BAB III. METODE PENELITIAN

3.1. Alat dan Bahan Penelitian

Alat dan bahan yang digunakan pada penelitian ini dikategorikan menjadi dua perangkat, yaitu berupa (1) perangkat keras dan (2) perangkat lunak. Secara umum, kategori perangkat keras yang digunakan berada pada kategori *high end*, agar dapat melakukan tugas dan proses secara cepat. Penjelasan rinci terkait spesifikasi perangkat lunak dan keras adalah sebagai berikut.

3.1.1. Alat Penelitian

A. Perangkat Keras (*Hardware*)

Penelitian ini menggunakan tiga perangkat keras yang terdiri dari komputer desktop, laptop, dan router internet. Ketiganya diintegrasikan menjadi *connected security-lab*. Penjelasan spesifikasi tiap perangkat adalah sebagai berikut.

1) Komputer Desktop

Komputer dekstop digunakan untuk menginstalasi perangkat lunak yang diperlukan pada penelitian. Komputer desktop digunakan untuk menginstalasi aplikasi *website server* Apache2 beserta modul dan pustaka-pustaka yang diperlukan. Adapun spesifikasi yang digunakan sebagai berikut.

Merk	:	Lenovo
Tipe	:	IdeaCentre G5 14AMR05
Prosesor	:	AMD Ryzen 7 3700X 8-Core Processor 3.59 GHz
RAM	:	16 GB
Storage	:	SSD 512 GB dan HDD 1TB

Sistem Operasi	:	Windows 11
Edisi	:	Windows 11 Home Single Language
Kartu Grafis	:	NVIDIA GeForce GTX 1660 6GB GDDR6

2) Laptop

Peneliti akan menggunakan satu laptop untuk menginstalasi aplikasi penyerang, yaitu ZAP dan Arachni. Instalasi *web server* dan aplikasi penyerang dipisahkan agar performa masing-masing perangkat tidak terpengaruh. Dengan kata lain, proses pengajuan dapat dilakukan lebih cepat. Adapun spesifikasi laptop yang digunakan untuk menyerang adalah sebagai berikut.

Merk	:	Acer
Tipe	:	Acer Nitro 7 (AN715-51)
Prosesor	:	Intel® Core™ i7-9750H Processor
RAM	:	16 GB
Storage	:	SSD 512 GB dan HDD 1TB
Sistem Operasi	:	Windows 11
Edisi	:	Windows 11 Home Single Language
OS Build	:	22000.318
Kartu Grafis	:	NVIDIA GeForce GTX 1660 6GB GDDR6

3) *Router* Internet

Untuk menghubungkan komputer desktop dan laptop atau *web server* dan aplikasi penyerang, peneliti menggunakan *router* dengan spesifikasi berikut.

Merk	:	TotoLink
Tipe	:	TotoLink A950RG AC1200
Jaringan Wireless	:	IEEE 802.11ac, IEEE 802.11a, IEEE 802.11b
Frekuensi Sinyal	:	2.4GHz/5GHz
Kecepatan Data	:	2.4GHz: Up to 300Mbps | 5GHz: Up to 867Mbps

B. Perangkat Lunak (Software)

Untuk melakukan pengujian terhadap metode deteksi dan mitigasi, penelitian ini menggunakan tujuh perangkat lunak, yaitu WSL2, Ubuntu Server, Apache, Python, MariaDB, dan Zed Attack Proxy, dan Arachni. Penjelasan terkait versi dan spesifikasi perangkat lunak yang dipergunakan adalah sebagai berikut.

a) *Windows Subsystem for Linux 2 (WSL*)

WSL adalah perangkat lunak berbasis Windows yang dapat digunakan untuk melakukan virtualisasi sistem operasi Linux. Dengan perangkat lunak ini, sistem operasi dapat diinstalasi di atas sistem operasi bawaan. Peneliti melakukan sistem operasi Linux virtualisasi karena tiga alasan, yaitu (1) mempermudah proses instalasi, terutama ketika terdapat kendala teknis; (2) memungkinkan pengelolaan sumber daya sistem operasi sesuai kebutuhan penelitian; dan (3) mempercepat proses *backup* dan *restore* apabila terjadi pergantian perangkat keras. Oleh karena itu, peneliti menggunakan WSL. Spesifikasi WSL yang digunakan sebagai berikut.

Versi : (GNU/Linux 5.10.102.1-microsoft-standard-WSL2 x86_64
Sistem Operasi : Windows 11
Tanggal Rilis : 22 November 2021

b) Sistem Operasi Ubuntu

Sistem operasi yang digunakan untuk menjalankan *web server* adalah sistem operasi berbasis Linux, yaitu Ubuntu. Peneliti memilih sistem operasi ini karena lima alasan, yaitu (1) memiliki banyak dokumentasi dan komunitas, terutama terkait instalasi dan petunjuk teknis lainnya; (2) pembaruan dilakukan lebih intensif dibandingkan sistem operasi Linux lainnya; (3) kemudahan dalam mencari modul atau

pustaka tambahan; (4) tingkat kompabilitas yang lebih tinggi; dan (5) kode atau perintah yang digunakan lebih mudah dipahami. Spesifikasi dan versi sistem operasi Ubuntu yang digunakan adalah sebagai berikut.

Versi	:	22.04 LTS
Penyimpanan	:	128 Gb
RAM	:	16 GB
Prosesor	:	4-6 *Cores*
Laman Unduhan	:	https://ubuntu.com/download/server

c) *Web Server* Apache2

OWASP *ModSecurity Firewall* adalah modul *web server*, sehingga akan diinstalasi pada *web server* Apache2. Adapun versi yang digunakan adalah 2.4.51 yang dirilis pada 07 Oktober 2021. Selain OWASP *ModSecurity Firewall,* peneliti juga melakukan instalasi Apache2 Mod WSGI (*Web Server Gateway Interface*) untuk menjalankan *web framework* Django.

d) Python dan Django

Web *framework* yang diujikan (untuk lapisan kedua) adalah Django. Pemilihan Django dilakukan karena berdasarkan studi komparasi yang dilakukan oleh penelitian [24] dan [23], Django disebut sebagai *web framework* yang memiliki tingkat dan fitur keamanan yang baik. Django adalah *web framework* berbasis Python. Oleh karena itu, peneliti juga akan menginstalasi Python. Selain digunakan untuk menjalankan Django, Python juga digunakan untuk keperluan *machine learning*.

