TECHNICAL REPORT

Fotoboks

Abstract

The results from a penetration test using OWASP Web Security Testing Guide

Magnus Baugerud, Henrik Mathias Berg, Lars Olsnes Østmo-Sæter

C 0

Document control

Version	Date	Editor	Comments
1.0	28/04/21	Magnus Baugerud Henrik Mathias Berg Lars Olsnes Østmo-Sæter	Finished version that will be sent to client.

Contents

Executive Summary

Introduction

The team have been given permission by NTNU IT to test the web application Fotoboks used for submitting photo id images for students' and employees' NTNU access cards. The team has not tested the production version of the system but has instead tested another instance hosted at innsidautv.ntnu.no/fotoboks so that none of the tests would affect the system's general workflow. Testing the production version would also require the team to have access to other Feide accounts than those they already have, otherwise each test posting data would request new student access cards, or would change the pin of the access cards.

The application was tested between the 25th of January and the end of the bachelor project on the 20th of May 2021. It was tested using some automated tests, but mostly the team has manually tested the system following methods described in the OWASP Web Security Testing Guide [1]. In order to cover as many tests as possible, all tests described in the OWASP guide have been considered and either been performed or given the not applicable status.

Overall security

Overall, the team believes the application has a good level of security. However, one critical issue was found, as well as some moderate and low risk issues. Many of these issues stem from the fact that the backend is too trusting of the frontend. Actions should be taken to fix these issues in order to prevent any potential future attacks. The team has detailed recommendations as to what actions should be taken under "Vulnerabilities and Recommendations".

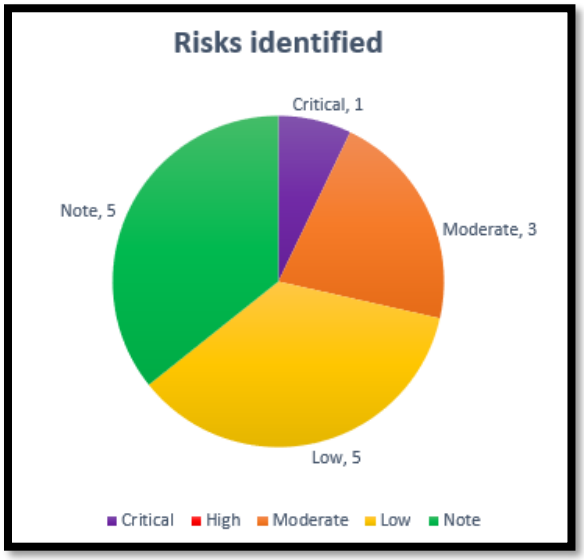


Figure 1: Pie chart presenting the distribution of risk severity.

High Severity Vulnerabilities

The critical-risk vulnerability in the application is related to the SMTP server. This server requires no authentication of the sender email, which means anyone can use it to send mail, and the sender address can be set to make it appear as if someone else sent it. The only requirements are that the sender is connected to the NTNU or Sit network, and that the receiver is a NTNU-email-address.

Moderate/Low Severity Vulnerabilities

Two of the moderate-risk vulnerabilities relate to lack of user input sanitization. The user may input files and text, and the backend will handle these as JPEGs, locations, and PIN-codes, with the only requirements being that the image format is JPEG, the pin is 10 character or less and the location is 50 characters or less. The frontend sanitizes these inputs, but the backends sanitation is lackluster. However, the input is only stored in a database by Fotoboks and used by other applications. The other applications have not been tested, so the exact impact of these vulnerabilities is not known.

The third moderate risk is the Apache server the application is using as a reverse proxy. This server has many known security vulnerabilities. Although the team has not been able to exploit any of these vulnerabilities, the server should be kept up to date to keep it as secure as possible.

Many of the low-risk vulnerabilities are misconfigurations of headers or cookies. Isolated, they pose little to no threat to the system. However, if some of these vulnerabilities are used together, they can pose a higher risk. Other low-risk vulnerabilities are outdated packages, lack of rate limiting and minimal logging.

Vulnerability Overview

Vulnerability Name	Severity
1. No Authentication on SMTP Server	Critical
2. Upload Malicious Files	Moderate
3. User-Input Not Sanitized	Moderate
4. CVEs for Apache HTTP Server Version	Moderate
5. Broken Logout	Low
6. Broken Cache-Control	Low
7. Cookie “session” Missing Secure Attribute	Low
8. Trusting Frontend to Limit POST	Low
9. No Automated Response to Repeated Errors	Low
10. CVE for Nodemailer	Note
11. Minimal Logging	Note
12. “SameSite” Cookie Attribute Missing	Note
13. Long Session Timeout	Note
14. Rpcbind on open port 111	Note

Table 1: Compact overview of findings.

System Overview

Description

Fotoboks is a web application where NTNU students and employees can upload an image and a pin for their access card. It is a react application connected to an Express server which is connected to a SQL database. The application was created in 2020 to accommodate infection prevention related to the COVID-19 pandemic. Fotoboks is hosted inhouse by NTNU and is available at innsida.ntnu.no/fotoboks/. The team was given access to and tested the test version of the application, which can be found at:

innsidautv.ntnu.no/fotoboks/

URL / Location

URL	innsidautv.ntnu.no/fotoboks/
IPv4	129.241.57.104/fotoboks
IPv6	[2001:700:300:31::104]/fotoboks

Table 2: The location of the application.

Environment

Users

Users of the system are students and employees of NTNU that have a Feide account. Users have access to the service from any web enabled device.

All users of the system have the same level of privileges. They can either update their pin or order a new student id card with a selected photo.

External firewall

Port	**Status**	**Protocol**	**Information**
80/TCP	open	HTTP	Apache httpd 2.4.29
111/TCP	open	Rpcbind	2-4 (RPC #100000)
443/TCP	open	SSL/HTTP	Apache httpd 2.4.29 ((Ubuntu))

Table 3: List of open ports and details from Nmap.

Reverse Proxy

The reverse proxy is an Apache server, and it hosts the portals for all of innsidautv.ntnu.no.

Goals

The primary goal of this penetration test is to validate that the system has implemented the necessary security measures to protect it from malicious actors. Any security vulnerabilities will be reported in this document and given recommendations on how to fix.

Threat Modelling

Attack Sources

The attack sources are first and foremost students and employees at NTNU, but the service is publicly available. To gain authenticated access to the system you are asked to sign in through Feide.

Motivation

Attackers would be motivated by denying students or employees access to NTNU buildings. To a lesser degree they might be motivated by the potential for malicious file upload and any exploits that such an attack might bring.

ATTACK TARGET

The attack target for this test is only the web service Fotoboks. There are other applications hosted on the same domain/IP, but these are out of scope.

RISK PROFILE

Information is sent by the user from the application to the database. This information can deny a user access to NTNU buildings if modified by an attacker. There is a chance that the information also could be harmful to other NTNU systems.

Tests Not Carried Out

DoS Attack

Denial of Service Attacks.

We have not attempted any tests attacking the service's capacity, since it was declared out of scope by NTNU IT.

Brute Force Login

Brute force login attempts on Feide.

We have not attempted any brute force attacks on the user login since it is a service provided by Feide.

Social Engineering

Attempt to gather information or get access through human channels.

Any social engineering attacks would be difficult to conduct as we are working on test instances and it was declared out of scope by NTNU IT.

Vulnerabilities and Recommendations

Definition of Risk Categories

Types

Likelihood Score

How likely it is that the vulnerability will be exploited. Both the ease of finding the vulnerability and the ease of exploitation will be taken into consideration.

Impact Score

How big of an impact the exploitation of the vulnerability will have. How much the attacker will gain and/or how much of the system will be affected will weigh in on the score.

Overall Risk Severity

How crucial it is that the vulnerability is fixed based on the likelihood and impact as seen in the diagram below.

	Impact		
Likelihood	Low	Moderate	High
Low	Note	Low	Moderate
Moderate	Low	Moderate	High
High	Moderate	High	Critical

Table 4: Chart used to calculate risk severity.

Levels

Likelihood Score

High	It is very likely that the vulnerability will be used by an attacker, due to it being easy to find and easy to use.
Moderate	It probable that this vulnerability will be used, but it is harder to find and/or exploit.

Low	Due to being very difficult to either find or exploit it is not likely that this vulnerability will be used by an attacker.

Table 5: Overview of likelihood scores and color codes.

Impact Score

High	May totally or partially compromise system.
Moderate	May compromise some users or smaller parts of the system.
Low	May result in compromising less important parts of the system, cannot compromise other users on its own.

Table 6: Overview of impact scores and color codes.

Overall Risk Severity

Critical	Must be fixed as soon as possible. It is very likely that this vulnerability will be exploited, and it will partially or totally compromise the system.
High	Should be fixed as soon as possible. This vulnerability will significantly compromise the system. It is harder to exploit, or it causes less damage than a critical issue.
Moderate	Should be addressed. The impact may be high, but then the likelihood will be low, and vice versa.
Low	Has the possibility to cause problems. Due to it having either a low likelihood or impact means it is not always a real issue.
Note	Should be looked at, but there is no immediate danger.

Table 7: Overview of risk severity scores and color codes.

Observations and Recommendations

1. No Authentication on SMTP Server

Description

The mail server requires no authentication of the sender email address. The only requirements are that the sender is connected to either the NTNU or Sit network, and that the receiver is an NTNU mail address. This means that an attacker can impersonate anyone using this server.

Risk Assessment

Likelihood Score: High
A quick google of "NTNU smtp" will tell us that this specific mail server is in use. Anyone testing to see if NTNU mail servers are vulnerable will quickly find out that there is no need to authenticate on NTNU networks.

Impact Score: High
Being able to send a mail to any NTNU-email with any sender address is a major vulnerability. There are many people with access to NTNU network and many who should not have access to NTNU network in Sit housing. Anyone of them can send a mail impersonating official NTNU channels of information.

An attacker can for example pose as an official NTNU channel to spread misinformation. During the COVID-19 pandemic it would be especially harmful to spread misinformation about the virus. An attacker could also try and steal sensitive information from people by claiming it is for contact tracing.

Another way an attacker could steal sensitive information is by posing as an official NTNU channel and claim the victims must update their password for their NTNU account. Updating your password is something NTNU requires regularly, so an attacker would probably be able to steal a lot of passwords this way.

Overall Risk Severity: Critical

Anyone with technical knowledge of SMTP, something that is taught to students at NTNU, can find and exploit this vulnerability. The vulnerability can be exploited in many ways and can most likely be used to steal sensitive information. Also, it does not seem like smtp.ansatt.ntnu.no should be available to use on the Sit network.

Recommendation

Implement authentication on the smtp server. Make sure the person sending an email is the person who owns the sender email. To limit accessibility to the SMTP server, block access from the Sit network. Limiting access to the SMTP server to the NTNU network will reduce the attack sources and reduce anonymity in the case of exploitation.

2. Upload Malicious Files

WSTG-BUSL-009

Description

The back-end check for uploaded files only looks at a short prefix in the base64 encoded files. This makes it possible to upload a malicious file crafted to pass as JPEG.

When the back end receives a base64 encoded JPEG it only decodes it to binary and saves it. This means that any metadata will be saved along with the image. There are 2 problems with this.

The first problem is that malicious code may be included in the metadata. The user is informed that someone will review the image, meaning it will be rendered at some point, and the image will also be printed on the student-card, meaning the image will be sent to a printer. The metadata may include malicious code crafted to exploit bugs in this specific renderer or printer, which can compromise system computers and/or printers.

The second problem is that the metadata may contain sensitive information about the user, like GPS-location and timestamp of the picture. The systems handling the image may not be aware of the sensitive information being stored at all. This type of information should not be leaked, and storing it adds unnecessary responsibility.

The front-end will ask the user to crop the image they upload and will at the same time convert accepted image-formats to JPEG and strip the metadata. This means that to upload a JPEG with metadata the user must bypass the front-end. This greatly reduces the likelihood of users sending their sensitive information as metadata unwillingly.

Risk Assessment

Likelihood Score: **Low**
exploiting this vulnerability requires knowledge of a vulnerability in either the specific renderer or specific printer in use, and the skills to exploit it. Since it is given information about neither it would be accomplished through repeated attempts with specifically crafted exploits.

Impact Score: **High**
Since the user-input file is saved in the database with potential malicious metadata or as an invalid JPEG containing malicious code the possible impact on systems further down the line is high.

Overall Risk Severity: **Moderate**

It would be difficult to exploit this vulnerability, since one does not know what systems are handling the uploaded data. Whether this vulnerability is exploitable is entirely dependent on the security of the other systems, which it should not be. The uploaded data should be properly sanitized in Fotoboks. However, if one is able to exploit this vulnerability it could cause a lot of harm to the systems in question.

