:

```
167     def TorSite(self):
168         if self.data['node']['onionservice']:
169             return 'http://' + self.data['node']['onionservice']
170 
171         return '[UNDEFINED]'
```

and `PlatformSignupKeyword::TorSite()`:

```
544     def TorSite(self):
545         return 'http://' + self.data['signup']['subdomain'] + '.' + self.data['node']
['onionservice']
```

and `PlatformSignupKeyword::ActivationUrl()`:

```
562     def ActivationUrl(self):
563         if self.data['node']['hostname']:
564             site = 'https://' + self.data['node']['hostname']
565         elif self.data['node']['onionservice']:
566             site = 'http://' + self.data['node']['onionservice']
567         else:
568             site = ''
569 
570         return site + '/#/activation?token=' + self.data['signup']['activation_token']
```

Even though Tor exit nodes cannot be trusted, HTTP links for Onion services can be accessed safely. This is because they are designed to be accessed through a circuit ending on the computer serving them, typically on the virtual local interface there. Therefore an SSL certificate, valid or not, does not provide additional security, and may even harm anonymity of the service.

Regardless, some Onion services may be still be using HTTPS instead of HTTP in spite of this. The links generated will then fail.

See also https://blog.torproject.org/facebook-hidden-services-and-https-certs for more information.

6 Future Work

- **Retest of fixed issues**

  Once the issues mentioned in this penetration test have been fixed, we recommend performing a retest with the fixes/patches applied - Sometimes fixes are inadequate or may introduce new vulnerabilities, so it is important to verify their proper implementation.
- **Test of external dependencies**

  The code base depends on third-party components, from public repositories. As explained in this report, the security of this project relies on the quality and stability of these repositories. It would therefore make sense to audit their code as well.
- **Regular security assessments**

  Security is an ongoing process and not a product, so we advise undertaking regular security assessments and penetration tests, ideally prior to every major release or every quarter.

7 Conclusion

The High and Elevated findings would allow an authenticated attacker to take full control over the application which would have a significant impact on its confidentiality, integrity and availability.

Several sensitive data exposure findings were found that should be addressed to protect sensitive user data against unauthorized users. During the review of the encryption protocol design no issues were found.

The source code audit found and documented security issues where server instances of GlobaLeaks may leak data back to the project, or execute arbitrary code within the continuous integration system. Even though both cases were deliberate decisions by the developers, this behaviour needed to be documented here.

A security issue with the handling of SSL/TLS connections was confirmed and then fixed within hours of reporting. Besides this, no major issues were found within the source code itself for what is perceived as typical deployments of the platform.

We recommend fixing all of the issues found, and to perform a retest in order to ensure that mitigations are effective and that new vulnerabilities have not been introduced.

Finally, we want to emphasize that security is a process – this penetration test is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security. 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 Testcases

The following tests were conducted:

- Verify that all pages and resources require authentication by default, except those specifically intended to be public (Principle of complete mediation).
- Verify that none of the password fields echo the user's password when it is entered.
- Verify that all authentication controls are enforced on the server side.
- Verify that the changing password functionality includes the old password, the new password, and a password confirmation.
- Verify that all authentication controls fail securely to ensure attackers cannot log in.
- Verify that account passwords make use of a sufficient strength encryption routine that withstands brute force attacks.
- Verify that credentials are transported using a suitably encrypted link and that all pages/functions that require a user to enter credentials do so using an encrypted link.
- Verify that the forgotten password function and other recovery paths do not reveal the current password and that the new password is not sent to the user in clear text.
- Verify there are no default passwords in use for the application framework or any components used by the application (such as "admin/password").
- Verify that information enumeration is not possible via login, password reset, or forgot password functionality.
- Verify that request throttling is in place to prevent automated attacks against common authentication attacks such as brute force attacks or denial of service attacks.
- Verify that forgotten password and other recovery paths use a soft token, mobile push, or an offline recovery mechanism.
- Verify that if knowledge based questions (also known as 'secret questions') are required, the questions should be strong enough to protect the application.
- Verify that measures are in place to block the use of commonly chosen passwords and weak passphrases.
- Verify that sessions are invalidated when the user logs out.
- Verify that administrative interfaces are not accessible to untrusted parties.
- Verify that sessions timeout/expire after a specified period of inactivity
- Verify that the session id is never disclosed in URLs, error messages, or logs. This includes verifying that the application does not support URL rewriting of session cookies.
- Verify that all successful authentication and re-authentication generates a new session and session id.
- Verify that session ids are sufficiently long, random and unique across the correct active session base.
- Verify that session ids stored in cookies have their path set to an appropriately restrictive value for the application, and authentication session tokens additionally set the 'HttpOnly' and 'Secure' attributes
- Verify that the application limits the number of active concurrent sessions.