Versi Django yang digunakan adalah 3.2 dan akan diinstalasi dalam *virtual environment* menggunakan Python 3.10. Dalam pengujian mitigasi, terdapat tiga *website* berbasis Django yang akan diujikan. Ketiga *website* tersebut dikonfigurasi-

kan menjadi tiga tingkatan keamanan, yaitu rendah, sedang, dan tinggi. Konfigurasi tersebut untuk melihat bagaimana efektivitas masing-masing tingkatan.

e) MariaDB

MariaDB merupakan *database management system* dengan sumber terbuka yang dikembangkan oleh Monty. Penelitian ini menggunakan MariaDB versi 10.6.5 pada arsitektur x86_64 yang berjalan dalam *systemd*.

f) Zed Attack Proxy (ZAP)

ZAP adalah salah satu perangkat lunak penyerang yang digunakan untuk melakukan simulasi penyerangan. Melalui aplikasi ini, tingkat efektivitas metode mitigasi berbasis *multi-layer security* dapat diukur secara mudah. Versi yang digunakan untuk penelitian ini adalah versi 2.11.0.

g) Arachni

Selain ZAP, perangkat lunak penyerang yang digunakan selanjutnya adalah Arachni. Peneliti tidak memilih versi yang terbaru, melainkan versi **2.0dev-1.0dev** atau *nightly version*, karena sering terjadi masalah pada saat mengelola serangan ketika menggunakan versi yang terbaru.

3.1.2. Bahan Penelitian

Sebelum memasuki penjelasan terkait bahan penelitian, berikut ini adalah penjelasan terkait objek penelitian. Penjelasan terkait objek penelitian ini diperlukan agar rasionalitas penggunaan bahan penelitian dapat terdefinisi lebih baik. Objek penelitian ini adalah teknik serangan siber berdasarkan daftar celah keamanan OWASP Top-10 *Common Web Vulnerabilities* tahun 2021.

Teknik serangan siber tersebut terdiri dari *Cross Site Scripting* (XSS), SQL *Injection*, dan *Remote Code Execution* (RCE). Ketiga teknik serangan tersebut telah definisikan oleh para ahli pada bab II. Untuk memperjelas objek penelitian ini, kelima teknik serangan tersebut secara singkat dapat didefinisikan sebagai berikut.

a) *Cross-Site Scripting* (XSS) adalah serangan yang memanfaatkan kode-kode Javascript untuk mengeksekusi perintah tertentu, seperti melakukan pencurian akun pengguna, data privasi, dan sejenisnya.
b) SQL *Injection* adalah serangan yang dilakukan melalui perantara *query* SQL yang dimodifikasi dan dikirimkan melalui permintaan *browser* atau teknologi sejenis yang memungkinkan pemrosesan *request* dan *response*.
c) *Remote Code Execution* (RCE) adalah adalah ancaman siber yang mengeksploitasi fungsional *web server* melalui skrip atau berkas.

Kelima objek penelitian tersebut diujikan oleh dua metode, yaitu metode deteksi dan metode mitigasi, seperti yang dijelaskan lebih rinci pada poin 3.5. Metode deteksi menghasilkan tingkat akurasi deteksi terhadap serangan, sedangkan metode mitigasi digunakan untuk mengukur tingkat efektivitas *multi-layer security*. Selain itu, kelima objek penelitian tersebut disematkan pada *website* uji coba.

Seperti yang diungkapkan pada penjelasan-penjelasan sebelumnya, terdapat tiga teknik serangan siber yang diujikan, yaitu (1) *Cross Site Scripting*; (2) SQL *Injection*; (3) *Remote Code Execution*. Karena setiap teknik memiliki karakteristik masing-masing, ketiga teknik serangan tersebut memiliki sumber dataset yang berbeda-beda. Secara umum, penelitian ini banyak menggunakan *public dataset* yang relevan. Penggunaan *public dataset* diharapkan akan mempermudah pene-

litian lanjutan, terutama bagi peneliti yang ingin melakukan penelitian dengan *dataset* yang sama. Adapun penjelasan terkait sumber data adalah sebagai berikut.

Tabel 3.1 Sumber Data Penelitian

No	Teknik Serangan	Sumber Data
1	*Cross Site Scripting (*XSS)	***Main XSS Attack Dataset*** **(Github)** https://github.com/fmereani/Cross-Site-Scripting-XSS ***Supportive XSS Attack Dataset*** **(Github)** https://github.com/payloadbox/xss-payload-list ***Chalengging Dataset*** **(Kaggle)** https://www.kaggle.com/datasets/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning Ketiga *dataset* di atas diambil atau diunduh pada 27 Januari 2022 (*latest version)*
2	SQL *Injection*	***Main SQLi Attack Dataset*** **V3 (Kaggle)** https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset ***Chalengging Dataset*** **V2 (Kaggle)** https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset *Dataset* tersebut diambil atau diunduh pada 27 Januari 2022 (*latest version)*
3	*Remote Code Execution*	***Main RCE Attack Dataset*** **(Github)** https://github.com/swisskyrepo/PayloadsAllTheThings https://www.kaggle.com/datasets/antonyj453/urldataset ***Chalengging Dataset*** https://github.com/carlospolop/Auto_Wordlists Kedua *dataset* di atas diambil atau diunduh pada 15 Januari 2022 (*latest version)*

Dalam kaitannya dengan metode deteksi, ketiga teknik serangan memiliki kesamaan yaitu, berbasis *string* atau teks. Dengan kata lain, implementasi metode deteksi pada penelitian ini berkaitan dengan pemrosesan dan pengenalan pola teks. Setiap serangan pada dasarnya adalah rangkaian-rangkaian teks perintah atau eksekusi. Oleh karena itu, untuk meningkatkan akurasi deteksi, peneliti menggunakan *machine learning* sebagai metode untuk mengenali serangan berbasis teks tersebut.

3.2. Tahapan Penelitian

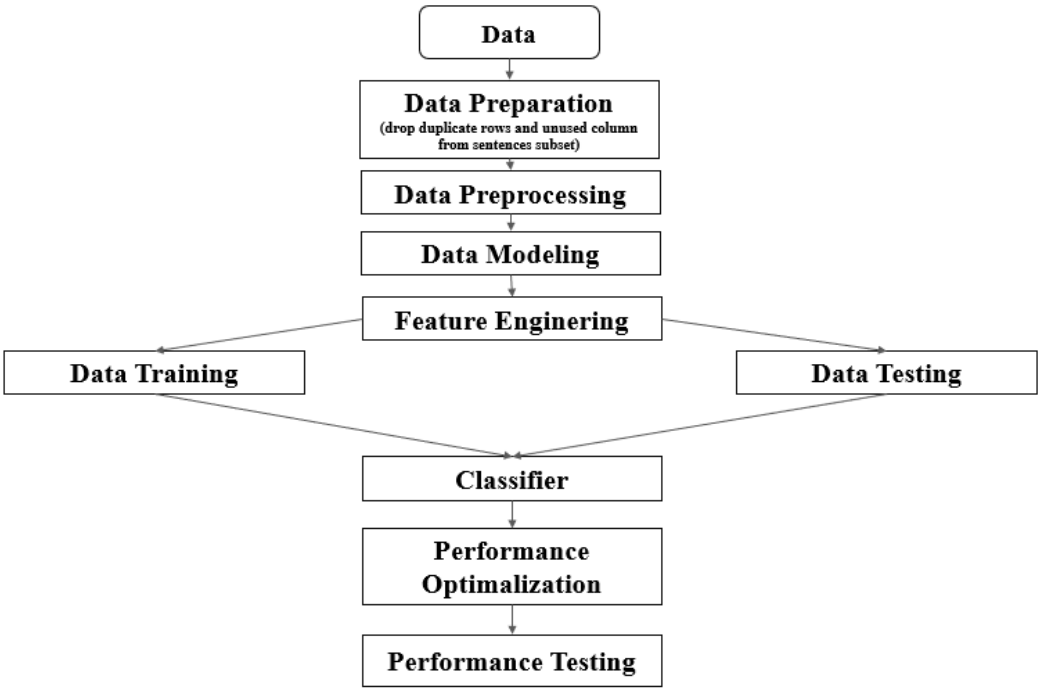
Karena memiliki dua objek penelitian, penelitian ini menerapkan dua skema tahapan penelitian. Penggunaan dua skema tahapan penelitian dilakukan agar proses penelitian dapat berjalan lebih terarah dan terukur. Skema pertama berkaitan dengan tahapan untuk mengimplementasikan, menguji, dan mengevaluasi metode deteksi. Sementara itu, skema kedua merupakan tahapan untuk mengimplementasikan, menguji, dan mengevaluasi metode mitigasi.