Recommendation

The back end should strip metadata from the image and check that it is a valid JPEG by loading it as an image and see if the format is accepted as JPEG.

3. User-input Not Sanitized

WSTG-INPV-15, WSTG-BUSL-01

Description

There are three user inputs that are sent to the back-end server: pin, image, and location.

The pin is supposed to be a four-digit number, which the front-end enforces. The front-end also prevents you from have four of the same digits or the combinations “1234” and “2580”. However, the backend does not perform any checks on or sanitizing of the pin at all. It can be any ten-character-long string, because the database does not allow a larger string.

The location input has the same issue. The front-end gives you options from a list to choose from, but the back-end does not check if the location is valid, nor does it sanitize it. The location can be any fifty-character-long string with max length of 50.

Since none of the user inputs are used by the application, not sanitizing them does not pose a threat to the application. However, if the data is used by another application, it could pose a security risk to that application. The pin and location can hold JavaScript which may be ran when used by another application.

Risk Assessment

Likelihood Score: Low

Discovering this vulnerability is easy. Intercepting and modifying the post request can for example be done with the Burp Suite Proxy. However, it is not easy to exploit it as an attacker cannot get any information on the systems that use the uploaded data from Fotoboks. It is possible that the systems that use this data are properly secured so that an exploit is not possible at all, but Fotoboks should sanitize this data just to be sure.

Impact Score: High

The impact may vary from wasting your own and administrators time, to denying yourself access to NTNU’s buildings, to performing an XSS attack on an admin interface or any other system that might use the data Fotoboks uploads.

Overall Risk Severity: Moderate

Finding this vulnerability is easy but performing an XSS attack with it is not. It would require knowledge of the systems that use the uploaded data, or trial and error. Same as with the malicious file upload vulnerability, the possibility of this vulnerability being exploitable is dependent on the systems that use the data. A successful exploit could mean JavaScript being executed, which can cause a lot of harm.

Recommendation

The back end should check that the pin is a four-digit-number, and that the location is one of the locations users can pick from the list on the front-end.

4. CVEs for Apache HTTP server version

WSTG-INFO-02, WSTG-ERRH-01

Description

Fingerprinting the server returned that it was running an Apache HTTP Server 2.4.29 on Ubuntu. This version was released in 2017 and has at least 18 CVEs published [2].

There are multiple places the server and version are leaked, for example in the server header in every HTTP response and on the error-pages shown for error codes 405 and 414. See WSTG-ERRH-01 for more info about the error pages.

Although the team tested the test version of Fotoboks, which is hosted on innsidautv.ntnu.no, it was checked whether innsida.ntnu.no uses the same Apache server version. It turns out it does.

Risk Assessment

Likelihood Score: Low

A lot of the CVEs have a low likelihood score. In addition, we cannot test for these vulnerabilities because they may affect other systems besides Fotoboks. Therefore, we do not know if any of these CVEs are applicable or exploitable on Fotoboks.

Impact Score: High

There are CVEs for the outdated server with high an impact score, including one for code execution.

Overall Risk Severity: Moderate

As this server version is being used for most of NTNUs web services it is important to secure it. However, as stated previously, we have not been able to test any of the CVEs and therefore do not know if any of them can be exploited on Fotoboks or any other system on Innsida.

Recommendation

Rigorously testing if the apache server is protected against the known vulnerabilities is out of scope, but it is recommended to review these CVEs to make sure the server is secure. Keeping the server up to date is the most recommended action.

It is also recommended to obscure the server version, both from error pages and response headers. Knowing the exact server version gives attackers the option to review CVEs related to the server version, making it easier to find possible exploits.

5. Broken Logout

WSTG-SESS-06

Description

When you log out the server does not properly invalidate the session. It sends a new session cookie that indicates that you are logged out, but the old cookie may still be used to authenticate and POST a new image to the database as well as updating the pin. This means that all authenticated sessions will be valid until they expire, which is 24 hours for this application.

Risk Assessment

Likelihood Score: Low

To exploit this vulnerability the attacker would have to get hold of a session cookie from before logout, from a less than a day-old session, which belongs to another user. It would require the existence of other vulnerabilities and/or access to the victim's computer.

Impact Score: Moderate

The direct impact is that an attacker who gets hold of another user's session cookie before it times out will have full access to the application as the authenticated user, even when the user has logged out and believes their session is invalidated.

Overall Risk Severity: Low

Although the impact is moderate this vulnerability requires that the attacker manages to obtain the victims cookie. It is unlikely the attacker will manage to obtain a cookie unless exploiting other vulnerabilities. Either way the logout function should invalidate the session properly.

Recommendation

Make sure the server invalidates the session for Fotoboks in addition to the Feide session on logout.

6. Broken Cache-Control

WSTG-ATHN-06, WSTG-SESS-04, ZAP

Description

There is one place in the application that cache control is explicitly set by the server and that is the home page. However, it is set to “public, max-age=0” which means that it can be stored in any cache, but it is immediately stale as it expires after 0 seconds. The problem with this is that setting the max-age to 0 does not always stop caches from serving the resource and you can go back to the main page after logout. If the goal is to always revalidate with the server to check if you are logged in the header should be set to “public, max-age=0, must-revalidate”; this will stop caches from serving stale resources.

The largest problem with the application concerning cache-control is that responses carrying session cookies as headers can be cached in private caches. This is a result of not using the Cache-Control header at all which gives it the default value of “private”.

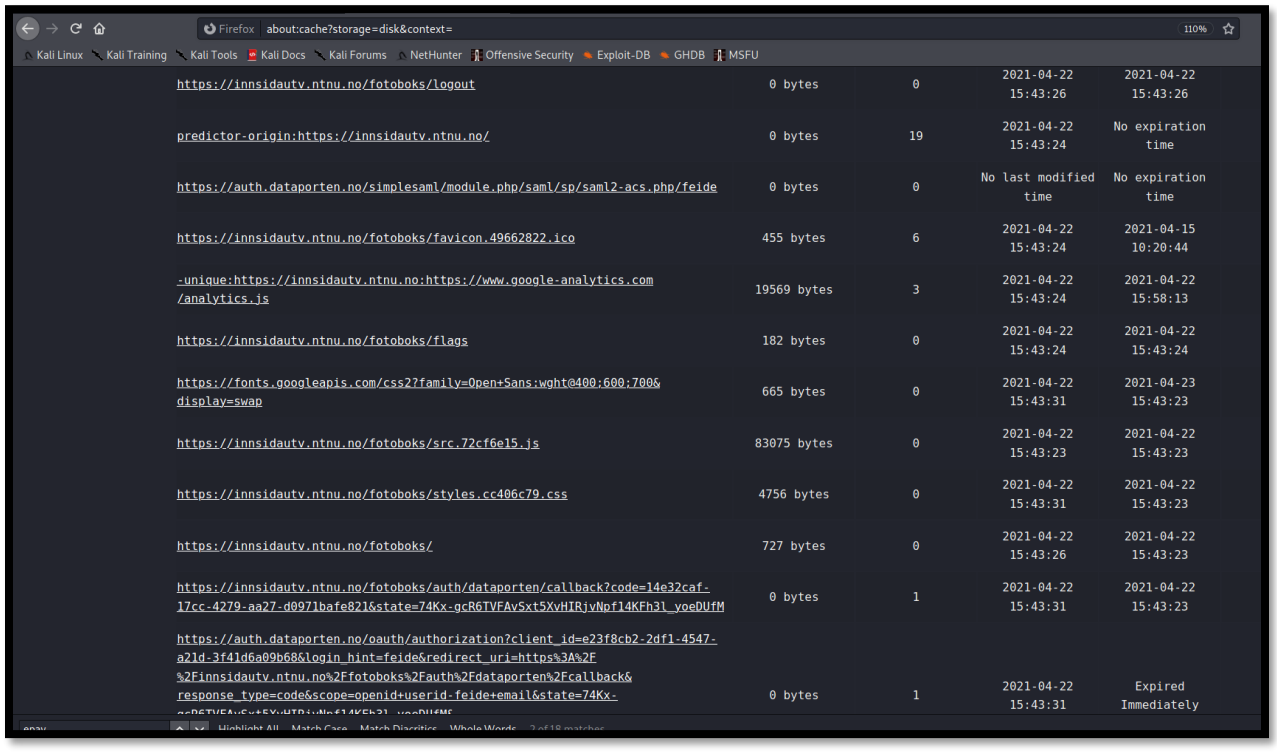


Figure 2 : The contents of the private cache for the Firefox browser

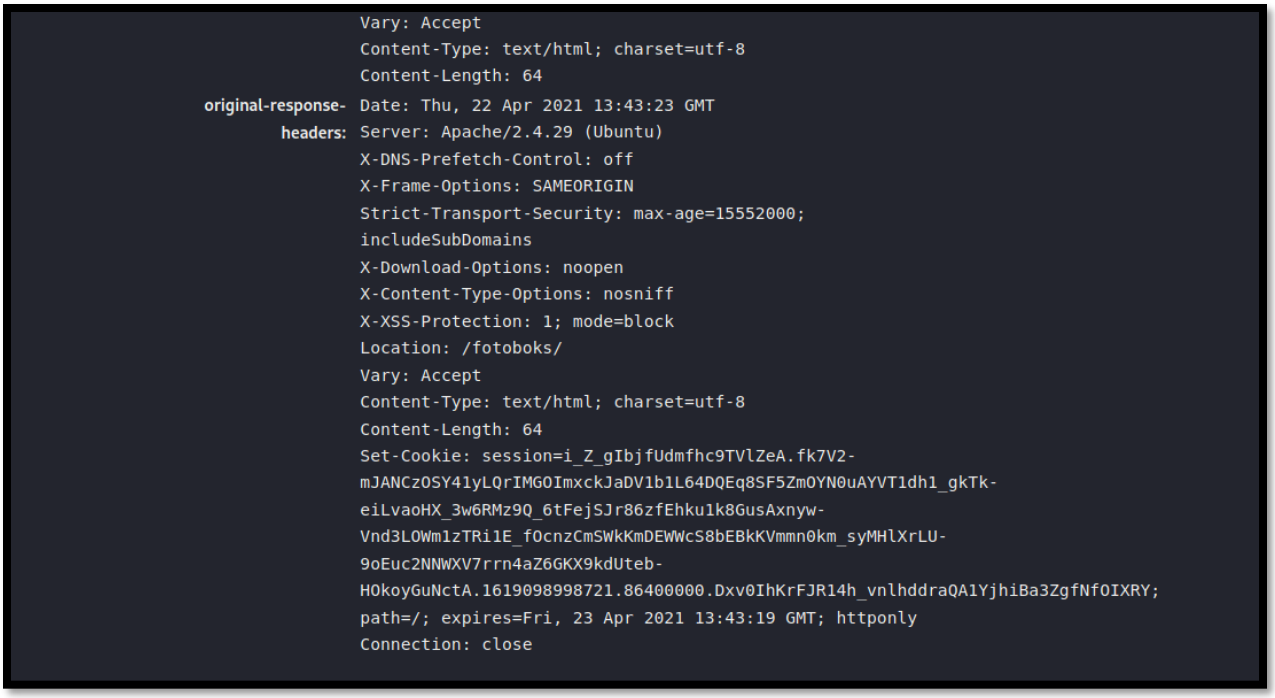


Figure 3 : Excerpt from the contents of a cache entry showing an authenticated session.

Figure 2 shows content from “about:cache” in the Firefox browser after logging in and out of Fotoboks. The cached resource from “/fotoboks/auth/dataporten/callback?” seems innocent with its 0 bytes body, but it contains the set-cookie header and shows a potentially authenticated session if you view it before the session has expired. As you can see on Figure 3, we can look at the session cookie and see when it expires.

This data should not be cached at all, the specific session cookie in the response is supposed to be accessed once and put in the browser.

Risk Assessment

Likelihood Score: Low

To exploit this issue, you need access to private caches between the user and the Fotoboks server. This would typically mean having access to the computer a potential victim used to access the application. Also, it would need to be done within 24 hours as the session cookie must be valid.

Impact Score: Moderate

If an attacker can manage to access a private cache, then they could get full authentication as another user. Allowing the attacker to do anything the application allows you to do authenticated and authorized as the victim.

Overall Risk Severity: Low

Although the impact is moderate this vulnerability requires that the attacker has access to the victim's computer and the risk severity is therefore set to low.

