



UNIVERSITEIT VAN PRETORIA  
UNIVERSITY OF PRETORIA  
YUNIBESITHI YA PRETORIA

FACULTY OF ENGINEERING, BUILT ENVIRONMENT  
AND INFORMATION TECHNOLOGY

DEPARTMENT OF ELECTRICAL, ELECTRONIC AND COMPUTER ENGINEERING

<http://www.up.ac.za/eece>

## **EPR 402 LAB BOOK**

*Compiled by*

**Mr. Jacobus M. Becker**  
20426471

*Generated on*

**Friday 7<sup>th</sup> June, 2024**

## **ISG@UP**

INTELLIGENT SYSTEMS GROUP

<http://isg.up.ac.za/>

# Contents

<b>1</b>	<b>2024 March</b>	<b>6</b>
	March Schedule . . . . .	6
	2024/03/01 . . . . .	7
	Project: Intelligent Web Application Firewall using JSON Web Token Inspection . . . . .	7
	Research Ideas . . . . .	7
	Literature Review . . . . .	7
	2024/03/04 . . . . .	10
	Research Ideas . . . . .	10
	Literature Review . . . . .	10
	2024/03/05 . . . . .	11
	JWTs . . . . .	11
	2024/03/08 . . . . .	13
	Research Ideas . . . . .	13
	WAF Basics . . . . .	13
	JWT authentication bypass . . . . .	13
	2024/03/11 . . . . .	14
	TryHackMe . . . . .	14
	DVWA . . . . .	14
	2024/03/12 . . . . .	15
	Research Ideas . . . . .	15
	2024/03/18 . . . . .	16
	2024/03/20 . . . . .	17
	Research Ideas . . . . .	17
	Testing block redaction . . . . .	18
	Testing local redaction . . . . .	18
<b>2</b>	<b>2024 April</b>	<b>19</b>
	April Schedule . . . . .	19
	2024/04/02 . . . . .	20
	2024/04/03 . . . . .	21
	2024/04/04 . . . . .	22
	2024/04/05 . . . . .	23
	2024/04/09 . . . . .	24
	2024/04/10 . . . . .	25
	2024/04/11 . . . . .	26
	2024/04/12 . . . . .	27
	2024/04/15 . . . . .	28
	2024/04/16 . . . . .	29
	2024/04/17 . . . . .	30

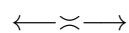
2024/04/18	31
2024/04/19	32
2024/04/22	33
2024/04/23	34
2024/04/24	35
2024/04/25	36
2024/04/26	37
2024/04/29	38
2024/04/30	39
This is a sample heading, typically activity related	40
Approach: Lab book	40
2024/04/04	41
Research Ideas	41
<b>3 2024 May</b>	<b>42</b>
May Schedule	42
2024/05/03	43
2024/05/06	44
2024/05/07	45
2024/05/08	46
2024/05/09	47
2024/05/10	48
2024/05/13	49
2024/05/14	50
2024/05/15	51
2024/05/16	52
2024/05/17	53
2024/05/20	54
2024/05/21	55
2024/05/22	56
2024/05/23	57
2024/05/24	58
2024/05/27	59
2024/05/28	60
2024/05/30	61
2024/05/31	62
<b>4 2024 June</b>	<b>63</b>
June Schedule	63
2024/06/03	64
2024/06/04	65
2024/06/05	66
2024/06/06	67
2024/06/07	68
<b>5 2024 July</b>	<b>69</b>
July Schedule	69
<b>6 2024 August</b>	<b>70</b>
August Schedule	70
<b>7 2024 September</b>	<b>71</b>
September Schedule	71
<b>8 2024 October</b>	<b>72</b>
October Schedule	72

<b>9 2024 November</b>	<b>73</b>
November Schedule . . . . .	73
<b>Appendices</b>	<b>75</b>
Acronyms . . . . .	75
<b>References</b>	<b>76</b>

# Schedule 2024

**Table 1:** EPR 402 Schedule for 2024

Week	Date	Part	Required reading / Assignment due date
1 – 7	4 Mar – 30 Apr	1	<ul style="list-style-type: none"> <li>• Project Proposal</li> </ul>
8 – 11	2 May – 31 May	2	<ul style="list-style-type: none"> <li>• Get a POC model working on WSL</li> </ul>
12 – 14	3 Jun – 28 Jun	3	<ul style="list-style-type: none"> <li>• Get an OTS model working on WSL</li> </ul>
	1 Jul – 19 Jul		<ul style="list-style-type: none"> <li>• Transfer model to embedded device</li> </ul>
15 – 18	22 Jul – 2 Aug	4	<ul style="list-style-type: none"> <li>• Swap out WAF components (Conventional)</li> </ul>
15 – 18	5 Jul – 16 Aug	4	<ul style="list-style-type: none"> <li>• Swap out WAF components (Unconventional)</li> </ul>
19 – 20	19 Aug – 30 Aug	5	<ul style="list-style-type: none"> <li>• Train baseline model</li> </ul>
21 – 23	2 Sept – 17 Sept	6	<ul style="list-style-type: none"> <li>• Analyse efficiency &amp; delay, debug.</li> </ul>
	18 Sept – 27 Sept		<ul style="list-style-type: none"> <li>• Debugging.</li> </ul>
24	30 Sept – 4 Oct	7	<ul style="list-style-type: none"> <li>• Capture final results for the report.</li> </ul>
25 – 29	7 Oct – 7 Nov	8	<ul style="list-style-type: none"> <li>• Write report</li> </ul>