3.2.1. Skema Tahapan Metode Deteksi

Terdapat lima tahapan yang dilakukan dalam menghasilkan metode deteksi. Pertama, pada tahap *preparation*, dataset disiapkan sesuai dengan kebutuhan, mulai dari *crawling*, penggabungan, dan sebagainya. Tahap kedua adalah *data preprocesing* dan *data modeling*. Pada tahapan ini, terdapat beberapa tindakan umum yang dilakukan yaitu (a) melakukan eliminasi pada baris dataset yang kosong atau tidak lengkap; (b) menghilangkan data yang duplikat; dan (c) melakukan konversi data sesuai dengan kebutuhan. Tahap ketiga adalah melakukan *feature enginering*, *data training*, dan *data testing*. Pada tahapan *feature enginering*, tindakan yang dilakukan adalah (a) kontruksi *corpus* dan (b) kontruksi fitur. Metode deteksi mengguna-

kan bantuan pustaka Python NLTK, sehingga setiap baris dataset dikonversikan menjadi corpus sebelum di-*training* dan *testing*. Setelah *classifier* dihasilkan, tahap berikutnya adalah melakukan *performance optimalization*.

Setiap algoritma diujikan tingkat akurasinya. Terdapat lima algoritma deteksi yang diujikan, yaitu (1) Naïve Bayes; (2) Logistic Regression; (3) Gradient Boosting; (4) Support Vector Machine; dan (5) K-Nearest Neighbor. Peneliti memilih algoritma deteksi berdasarkan tingkat akurasi tertinggi. Kemudian, peneliti juga melakukan *performance testing*. Artinya, algoritma dengan tingkat akurasi tertinggi diujikan untuk mendeteksi baris dataset baru atau yang berbeda dengan dataset yang digunakan pada proses *training* dan *testing*. Jika terdapat serangan yang tidak terdeteksi secara baik, serangan tersebut akan diproses oleh tahapan mitigasi.



Gambar 3.1 Skema Tahapan Implementasi Metode Deteksi

Pada tahapan *performance optimalization*, peneliti juga melakukan kalkulasi *confussion matrix*, sehingga efektivitas model pembelajaran mesin yang telah diha-

silkan dapat diukur lebih komprehensif. Tahapan ini mengevaluasi kinerja atau performa model, sehingga model menghasilkan metode deteksi yang akurat. Selain itu, pada *confussion matrix* juga ditampilkan *accuracy, precision, recall,* dan juga *f-measure*. Berikut ini tabel *confussion matrix* dengan pustaka NLTK.

Tabel 3.2 Tabel *Confussion Matrix* pada Metode Deteksi

	Payload	***Non-Payload***
Payload	*<valid>*	*invalid*
Non-Payload	*invalid*	*<valid>*

Pada *confussion matrix* 3.2 di atas, baris atau *row* diartikan sebagai referensi atau *actual label*, sementara kolom merepresentasikan data yang diujikan. Klasifikasi yang digunakan adalah *binary*, karena memprediksi *payload* dan *non-payload.* *Payload* diartikan sebagai kode yang digunakan untuk melakukan serangan. Pada *matrix* ini dikenal istilah TP, TN, FP, FN, yang akan dijelaskan sebagai berikut.

1) TP (*True Positive)*: adalah data positif yang diprekdisi benar. Artinya, data tersebut merupakan *payload* dan terdeteksi sebagai *payload*.
2) TN (*True Negative*): adalah data negatif yang diprediksi benar. Artinya, data tersebut bukan merupakan *payload* dan terdeteksi *non-payload*.
3) FP (*False Positive*): adalah data negatif yang diprediksi sebagai data positif. Artinya, data *non-payload* dideteksi sebagai *payload.*
4) FN (*False Negative*): adalah data positif yang diprediksi sebagai data negatif. Artinya, data *payload* namun dideteksi sebagai *non-payload*.

Secara singkat, berdasarkan penjelasan di atas, TP dan TN berarti bahwa model pembelajaran mesin yang dihasilkan telah mampu mendeteksi data serangan secara benar. Sementara itu, FP dan FN berarti bahwa model tersebut salah dalam

memprediksi atau mendeteksi suatu serangan. Berdasarkan pada kaidah tersebut, *precision, recall,* dan *f-measure* dihitung menggunakan rumus berikut ini.

$$\boldsymbol{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Rumus 3.1 Mencari Tingkat Akurasi Model Deteksi

Sesuai pada rumus 3.1 di atas, nilai akurasi didapat dari penjumlahan prediksi *payload* dan *non-payload* yang terdeteksi secara benar, kemudian dibagi dengan penjumlahan TP, TN, FP, dan FN. Akutasi berkaitan dengan seberapa akurat model deteksi melakukan klasifikasi secara benar. Sementara itu, formulasi *precision* yang diguna-kan pada penelitian ini adalah sebagai berikut.

$$\boldsymbol{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Rumus 3.2 Mencari Tingkat Presisi Model Deteksi

Selain akurasi, penelitian ini juga menghitung *precision* seperti yang digambarkan pada rumus 3.2. *Precision* diartikan sebagai tingkat keakuratan antara data yang diharapkan dengan hasil prediksi yang dihasilkan model. Artinya, pada formulasi ini, *precision* fokus pada penghitungan rasio hasil prediksi benar positif yang diberikan model, sementara tingkat akurasi pada prediksi keseluruhan. Selain *precision*, formula untuk mencari *recall* digambarkan pada rumus 3.3 di bawah.

$$\boldsymbol{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Rumus 3.3 Mencari Nilai *Recall* Model Deteksi