Recommendation

It is especially important that the responses setting the session cookie has cache-control set to "no-store, max-age=0" to prevent caching entirely. Setting "max-age=0" will force caches to revalidate existing entries and clear the cache for these responses. The main page should set cache control to "public, max-age=0, must-revalidate" if it is intended that a user should always be sent to login when accessing the frontpage unauthenticated. [3]

7. Cookie "session" Missing Secure Attribute

WSTG-SESS-02, WSTG-CRYP-03, WSTG-ATHN-01

Description

The session cookie gives authorized access to the application but does not have the "Secure"-attribute set, meaning the cookie does not require an encrypted connection to be sent. When accessing "http://innsidautv.ntnu.no/fotoboks" the cookie will be sent over the unencrypted HTTP connection before redirecting to HTTPS. This vulnerability only works if the browser does not know that the application uses HSTS. After the browser encounters a HSTS header it will only use HTTPS when communicating with the web service. For the cookie to be sent over HTTP the browser needs to forget the HSTS header, for example through deleting the browser data while keeping the cookies.

Risk Assessment

Likelihood Score: Low

The likelihood of this vulnerability being exploited is low. After connecting to innsidautv.ntnu.no once, the browser will remember the HSTS header, and that the connection is supposed to go over HTTPS. In order for this vulnerability to be exploited the victim must delete its browser data, but not their cookie data, after they have logged in to innsida.ntnu.no or innsidautv.ntnu.no.

Impact Score: Moderate

Stealing the session cookie gives full access to the victims Fotoboks. Worst case for the victim is that it has to cancel a card ordered by the attacker, or has the pin changed to an unknown value.

Overall Risk Severity: Low

The likelihood of this being exploited is very low but it does make it possible to steal sessions and should therefore be addressed.

Recommendation

Add the “Secure”-attribute to the session cookie. This attribute will require the connection to be encrypted for the cookie to be sent.

8. Trusting Frontend to Limit POST

WSTG-BUSL-05

Description

The front-end has a limit of 1 uploaded image per session, but this limit is easy to bypass. This limit is enforced by updating a variable in the cookie “session”, after this the frontend will notify you that you have already uploaded and prompt you to log out. If you circumvent the frontend, you will see that you can repeatedly upload images with no limit as the server still accepts the “session” cookie. It works to post with both the old and the updated session cookie.

Risk Assessment

Likelihood Score: Moderate

Discovering this vulnerability is easy. Intercepting and modifying the post request can for example be done with the Burp Suite Proxy. The likelihood is lowered by the fact that the exploitation of this vulnerability is easily tracked to the exploiter’s account.

Impact Score: Low

By not limiting the rate of image uploads the service is more vulnerable to DoS-attacks.

Overall Risk Severity: Low

It is easy to discover this vulnerability, but it is traceable to an account and would only affect server-load.

Recommendation

The user should be logged out of Fotoboks on a successful post, which should invalidate the session cookie. The user does not have to be logged out from Feide however but can be presented with a prompt to do so.

9. No Automated Response to Repeated Errors

WSTG-BUSL-07

Description

It appears to be no active defenses against misuse of the application. Although there is some logging on all requests, there is no rate limiting on errors. Also, as this application is only meant to be used once by each individual user to send the data needed to make one access-card, it should be easy to detect and block fuzzing attempts.

Risk Assessment

Likelihood Score: Moderate

It is easily discovered that the attempted rate limiting does not actually work as you can confirm it by sending the same POST request twice.

Impact Score: Low

No automated responses make probing and fuzzing for weaknesses much faster/easier. This vulnerability mostly affects the information gathering and preparation phase. It might also be exploited to launch a DoS attack.

Overall Risk Severity: Low

This is not a difficult vulnerability to find, but it does not have any direct system impact, apart from possibly being used in a DoS attack. The OWASP Web Security Testing Guide [1] list the lack of automated responses as a noteworthy issue.

Recommendation

Since this application only requires one POST request per user, it would be recommended to implement some form of rate limiting on general use.

10. CVE for Nodemailer

WSTG-CONF-01

Description

The version of Nodemailer was found through reviewing the source code. Searching for the Nodemailer version shows there is one CVE: CVE-2020-7769 [4]. The issue described is: “Use of crafted recipient email addresses may result in arbitrary command flag injection in sendmail transport for sending mails.”

The application does not allow users to decide recipient email, the recipient is decided by data acquired from Feide. Because of this it is not likely that the application is vulnerable to this issue, but it is still of note if the situation changes, and users can for any reason set recipient for the sent email.

Risk Assessment

Likelihood Score: Low
There is no way to exploit this vulnerability since the users can not set the recipient for emails.

Impact Score: Low

The impact is low as there is no way to exploit this vulnerability.

Overall Risk Severity: Note

This vulnerability may not be exploited as the application does not use the exploitable features in Nodemailer, but it is noteworthy since the application may use these features in the future.

Recommendation

There is no immediate action required for this vulnerability, but it should be kept in mind if the app is ever expanded with new functionality. Updating Nodemailer to version 6.4.16 will eliminate this vulnerability completely.

11. Minimal Logging

WSTG-CONF-02

Description

By reviewing the source code, the team could see exactly what was logged for each request sent to the server. As it is now there is not a lot that gets logged: standard info about the remote user and the request, along with the status code. If there are any errors, it will be possible to detect it by looking at the response code. However, it is not possible to see what caused the error. As the data sent to the database is supposed to be sanitized user data, the application should notify or at least log when there is an error with the SQL.

Risk Assessment

Likelihood Score: Low

The likelihood of this vulnerability being exploited by an attacker is low. This vulnerability would not be directly exploited by an attacker, but it could hamper the developer's ability to detect abnormalities. Logging is a tool used by the developers of the system to monitor it, and a lack of logging makes monitoring difficult.

Impact Score: Low

Minimal logging will impact the developers when trying to determent what caused errors.

Overall risk severity: Note

There is no direct impact on the system and the likelihood of finding a way to exploit this is very low.

Recommendation

Some user input should be included in the log when errors are encountered.

12. SameSite Cookie Attribute Missing

ZAP

Description

The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks. Fotoboks uses the custom header “x-token” as an anti-CSRF token, but it is still good practice that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.

Risk Assessment

Likelihood score: **Low**

The service is using x-token as an anti-CSRF token; therefore, the likelihood of this vulnerability being exploited is low to none.

Impact score: **Low**

This vulnerability may be exploited in a case where x-token is missing. However, in the current version of the application, the x-token header is always properly set.

Overall risk severity: **Note**

Setting the SameSite attribute is to take extra precautions in case of further development of the system, and to follow good practice.

Recommendation

The SameSite attribute should be set to either 'lax' or ideally 'strict' for all cookies.

13. Long Session Timeout

WSTG-SESS-07

Description

The application uses the same session cookie to authenticate requests for 24 hours. This is much longer than what it needs to be, which can lead to sessions being hijacked long after a user used the session.

Risk Assessment

Likelihood score: **Low**

It is not likely that this vulnerability will be exploited. It only makes the timeframe in which the session can be hijacked larger. For example, if an attacker has access to a computer where a victim is logged in, the attacker could use that session up to 24 hours after the victim logged in.

Impact score: **Low**

This vulnerability does not directly make the act of hijacking a session easier. As mentioned in the likelihood score, it only increases the timeframe in which the session can be stolen.

Overall Risk Severity: **Note**

It is not likely that this vulnerability will be exploited, and the impact would be low. Even so, the long expire time puts the cookie at unnecessary risk.

Recommendation

Reduce the time for the session cookie to expire. OWASP recommends it being set to somewhere between 15 and 30 minutes. [5]

14. Rpcbind on open port 111

Nmap

Description

The Nmap scan revealed an rpcbind service running on port 111 and is open to the public.

Risk Assessment

Likelihood Score: Low

There are no vulnerabilities known to the team for this port.

Impact Score: Low

The open port increases the attack surface.

Overall Risk Severity: Note

The attack surface is increased, but there are no vulnerabilities known to the team. In the future, vulnerabilities might be discovered that exploits the fact that port 111 is open.

Recommendation

Evaluate if the rpcbind port 111 is required to be open to the public. If not, it should be closed.

Finding Summary

Overview

Vulnerability Name	Likelihood	Impact	Severity
1. No Authentication on SMTP Server	High	High	Critical
2. Upload Malicious Files	Low	High	Moderate
3. User-input Not Sanitized	Low	High	Moderate
4. CVEs for Apache HTTP server version	Low	High	Moderate
5. Broken Logout	Low	Moderate	Low
6. Broken Cache-Control	Low	Moderate	Low
7. Cookie “session” Missing Secure Attribute	Low	Moderate	Low
8. Trusting Frontend to Limit POST	Low	Moderate	Low
9. No Automated Response to Repeated Errors	Moderate	Low	Low
10. CVE for Nodemailer	Low	Low	Note
11. Minimal Logging	Low	Low	Note
12. “SameSite” Cookie Attribute Missing	Low	Low	Note
13. Long Session Timeout	Low	Low	Note
14. Rpcbind on open port 111	Low	Low	Note

Table 8: Overview of findings including likelihood score, impact score and risk severity.

Combining Vulnerabilities

The broken logout means an authenticated session cookie is never invalidated before it expires, this combined with a long session timeout of 24 hours leaves the cookie vulnerable. On top of that the cookie is saved in the cache because of broken cache-control. An attacker with access to the machine of a victim has a 24-hour window to extract the cookie from the cache, even when the user logged out, and may hijack the session. The fact that these vulnerabilities can be combined were not taken into consideration when giving them individual likelihood and impact scores, but they are more likely to be exploited when used together.

Web Application Penetration Test Report

Test IDs reference to the OWASP Web Security Testing Guide and results

The following tests are executed as described in the OWASP Web Security Testing Guide [1]. The tests' reference codes will match a code in the guide. The tests also have a title, a description of the test performed with results, and a status. The title of the tests also matches a title in the guide. The status is explained in the table below.

Table 9

Status	Description
Pass	Test results revealed no issues
Issue	Test results revealed issues
N/A	Test not applicable for this application/service

Table 10: Overview of test status and color codes.

Information Gathering

WSTG-INFO-01

Conduct Search Engine Discovery and Reconnaissance for Information Leakage

Pass

All public info that could be found about Fotoboks is that it is a digital photo box that you use to set the photo for you student id or your employee id card. This information was found on public pages meant for new students and employees; it did not reveal any sensitive info about the application itself.

WSTG-INFO-02

Fingerprint Web Server Issues

Issue

Checked the response headers with Burp Suite Proxy and found that the service runs on an Apache HTTP Server 2.4.29. This version was released October 23, 2017. This server should be updated to version 2.4.46, released August 7, 2020.

Fingerprint through nmap gave:

80/tcp	open	http	Apache httpd 2.4.29
443/tcp	open	ssl/http	Apache httpd 2.4.29 ((Ubuntu))

Table 11: Nmap results, see Nmap.

More details on this vulnerability and the risk severity can be found under CVEs for Apache HTTP server version.

WSTG-INFO-03

Review Webserver Metafiles for Information Leakage

Pass

Found robots.txt, but the contents were empty.

```
User-Agent: *
Disallow:
Sitemap: http://innsida.ntnu.no/sitemap.xml
```

Figure 4: The contents of robots.txt

WSTG-INFO-04

Enumerate Applications on Webserver

N/A

We are only to test https://innsidautv.ntnu.no/Fotoboks/. Meaning the application itself.

WSTG-INFO-05

Review Webpage Comments and Metadata for Information Leakage

Pass

Read through source code, there are no revealing comments.