## 1

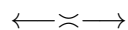
# March 2024

## March Schedule

Table {1.1} Shows the EPR 402 schedule for March 2024.

**Table 1.1:** EPR 402 Schedule for March 2024

Week	Date	Part	Required reading / Assignment due date
1	4 Mar – 8 Mar	1	Literature study and reading.
2	11 Mar – 15 Mar	1	Literature study and reading
3	18 Mar – 20 Mar	1	Project proposal draft
	20 Mar – 31 Mar		Recess



## Friday, 1 March 2024

### Project: Intelligent Web Application Firewall using JSON Web Token Inspection

Next-generation firewalls, also called Web Application Firewalls (WAFs), have significantly improved the defence capabilities of organisations against common web application attacks. These devices perform deep-packet inspection of traffic to detect and alert on anomalies in the HTTP requests and responses. Although these devices can detect common attacks such as cross-site scripting and SQL injection, they still lack the intelligence required to detect attacks masquerading as normal behaviour such as business logic flaws or authorisation bypasses.

However, with the latest move towards decentralised authentication mechanisms, such as JSON Web Tokens (JWTs), which contain information such as user claims, it may be possible to incorporate this information to detect more complex attacks.

The project student is expected to deliver a standalone device that can leverage existing detection techniques and new information that can be found in the claims of tokens to detect more complex attacks such as authorisation bypasses. Using machine learning techniques that incorporate this new information, the student should develop a system that can better learn normal behaviour to detect and alert on complex anomalies.

#### Nature of the work

- A standalone device that can act as a WAF and detect more complex attacker techniques such as authorisation bypasses will be implemented.

#### Research Ideas

- Web Application Firewalls
- Heuristic-based anomaly detection
- Token-based access control
- Read and summarise: Next Generation Firewall for Network Security: A Survey [1]
- Read and summarise: Critical Analysis on Web Application Firewall Solutions [2]

Should do a search to identify any relevant papers and articles

#### Literature Review

##### Next Generation Firewall for Network Security: A Survey [1]

- The paper discusses the security requirements and objectives summarized in Table I related to network security goals.
- It highlights the advantages of next-generation firewall (NGFW) over traditional firewalls, including more accessibility to network traffic, operability across OSI layers, and advanced features to protect against emerging threats.
- The paper provides motivation and the need for NGFW for network security, along with a survey on recent advances in security threats and countermeasures using NGFW.
- Different firewall technologies and their benefits are presented, emphasizing the advantages of NGFW and comparing them with first-generation firewalls.
- The paper concludes by discussing the current state-of-the-art techniques of NGFW from various vendors like Palo Alto, Fortinet, and Check Point in terms of security functions and performance.

### Critical Analysis on Web Application Firewall Solutions [2]

- The paper investigates the impact of climate change on marine biodiversity, specifically focusing on how rising sea temperatures and ocean acidification are affecting marine ecosystems.
- The authors discuss the potential consequences of these changes, including shifts in species distributions, loss of biodiversity, and disruptions to ecosystem functioning.
- They highlight the importance of implementing conservation measures to protect marine biodiversity, such as creating marine protected areas and reducing greenhouse gas emissions.
- The paper also emphasizes the need for further research to better understand the specific mechanisms driving these changes and to develop effective strategies for mitigating their impacts.
- Overall, the authors stress the urgency of addressing climate change and its effects on marine biodiversity in order to prevent irreversible damage to ocean ecosystems.

### Fast Pattern Matching in Compressed Data Packages [3]

- The paper addresses pattern matching for deep packet inspection applications.
- It proposes a method to overcome limitations of pattern matching in compressed data packages for HTTP traffic.
- The proposed scheme aims to avoid searching in compressed data by compressing only necessary parts at the boundaries of back-references.
- The method involves handling back-references and avoiding decompression except for necessary cases.
- The paper highlights the complexity of GZIP/deflate compression algorithms used in HTTP compression and the benefits of the proposed scheme.