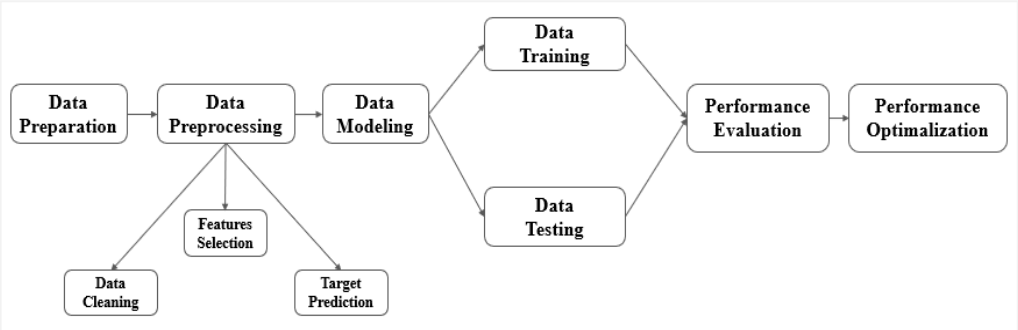
Sesuai pada rumus 3.3 di atas, *Recall* pada dasarnya mencari tingkat keberhasilan model dalam *refind* informasi, sehingga pada tahapan ini *recall* menghitung rasio prediksi benar dibandingkan keseluruhan data benar positif. Penhitu-

ngan selanjutnya yang dilakukan pada *confussion matrix* adalah F-*Measure* atau juga disebut sebagai F_1-Score. Rumus yang digunakan untuk mencari nilai F-*Measure* pada penelitian ini adalah sebagai berikut.

$$\boldsymbol{F - Measure} = 2 \text{ x } \frac{precison \text{ x } recall}{\text{precision} + \text{recall}}$$

Rumus 3.4 Mencari Nilai *F-Measure* pada Model Deteksi

Nilai F-*Measure* digunakan untuk melihat perbandingan rata-rata nilai presisi dan *recall*. Nilai ini digunakan apabila model kinerja atau performa deteksi data dengan jumlah data FN dan FP tidak mendekati (*non symetric*). Artinya, nilai F sebaiknya mendekati atau mencapai 1 sehingga model klasifikasi yang digunakan dikatakan memiliki tingkat presisi dan *recall* yang baik. Oleh karena itu, tahapan *confussion matrix* merupakan komponen penting pada tahap evaluasi dan optimalisasi performa model deteksi yang dihasilkan.

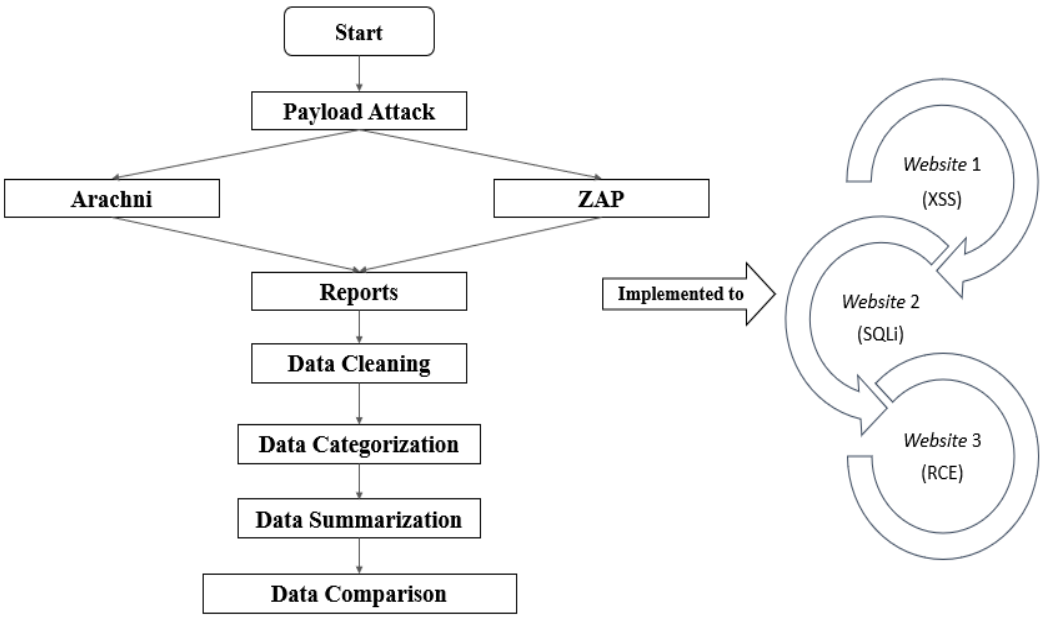


Gambar 3.2 Tahapan Alternatif Implementasi Metode Deteksi

Selain alur yang tertuang pada gambar 3.1, terdapat tahapan yang juga dilakukan peneliti dalam menguji metode deteksi, yaitu (1) *data preparation*; (2) *data training and testing*; dan (3) optimalisasi kinerja deteksi. Alur ini digunakan karena penelitian ini mengujikan tiga teknik serangan atau celah keamanan yang memiliki karakter yang berbeda-beda, yaitu XSS, SQLi, dan RCE. Oleh karena itu, setiap alur tidak dapat digeneralisasikan dan memerlukan alternatif alur.

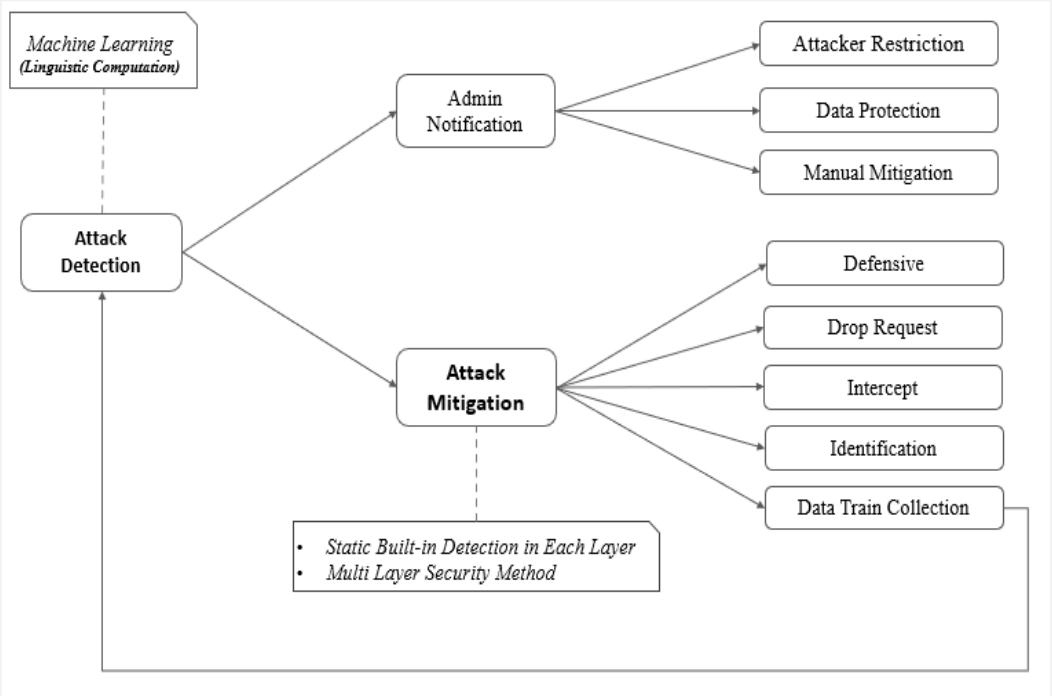
3.2.2. Skema Tahapan Metode Mitigasi

Berbeda dengan skema tahapan metode deteksi, tahapan pada implementasi metode mitigasi terdiri dari enam tahapan yang meliputi (1) uji coba penyerangan (*payload attack*); (2) menghasilkan laporan (*report)*; (3) pembersihan data laporan (*data cleaning*); (4) melakukan kategorisasi data temuan yang telah diungkapkan pada *report (data categorization*); (5) meringkas temuan; dan (6) melakukan komparasi. Terdapat dua komparasi yang digunakan yaitu (1) komparasi antara sebelum dan sesudah implementasi metode mitigasi dan (2) komparasi efektivitas antara metode mitigasi pada penelitian ini dan penelitian sejenis lainnya. Selain itu, tahapan ini dilakukan pada lebih dari satu *website*, karena metode mitigasi yang diajukan pada penelitian ini diarahkan pada penggunaan *multi-website* (gambar 3.3).