WSTG-INFO-06

Identify application entry points

Pass

page request	Where to find	request type	interesting parameters	interesting headers	authenticated/unauthenticated	TLS	multi-step process	websockets	other notes
innsidautv.ntnu.no/fotboks/flags		GET	Cookie: session		Cookie: session	yes		no	Only to keep connection open?
innsidautv.ntnu.no/fotboks/	Save button in "Confirm image and choose PIN" page	POST	Cookie: session	x-token, Content-Type: application/json	Cookie: session	yes		no	sends json object containing file, pin and location. Is x-token a custom header?
innsidautv.ntnu.no/fotboks/logout	logout button	GET	Cookie:JSESSIONID=…; session=…	Upgrade-Insecure-Requests	Cookie: session	yes	beginning of loggout	no	initiates logout
innsidautv.ntnu.no/fotboks/pin	save button under "just change PIN" option	POST	Cookie: session	x-token, Content-Type:application/json	Cookie: session	yes		no	sends json object containing pin. X-token again
innsidautv.ntnu.no/fotboks/	Save button in "Confirm image and choose PIN" page after taking foto with web cam	POST	Cookie: session	x-token, Content-Type: application/json	Cookie: session	yes		no	sends json object containing file, pin and location. Is x-token a custom header?

innsida.ntnu.no/studentkort	"For students" link after submitting image and pin	GET	Cookie: JSESSIONID=…; COOKIE_SUPPORT=true; GUEST_LANGUAGE_ID=nb_NO; ga=...; _gid=...; nmstat=...; LFR_SESSION_STATE_10135=… referer: https://innsidautv.ntnu.no/			yes	beginning of redirect. Leaves innsidautv.ntnu.no/fotoboks/	no	redirects to innsida info page on student cards
innsida.ntnu.no/adgangskort	"For employees" link after submitting image and pin	GET	Cookie: JSESSIONID=…; COOKIE_SUPPORT=true; GUEST_LANGUAGE_ID=nb_NO; ga=...; _gid=...; nmstat=...; _gat=1; LFR_SESSION_STATE_10135=… referer: https://innsidautv.ntnu.no/			yes	beginning of redirect. Leaves innsidautv.ntnu.no/fotoboks/	no	redirects to innsida info page on employee cards
innsida.ntnu.no/wiki/-/wiki/English/Photo+requirements+for+access+cards	"For employees" link after submitting image and pin	GET	Cookie: JSESSIONID=…; COOKIE_SUPPORT=true; GUEST_LANGUAGE_ID=nb_NO; ga=...; _gid=...; nmstat=...; LFR_SESSION_STATE_10135=… referer: https://innsidautv.ntnu.no/			yes	beginning of redirect. Leaves innsidautv.ntnu.no/fotoboks/	no	redirects to innsida info page on accesss card guidelines

Table 12: Overview of all identified non-admin functionality entry points in the application.

WSTG-INFO-07

Map execution paths through application

Pass

These are the endpoints used by the application during normal usage:

```
/fotoboks (GET, POST)
/fotoboks/logout
/fotoboks/auth/dataporten
/fotoboks/auth/dataporten/callback?[code]&[state]
/fotoboks/flags
/fotoboks/pin (POST)
/fotoboks/styles.cc406c79.css
/fotoboks/src.72cf6e15.js
/fotoboks/favicon.49662822.ico
```

Figure 5: The endpoint in the application

WSTG-INFO-08

Fingerprint Web Application Framework

Pass

Identified node.js by generating 404 error, see WSTG-ERRH-01.

The session cookie is generic.

Wappalyzer fingerprinted OS to be Ubuntu.

WSTG-INFO-09

Fingerprint Web Application

N/A

Merged with WSTG-INFO-08.

WSTG-INFO-10

Map Application Architecture

Pass

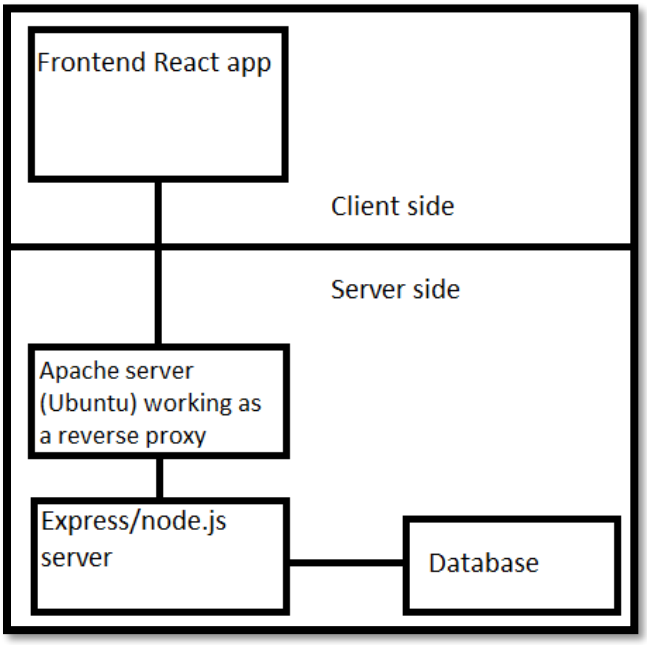


Figure 6: An overview of the different components of the system.

Configuration and Deploy Management Testing

WSTG-CONF-01

Test Network/Infrastructure Configuration

Issue

There are known exploits for the version of Nodemailer used in this program. The usage of Nodemailer was found by reviewing the source code.

More details on this vulnerability and the risk severity can be found under Trusting Frontend to Limit POST.

WSTG-CONF-02

Test Application Platform Configuration

Issue

Read source code. Back-end does not log input that caused errors, and it does not react to repeated errors. The lack of rate limiting is experienced while performing automated scans.

More details on this vulnerability and the risk severity can be found under Minimal Logging.

WSTG-CONF-03

Test File Extensions Handling for Sensitive Information

Pass

A Nikto scan was performed. It revealed no files or sensitive information.

A DirBuster scan was also performed but it did not find anything either.

WSTG-CONF-04

Backup and Unreferenced Files for Sensitive Information

Pass

Found no backups or unreferenced files in source code.

WSTG-CONF-05

Enumerate Infrastructure and Application Admin Interfaces

Pass

Ran a Nikto and a DirBuster scan and found no admin-pages nor endpoints.

WSTG-CONF-06

Test HTTP Methods

Pass

The application consistently allows OPTIONS and HEAD. It will allow GET and/or POST if there is functionality that requires it, if not it will return the standard 404 from the node.js server. If it is a recognized HTTP method the server will reply with a 404 message if it is not one of the methods specified in OPTIONS.

The exceptions are TRACE and CONNECT. TRACE returns the error code 405 “method not allowed” from the reverse proxy and CONNECT returns a 400 bad request. If the server receives a request without an HTTP method or with a non-recognized HTTP method, it will return the error code 400 “bad request”.

WSTG-CONF-07

Test HTTP Strict Transport Security

Pass

All responses from the application were checked with Burp proxy. The HSTS header is set correctly.

WSTG-CONF-08

Test RIA cross domain policy

Pass

No policy files were found. “crossdomain.xml” and “clientaccesspolicy.xml” was searched for. An automated scan by DirBuster did not reveal any policy files either.

Identity Management Testing

N/A

Identity management is handled by Feide. We have not performed any tests on the Feide system.

Authentication Testing

WSTG-ATHN-01

Testing for Credentials Transported over an Encrypted Channel

Issues

The application was checked for the HSTS in WSTG-CONF-07. It is set properly. However, WSTG-SESS-02 revealed that the “secure” attribute is not set on the “session” cookie. This means it exists an edge case where the session cookie can be sent over an unencrypted connection. This is detailed under Cookie “session” missing secure attribute.

WSTG-ATHN-02

Testing for default credentials

N/A

This part of the application is handled by Feide.

WSTG-ATHN-03

Testing for Weak lock out mechanism

N/A

Login is handled by Feide.

WSTG-ATHN-04

Testing for bypassing authentication schema

Pass

There is no way to bypass the authentication schema. This was tested by intercepting request with Burp Suite proxy and tampering with the session cookie and the “x-token” header. Tampering with the “x-token” header will only display the following message: “Sorry! Saving failed (server error)”.
Tampering with the session cookie will only display this message: “You are no longer logged in” when sending a POST request.
Sending a GET request to innsidautv.ntnu.no/fotoboks/flags with an invalid session cookie will only return an empty object like this: “{}”.

WSTG-ATHN-05

Test remember password functionality

N/A

Fotoboks does not handle login credentials.

WSTG-ATHN-06

Testing for Browser cache weakness

Issues

The cache-control header was checked using Burp Suite proxy.

The main page sets Cache-control header to “public, max-age=0” meaning that it will quite often be cached, setting the max-age to 0 does not stop the browser from serving cached sites as shown in the image below.

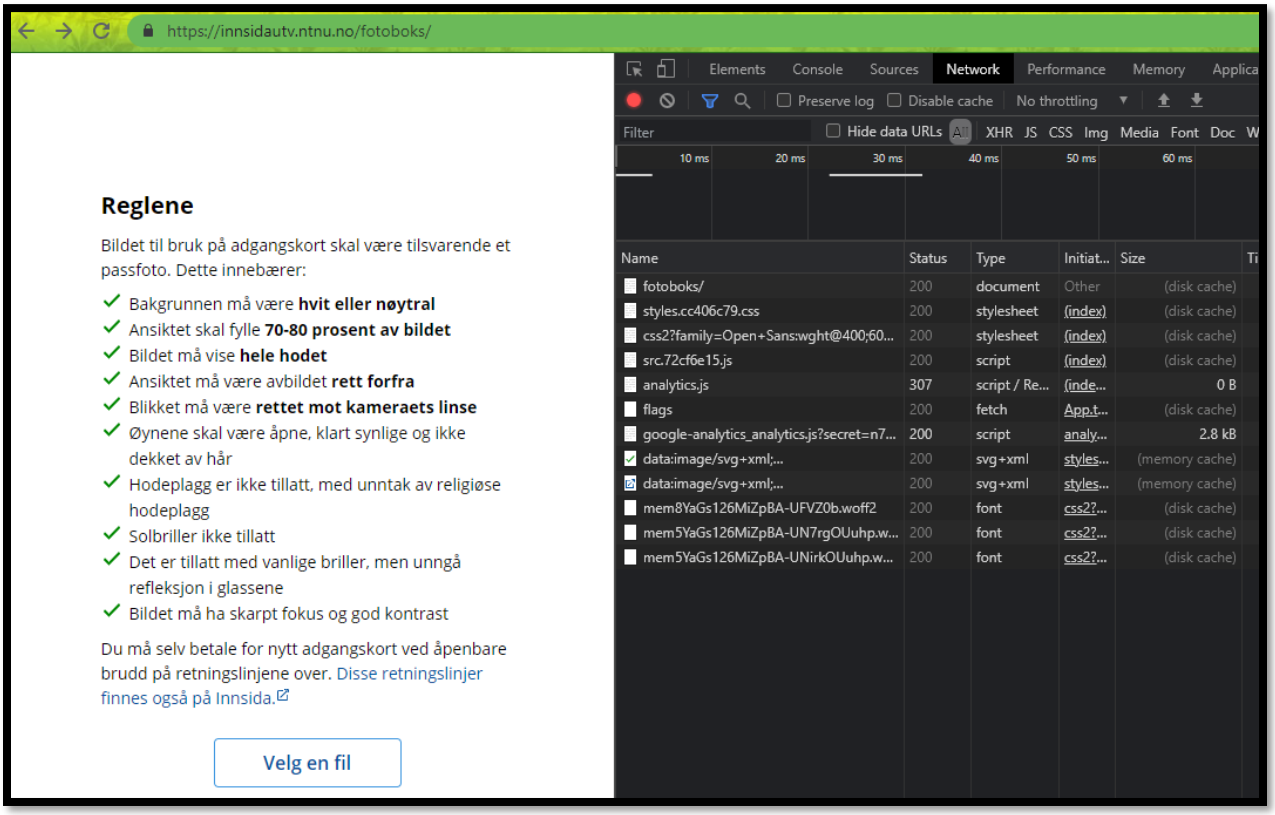


Figure 7: A screenshot of the Chrome browser serving the Fotoboks main page from disk cache.

On login and logout, where the session cookie is set there is no cache control header, the server does not specify any policy on web-caching. It will then default to setting the cache control to private, which means that your own browser will always store the session cookie in the browser's web-cache when you get it. This is usually not a big issue but the case where you access Fotoboks on a system other people have access to. As a best practice any response sending session cookies should not be cached at all, as they are supposed to identify a specific session and should be different for each login.

More details on this vulnerability and the risk severity can be found under Broken Cache-Control.

WSTG-ATHN-07

Testing for Weak password policy

N/A

Fotoboks does not handle passwords.

WSTG-ATHN-08

Testing for Weak security question/answer

N/A

This type of functionality is not used, and if it were used it would have been handled by Feide.

WSTG-ATHN-09

Testing for weak password change or reset functionalities

N/A

Fotoboks does not handle passwords.

WSTG-ATHN-10

Testing for Weaker authentication in alternative channel

N/A

Authentication handled by Feide.

Authorization Testing

WSTG-ATHZ-01

Testing Directory traversal/file include

Pass

The application uses express and has no endpoints with parameters, therefore, there is no access to the directory.

WSTG-ATHZ-02

Testing for bypassing authorization schema

Pass

Resources cannot be access without authentication, which was tested in WSTG-ATHN-04. Given that there is only one level of authorization, this cannot be bypassed.