### A Fully Automated Deep Packet Inspection Verification System with Machine Learning [4]

- The paper discusses the concept of including and excluding specific application flows based on manual analysis and self-learning algorithms.
- It mentions a verification system that identifies false negative cases when remaining relevant application flows are marked as HTTP/HTTPS/Undetected.
- A sample configuration file for the Times of India app is shown, including include/exclude lists for different parsers like HTTP, SSL, UDP, and Application.
- Specific examples of entries in the exclude and include lists for different parsers are provided, such as Facebook, DoubleClick, AppsFlyer, Yahoo, Flurry, Microsoft, and Amazon.
- The paper also mentions auto signature suggestions from their tool, as shown in Table 5.

## Core Problems

### Primary Problems

#### 1. JWT Inspection:

The product needs to parse incoming JWT tokens to extract claims and verify their integrity and authenticity, validate JWT signatures to ensure they haven't been tampered with and check the expiration time, issuer, and audience of the JWT token to prevent token replay attacks. It also needs to monitor JWT usage patterns to detect abnormal JWT behaviour, such as excessive token creation or usage from unexpected locations.



**2. Heuristic-Based Anomaly Detection:**

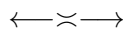
The product needs heuristics to identify abnormal patterns in incoming HTTP requests which may include request rate, payload size, header analysis, HTTP method, user agent analysis, etc. It also needs to implement algorithms to detect anomalies based on these heuristics. Machine learning algorithms such as clustering, decision trees, or neural networks can be used for anomaly detection.

**3. Integration with Web Application Firewall:**

The product needs ensure the WAF can intercept incoming HTTP requests before they reach the application server and implement rules to block or allow requests based on the results of anomaly detection and JWT inspection.

**Secondary Problems**

1. **Continuous Monitoring and Updates:** Continuously monitor the performance of the WAF and adjust heuristics and detection algorithms as needed.
2. **Logging and Reporting:** Log all WAF actions, including blocked requests, allowed requests, and detected anomalies.
3. **Testing and Deployment:** Thoroughly test the WAF in a staging environment, monitoring its performance and effectiveness in real-world scenarios.
4. **Documentation and Training:** Document the WAF's configuration, rules, and operational procedures to train administrators and developers on how to use and configure the WAF effectively.



## Monday, 4 March 2024

### Research Ideas

- Research some more papers with a more technical approach to WAFs.
- Find out more about JWTs on Google.
- Find out which features and functions a WAF has.

Should do a search to identify and summarise any relevant technical papers

### Literature Review

#### A survey of network anomaly detection techniques [5]

This paper proposes an in-depth analysis of classification, statistics, information theory and clustering in anomaly detection techniques. The authors describe the generic framework for network anomaly detection and how it is utilized in most solutions. They also indicate that security incidents had exponential growth in 2009-2014 and anomaly detection has become more important.

The authors describe types of anomalies like point anomalies, contextual anomalies and collective anomalies, how these different anomalies can be detected and the output of different anomaly detection techniques. The authors have a heuristic score where if the anomaly score is above a certain threshold it gets flagged. The types of network attacks like Denial of Service (DoS), Probe, User to Root (U2R) and Remote to User (R2U) are described and how these attacks can be measured by heuristics and how sophisticated the attacks can get.

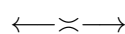
The paper then follows by describing classification-based techniques, statistical anomaly detection techniques, Information theory and cluster-based techniques and their respective complexities. In conclusion, depending on the attack preference, different techniques can be used to detect the anomalies.

#### Deep learning methods in network intrusion detection: A survey and an objective comparison [6]

This paper investigates the efficiency of different deep learning models on different internet traffic datasets. The paper lists supervised instance learning, supervised sequence learning, semi-supervised instance learning and other learning paradigms as deep learning models. The Internet traffic datasets are of different sizes and are real-world data from web servers with the datasets varying in size from 100 000 instances to 16 million instances.

The weighted macro average of the metrics is taken as the heuristic of the neural network and the results are based on the accuracy of the neural network. The authors explain how they do pre-processing on the data and hyper-parameter configuration. The authors list using legacy datasets as one of the several weaknesses, as well as inadequate details of models leading to inaccurate results that cannot be reproduced.

The results show that deep feed-forward neural networks perform the best overall datasets and are perfect for WAF's that have a large number of packets incoming since the ANN gets more efficient on more data. Since this is application-specific, the ANN does not do well on small web servers.



Tuesday, 5 March 2024

## JWTs

JWTs have the following structure according to [7]:

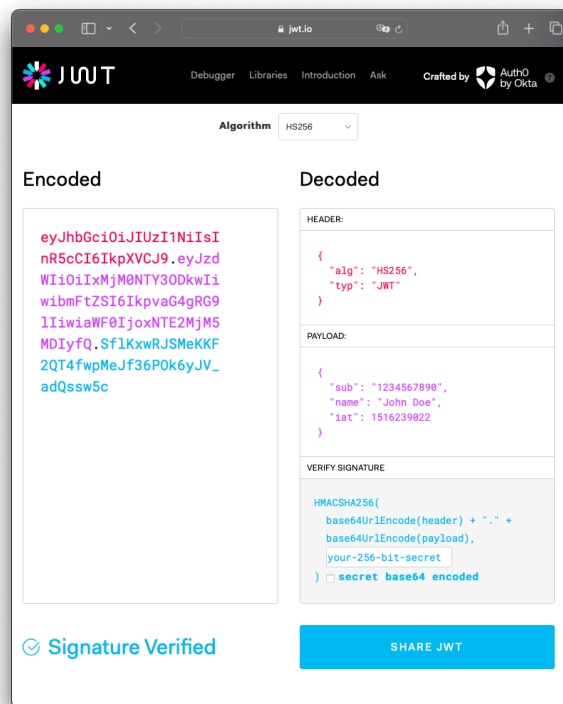


Figure 1.1: JWT Structure

The JWT contains three elements, namely the header, payload and signature. These are individually base64 encoded in a xxx.yyy.zzz format as can be seen above.

### JWT Header

The JWT header contains the following information.

```
1 {  
2   "alg": "HS256",  
3   "typ": "JWT"  
4 }
```

The header consists of two parts. The first part is the type of the token, which is JWT in this case, and the second part is the signing algorithm used, such as HMAC SHA256 or RSA.

## JWT Payload

The JWT payload contains the following information:

```
1 {  
2   "sub": "1234567890",  
3   "name": "John Doe",  
4   "admin": true  
5 }
```

The payload contains the claims of the JWT. Claims are statements about the user and some additional data. Claims fall into one of three categories, namely registered-, public- and private claims.

- **Registered claims** are a set of predefined claims that are not mandatory but recommended. It provides a set of useful and interoperable claims such as **iss**(issuer), **exp**(expiration time), **sub**(subject), **aud**(audience) among others.
- **Public claims** are claims that can be identified at will by the user of the JWT. They are defined in the JWT registry to avoid collisions in the namespace.
- **Private claims** are custom claims created to share information between parties that agree on them and are not registered or public claims.

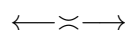
**These claims can be used to extract useful information about the user that is not available in normal HTTP requests.** Since the claim contains information about the user, a portfolio about the user can be built up to predict what a user will or will not do and if a user acts outside of this framework, it can be logged as an anomaly.

## JWT Signature

The signature, as shown below, is a combination of the encoded header, the encoded payload and a secret which is signed with the algorithm specified in the header.

```
1 HMACSHA256(  
2   base64UrlEncode(header) + "." +  
3   base64UrlEncode(payload),  
4   secret)
```

The signature verifies that the message was not changed along the way between the client and server. If the token is signed with a private key, it also verifies that the sender of the token is the client and not a third party.



## Friday, 8 March 2024

### Research Ideas

- Do more practical work, try to hack JWTs.
- Try and get a DVWA (Damn Vulnerable Web Application) going on a VM.

Should try and hack a JWT using TryHackMe

### WAF Basics

WAFs block the following attacks according to [8]

- DDOS
- SQL Injection
- Cross-Site Scripting (XSS)
- Zero-day
- Business Logic (Circumstantial vulnerabilities)
- Man-in-the-middle
- Malware
- Defacements

### JWT authentication bypass

According to [9], JWT authentication bypasses occur when user claims are modified to gain unauthorised access.

**JWT authentication bypasses can be prevented by:**

- Using modern, up-to-date JWT libraries.
- Perform robust signature verification on JWTs and account for edge cases such as using unexpected signing algorithms.
- Enforce a strict whitelist of permitted hosts for the **jku**(JWK Set URL) header.
- Ensure the website is not vulnerable to path traversal or SQL injection via the **kid**(Key ID) header parameter.

← ∞ →

## Monday, 11 March 2024

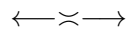
### TryHackMe

Using the tutorial provided by [10], I tried to hack a JWT. It worked for all the tasks. I learned that it would be difficult to use conventional defence methods to stop JWT authentication bypasses since they are legitimate requests that exploit flaws in web applications.

### DVWA

Next, I tried to get the DVWA going on an Ubuntu VM as provided by [11] in order to try and hack the web application on my local network.

I learned that more work will be necessary in this regard since the local network blocks some attacks before they can even reach the host machine.



## Tuesday, 12 March 2024

### Research Ideas

- Start looking at the Project Proposal.
- Try and write the introduction.
- Look at the OWASP Top 10.
- Look at the OWASP testing suite.
- Look at open-source WAF implementations.

Work on Project Proposal

←—∞—→

## Monday, 18 March 2024

### OWASP Top 10 Vulnerabilities

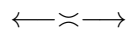
The OWASP Top 10 are the 10 most common vulnerabilities in web applications at the current time. In 2021, which is the most up-to-date version [12], it was indicated as follows:

1. Broken Access Control
2. Cryptographic Failures
3. Injection
4. Insecure Design
5. Security Misconfiguration
6. Vulnerable and Outdated Components
7. Identification and Authentication Failures
8. Software and Data Integrity Failures
9. Security Logging and Monitoring Failures
10. Server-Side Request Forgery

**JWT authentication bypasses** are part of the **Identification and authentication failures**

### Embedded Device

Some possible embedded devices include the O-droids. This includes the M1, which is commonly used or the M1S which is a version with the same processor but different GPU cores.