Gambar 3.3 Skenario Uji Coba Mitigasi Serangan (*Multi-Website*)

Tahapan penelitian juga akan berkaitan dengan skenario implementasi metode deteksi dan mitigasi. Secara umum, skenario implementasi atau integrasi metode deteksi dan mitigasi pada penelitian ini adalah sebagai berikut.



Gambar 3.4 Integrasi Metode Deteksi dan Mitigasi Serangan Siber

Seperti yang telihat pada gambar 3.4 di atas, metode deteksi dan mitigasi saling sinergi dalam mengatasi serangan siber. Sebagai respon metode deteksi, mesin deteksi dapat mengirimkan pesan kepada admin dan mengaktifkan metode mitigasi. Dalam kasus telah terjadi serangan, metode mitigasi serangan memiliki lima tindakan, yaitu (1) *devensive* (bertahan); (2) *drop request* (tidak melayani *request*); (3) *intercept* (menghambat konektivitas); (4) *identification* (mengidentifikasi data penyerang); dan (5) *data train collection* (mengumpulkan data untuk memperkuat deteksi). *Drop requests* (tindakan kedua) kebanyakan dilakukan melalui pengiriman sinyal HTTP 404 kepada penyerang. Pada tindakan identifikasi, mesin dapat menyimpan alamat IP, *user-agent*, dan data pendukung lainnya.

3.3. Metode Penelitian

Terdapat dua metode yang digunakan pada penelitian ini, yaitu (1) metode deteksi dan (2) metode mitigasi. Metode deteksi yang digunakan pada penelitian ini

adalah metode deteksi serangan berbasis *machine learning* atau secara lebih khusus disebut sebagai *NLP based machine learning*. Metode ini digunakan untuk mendeteksi pola-pola serangan yang berbasis teks. Dalam konteks *machine learning*, pola pembelajaran penelitian ini termasuk dalam *supervised learning*. Untuk melakukan deteksi, peneliti menyiapkan data *training* sehingga mesin dapat mempelajari pola setiap dataset, baik teks dan labelnya. Kemudian, tahapan yang dilakukan setelah *training* adalah *testing*. Tahapan ini dilakukan untuk menguji kualitas *data training.* Pada pengujian, tingkat akurasi metode deteksi dapat diketahui.

Sementara itu, metode mitigasi pada penelitian ini diartikan sebagai suatu metode untuk mengatasi atau setidak-tidaknya meminimalisasi dampak atau kerugian serangan dan keberhasilan serangan. Metode mitigasi penelitian ini menggunakan *multi-layer security* yang terdiri dari lima lapisan: (1) OWASP ModSecurity; (2) HTTP *Middleware*; (3) *Template Engine*; (4) *Data Sanitizer* dan (5) *Framework*/CMS *Built in Security*. Setiap lapisan di dalam *multi-layer security* bekerja pada cakupan masing-masing.

3.3.1. Metode Deteksi

Metode deteksi penelitian ini menggunakan *NLP based machine learning*. Pada pembelajaran mesin ini dilakukan pengenalan teks atau pola serangan menggunakan bantuan Python NLTK. Terdapat lima algoritma klasifikasi yang diujikan, yaitu Naïve Bayes, Logistic Classifier, Gradient Boosting, Support Vector Machine, dan K-Nearest Neighbor. Setelah setiap algoritma diujikan, metode deteksi akan memilih algoritma yang mampu mencapai tingkat akurasi tertinggi. Berikut adalah penjelasan terkait metode deteksi yang digunakan pada penelitian.

3.3.1.1. Model Deteksi

Seperti yang telah dijelaskan pada poin 3.3.1, penelitian ini mengujikan lima algoritma klasifikasi untuk mendapatkan metode deteksi dengan tingkat akurasi tertinggi. Pengujian kelima algoritma tersebut dilakukan agar metode deteksi benar-benar menggunakan algoritma klasifikasi deteksi yang sesuai dengan karakteristik teknik serangan yang dideteksi. Pemilihan kelima algoritma tersebut didasari oleh hasil penelitian sebelumnya. Berdasarkan penelitian metode deteksi serangan sebelumnya, kelima algoritma klasifikasi tersebut memiliki tingkat akurasi yang baik, terutama dalam mendeteksi atau mengenali pola-pola serangan [87], [88]. Berikut ini disampaikan rumus yang digunakan pada masing-masing algoritma.

$$\mathbf{P(H|X)} = \frac{\mathrm{P(X|H)}}{\mathrm{P(X)}} * \mathrm{P(H)}$$

Rumus 3.5 Klasifikasi Menggunakan *Naïve Bayes Classifier*

Keterangan:

X	= Data kelas	P(X)	= Probabilitas X
H	= Hipotesis kelas spesifik	P(H)	= Prob. hipotesis H
P(H|X)	= Prob. hipotesis H berdasarkan X		
P(X|H)	= Prob. X berdasarkan hipotesis H		

Berdasarkan rumus tersebut, H adalah label (*payload* atau *non-payload*) dan X adalah kelas atau fitur. Secara operasional, formulasi algoritma ini adalah.

$$\mathbf{P(label|feature)} = \frac{\mathrm{P}(label) * \mathrm{P}(features|label)}{\mathrm{P}(features)}$$

Rumus 3.6 Eksekusi *Naïve Bayes Classifier* pada NLTK *Step 1*

Algoritma ini akan membuat asumsi yang *naïve*, dimana semua fitur yang telah ditentukan dipandang sebagai sesuatu yang independen. Dalam konteks tersebut, masing-masing fitur dikaitkan atau dikorelasikan dengan label yang muncul.

$$\mathbf{P(label|feature)} = \frac{\mathrm{P}(label) * \ \mathrm{P}(\mathrm{f1}|label) * \ldots * \mathrm{P}(\mathrm{fn}|label)}{\mathrm{P}(features)}$$

Rumus 3.7 Eksekusi *Naïve Bayes Classifier* pada NLTK *Step 2*

Algoritma kemudian menghitung P(*feature*) secara implisit, sehingga algoritma ini akan menghitung pembilang untuk semua label (*payload* atau *non-payload*), dan melakukan normalisasi sehingga didapatkan persamaan seperti di bawah ini.