WSTG-ATHZ-03

Testing for Privilege Escalation

Pass

The possibility of privilege escalation in this application is dependent on either an SQL injection to do unauthorized operations in the database, or by changing the ID used to identify each user in the database. This ID is stored in the “session” cookie, attempts at forging this was unsuccessful.

WSTG-ATHZ-04

Testing for Insecure Direct Object References

Pass

No user input was detected that could be used as a direct object reference.

Session Management Testing

WSTG-SESS-01

Testing for Bypassing Session Management Schema

Pass

Session cookies were collected from different members of the team with the Cookie/Token list Script. The cookies seem to be completely random except for dots and the repeated “86400000”. Following is a small sample of session cookies.

G1jPZDRrtYUp50wKyclMuw. FD4KORIbSyCJYx3Ps1R1fkPu3vXuqacsFtaaC-8l0vLBNs0vT6jXNGCKVyfqf3PPPZFwMCXdBU3lYiCdcnRckCn6LbblSZY3lMhBikxa-attn4f2ejAMp3Mq6l3zLjPqWcuDMfDLIaYVva7dn0RiFouRedA7bXIRh2f6Zim5JxcOzJiCrcAO5mOQXgph2yEHEEo0iUsefwDcuF2t429QDyAu715gSAs8SLlbY7lu1y8. 1612950446586. 86400000. 8PczOjmd1khXF4v9N6QNNoIP_2TMGCUE7WQkxiYxBA0
c3u3BrR85kFO7zmbIc3QrA. ZDmwNrGuaZ9ClONvzpLVd1rgnqMex9FwDM3laCbbkKaTGpg2Hlyz2gxXhXjvJ2BLhQrs91R7X0YsVrcCXF14y6cM5945jTWp6I08Wc0-jsTgv0hKHx5x8bPF4ok0bCfmJeTwj7ktsn4LYirRO7w2b18pQ-webmTKxa7K3bh2UnFh6ZY3iSOMKl8bnRlk2D9V0oaDjLZz9MadDZMhbFm3NNLhRlDsOjc_4YR19s3Y248. 1612950450546. 86400000. jBTn94-HajF3EEP6y28rtG1hizdTnE4977UVaeI_XKc
tq9HrWxIGk6xOZYRC_yKsQ. _hYtYy4Xaqu9fIn4tFLnUxc9utDG3s2oQh75UpggR1WpMmyKT4DT7CbWvMeTlyThG37nTUiPXrnuk-rziQwqc1nfYDzSH8E5XPyVrgciQhfNYOUBsFqyM3n3WR8Fy6L7FICRVkOoEuYs_vFdOHVoiTdLXHu0vUpIJ9wKX-9sX-qCJHniNA-Aac_zdvvbTlts_iyrQ5QikkeNNGvvqtkgQHeTGKNsm9qlcdajtesYHo0. 1612950452466. 86400000. ASLuDVBj--i7Nxu33MUnQKRN6bBv8KNmiNuUVmtJuw4

Table 13: A collection of session cookies.

There is no use looking at the x-token since it always requires valid session cookie, and with a valid session cookie you may ask for a x-token by sending a request to the */fotoboks/flags* endpoint.

SESSION cookie variation

Different characters count on each index of the SESSION cookie over multiple cookies

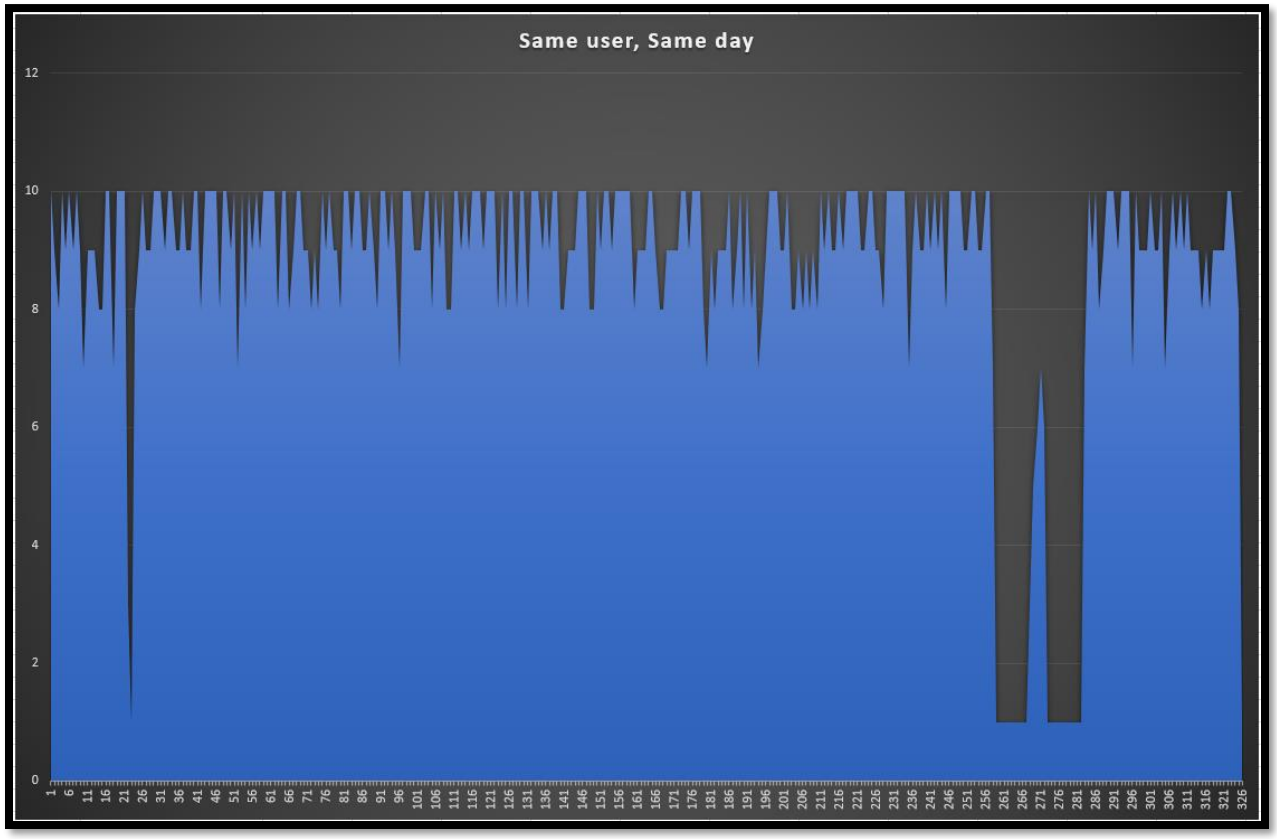


Figure 8: 10 sessions from the same user from the same day.

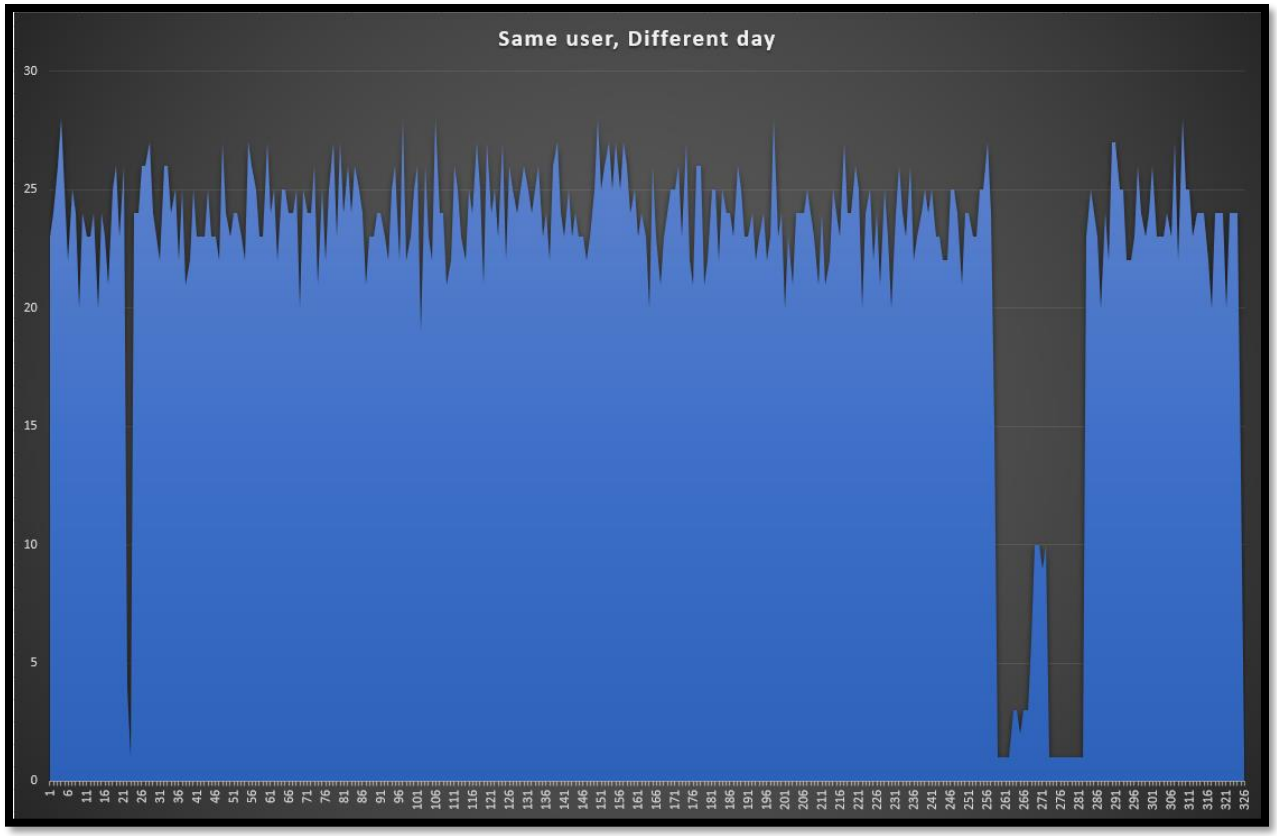


Figure 9: 40 sessions from the same user from different days.

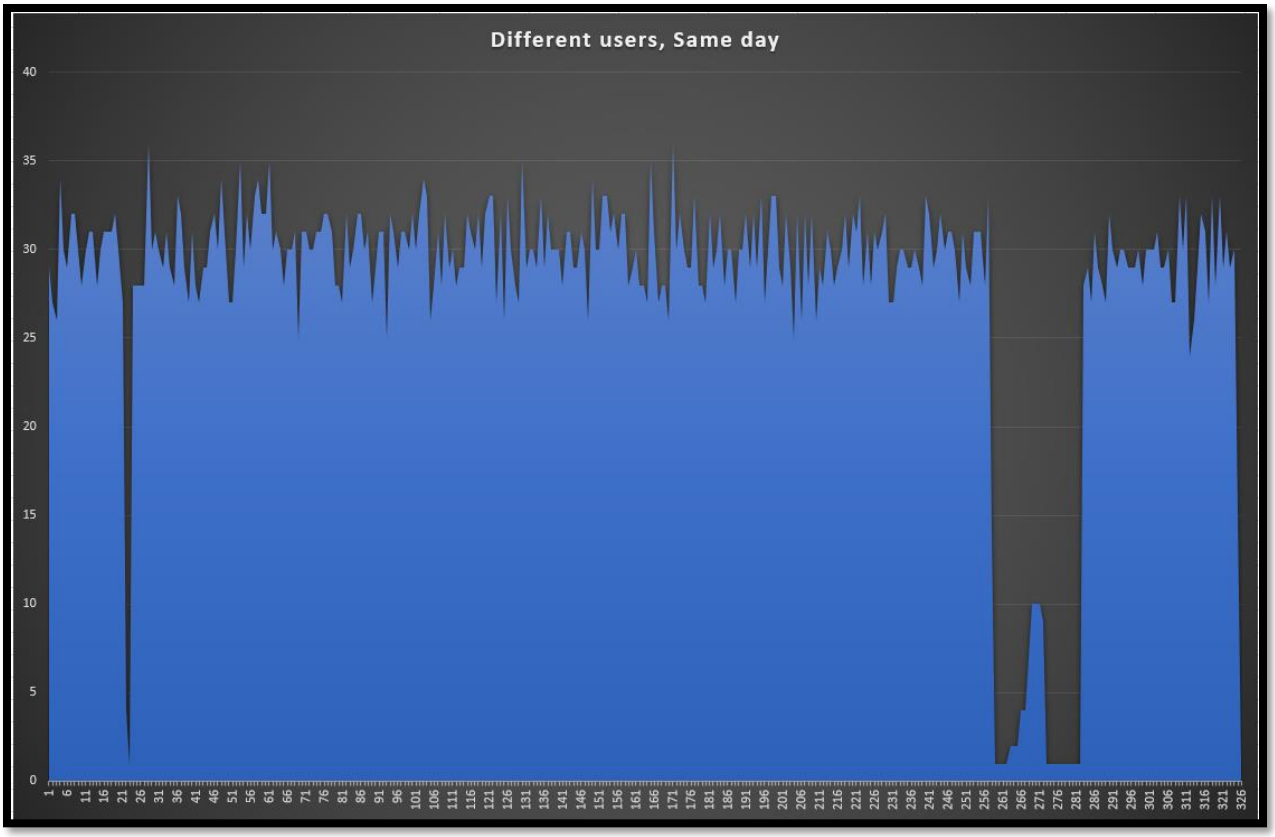


Figure 10: 40 sessions from 2 different users from the same day.