## Wednesday, 20 March 2024

### Research Ideas

- Look at more vulnerabilities outside of the OWAPS Top 10.
- Focus research on WAF accuracy thresholds
- Look into one WAF serving multiple web applications.
- Keep in mind that both client and server-side attacks need to be included.
- Focus on WAF components

Do more TryHackMe tutorials

Complete Project Proposal

Focus on functional blocks of a WAF

Focus on accuracy of WAF, split into conventional and non-conventional attacks

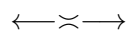
### OWASP Testing Suite

The OWASP Testing Suite is a web application penetration tool to scout for vulnerabilities in web applications, as indicated in [13]. It could be useful to test against existing web applications and WAFs.

### Open-source WAFs

Some open-source WAFs according to [14], include:

- NAXSI
- WebKnight
- Shadow Daemon
- Coraza
- OctopusWAF
- IronBee
- ModSecurity



Redacted, see page 20 for correct version.

### Testing block redaction

Testing redaction of entries that contain errors or are no longer relevant.

### Testing local redaction

As an alternative, if just a local change must be made to a sentence, the follow example shows how this should be done: ~~Example of a redacted sentence~~Replacement text.

← ∞ →

## 2

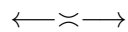
# April 2024

## April Schedule

Table {2.1} Shows the EPR 402 schedule for April 2024.

**Table 2.1:** EPR 402 Schedule for April 2024

Week	Date	Part	Required reading / Assignment due date
4	2 Apr – 5 Apr	1	Research different processing platforms.
5	8 Apr – 12 Apr	1	Test Week - Project proposal draft 1st Revision
6	15 Apr – 19 Apr	1	Project Proposal draft 2nd revision
7	22 Apr – 26 Apr	1	Project Proposal draft final revision
8	29 Apr – 30 Apr	1	Project proposal due



## Tuesday, 2 April 2024

Edit Project Proposal Draft - Functional Block Diagram

Research different processing platforms

Visualise final system

Specifications from requirements

← ∞ →

**Wednesday, 3 April 2024**

Research Reverse Proxies

Project Proposal Draft Revise FBD layout

←— ∞ —→

## Thursday, 4 April 2024

Research Load Balancers

Research WAF efficacy comparison

←— ∞ —→

## Friday, 5 April 2024

Project Proposal Draft - Update System Requirements

Research **Shibboleth Authentication Requests**

←— ∞ —→

## Tuesday, 9 April 2024

Project Proposal Draft - Update Project Description

Project Proposal Draft - Update Primary Design and Implementation Challenges

←—∞—→



**Wednesday, 10 April 2024**

Project Proposal Draft - Update Functional Description

← ∞ →

## Thursday, 11 April 2024

Project Proposal Draft - Update Functional Description  
Upgrade WSL distro version

←— ∞ —→

## Friday, 12 April 2024

Project Proposal Draft - Update Field Conditions

Project Proposal Draft - Upload changes to Git repository on [Merge Request #8](#).

Project Proposal Draft - Remove Unnecessary Information on [Merge Request #9](#).

←—(—→

## Monday, 15 April 2024

Change local repository folder structure

Research [OPNsense Nginx Load Balancing](#)

Research [OPNsense Nginx WAF](#)

Research [A Deep Learning Approach to Web Application Firewall](#)

← — ∞ — →

## Tuesday, 16 April 2024

Research frameworks to measure WAF efficacy [here](#)

Look at code to measure WAF efficacy [here](#)

Research WAF efficacy comparison [here](#)

Research [Exploring the Effectiveness of Web Application Firewalls Against SQL Targeted Attack Vectors](#)

Research [how you can ensure your web applications meet Industry standards?](#)

Research [NGINX App Protect WAF alternatives](#)

Research [NS Web Application Security](#)

Research [Best WAF Solutions in 2023: Real-World Comparison](#)

Research [OWASP ModSecurity Core Rule Set](#)

Research [Three types of WAFs and Key Capabilities](#)

Upload Project Proposal Draft to Git repository on [Merge Request #10](#)

←— ∩ —→

## Wednesday, 17 April 2024

Research WAF comparion [here](#)

Research Risk of False positives [here](#)

Research [All about WAFs](#)

Research [How WAFs work](#)

← ∞ →

## Thursday, 18 April 2024

Research [2020 Top 6 Threats WAFs can prevent](#)

Research [WAF vs NGFW](#)

Research [Comparing F5 Advanced WAF and BIG-IP ASM profiles and features](#)

Research [Next Generation AI-Based Firewalls: A Comparative Study](#)