$$\mathbf{P(label|feature)} = \frac{\mathrm{P(label)} * \ \mathrm{P(f1|label)} * \ldots * \mathrm{P(fn|label)}}{\mathrm{P(features)}}$$

Rumus 3.8 Eksekusi *Naïve Bayes Classifier* pada NLTK *Step 3*

Selain Naïve Bayes, algoritma kedua yang diujikan pada penelitian ini adalah *Logistic Regression*. Adapun rumus yang digunakan adalah sebagai berikut.

$$\mathbf{Y} = \frac{\exp\ (\mathrm{B_0} + \mathrm{B_1X})}{(1 + \exp(\mathrm{B_0} + \mathrm{B_1X}))}$$

Rumus 3.9 Klasifikasi Menggunakan *Logistic Regresion Classifier*

Keterangan:

Y = luaran yang diprediksi B_0 = Bias

B_1 = Koefisien nilai input X

Klasifikasi regresi logistik memodelkan variabel target diskrit sebagai fungsi beberapa variabel fitur. Klasifikasi ini menggunakan variabel **y** diskrit. Untuk setiap observasi, probabilitas yang menyatakan bahwa **y = 1** dimodelkan sebagai fungsi logistik atas kombinasi linear dari nilai-nilai fitur. Sekumpulan fitur $\mathbf{x_i}$ akan diikuti oleh sebuah label $\mathbf{y_i}$. Artinya, regresi logistik akan menginterpretasikan probabilitas bahwa label berada dalam satu kelas sebagai fungsi logistik dari kombinasi fitur. Pada pustaka Turicreate, proses ini tertuang pada persamaan berikut.

$$\int \emptyset = \mathrm{p}(yi = 1|x) = \frac{1}{1 + \exp\ (-\emptyset^T x)}$$

Rumus 3.10 Klasifikasi *Logistic Regresion Classifier* Turicreate

Gradient Boosting memiliki beberapa jenis basis. Penelitian ini menggunakan *Gradient Boosting* berbasis *decision tree*. Algoritma ini memproses dataset secara sekuensial, karena menambahkan prediktor sebelumnya ke dalam data *ensemble*. Dengan pola kerja seperti ini, kesalahan-kesalahan prediksi sebelumnya diperbaiki. Dalam hal ini, *ensemble* diartikan sebagai daftar keputusan prediksi yang dihasilkan *machine learning*. Setiap baris data diprediksi oleh kelas dominan.

$$-\log L1 = -\sum_{i=1}^{N} y_i \log(odds) + \log(1 + e^{\log(odds)})$$

Rumus 3.11 Rumus *Gradient Boosting*

Support Vector Machine adalah algoritma yang digunakan untuk menentukkan *decision boundary*. Klasifikasi algoritma ini ditentukan oleh *decision boundary* tersebut. *Support Vector Machine* memanfaatkan model linear sebagai *decision boundary*. Adapun bentuk umum proses ini adalah sebagai berikut.

$$\boldsymbol{y(x) = w^T \emptyset(x) + b}$$

Rumus 3.12 *Decision Boundary* Menggunakan Model Linear

Berdasarkan rumus 3.12 tersebut, w diartikan sebagai parameter bobot, ∅(x) adalah fungsi basis, dan *b* adalah bias. Bentuk model linear yang paling sederhana untuk *decision boundary* adalah $\mathbf{y(x) = w^t x + w_0}$, dimana x merupakan vektor dan **w** merupakan vektor bobot, serta $\mathbf{w_0}$ adalah bias. Dengan demikian, *decision boundary* didefinisikan sebagai y(x) = 0 yang merupakan *hyperplane* berdimensi (D-1). Sementara itu, algoritma terakhir yang diujikan adalah K-Nearest Neighbor. Algoritma ini menggunakan metode *Euclidean Distance* untuk melihat jarak terde-

kat prediksi pada label-label terdefinisi. Secara umum, formula *Euclidean Distance* tersebut dapat digambarkan sebagai berikut.

$$\text{dis}\,(\mathrm{x}_1, \mathrm{x}_2) = \sqrt{\sum_{i=0}^{n} (x_{1i} - x_{2i})2}$$

Rumus 3.13 Formula *Euclidean Distance* 1-*Dimensional Space*

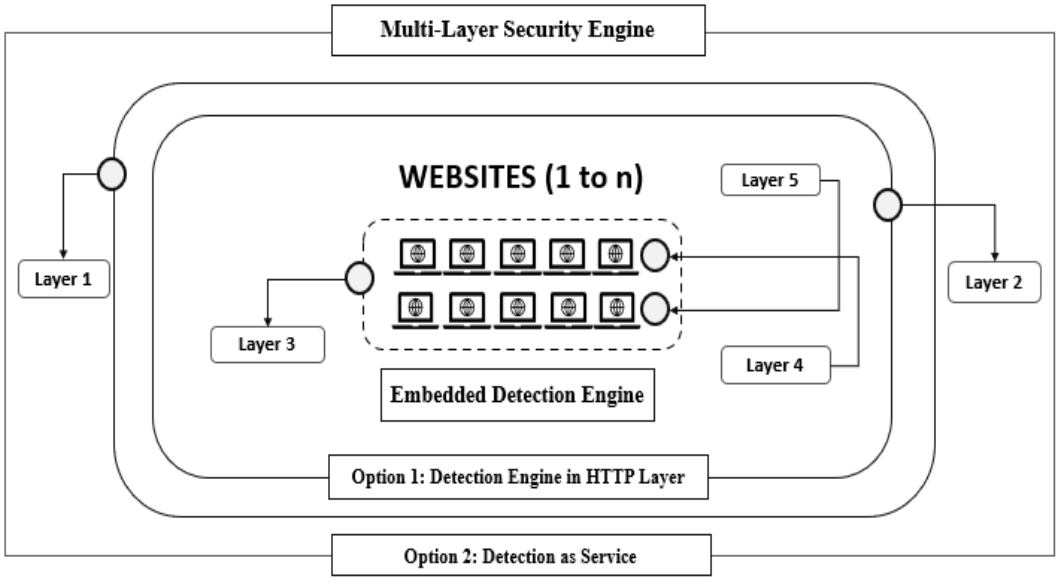
Sementara itu, jika lebih dari 1 *dimensional space*, formula yang digunakan adalah seperti berikut ini.

$$dis = \sqrt{\sum_{i=0}^{n} (x_{1i} - x_{2i})2 + (\mathrm{y}_{1i} - \mathrm{y}_{2i})2 + \cdots}$$

Rumus 3.14 Formula *Euclidean Distance* Lebih Dari 1-*Dimensional Space*

3.3.1.2. *Detection Placement* (Penempatan Deteksi)

Penempatan metode deteksi digambarkan pada gambar 3.6 di bawah ini.