By comparing Figure 8 and Figure 9 we see that there is no significant difference in variation based on time, meaning it is no way to predict parts of the cookie based on time. By comparing Figure 8 and Figure 10 we see that there is no significant difference in variation based on the user, meaning there is no way to predict parts of the cookie based on user-id or something similar.

WSTG-SESS-02

Testing for Cookies attributes

Issues

Used Burp Suite Proxy to look at the set-cookies attributes. The "session"-cookie does not use the "Secure"-attribute. This cookie gives authorized access to Fotoboks. The Strict-Transport-Security header is set, so there are only very special cases where the cookie may be exposed. These special cases are described under the Cookie “session” missing secure attribute section.

ZAP found that the cookie “session” is also missing the SameSite attribute. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.

More details on this vulnerability and the risk severity can be found under SameSite Cookie Attribute Missing.

WSTG-SESS-03

Testing for Session Fixation

Pass

The application does not validate existing sessions, but instead creates a valid new session if your SSO session (Feide session) is valid.

WSTG-SESS-04

Testing for Exposed Session Variables

Issues

The application does not make sure that sessions are not cached. To make sure the session cookie is not cached under any circumstances the Cache-Control header should be set to “no-cache, max-age=0” on any responses carrying the Set-Cookie header. This will ensure that it will not be cached.

Also as mentioned in WSTG-SESS-02 the secure attribute is missing on the session cookie.

More details on these vulnerabilities and the risk severity can be found under Broken Cache-Control and Cookie “session” Missing Secure Attribute.

WSTG-SESS-05

Testing for Cross Site Request Forgery

Pass

The application has a custom header called x-token. This header protects the application against CSRF as JavaScript cannot set custom headers if it is cross origin. This header was viewed by intercepting a request to the application with Burp Suite Proxy.