← ⇐ →

## Friday, 19 April 2024

Project Proposal Draft - Update Functional Block Diagram

Research **2022 Cloud Web Application Firewall (WAF) CyberRisk Validation Comparative Report**

Research **Estimates on the effectiveness of WAFs against targeted attacks**

Research **Web Application Security**

← ∞ →



## Monday, 22 April 2024

Research **How many applications can be hosted on the same server?**

Research **How much RAM is needed to host 10 app pools per web server?**

Research **Security through Optimization Techniques of Firewall Rule Sets**

Research **Correlation Analysis between inference accuracy and Inference parameters for stateless firewall policy**

Add Project Proposal Revision draft milestone to Git

Update Project Proposal Revision Draft **Merge Request #12**

←—∞—→

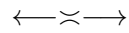
## Tuesday, 23 April 2024

Research [Working of Web Application Firewall](#)

Research [How to measure Database Firewall Effectiveness](#)

Research [CyberRatings Enterprise Firewall Comparative Report](#)

Research [Fortinet Achieves 99 Percent Security Effectiveness Score on Independent Third-Party Next-Generation Firewall Test](#)



**Wednesday, 24 April 2024**

Research [Azure Firewall Metrics](#)

Research [ISA-95 Explained](#)

Research [Huawei Security Products](#)

← ∩ →

**Thursday, 25 April 2024**

Update Project Proposal Draft - Update Functional Description and Functional Block Diagram

Check Project Proposal Draft spelling and grammar on Grammarly

Project Proposal Draft - Update Functional Description and Functional Block Diagram  
←—∩—→

## Friday, 26 April 2024

Update Project Proposal Final Draft - Language Edited

Upload Project Proposal Final Draft to Git repository on [Merge Request #13](#)

Update Project Proposal Final Draft signatures [Merge Request #15](#)

Update README file on Git repository in [Merge Request #16](#)

← — ∞ — →

**Monday, 29 April 2024**

Upload signed project proposal to AMS.

← ∞ →

**Tuesday, 30 April 2024**

Research [TensorFlow](#) Add folder structure to Git repository on [Merge Request #15](#)

← ∞ →

**This is a sample heading, typically activity related**

Some normal text....

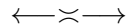
**Approach: Lab book**

Some normal text....

*Keeping a lab notebook is generally considered best practice, both for engineering work and also research projects such as this. A paper based approach is however less than ideal as the resultant work cannot be easily searched. For work of this nature it has further disadvantages with respect to figures, plots and code fragments being harder to manage given that all three are to be generated automatically.*

*A “year+month” based hierarchy was decided on in order to balance the complexity of organisation and the level of nesting. For the cases where additional files must be included. As a result of the nesting however the  $\LaTeX$  `\input` command must be used to handle the inclusion.*

Some normal text....







## 3

## May 2024

## May Schedule

Table {3.1} Shows the EPR 402 schedule for May 2024.

**Table 3.1:** EPR 402 Schedule for May 2024

Week	Date	Part	Required reading / Assignment due date
8	2 May – 3 May	2	Start computer simulations VM
9	6 May – 10 May	2	<ul style="list-style-type: none"><li>• Vulnerable web servers</li><li>• Traffic simulator</li></ul>
10	13 May – 17 May	2	Test week - User interface
11	20 May – 24 May	2	<ul style="list-style-type: none"><li>• Traffic interceptor</li><li>• JWT extractor</li><li>• Traffic proxy</li></ul>
12	27 May – 31 May	2	<ul style="list-style-type: none"><li>• Basic A/D logic</li><li>• Traffic analyser</li><li>• Anomaly logger</li></ul>

## Friday, 3 May 2024

Upload schedule for project to git repository in [Merge Request 17](#)

Upload schedule PDF to [Merge Request #18](#)

←— ∞ —→

## Monday, 6 May 2024

Read up on [JWT Authentication Bypass](#)

Read up on the [OWASP Top 10](#)

Implement the [OWASP Web Security Testing Guide](#)

Research the [Best Open Source Web Application Firewalls](#) Research the [OWASP Cheat Sheet Series](#)

Read up on the [Web Application Firewall Evaluation Criteria](#)

← ∞ →

## Tuesday, 7 May 2024

Research [Juggling with Tokens: Securing JWTs](#) Read up on [VulnHub](#) vulnerable web applications  
Log Milestones and Issues on Git repo for the OTS implementation

←—∞—→

## Wednesday, 8 May 2024

Upload [DVWS](#) on the Odroid M1.

