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

Wednesday 5th June, 2024

ISG@UP
INTELLIGENT SYSTEMS GROUP
http://isg.up.ac.za/

Contents

Schedule 2024

Table 1: EPR 402 Schedule for 2024

Week	Date	Part	Required reading / Assignment due date
1 – 7	4 Mar – 30 Apr	1	• Project Proposal
8 – 11	2 May – 31 May	2	• Get a working hybrid OTS model on a PC.
12 – 14	3 Jun – 28 Jun	3	• Transfer model to the embedded device.
15 – 18	22 Jul – 16 Aug	4	• Start swapping out WAF components
19 – 20	19 Aug – 30 Aug	5	• Train baseline model
21 – 23	2 Sept – 17 Sept	6	• Analyse efficiency & delay, debug.
24	30 Sept – 4 Oct	7	• Capture final results for the report.
25 – 29	7 Oct – 7 Nov	8	• Write report

1

March 2024

March Schedule

Table **{??}** 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 [**?**]
- Read and summarise: Critical Analysis on Web Application Firewall Solutions [**?**]

Should do a search to identify any relevant papers and articles

Literature Review

Next Generation Firewall for Network Security: A Survey [?]

The authors of this paper attempt to understand what differentiates next-generation firewalls (WAF's) from traditional firewalls we are generally used to. They go into depth to see which features these firewalls use that make them desirable and what traditional firewalls lack in the modern age. These authors list commonly occurring cyber attacks and how each firewall deals with each attack. The authors also compare the advantages and disadvantages of each firewall.

The performance benchmarks used by the authors are Advanced Evasion Techniques, Targeted Cyber-Attack, Web Application Attacks, and Data Focused Attacks. The authors also compare the differences between the two firewalls in the following areas: Preventing Advanced Persistent Attacks, Inspecting SSL Traffic, Controlling Web Applications, Managing Users & Use Policy and Trade off Security vs Performance.

Overall this paper does not go into too much technical detail about what WAF's do and how they operate, but the authors make the point that it is necessary because of its advantages over traditional firewalls. Finally, the authors

look at an example of the Palo Alto Next-Generation Firewall and how it manages to extend its functionality over traditional firewalls.

Critical Analysis on Web Application Firewall Solutions [?]

This paper compares existing WAF solutions to see which gaps exist in which solutions. The authors make the argument that certain solutions have certain features that make them a better fit for certain scenarios. Solutions that are being compared include Mod Security, Imperva's Secure Sphere, Barracuda network application gateway etc.

The paper goes into some technical detail on how and where WAF's operate. WAF's work on OSI layer 7, the application layer by deeply inspecting the HTTP packet and directing and analyzing it for malicious strings and configuration error problems. The authors compare all the options using metrics like Time efficiency, how well its organised, Effectiveness, if it has Monitoring, Blinking, Response Filtering, Attack Prevention, Web Site Cloaking, Authentication and Web SSO, Deep Inspection, Session Protection and finally evaluate the Overall Security Performance.

The researchers also compare the management interface of the WAF applications using the following metrics: Web Based, Command Line, Desktop Based, Ease-of-use, Comprehensiveness and Flexibility. The authors make a final observation that WAF's have become essential for each website and can become highly customized, but depending on your business needs a best fit solution can be found.

Fast Pattern Matching in Compressed Data Packages [?]

The authors of this paper explore Fast Pattern Matching which can significantly speed up Firewalls and intrusion prevention systems by leveraging deep packet inspection of compressed data packages in high-speed systems. The authors attempt to leverage hardware Huffman encoding by only decompressing data once and then filtering similar packets since the system does not need to decompress packets with the same source and destination address as the first decompressed packet.

The authors explain that this is also an important step in ad filtering since keyword searches for ad-related words in the decompressed packages can also be searched for. According to the authors, this method has enormous speed increases compared to pure software encoding.

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

The authors of this paper created a deep packet inspection (DPI) verification system for growing network density and Quality of Service. The authors use mobile automation tools like Monkey Runner, UI Automator, Monkey Talk, Appium and GUITAR to simulate application web traffic. This is easier than generating application web traffic using a mobile web browser since not a lot of tools for this exist.

The authors explore doing Automated Application Signature Verification for the HTTP Packets. The solution works by running a command-line packet sniffer and network protocol analyzer to see all the HTTP traffic. From this an include/exclude list which contains specific keywords that are either blocked or allowed. The packets are inspected and allowed or blocked based on the specific keyword. This is done differently for HTTP, SSL, TCP and UDP.

The proposed script generates a list of relevant flows based on the SSL, TCP or UDP flow size and rate. The verification system detects the flow rates and flags any suspicious flow rates. The final result is a fully automated signature verification system of the flow rates.

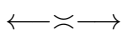
Core Problems

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 [?]

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.

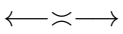
The paper then follows by describing classification-based techniques, statistical anomaly detection techniques, Information theory and cluster-based techniques and their respective complexities. In conclusion, depending on the attack preference, different techniques can be used to detect the anomalies.