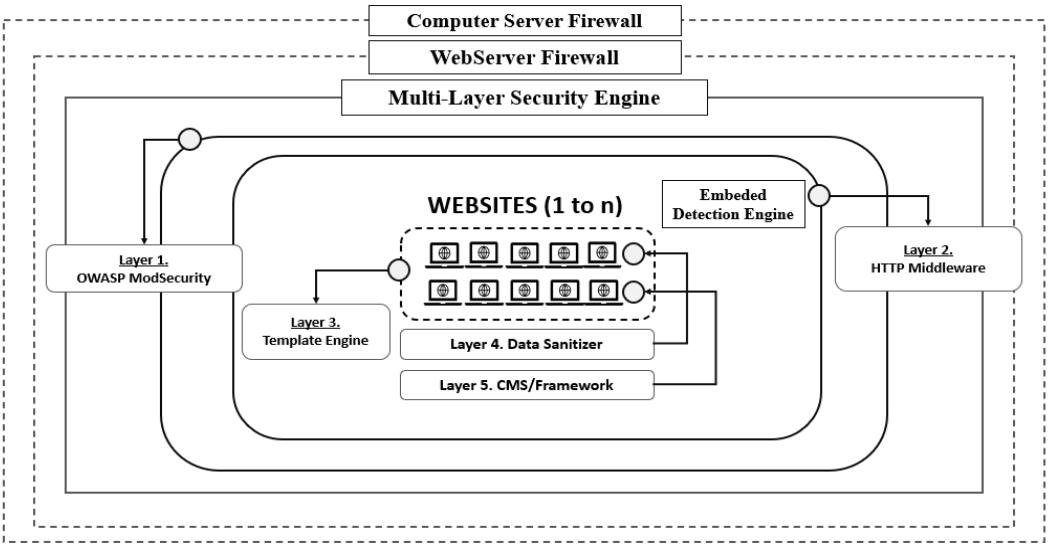
Gambar 3.5 Mekanisme Penempatan Mesin Deteksi

Dalam proses implementasi, metode deteksi dapat diletakan pada 3 alternatif peletakan, yaitu (1) sebagai *service* pada tingkat *web server*; (2) lapisan HTTP *mi-*

ddleware; dan (3) pada tingkatan aplikasi. Pada penelitian ini, mesin deteksi diletakkan pada tingkatan aplikasi, yaitu HTTP Middleware. Hal ini dilakukan karena eksekusi mesin *machine learning* yang digunakan berada pada tingkatan aplikasi.

3.3.2. Metode Mitigasi

Metode mitigasi yang digunakan adalah *multi-layer security* yang terdiri dari lima lapisan keamanan. Tiap lapisan memiliki peran dan *workspace* secara tersendiri. Dalam kaitan dengan penelitian sebelumnya, metode mitigasi ini telah mempertimbangkan kelemahan lapisan pertama yang masih terjadi pada penelitian Ayunda *et al* [21], Zavarsky [22], dan Akbar *et al* [11]. Peneliti memperkuat keamanan melalui penambahan lapisan kedua sampai dengan kelima. Secara garis besar, metode ini dapat digambarkan sebagai berikut.



Gambar 3.6 Mekanisme *Multi-Layer Security*

Penjelasan terkait masing-masing lapisan telah diungkapkan secara tersendiri pada Bab 2. Terkait dengan model skenario implementasi metode *multi-layer security* yang dijelaskan pada tabel 2.10, urutan tiap lapisan dapat disesuaikan dengan arsitektur atau karakteristik *server* atau aplikasi yang digunakan. Metode mitigasi

diujikan oleh dua aplikasi penyerang yaitu Arachni dan ZAP. Pengujian tersebut mengukur tingkat efektivitas *multi-layer security* terhadap serangan. Peneliti juga melakukan komparasi terhadap metode mitigasi yang digunakan sebelumnya.

3.3.2.1. *OWASP Mod Security Firewall*

OWASP ModSecurity bekerja berdasarkan *core rule set* atau seperangkat aturan inti yang diaktifkan pada aplikasi. Pada penelitian ini, peneliti menggunakan *core rules set* standar, tanpa melakukan perubahan atau penambahan *rules* apapun. Daftar *rules* yang digunakan pada penelitian ini adalah *core rule set* serangan XSS, SQLi, dan RCE dengan nama berkas sebagai berikut.

1) REQUEST-941-APPLICATION-ATTACK-XSS.conf
2) REQUEST-932-APPLICATION-ATTACK-RCE.conf
3) REQUEST-932-APPLICATION-ATTACK-RCE.conf

3.3.2.2. *HTTP Middleware*

Seperti yang telah diungkapkan pada poin 3.4.2.2, fitur keamanan pada *HTTP middleware* tidak akan diaktifkan untuk melihat efektivitas *multi-layer security* sebelum dan sesudah diimplementasikan. Setelah diaktifkan, *layer* ini akan melindungi lalu lintas *website*. Logika yang digunakan untuk memitigasi serangan secara sederhana dapat diungkapkan melalui baris kode Python sebagai berikut.

```
1.  import os
2.  from django.http import Http404
3.  from django.conf import settings
4.  from aurora.backend.library.mlsm.xss.classifier import XSSClassifier
5.  from aurora.backend.library.mlsm.rce.classifier import RCEClassifier
6.  from aurora.backend.library.mlsm.sqli.classifier import SQLiClassifier
7.
8.  class MSLMMiddleware(object):
9.
10.     def __init__(self, get_response):
```

```
        self.get_response = get_response


    def __call__(self, request):
        response = self.get_response(request)
        return self.process_request(request, response)


    def process_request(self, request, response):
        GET_DATA  = request.get_full_path()
        POST_DATA = request.POST
        if settings.LAYER_2_MIDLEWARE == True:
            # Cross Site Scripting Detection
            # Remote Code Execution Detection
            # SQL Injection Detection
            classifiers = {
                'xss'   : XSSClassifier(),
                'rce'   : RCEClassifier(),
                'sqli'  : SQLiClassifier()
            }
            for data in [GET_DATA, POST_DATA]:
                for name, classifier in classifiers.items():
                    label = classifier.classify(data)
                    if label == 'payload':
                        raise Http404()
        return response

```

Gambar 3.7. Kode Python untuk Mengaktifkan Metode Deteksi pada *Middleware*

3.3.2.3. *Template Engine*

Lapisan *template engine* digunakan untuk mengatasi serangan pada sisi *front-end* seperti XSS. Pada dasarnya lapisan ini akan melakukan *escape* pada karakter-karakter tertentu agar tidak dapat dieksekusi oleh *browser*. Adapun aturan *escape* yang digunakan adalah sebagai berikut.

Tabel 3.3 Daftar Karakter yang di-*Escape* oleh *Template Engine*

No	Karakter HTML	*Escaped*
1	&	&amp;
2	<	<
3	>	>
4	“	"
5	‘	&#39

3.3.2.4. *Data Sanitizer*

Apabila lapisan pertama sampai ketiga telah memiliki sanitasi data yang baik, lapisan ini berperan menjadi lapisan pelengkap. Namun, apabila lapisan sebelumnya belum memiliki fitur *data sanitizer*, maka lapisan ini menjadi lapisan yang sangat penting untuk diaktifkan. Lapisan ini akan melakukan sanitasi pada beberapa komponen atau elemen aplikasi, yaitu kueri SQL, parameter pada *request*, dan isian pada formulir yang dikirimkan oleh *browser*.