Upload [DVWA](#) on the Odroid M1

Update README in [Merge Request 16](#)

Read up on [MinU: v1 and v2](#)

Read up on [How to run OVA file in Ubuntu](#)

Research [JWT Authentication Bypass via Algorithm Confusion](#)

←—×—→

**Thursday, 9 May 2024**

Split implementation between conventional and unconventional

←—×—

**Friday, 10 May 2024**

Setup Libraries on Odroid M1

←—×—→



## Monday, 13 May 2024

Setup DVWA's on WSL Web server

← ∞ →

**Tuesday, 14 May 2024**

Setup WSTG on WSL Web server

←—∞—→

**Wednesday, 15 May 2024**

Setup Git integration on Odroid M1

← ∞ →

**Thursday, 16 May 2024**

Setup DVWA's on Odroid M1 Web server

←—×—→

**Friday, 17 May 2024**

Test system

←— ∞ —→

## Monday, 20 May 2024

Complete setup of **WSTG**

Research **OpenAppSec WAF and API security**

←— ∞ —→

## Tuesday, 21 May 2024

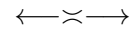
Add Web Servers and WSTG to git in [Merge Request 19](#)

Read up on [deploying to web servers](#)

←— ∞ —→

**Wednesday, 22 May 2024**

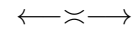
Try traffic proxy





**Thursday, 23 May 2024**

Try JWT decoding using base64 decoder



**Friday, 24 May 2024**

Start with conventional attack detection

←→

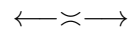
**Monday, 27 May 2024**

Setup VSCode Server with WSL

←→

**Tuesday, 28 May 2024**

Setup VSCode Server with SSH on Odroid M1



## Thursday, 30 May 2024

Smooth out communication errors between Odroid M1 and PC

←—×—

**Friday, 31 May 2024**

Project Proposal Revision - Update Requirements

←— ∞ —→

## 4

## June 2024

### June Schedule

Table {4.1} Shows the EPR 402 schedule for June 2024.

**Table 4.1:** EPR 402 Schedule for June 2024

Week	Date	Part	Required reading / Assignment due date
13	3 Jun – 7 Jun	2	<ul style="list-style-type: none"><li>• Baseline trainer</li><li>• Use TensorFlow</li></ul>
14	10 Jun – 14 Jun	3	<ul style="list-style-type: none"><li>• Obtain embedded device</li><li>• Transfer code to embedded device</li></ul>
	17 Jun – 21 Jun		Exam
15	24 Jun – 28 Jun	3	<ul style="list-style-type: none"><li>• Ensure all libraries are installed</li><li>• Debug code on embedded device</li></ul>

## Monday, 3 June 2024

Add traffic interceptor to the project in [Merge Request 20](#).

Project Demo 1

Show Vulnerable web application and how it can be exploited.

←—(—→



## Tuesday, 4 June 2024

Read up on [Web Application Firewalls](#).

Project Proposal Revision 1 - Update Functional Description and Functional Block Diagram Upload Project

Proposal Revision 1 to git in [Merge Request #21](#).

←—(—→

## Wednesday, 5 June 2024

Upload Project Meeting Minutes to git in [Merge Request #22](#).

Folder structure updated and labbook uploaded to git.

←—⋈—→

**Thursday, 6 June 2024**

← ∞ →

**Friday, 7 June 2024**

← ∞ →

## 5

# July 2024

## July Schedule

Table {5.1} Shows the EPR 402 schedule for July 2024.

**Table 5.1:** EPR 402 Schedule for July 2024

Week	Date	Part	Required reading / Assignment due date
	1 Jul – 5 Jul		Recess
	8 Jul – 12 Jul		Recess
	15 Jul – 19 Jul		Recess
16	22 Jul – 26 Jul	4	<ul style="list-style-type: none"><li>• Swap out traffic interceptor</li><li>• Swap out traffic proxy</li></ul>
17	29 Jul – 31 Jul	4	<ul style="list-style-type: none"><li>• Swap out JWT Extractor</li><li>• Swap out basic logic unit</li></ul>

## 6

## August 2024

### August Schedule

Table {6.1} Shows the EPR 402 schedule for August 2024.

**Table 6.1:** EPR 402 Schedule for August 2024

Week	Date	Part	Required reading / Assignment due date
18	5 Aug – 8 Aug	4	<ul style="list-style-type: none"><li>• Swap out traffic analyser</li><li>• Swap out anomaly logger</li></ul>
19	12 Aug – 16 Aug	4	<ul style="list-style-type: none"><li>• Swap out baseline trainer</li></ul>
20	19 Aug – 23 Aug	5	<ul style="list-style-type: none"><li>• Obtain captured traffic</li><li>• Train baseline model</li></ul>
21	26 Aug – 30 Aug	5	<ul style="list-style-type: none"><li>• Train baseline model</li><li>• Capture results</li></ul>

## 7

## September 2024

### September Schedule

Table {7.1} Shows the EPR 402 schedule for September 2024.