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

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 [**?**]:

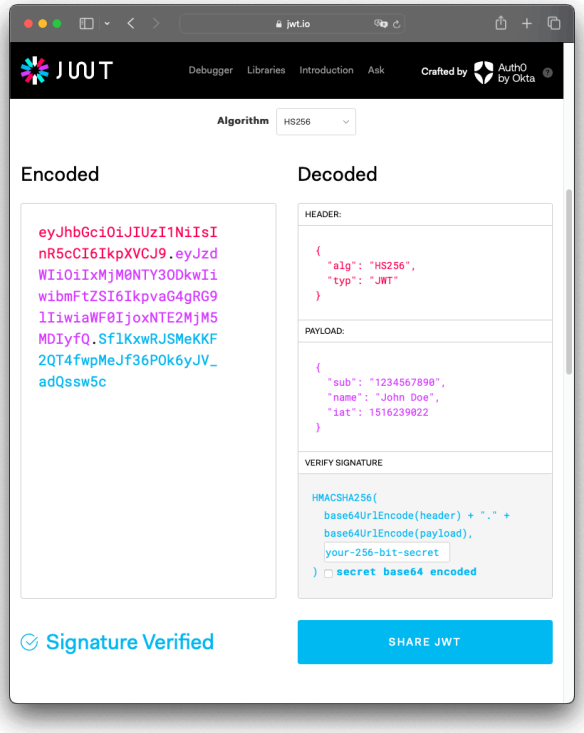


Figure 1.1: JWT Structure

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:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

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.

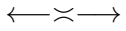
These claims can be used to extract useful information about the user that is not available in normal HTTP requests. Since the claim contains information about the user, a portfolio about the user can be built up to predict what a user will or will not do and if a user acts outside of this framework,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

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  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 [**?**]

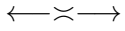
- DDOS
- SQL Injection
- Cross-Site Scripting (XSS)
- Zero-day
- Business Logic (Circumstantial vulnerabilities)
- Man-in-the-middle
- Malware
- Defacements

JWT authentication bypass

According to [**?**], 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 [**?**], 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 [**?**] 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

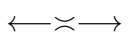
The OWASP Top 10 are the 10 most common vulnerabilities in web applications at the current time. In 2021, which is the most up-to-date version [**?**], it was indicated as follows:

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 [**?**]. It could be useful to test against existing web applications and WAFs.

Open-source WAFs

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

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

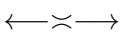
Redacted, see page ?? for correct version.

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 **{??}** 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

This is a sample heading, typically activity related

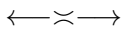
Some normal text....

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....



Thursday, 3 April 2024

Research Ideas

- Idea 1... testing acronym **EM!** (**EM!**)
- Idea 2... testing index SLAM
- Idea 3... testing href and reference David Marr [**?**]

Should do a search to identify any relevant ...

Eq. **(??)** shows an example of a numbered equation.

$$\Delta w_{i,j} = \alpha a_i \Delta[j] \tag{2.1}$$

A subsection a subsection a subsection a subsection a subsection a subsection a subsection a subsection a subsection a subsection a subsection a subsection a subsection a subsection a subsection a subsection a subsection.

Fig. **[??]** shows an example of a figure.

Figure 2.1: This is a figure caption

Third theme of literature study third theme of literature study third theme of literature study third theme of literature study third theme of literature study. See Section **??** for more information.

3

May 2024

May Schedule

Table **{??}** 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	• Vulnerable web servers • Traffic simulator
10	13 May – 17 May	2	Test week - User interface
11	20 May – 24 May	2	• Traffic interceptor • JWT extractor • Traffic proxy
12	27 May – 31 May	2	• Basic A/D logic • Traffic analyser • Anomaly logger

Tuesday, 7 May 2024

JWTs

4

June 2024

June Schedule

Table **{??}** 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	• Baseline trainer • Use TensorFlow
14	10 Jun – 14 Jun	3	• Obtain embedded device • Transfer code to embedded device
	17 Jun – 21 Jun		Exam
15	24 Jun – 28 Jun	3	• Ensure all libraries are installed • Debug code on embedded device

5

July 2024

July Schedule

Table **{??}** 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	• Swap out traffic interceptor • Swap out traffic proxy
17	29 Jul – 31 Jul	4	• Swap out JWT Extractor • Swap out basic logic unit

6

August 2024

August Schedule

Table **{??}** 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	• Swap out traffic analyser • Swap out anomaly logger
19	12 Aug – 16 Aug	4	• Swap out baseline trainer
20	19 Aug – 23 Aug	5	• Obtain captured traffic • Train baseline model
21	26 Aug – 30 Aug	5	• Train baseline model • Capture results

7

September 2024

September Schedule

Table **{??}** 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	• Analyse Efficiency • Analyse Delay
23	9 Sept – 13 Sept	6	• Debug errors • Debug inefficiencies
24	16 Sept – 17 Sept	6	• Debug errors • Debug inefficiencies
	18 Sept – 27 Sept		Recess
25	30 Sept	7	• Start capturing final results for report

8

October 2024

October Schedule

Table **{??}** 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	• Report - Start and finish introduction section • Report - Start theoretical background section
27	14 Oct – 18 Oct	8	• Report - Finish theoretical background section • Report - Start method section
28	21 Oct – 25 Oct	8	• Report - Finish method section • Report - Start results section
29	28 Oct – 31 Oct	8	• Report - Finish results • Report - Start conclusion

9

November 2024

November Schedule

Table **{??}** 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	• Report - Finish conclusion
30	4 Nov – 7 Nov	4	• Report - Revise • Report - Language Edit

Appendix

Acronyms

$\longleftarrow\asymp\longrightarrow$