3.3.2.5. *CMS/Framework Built-in Security*

Web framework yang digunakan pada penelitian ini adalah Django, suatu *web framework* berbasis Python. Versi Django yang digunakan adalah versi 4.0. Django dipilih berdasarkan tiga pertimbangan, yaitu (1) memiliki fitur HTTP *middleware;* (2) memiliki fitur *data sanitizer*; dan (3) memiliki fitur *template engine*. Melalui fitur-fitur tersebut pengujian keamanan dapat dilakukan secara fleksibel. Pada saat pengujian, konfigurasi keamanan tidak diaktifkan atau dengan konfigurasi berikut.

Auto Escape	: *Off*
HTTP Middleware	: *Inactive*
Data sanitizer	: *Unused*

3.3.2.6. Komparasi Data Mitigasi

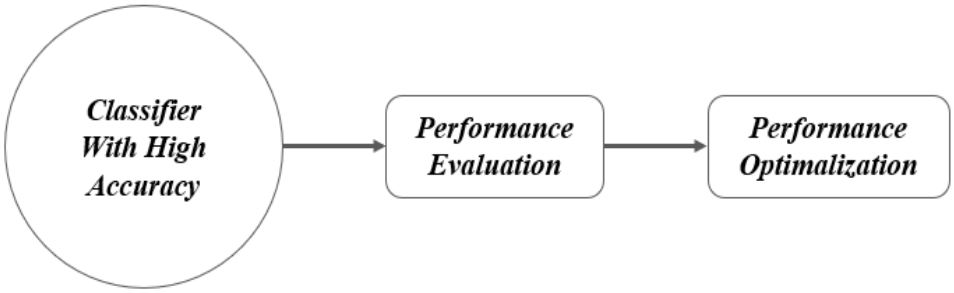
Untuk menentukan apakah metode mitigasi dapat dikatakan efektif atau tidak efektif, peneliti menggunakan komparasi kinerja mitigasi. Terdapat dua pola komparasi yang digunakan pada penelitian ini. Pertama, komparasi yang dilakukan pada saat sebelum dan sesudah menerapkan metode mitigasi. Peneliti mengukur berapa persen persentase efektivitas metode mitigasi dalam mengatasi serangan siber keti-

ka telah diaktifkan. Kedua, komparasi dengan penelitian relevan sebelumnya. Hal ini dilakukan untuk mendapatkan hasil yang lebih kuat, yaitu hasil yang membuktikan bahwa metode mitigasi pada penelitian ini lebih efektif dalam mengatasi serangan siber yang diujikan, dibandingkan metode yang digunakan sebelumnya.

3.4. Evaluasi Metode

3.4.1. Evaluasi Metode Deteksi

Selain menggunakan *confussion matrix*, untuk semakin meningkatkan kualitas pengujian algoritma dalam mendeteksi pola atau vektor serangan, terdapat dua tindakan yang dilakukan peneliti, yaitu *performance optimalization* dan *performance testing.* Pada tahapan *performance optimalization*, peneliti menguji kelima algoritma dengan *margin*, aturan korpus, dan parameter konfigurasi yang berbeda-beda sampai diperoleh tingkat akurasi deteksi yang tinggi. Sementara itu, pada tahapan *performance testing*, algoritma deteksi terpilih dengan tingkat akurasi tertinggi diujikan kualitas deteksinya pada dataset yang berbeda.



Gambar 3.8. Kode Python untuk Mengaktifkan Metode Deteksi pada *Middleware*

Berbeda dengan kebanyakan pola pembelajaran mesin umumnya yang hanya berpijak pada tingkat akurasi pada proses *data training* dan *data testing,* penelitian ini menerapkan *performance testing* untuk menguji apakah metode deteksi benar-benar mampu mendeteksi vektor serangan yang *"asing"* dari dataset yang diguna-

kan pada proses *training* dan *testing*. Dataset yang digunakan pada tahapan ini disebut sebagai *challenge dataset*. Pada tahapan ini, peneliti juga mengukur tingkat akurasi, *time of process*, total prediksi valid, dan total prediksi tidak valid. Jika metode deteksi masih menyisakan vektor serangan yang dideteksi secara salah, vektor tersebut akan kembali diproses oleh metode mitigasi lapisan berikutnya.

3.4.2. Evaluasi Metode Mitigasi

Dalam metode mitigasi, peneliti akan melakukan dua studi komparasi. Studi komparasi yang pertama berupa komparasi terhadap *website* yang belum dan sudah diimplementasikan metode mitigasi. Artinya, peneliti mengukur dan membandingkan jumlah serangan yang dapat dilakukan sebelum dan sesudah implementasi metode mitigasi. Lalu, studi komparasi yang kedua adalah membandingkan hasil metode mitigasi penelitian ini dengan metode mitigasi yang diusulkan atau diimplementaskan pada penelitian sebelumnya. Hal ini dilakukan untuk melihat apakah metode mitigasi ini lebih efektif dalam mengatasi serangan siber sejenis atau tidak. Berdasarkan perbandingan metode tersebut, keunggulan dari masing-masing metode dapat diukur secara jelas dan terukur.

3.5. *Timeline* Penelitian

Penyusunan *timeline* atau jadwal pada penelitian ini telah mempertimbankan berbagai aspek, terutama terkait tingkat kesulitan penelitian ini. Tingkat kesulitan penelitian memengaruhi rentang waktu pelaksanaan prosedur di dalamnya. Peneliti menyadari bahwa terdapat setidak-tidaknya tiga tantangan dalam penelitian ini, yaitu (1) penggunaan dua metode (deteksi dan mitigasi); (2) penerapan pada *multi-*

website, bukan *single website*; dan (3) uji coba dilaksanakan untuk deteksi dan mitigasi *multi-attack* yaitu XSS, *SQL Injection*, dan RCE. Berdasarkan tingkat kesulitan tersebut, *timeline* penelitian adalah sebagai berikut.

Bulan Pertama (Metode Deteksi):

No	Nama Kegiatan	Pelaksanaan Minggu ke -			
		01	02	03	04
1	Mempersiapkan data-data penelitian				
2	Mempersiapkan *research environment*				
3	Implementasi metode deteksi XSS				
4	Implementasi metode deteksi SQL *Injection*				
5	Implementasi metode deteksi CSRF				
6	Implementasi metode deteksi IDOR				
7	Implementasi metode deteksi RCE				

Bulan Kedua (Metode Mitigasi):

No	Nama Kegiatan	Pelaksanaan Minggu ke -			
		01	02	03	04
1	Mempersiapkan data-data penelitian				
2	Mempersiapkan *research environment*				
3	Implementasi metode mitigasi XSS				
4	Implementasi metode mitigasi SQL *Injection*				
5	Implementasi metode mitigasi CSRF				
6	Implementasi metode mitigasi IDOR				
7	Implementasi metode mitigasi RCE				

Bulan Ketiga (Evaluasi, Optimalisasi, dan Integrasi):

No	Nama Kegiatan	Pelaksanaan Minggu ke -			
		01	02	03	04
1	Melakukan evaluasi pada kedua metode				
2	Melakukan optimalisasi pada kedua metode				
3	Melakukan Integrasi				