**Table 7.1:** EPR 402 Schedule for September 2024

Week	Date	Part	Required reading / Assignment due date
22	2 Sept – 6 Sept	6	<ul style="list-style-type: none"><li>• Analyse Efficiency</li><li>• Analyse Delay</li></ul>
23	9 Sept – 13 Sept	6	<ul style="list-style-type: none"><li>• Debug errors</li><li>• Debug inefficiencies</li></ul>
24	16 Sept – 17 Sept	6	<ul style="list-style-type: none"><li>• Debug errors</li><li>• Debug inefficiencies</li></ul>
	18 Sept – 27 Sept		Recess
25	30 Sept	7	<ul style="list-style-type: none"><li>• Start capturing final results for report</li></ul>

## 8

# October 2024

## October Schedule

Table {8.1} Shows the planned schedule for EPR 402 in 2024 until the project proposal is due. This is also on my Google calendar for easy access.

**Table 8.1:** EPR 402 Schedule for October 2024

Week	Date	Part	Required reading / Assignment due date
25	1 Oct – 4 Oct	7	Finish capturing final results
26	7 Oct – 11 Oct	8	<ul style="list-style-type: none"><li>• Report - Start and finish introduction section</li><li>• Report - Start theoretical background section</li></ul>
27	14 Oct – 18 Oct	8	<ul style="list-style-type: none"><li>• Report - Finish theoretical background section</li><li>• Report - Start method section</li></ul>
28	21 Oct – 25 Oct	8	<ul style="list-style-type: none"><li>• Report - Finish method section</li><li>• Report - Start results section</li></ul>
29	28 Oct – 31 Oct	8	<ul style="list-style-type: none"><li>• Report - Finish results</li><li>• Report - Start conclusion</li></ul>



## 9

## November 2024

### November Schedule

Table {9.1} Shows the planned schedule for EPR 402 in 2024 until the project proposal is due. This is also on my Google calendar for easy access.

**Table 9.1:** EPR 402 Schedule for November 2024

Week	Date	Part	Required reading / Assignment due date
29	1 Nov	8	<ul style="list-style-type: none"><li>• Report - Finish conclusion</li></ul>
30	4 Nov – 7 Nov	4	<ul style="list-style-type: none"><li>• Report - Revise</li><li>• Report - Language Edit</li></ul>

# Appendix

## Acronyms

<b>EM</b>	Expectation Maximisation . . . . .	41
-----------	------------------------------------	----

← ∞ →

# Bibliography

- [1] K. Neupane, R. Haddad, and L. Chen, “Next generation firewall for network security: A survey,” in *SoutheastCon 2018*, pp. 1–6, 2018. 7
- [2] A. Razzaq, A. Hur, S. Shahbaz, M. Masood, and H. F. Ahmad, “Critical analysis on web application firewall solutions,” in *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*, pp. 1–6, 2013. 7, 8
- [3] M. S. Berger and B. B. Mortensen, “Fast pattern matching in compressed data packages,” in *2010 IEEE Globecom Workshops*, pp. 1591–1595, 2010. 8
- [4] U. Trivedi and M. Patel, “A fully automated deep packet inspection verification system with machine learning,” in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1–6, 2016. 8
- [5] M. Ahmed, A. Naser Mahmood, and J. Hu, “A survey of network anomaly detection techniques,” *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016. 10
- [6] S. Gamage and J. Samarabandu, “Deep learning methods in network intrusion detection: A survey and an objective comparison,” *Journal of Network and Computer Applications*, vol. 169, p. 102767, 2020. 10
- [7] “Jwt debugger.” <https://jwt.io/>. Accessed: 05 March 2024. 11
- [8] “8 types of cyberattacks a waf is designed to stop.” <https://www.indusface.com/blog/8-types-of-cyberattacks-a-waf-designed-to-stop/>. Accessed: 08 March 2024. 13
- [9] G. Patil, “Jwt authentication bypass,” 2022. 13
- [10] “Authenticate - tryhackme walkthrough.” <https://medium.com/@buddhsen/authenticate-tryhackme-walkthrough-e25f8731d371>. Accessed: 11 March 2024. 14
- [11] “Dvwa.” <https://github.com/digininja/DVWA?tab=security-ov-file>. Accessed: 11 March 2024. 14
- [12] “Owasp top 10.” [https://owasp.org/Top10/A00\\_2021\\_Introduction/](https://owasp.org/Top10/A00_2021_Introduction/). Accessed: 18 March 2024. 16
- [13] “Owasp web security testing guide.” <https://owasp.org/www-project-web-security-testing-guide>. Accessed: 20 March 2024. 17

- [14] “Open-source wafs.” <https://www.zenarmor.com/docs/network-security-tutorials/best-open-source-web-application-firewalls>. Accessed: 20 March 2024. 17
- [15] D. Marr, *Vision: A Computational Investigation Into the Human Representation and Processing of Visual Information*. MIT Press, 2010. 41
- [16] F. Dellaert and C. Thorpe, “Robust car tracking using Kalman filtering and Bayesian templates,” in *Proceedings of SPIE: Intelligent Transportation Systems*, vol. 3207, October 1997.