Example of what the x-token looks like:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI2OGMwZjI1Yi03YjBmLTRkNmUtODMwMy02NjQ3YmVhZmY2YzEiLCJleHAiOjE2MTgzODg0OTcwNzN9.s2Wm-KOauLgpypxhJIsMqBrwJN4kG3VdBIL_5VoQ2y4
```

Figure 11: An x-token.

WSTG-SESS-06

Testing for logout functionality

Issues

The logout button is always visible.

The session cookie is not invalidated on the server side on logout. This was tested with Burp Suite Repeater. The server trusts that the client discards the old session cookie on logout and supplies a new one that is not authenticated. Using the old session cookie after having been logged out will allow you to post new data. The x-token times out after 15min, and the session-cookie times out after 24 hours and can generate new tokens while it is valid. An authenticated session will be valid until it times out, logging out does nothing to hinder this.

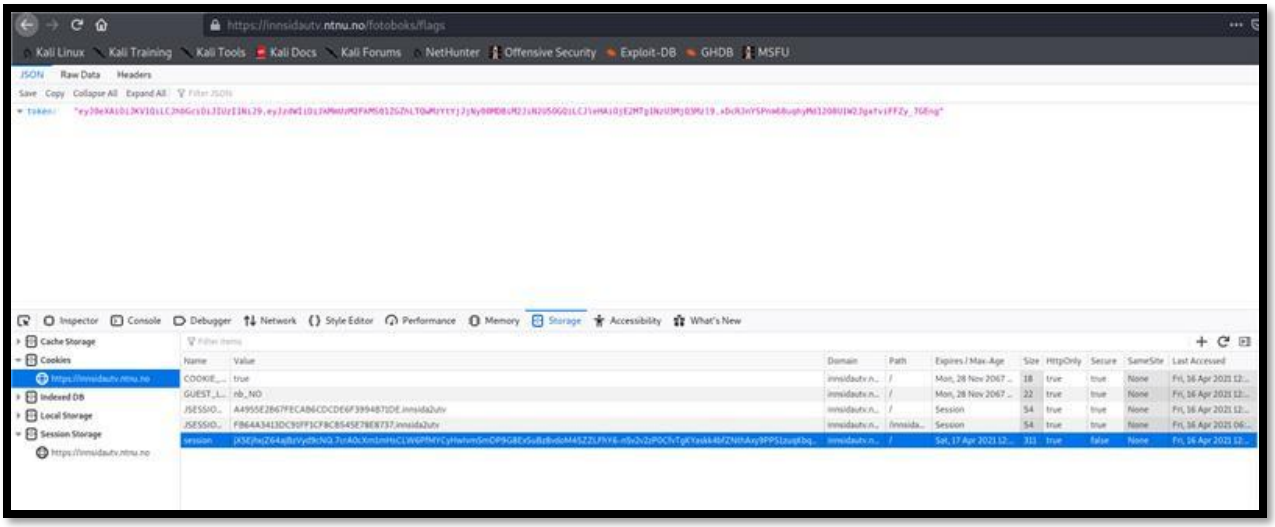


Figure 12: Entering the old cookie after logout allows you to extract new tokens.

If we replace the session cookie with the cookie we had before logging out, we can still get the /flags endpoint and receive a new valid token.

After reading source code we can see that one session is invalidated and that is the Feide session. This was confirmed by checking innsida.ntnu.no. If you were logged in to Fotoboks and try to access Innsida you will be authenticated through Feide. If you try the same after having logged out of Fotoboks you will not be authenticated and prompted to enter you Feide credentials again. If replicating this make sure to be logged out of Innsida when testing both cases as you can be logged in to Innsida without having an active SSO session with Feide.

More details on this vulnerability and the risk severity can be found under Broken Logout.

WSTG-SESS-07

Test Session Timeout

Issues

The “session” cookie is valid for 24 hours. This increases the risk of exploiting the exposed session cookie and broken logout vulnerabilities described in WSTG-SESS-04 and WSTG-SESS-06

The sessions do timeout, attempting to post using a timed-out session returns 401.

More details on this vulnerability and the risk severity can be found under Long Session Timeout.

WSTG-SESS-08

Testing for Session puzzling

Issues

No variables are used in two different ways. The “session” cookie is updated on logout and on POST requests for the front-end to enforce change. If you use the old cookies in POST requests, you may bypass POST limit and still be authenticated after logout. It is also possible to use the updated cookie after a POST request if you bypass the front-end.

More details on these vulnerabilities and the risk severity can be found under Broken Logout and Trusting Frontend to Limit POST.

Data Validation Testing

WSTG-INPV-01

Testing for Reflected Cross Site Scripting

Pass

No user input, to the backend, is ever presented to the user. It only saves the information posted. The only user input presented to the user in the front-end is an image.

WSTG-INPV-02

Testing for Stored Cross Site Scripting

Pass

No user input is stored and then presented on the site.

WSTG-INPV-03

Testing for HTTP Verb Tampering

Pass

Read source code and tested with Burp Suite Proxy. Endpoints configured correctly.

WSTG-INPV-04

Testing for HTTP Parameter pollution

Pass

During POST requests the input is only read once. The sanitizing that could potentially be bypassed through HTTP parameter pollution is the check on the filetype. By looking at the source code we can see that data from the request is immediately converted from base64 and put in another variable before it is sanitized. Other data in the request is also read only once when put in the SQL query. This means all data is only used once, so HTTP parameter pollution is not possible in this application.

WSTG-INPV-05

Testing for SQL Injection

Pass

Read source code. The service only updates 1 row in the database and is using prepared statements.

WSTG-INPV-06

Testing for LDAP Injection

Pass

The application never uses input from the user, it only stores it in a database. There it will not be possible to execute queries needed for a LDAP injection attack.

WSTG-INPV-07

Testing for XML Injection

N/A

The application never takes any XML as input.

WSTG-INPV-08

Testing for SSI Injection

Pass

The user input is only stored in a database, and prepared statements are used to store them. Therefore, SSI injection is not possible. This was tested by analyzing the source code.

WSTG-INPV-09

Testing for XPath Injection

N/A

No XML is used in requests or responses in this application.

WSTG-INPV-10

IMAP/SMTP Injection

Pass

Read source code. Mail service takes no user input, only uses users email from session cookie.

WSTG-INPV-11

Testing for Code Injection

Pass

Testing code injection payloads on the image and pin input using Burp Suite Proxy has not been successful.

WSTG-INPV-12

Testing for Command Injection

Pass

User input is only stored in a database using prepared statements, so there are no entry points for command injection.

WSTG-INPV-13

Testing for Format String

Pass

This was tested by sending in strings such as “%s%s%s%s%s%s%s%s” to the server, but it gave no result. The inputs are only stored in the database.

WSTG-INPV-14

Testing for Incubated Vulnerabilities

Issues

Read source code and found that the service does not sanitize user input. We do not have access to the system that retrieves JPEG, PIN or location from the database, and may therefore not test for vulnerabilities. Based on our knowledge and access we think it might be possible to enter “js([code])” as PIN/location, and since the input is not sanitized, cause issues down the line. See WSTG-BUSL-09.

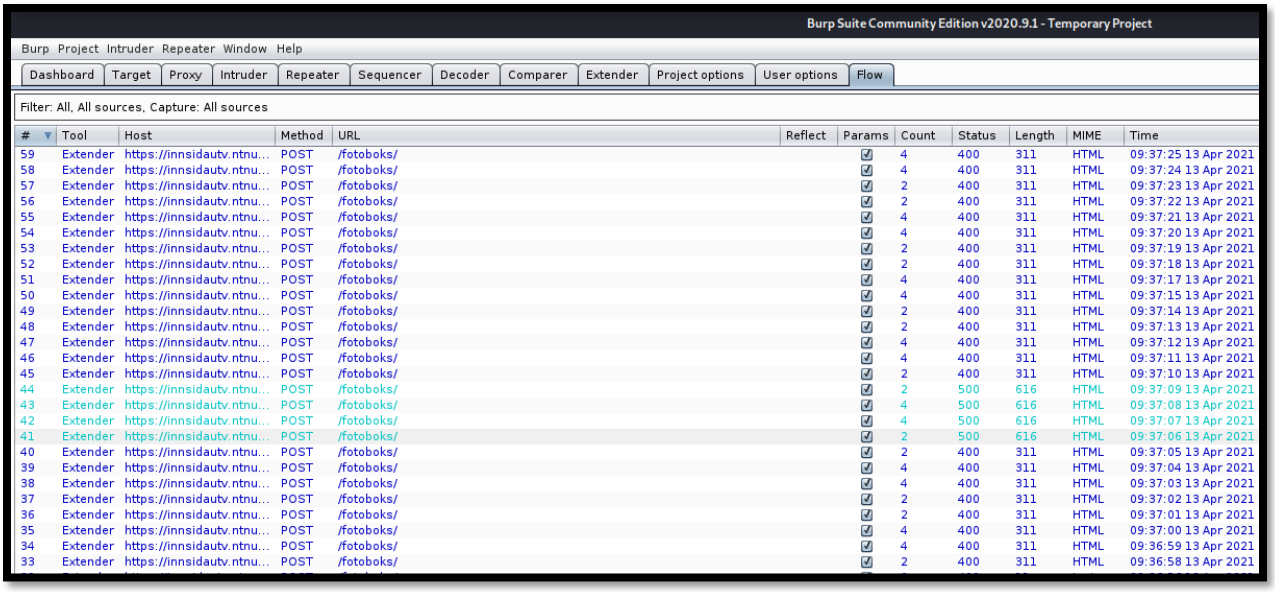
More details on these vulnerabilities and the risk severity can be found under User-input Not Sanitized and Upload Malicious Files.

WSTG-INPV-15

Testing for HTTP Splitting/Smuggling

Pass

HTTP Splitting/Smuggling was tested by running “HTTP Request Smuggler”, which is an automated test extension for Burp Suite. “Flow”, which is another extension for Burp suite, was also used to see the responses to the Request Smuggler’s requests. No vulnerabilities were found.



#	Tool	Host	Method	URL	Reflect	Params	Count	Status	Length	MIME	Time
59	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:37:25 13 Apr 2021
58	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:37:24 13 Apr 2021
57	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	400	311	HTML	09:37:23 13 Apr 2021
56	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	400	311	HTML	09:37:22 13 Apr 2021
55	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:37:21 13 Apr 2021
54	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:37:20 13 Apr 2021
53	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	400	311	HTML	09:37:19 13 Apr 2021
52	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	400	311	HTML	09:37:18 13 Apr 2021
51	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:37:17 13 Apr 2021
50	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:37:15 13 Apr 2021
49	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	400	311	HTML	09:37:14 13 Apr 2021
48	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	400	311	HTML	09:37:13 13 Apr 2021
47	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:37:12 13 Apr 2021
46	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:37:11 13 Apr 2021
45	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	400	311	HTML	09:37:10 13 Apr 2021
44	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	500	616	HTML	09:37:09 13 Apr 2021
43	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	500	616	HTML	09:37:08 13 Apr 2021
42	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	500	616	HTML	09:37:07 13 Apr 2021
41	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	500	616	HTML	09:37:06 13 Apr 2021
40	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	400	311	HTML	09:37:05 13 Apr 2021
39	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:37:04 13 Apr 2021
38	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:37:03 13 Apr 2021
37	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	400	311	HTML	09:37:02 13 Apr 2021
36	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	400	311	HTML	09:37:01 13 Apr 2021
35	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:37:00 13 Apr 2021
34	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	4	400	311	HTML	09:36:59 13 Apr 2021
33	Extender	https://innsidautv.ntnu...	POST	/fotoboks/		☑	2	400	311	HTML	09:36:58 13 Apr 2021

Figure 13: flow gives an overview of all the requests.

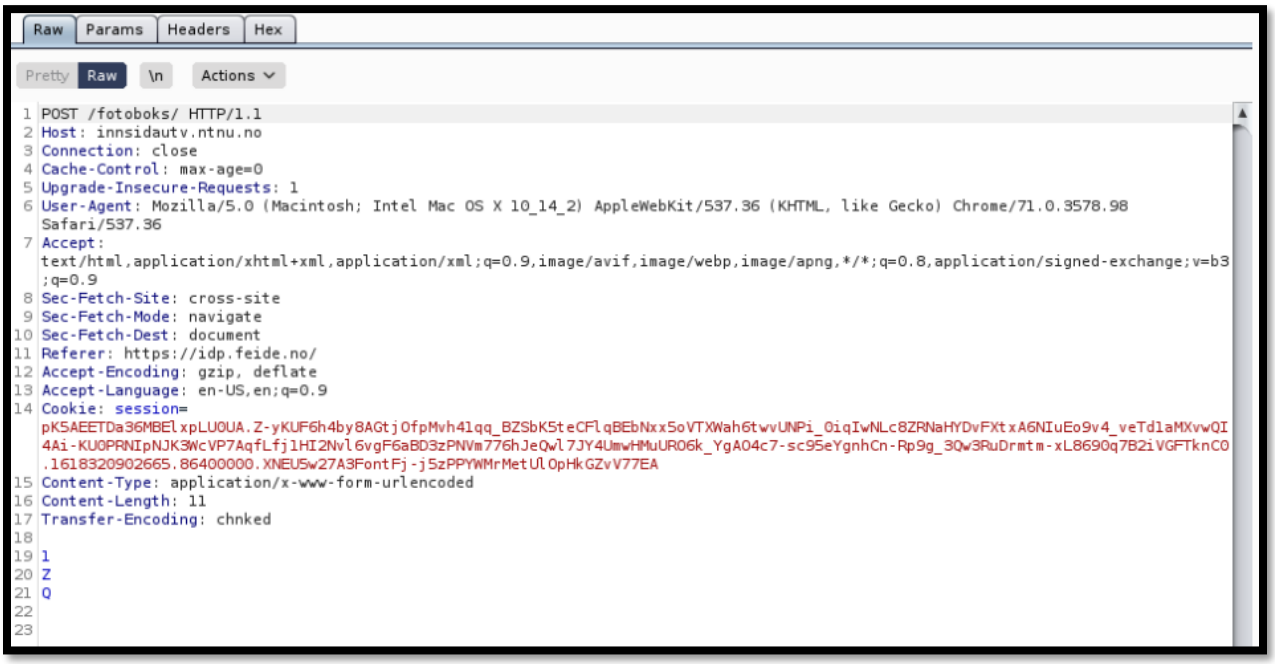


Figure 14: Example of a request sent by HTTP Request Smuggler.

HTTP splitting is not possible as the application never uses user input to generate headers in the response.

WSTG-INPV-16

Testing for HTTP Incoming Requests

N/A

The team does not have access to the machine running the server, so this test cannot be performed.

WSTG-INPV-17

Testing for Host Header Injection

Pass

This was tested by intercepting requests with Burp Suite Proxy and changing the host to a domain controlled by the team. However, this only resulted in a 403 error.

WSTG-INPV-18

Testing for Server-side Template Injection

Pass

This was tested by reviewing requests and responses in Burp Suite Proxy. The application never uses the input sent in by the user, it only stores it in a database. Therefore, the application has no dynamic application responses.

WSTG-INPV-19

Testing for Server-Side Request Forgery

Pass

This was tested using burp proxy and no entry points were found for SSRF.

Error Handling

WSTG-ERRH-01

Analysis of Error Codes

Issues

The error pages for error codes 400 and 404 show the Node.js standard error pages. The error pages for error code 405 and for 414 return error pages including "*Apache/2.4.29 (Ubuntu) Server at innsidautv.ntnu.no Port 443*". Error code 400, 404 and 405 found with Nikto, and error code 414 found by manual testing.

Error 413 is just an empty page with the text "Payload Too Large". Found with Fuzzer Script.

Error	**Example of how to recreate**
400--Bad Request	GET https://innsidautv.ntnu.no/fotoboks/%a%s%p%d
404--Not Found	GET https://innsidautv.ntnu.no/fotoboks/[anything invalid]
405--Method Not Allowed	TRACE https://innsidautv.ntnu.no/fotoboks/
413--Payload Too Large	POST https://innsidautv.ntnu.no/fotoboks/ with a large payload
414--Request-URI Too Long	GET https://innsidautv.ntnu.no/fotoboks/[long string]

Table 14: Overview of error codes and how to recreate the errors.

More details on this vulnerability and the risk severity can be found under CVEs for Apache HTTP server version.

Cryptography

WSTG-CRYP-01

Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection

Issues

Certificate:
Signature Key Length: 4096 (Min 2048)
Public Key Length: 2048
Signature Algorithm: sha384RSA (Min SHA-256)
Valid 366 days (<397 days)
Valid from 28/10/2020 to 29/10/2021
CA: GEANT OV RSA CA 4.
SAN: innsidautv.ntnu.no (Match)

Figure 15: Certificate details.

The certificate is ok.

See WSTG-SESS-02 for issue with insufficient transport layer protection.

WSTG-CRYP-02

Testing for Padding Oracle

Pass

By analyzing the source code, we found that the “session” token is made with the “client-sessions” library, and the “x-token” token is made with “jwt-simple”. None of these libraries have any known vulnerabilities related to padding oracle.

WSTG-CRYP-03

Testing for Sensitive Information Sent Via Unencrypted Channels

Issues

Session cookie can be sent over unencrypted channel. See WSTG-SESS-02 and Cookie “session” Missing Secure Attribute.

Business logic Testing

WSTG-BUSL-01

Test Business Logic Data Validation

Issues

This was tested by reading the source code. There is no validation nor is the user input sanitized. The user input is saves to the database using prepared statement to prevent SQL-injection. The database accepts a “PIN” of max length 10 and “location” of max length 50. The backend checks that uploaded files are in JPEG format by making sure the base64 encodings of the files start with "file":",/9j/". Metadata on uploaded files is not stripped backend.

The frontend does some input sanitizing. The files accepted is only image files, and they will be converted to JPEG and stripped of metadata before being sent to the backend. The PIN is required to be 4 digits long, and may not be “1234”, “2580”, or the same number 4 times. The location text input will be selected from a list.

More details on this vulnerability and the risk severity can be found under User-input Not Sanitized.

WSTG-BUSL-02

Test Ability to Forge Requests

Pass

This was tested by reading the source code. There are no hidden fields.

WSTG-BUSL-03

Test Integrity Checks

N/A

The application has no hidden fields, and the logs are predefined.

WSTG-BUSL-04

Test for Process Timing

Pass

The only parts of the application dependent on time are the tokens. However, expired tokens are properly declined. This was tested by using Burp Suite Proxy and injecting intercepted requests with expired tokens.

WSTG-BUSL-05

Test Number of Times a Function Can be Used Limits

Issues

Under normal use the application will stop you from posting twice, but the method used for limiting the number of posts is ineffective. It works by updating a variable in the session cookie, but the cookie is still accepted. As long as your session has not timed out you can post as many times as you like.

This makes it very easy to fuzz the input, see Fuzzer Script.

More details on this vulnerability and the risk severity can be found under Trusting Frontend to Limit POST.

WSTG-BUSL-06

Testing for the Circumvention of Work Flows

Pass

This test was performed simply by exploring the application and trying to circumvent the workflow. No circumventions were found.

WSTG-BUSL-07

Test Defenses Against Application Mis-use

Issues

The application's rate limiting does not work. There is also no response to repeater errors.

More details on these vulnerabilities and the risk severity can be found under No Automated Response to Repeated Errors and Trusting Frontend to Limit POST.

See also WSTG-BUSL-05 .

WSTG-BUSL-08

Test Upload of Unexpected File Types

Pass

The image upload should only accept the format JPEG. It would not accept other file types. Reading the source code reveals that if the mime header is missing or if the mime header is wrong it will not accept the file.

WSTG-BUSL-09

Test Upload of Malicious Files

Issues

By reading the source code we found that Fotoboks does not strip extra image information from the JPEGs unless it is posted through the front-end, which means it is possible to upload a JPEG containing malicious code that is crafted to exploit a certain renderer or printer. The user is informed that a human will review the picture, which means that the JPEG will be decoded and rendered at some point. If the renderer has a flaw, it might lead to malicious code executing on a system computer.

The back-end also trusts that all JPEGs received are valid, which it should not. This was found using the Image post Script. See also WSTG-BUSL-01.

More details on this vulnerability and the risk severity can be found under Upload Malicious Files.

Client-Side Testing

WSTG-CLNT-01

Testing for DOM based Cross Site Scripting

Pass

By studying requests and responses with Burp Suite proxy it was found that the application never uses any input from the user, it only stores it. Therefore, a DOM base XSS attack is not possible.

WSTG-CLNT-02

Testing for JavaScript Execution

Pass

The application does not use any user input.

WSTG-CLNT-03

Testing for HTML Injection

Pass

This was tested by sending requests to the application and studying the responses using Burp Suite proxy. The application never uses any input from the user, it only stores it, so HTML injection is not possible.

WSTG-CLNT-04

Testing for Client Side URL Redirect

Pass

This test was tested the same way as WSTG-CLNT-03. The application never uses any user input, it only stores it, so client-side URL redirection is not possible.

WSTG-CLNT-05

Testing for CSS Injection

Pass

From reading the source code we learn that the application never uses any user input, it only stores it.

WSTG-CLNT-06

Testing for Client Side Resource Manipulation

Pass

Paths to all resources used are defined either by the “index.html” at “/fotoboks/” or by resources that were defined by the “index.html” initially. No user input can be used to change which resources will be loaded.

WSTG-CLNT-07

Test Cross Origin Resource Sharing

Pass

There are no endpoints that implements CORS.

WSTG-CLNT-08

Testing for Cross Site Flashing

N/A

This application does not use Flash.

WSTG-CLNT-09

Testing for Clickjacking

Pass

The “X-Frame-Options”-header is set to “sameorigin” and therefore Fotoboks will not be rendered inside an iframe.

WSTG-CLNT-10

Testing WebSockets

N/A

The application does not use WebSockets.

WSTG-CLNT-11

Test Web Messaging

Pass

By exploring requests and responses with Burp Suite Proxy it was found that all connection with other sites is done through hardcoded redirects.

WSTG-CLNT-12

Test Local Storage

Pass

All sensitive information handled by the application is in the session cookie.

WSTG-CLNT-13

Testing for Cross Site Script Inclusion

Pass

The only sensitive data that should not be leaked is the session cookie, and it has the http only attribute, and cannot be leaked through JavaScript.

Automated Tests

Nikto

2021-04-13 03:36:20 (GMT-4)

```
+ Target IP:          129.241.57.104
+ Target Hostname:    129.241.57.104
+ Target Port:        80
+ Start Time:         2021-04-13 03:36:20 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.29 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Root page / redirects to: https://innsidautv.ntnu.no/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ 7916 requests: 0 error(s) and 4 item(s) reported on remote host
+ End Time:          2021-04-13 03:39:13 (GMT-4) (173 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Table 15: Terminal output from Nikto.