- Verify that the principle of least privilege exists - users should only be able to access functions, data files, URLs, controllers, services, and other resources, for which they possess specific authorisation. This implies protection against spoofing and elevation of privilege.
- Verify that access to sensitive records is protected, such that only authorised objects or data is accessible to each user (for example, protect against users tampering with a parameter to see or alter another user's account).
- Verify that directory browsing/indexing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.
- Verify that the same access control rules implied by the presentation layer are enforced on the server side.
- Verify that all SQL queries, HQL, OSQL, NOSQL and stored procedures, calling of stored procedures are protected by the use of prepared statements or query parameterization, and thus not susceptible to SQL injection.
- Verify that the application is not susceptible to LDAP Injection, or that security controls prevent LDAP Injection.
- Verify that the application is not susceptible to OS Command Injection, or that security controls prevent OS Command Injection.
- Verify that the application is not susceptible to Remote File Inclusion (RFI) or Local File Inclusion (LFI) when content is used that is a path to a file.
- Verify that the application is not susceptible to common XML attacks, such as XPath query tampering, XML External Entity attacks, and XML injection attacks.
- Ensure that all string variables placed into HTML or other web client code is either properly contextually encoded manually, or utilise templates that automatically encode contextually to ensure the application is not susceptible to reflected, stored and DOM Cross-Site Scripting (XSS) attacks.
- Verify that all input data is validated, not only HTML form fields but all sources of input such as REST calls, query parameters, HTTP headers, cookies, batch files, RSS feeds, etc; using positive validation (whitelisting), then lesser forms of validation such as greylisting (eliminating known bad strings), or rejecting bad inputs (blacklisting).
- Verify that unstructured data is sanitised to enforce generic safety measures such as allowed characters and length, and characters potentially harmful in given context should be escaped (e.g. natural titles with Unicode or apostrophes).
- Verify that authenticated data is cleared from client storage, such as the browser DOM, after the session is terminated.
- Verify that all forms containing sensitive information have disabled client side caching, including autocomplete features.
- Verify that all sensitive data is sent to the server in the HTTP message body or headers (i.e., URL parameters are never used to send sensitive data).
- Verify that the application sets appropriate anti-caching headers as per the risk of the application, such as the following: Expires: Tue, 03 Jul 2001 06:00:00 GMT; Last-Modified: {now}; GMT Cache-Control: no-store, no-cache, must-revalidate, max-age=0; Cache-Control: post-check=0, pre-check=0; Pragma: no-cache.
- Verify that data stored in client side storage - such as HTML5 local storage, session storage, IndexedDB, regular cookies or Flash cookies - does not contain sensitive data or PII.

- Verify that HTTP Strict Transport Security headers are included on all requests and for all subdomains, such as Strict-Transport-Security: max-age=15724800; includeSubdomains
- Verify that only strong algorithms, ciphers, and protocols are used, through all the certificate hierarchy, including root and intermediary certificates of your selected certifying authority. This includes verifying that weak RC4 cipher or cipher less than 128bits is not in use.
- Verify that the TLS settings are in line with current leading practice, particularly as common configurations, ciphers, and algorithms become insecure.
- Verify that the application accepts only a defined set of required HTTP request methods, such as GET and POST are accepted, and unused methods(e.g. TRACE, PUT, and DELETE) are explicitly blocked.
- Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8, ISO 8859-1).
- Verify that the HTTP headers or any part of the HTTP response do not expose detailed version information of system components.
- Verify that all API responses contain X-Content-Type-Options: nosniff and Content-Disposition: attachment; filetitle = 'api.json' (or other appropriate filetitle for the content type).
- Verify that the X-XSS-Protection: 1; mode=block header is in place.
- Verify that URL redirects and forwards only allow whitelisted destinations, or show a warning when redirecting to potentially untrusted content.
- Verify that untrusted file data submitted to the application is not used directly with file I/O commands, particularly to protect against path traversal, local file include, file mime type, and OS command injection vulnerabilities
- Verify that files obtained from untrusted sources are validated to be of expected type and scanned by antivirus scanners to prevent upload of known malicious content.
- Verify that untrusted data is not used within cross-domain resource sharing (CORS) to protect against arbitrary remote content.
- Verify the application code does not execute uploaded data obtained from untrusted sources.
- Verify that access to administration and management functions within the application is limited to administrators.
- Verify the use of session-based authentication and authorisation.
- Verify that the application is protected from Cross-Site Request Forgery (CSRF).
- Verify the application explicitly checks the incoming Content-Type to be the expected one, such as application/xml or application/json.
- Verify application does not utilise third-party scripts from different origins.
- Verify application sets appropriate X-Frame-Options header for all application responses, such as DENY option.
- Verify application is not running on an outdated version of web server.
- Verify application does not utilise hardcoded credentials or passwords.
- Verify application does not utilise self-signed certificate.
- Verify application does not utilise predictable location for uploaded files.
- Verify application uses transport layer protection if transmitting sensitive information, such as authenticated requests.

- Verify application enforces password security policy and/or requirements.
- Verify __VIEWSTATE parameter is encrypted.
- Review the application encryption and design.

Appendix 2 Testing team

Pierre Pronchery	Pierre Pronchery is a Senior IT-Security Consultant and an accomplished developer. Freelancing for about a decade now, he could be found auditing major companies in the Telecommunications and Finance sectors, or supporting the Open Source Software and Hardware movements. He is a developer for the NetBSD Foundation since 2012, serving on its Board of Directors since 2017, and more recently, the founder of Defora Networks GmbH in Germany.
Stefan Vink	Stefan is an IT professional with a passion for IT security and automation. With 20 years hands-on experience in a diverse range of IT roles such as automation / scripting / monitoring / web development / system and network management in Windows and Linux environments. He worked for organisations such as the Central Bank of the Netherlands (DNB) and is MCITP, CCNA, LPIC, OSCP certified and passed the CISSP exam. He loves to travel, hike, tennis, chess, automation, and lives with his wife and daughter 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.