Nikto scans showing all errors were also ran, and the results can be boiled down to the error codes encountered.

Error	**Example of how to recreate**
400—Bad Request	GET https://innsidautv.ntnu.no/fotoboks/%a%s%p%d
404--Not Found	GET https://innsidautv.ntnu.no/fotoboks/[anything invalid]
405--Method Not Allowed	TRACE https://innsidautv.ntnu.no/fotoboks/

Table 16: Overview of error codes found with Nikto and how to recreate the errors.

Summary

This scan revealed multiple missing security headers.

DirBuster

```
DirBuster 1.0-RC1 - Report
http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
Report produced on Tue Apr 13 10:18:43 EDT 2021
--------------------------------
https://innsidautv.ntnu.no:443
--------------------------------
Directories found during testing:
Dirs found with a 302 response:
/fotoboks/
/fotoboks/logout/
/fotoboks/Logout/
Dirs found with a 200 response:
/fotoboks/flags/
/fotoboks/Flags/
Dirs found with a 502 response:
/fotoboks/cd-emulator/
/fotoboks/Flags/13643/
--------------------------------
Files found during testing:
Files found with a 502 responce:
/fotoboks/flags/macromedia.xml
/fotoboks/logout/products_on.php
/fotoboks/flags/131843.xml
/fotoboks/logout/worcestershire.php
/fotoboks/Flags/13643/19638.xml
```

Figure 16: Output from DirBuster.

Summary

DirBuster found some files and directories the team did not know about. However, these were found with a 502 response. This means the responses from the server were invalid, so these directories and files are probably not interesting.

Nmap

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -p1-65535 129.241.57.104
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-13 03:19 EDT
Nmap scan report for lvs57vip04.it.ntnu.no (129.241.57.104)
Host is up (0.048s latency).
Not shown: 65532 filtered ports
PORT    STATE SERVICE  VERSION
80/tcp  open  http     Apache httpd 2.4.29
```

```
111/tcp open  rpcbind  2-4 (RPC #100000)
443/tcp open  ssl/http Apache httpd 2.4.29 ((Ubuntu))
Nmap done: 1 IP address (1 host up) scanned in 1035.41 seconds
```

Figure 17: Terminal output from Nmap.

Summary

Port 80 and 443 open for communication with the Apache Server and 111 for the RPC server.

Zap

2021-04-13

ZAP gave these results:

Risk Level	Title	Description	Solution
Medium FALSE POSITIVE	X-Frame-Options Header Not Set	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.
Low FALSE POSITIVE The service uses x-token as anti-CSRF token	Absence of Anti-CSRF Tokens	No Anti-CSRF tokens were found in a HTML submission form. A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, use anti-CSRF packages such as the OWASP CSRFGuard. Phase: Implementation Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script. Phase: Architecture and Design

		CSRF attacks are effective in a number of situations, including: * The victim has an active session on the target site. * The victim is authenticated via HTTP auth on the target site. * The victim is on the same local network as the target site. CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.	Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330). Note that this can be bypassed using XSS. Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation. Note that this can be bypassed using XSS. Use the ESAPI Session Management control. This control includes a component for CSRF. Do not use the GET method for any request that triggers a state change. Phase: Implementation Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.
Low The service uses x-token as anti-XSRF token	Cookie Without SameSite Attribute	A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.	Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.
Low	Cookie Without Secure Flag	A cookie has been set without the secure flag, which means that the cookie can be accessed via unencrypted connections.	Whenever a cookie contains sensitive information or is a session token, then it should always be passed using an encrypted channel. Ensure that the secure flag is set for cookies containing such sensitive information.
Low	Incomplete or No Cache-control and Pragma	The cache-control and pragma HTTP header have not been set properly or are missing allowing the browser and proxies to cache content.	Whenever possible ensure the cache-control HTTP header is set with no-cache, no-store, must-revalidate; and that the pragma HTTP header is set with no-cache.

	HTTP Header Set		
Low	Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.	Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.
Low FALSE POSITIVE	X-Content-Type-Options Header Missing	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Note	Timestamp Disclosure - Unix	A timestamp was disclosed by the application/web server - Unix	Manually confirm that the timestamp data is not sensitive, and that the data cannot be aggregated to disclose exploitable patterns.

Figure 18: Report from ZAP.

Scripts

Fotoboks Login

Input username and password and returns session-cookie and x-token.

```
import requests
from bs4 import BeautifulSoup

def login(username, password):
    host = "https://innsidautv.ntnu.no/fotoboks/"
    s = requests.Session()
    r = s.get(host)
    url = r.url
    auth_state = url.split('AuthState=')[1]
    url = 'https://idp.feide.no/simplesaml/module.php/feide/login?AuthState=' + auth_state + '&org=ntnu.no'
    data = {'feidename':username,'password':password}
    cookies = {'SimpleSAMLSessionID': s.cookies.get('SimpleSAMLSessionID', domain='idp.feide.no'), 'SimpleSAMLSessionID_n
ss': s.cookies.get('SimpleSAMLSessionID_nss', domain='idp.feide.no')}
    r = s.post(url, data=data, cookies=cookies)
    soup = BeautifulSoup(r.content)
    saml_response = soup.find('input', attrs={'name':'SAMLResponse', 'type':'hidden'})['value']
    url = 'https://auth.dataporten.no/simplesaml/module.php/saml/sp/saml2-acs.php/feide'
    data = {'SAMLResponse':saml_response}
    cookies = {'SimpleSAMLSessionID': s.cookies.get('SimpleSAMLSessionID', domain='auth.dataporten.no'), 'SimpleSAMLSessi
onID_nss': s.cookies.get('SimpleSAMLSessionID_nss', domain='auth.dataporten.no')}
    r = s.post(url, data=data, cookies=cookies)
    print('session: ' + s.cookies.get('session'))
    url = 'https://innsidautv.ntnu.no/fotoboks/flags'
    cookies = {'session': s.cookies.get('session')}
    r = s.get(url, cookies=cookies)
    token = r.json()['token']
    print('x-token: ' + token)
    json = {'session': s.cookies.get('session'), 'x-token': token}
    return json
```

Figure 19: Python script for Fotoboks login.

Used by other scripts.

Fuzzer

Fuzzes PIN input to find accepted ASCII characters and max length.

```
import requests
from bs4 import BeautifulSoup
from fotoboks_login import login

username = ''
password = ''
json = login(username, password)

host = "https://innsidautv.ntnu.no/fotoboks/"
s = requests.Session()
```

```
url = host + 'pin'
cookies = {'session': json['session']}
headers = {'x-token': json['x-token']}

print('ASCII Table Fuzzing')
for x in range(0, 128):
    data = {"pin":chr(x)}
    r = s.post(url, cookies=cookies, headers=headers, json=data)
    print(x, data, r, r.text)

print('String Length Fuzzing, 1-99')
for x in range(1, 100):
    pin = 'A' * x
    data = {"pin":pin}
    r = s.post(url, cookies=cookies, headers=headers, json=data)
    print('A*' + str(x), r, r.text)

print('String Length Fuzzing')
for x in range(1, 10):
    pin = '1' * (pow(10, x))
    data = {"pin":pin}
    r = s.post(url, cookies=cookies, headers=headers, json=data)
    print('Input 1*10^' + str(x), r, r.text)

print('0000-9999 Fuzzing')
for x in range(0, 10):
    for y in range(0, 10):
        for z in range(0, 10):
            for i in range(0, 10):
                pin = chr(48+x) + chr(48+y) + chr(48+z) + chr(48+i)
                data = {"pin":pin}
                r = s.post(url, cookies=cookies, headers=headers, json=data)
                print(pin, r, r.text)
```

Figure 20: Python script for fuzzing PIN input.

All ASCII characters were accepted.

Max PIN length is 10 characters.

PIN > 10: [500] Internal Server Error
PIN > 10^7: [413] Payload Too Large
PIN > Apache Server answers

No limit for posting PIN, while front-end limits a login to 1 post.

No automated responses of repeated errors/fuzzing.

Cookie/Token list

Writes a list of 10 session-cookies to file and the corresponding x-tokens to another file.

```
from fotoboks_login import login

username = ''
password = ''
sessions = ''
tokens = ''

for x in range(10):
    json = get_login(username, password)
    sessions += json['session'] + '\n'
    tokens += json['x-token'] + '\n'
    print(json)

sessions_file = open('sessions.txt', "a+")
sessions_file.write(sessions)
sessions_file.close()
tokens_file = open('tokens.txt', "a+")
tokens_file.write(tokens)
tokens_file.close()
```

Figure 21: Python script for retrieving a list of session cookies and a list of x-tokens.

Ran the script February 10., March 2. and April 8. by 2 different users.

Image post

Posts a base64 encoded image to the */fotoboks/* endpoint.

```
import requests
from pathlib import Path
import sys
from fotoboks_login import login

path = Path(input('Input text-file with base64 to send: ') or 'base64.txt')
if not path.is_file:
    print('No file at ' + path)
    sys.exit()
pin = input('Input PIN: (1-10 chars) ') or '1337'
if len(pin) > 10 or len(pin) < 1:
    pin = '1337'
    print('Bad PIN. Changed to ' + pin)
location = 'Sentralbygg'
with open(path, "rb") as image_file:
    file = image_file.read()
url = "https://innsidautv.ntnu.no/fotoboks/"
s = requests.Session()
username = ''
password = ''
json = get_login(username, password)
cookies = {'session': json['session']}
headers = {'x-token': json['x-token']}
```

```
data = {'file': str(file), 'pin': pin, 'location': location}
r = s.post(url, cookies=cookies, headers=headers, json=data)
print(r, r.text)
```

Figure 22: Python script for posting base64 encoded file as image.

Accepts invalid JPEGS.

Image to base64

Base64 encodes a JPEG, adds prefix that is checked server side and writes to file.

```
import base64
from pathlib import Path
import sys

inp = input('Input name of file to encode: ') or '123.jpeg'
ext = inp.split('.')[-1]
prefix = ''
path = Path(inp)
if not path.is_file:
    print('No file at ' + path)
    sys.exit()
outp = Path(input('Input name of base64 encoded file: ') or 'base64.txt')
if outp.is_file:
    verify = input('File exists, overwrite?: (y/n) ')
    if verify != 'y':
        sys.exit()
with open(path, "rb") as image_file:
    encoded = base64.b64encode(image_file.read())
    b64_file = open(outp, "w")
    b64_file.write(prefix + str(encoded)[2:-1])
    b64_file.close()
```

Figure 23: Python script for base64 encoding of files.

Used to test image post.

Logout Test

Tries to use the old session cookie after logout.

```
import requests
from pathlib import Path
import sys
from fotoboks_login import login

username = ''
password = ''
json = login(username, password)
host = "https://innsidautv.ntnu.no/fotoboks/"
s = requests.Session()
```

```
url = host + 'pin'
cookies = {'session': json['session']}
headers = {'x-token': json['x-token']}
data = {"pin":"1233"}
url3 = 'https://innsidautv.ntnu.no/fotoboks/flags'
r = s.post(url, cookies=cookies, headers=headers, json=data)
print(data, r, r.text)
url2 = host + 'logout'
r = s.get(url2, cookies=cookies, headers=headers, json=data)
print(data, r, r.text)
s = requests.Session()
r = s.get(url3, cookies=cookies)
print(data, r, r.text)
```

Figure 24: Python script for testing logout functionality.

Session cookie not invalidated on logout.

References

[1] "OWASP Web Security Testing Guide," [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/. [Accessed 27 04 2021].

[2] "Apache Http Server 2.4.29 Security Vulnerabilities," [Online]. Available: https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-66/version_id-241078/Apache-Http-Server-2.4.29.html. [Accessed 27 04 2021].

[3] "developer.mozilla.org," 17 03 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control. [Accessed 27 04 2021].

[4] "Nodemailer CVE," [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-7769. [Accessed 27 04 2021].

[5] "Session Management Cheat Sheet," [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html. [Accessed 27 04 2021].

[6] "MDN Web Docs," 12 March 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options. [Accessed 27 04 2021].