# BAB II. LANDASAN TEORI

## 2.1. Penelitian Sebelumnya

Seperti yang telah disebutkan sebelumnya, beberapa peneliti pernah mengawali pelaksanaan penelitian yang sebidang dengan penelitian ini. Namun, penelitian tersebut dilakukan secara terpisah. Dalam hal ini, penelitian terkait metode deteksi berbasis *machine learning* dan metode mitigasi berbasis *multi-layer security* dilakukan secara parsial. Penelitian sebelumnya secara umum bertujuan untuk (1) mengukur akurasi metode deteksi; (2) mengukur efektivitas mitigasi; dan (3) memberikan gambaran terkait implementasi metode deteksi dan mitigasi yang dikembangkan. Berikut ini adalah rangkuman dari pelaksanaan penelitian sebelumnya.

**Tabel 2.1 Rangkuman Pelaksanaan Penelitian Sebelumnya**

| No | Peneliti | Judul dan Tahun Penelitian | Metode dan Hasil Penelitian | Kelebihan dan Kelemahan |
|---|---|---|---|---|
| **Terkait pengujian terhadap tingkat akurasi metode deteksi serangan siber** | | | | |
| 1 | Prasetio *et al* [16] | **Judul:**<br>*Cross-Site Scripting Attack Detection Using Machine Learning With Hybrid Features* | **Metode:**<br>Peneliti menggunakan *hybrid features* dengan metode *Logistic Classifier* dan didukung oleh analisis komputasi | **Kelebihan:**<br>Metode yang dipergunakan peneliti dapat mendeteksi serangan XSS secara akurat. |

| No | Peneliti | Judul dan Tahun Penelitian | Metode dan Hasil Penelitian | Kelebihan dan Kelemahan |
|---|---|---|---|---|
| | | **Tahun Publikasi**: 2021 | berbasis linguistik untuk mendeteksi serangan XSS.<br><br>**Hasil:**<br>Penggunaan *hybrid features* dan analisis komputasi berbasis linguistik dapat semakin meningkatkan akurasi dalam mendeteksi serangan XSS. | **Kelemahan:**<br>Teknik serangan yang dikaji dan diteliti hanya serangan XSS dan hanya sebatas untuk deteksi, tidak ada metode mitigasi. Selain itu, model yang dihasilkan juga belum cukup efektif melakukan prediksi serangan XSS. |
| 2 | Khalid *et al* [18] | **Judul:**<br>*Predicting Web Vulnerabilities in Web Applications Based on Machine Learning*<br><br>**Tahun Publikasi:** 2018 | **Metode:**<br>Penelitian ini menggunakan suatu metode yang bernama NMPREDICTOR, yaitu kerangka kerja untuk mengatasi celah keamanan berdasarkan *machine learning*. Metode ini digunakan untuk mendeteksi apakah kode sumber aman atau tidak. | **Kelebihan:**<br>Metode *machine learning* dapat digunakan untuk mendeteksi serangan siber,<br><br>**Kelemahan:**<br>a) tidak merinci teknik serangan apa saja yang berhasil dideteksi;<br>b) metode tidak diuji menggunakan ban-tuan aplikasi penyerang otomatis; |

| No | Peneliti | Judul dan Tahun Penelitian | Metode dan Hasil Penelitian | Kelebihan dan Kelemahan |
|---|---|---|---|---|
| | | | **Hasil:**<br>NMPREDICTOR sukses dalam melakukan deteksi celah celah keamanan dengan F-*Measure* yang lebih baik dari penelitian yang sebelumnya | |
| 3 | Jemal [20] | **Judul:**<br>*SQL Injection Attack Detection and Prevention Techniques Using Machine Learning*<br><br>**Tahun Publikasi:** 2020 | **Metode:**<br>Penelitian ini mengkaji tentang akurasi deteksi serangan SQL *Injection* menggunakan beberapa metode klasifikasi berbasis *machine learning*,<br><br>**Hasil:**<br>Berdasarkan komparasi yang telah dilakukan, metode klasifikasi yang direkomendasikan adalah Naïve Bayes | **Kelebihan:**<br>Peneliti melakukan komparasi yang cukup memadai menggunakan beberapa metode klasifikasi, termasuk dataset yang digunakan dan tingkat akurasinya.<br><br>**Kelemahan:**<br>Hanya fokus pada bagaimana mendeteksi, tanpa ada upaya mitigasi dan serangan juga hanya difokuskan pada SQL *Injection*. |
| | | | | |

| No | Peneliti | Judul dan Tahun Penelitian | Metode dan Hasil Penelitian | Kelebihan dan Kelemahan |
|---|---|---|---|---|
| **Terkait pengujian terhadap tingkat efektivitas metode mitigasi serangan siber** | | | | |
| 4 | Ayunda *et al* [21] | **Judul:**<br>*Implementation and Analysis ModSecurity on Web-Based Application with OWASP Standards*<br><br>**Tahun Publikasi:** 2021 | **Metode:**<br>Penelitian menggunakan metode eksperimen. Tingkat keamanan OWASP ModSecurity diujikan dengan lima tahapan untuk melihat efektivitasnya.<br><br>**Hasil:**<br>Tingkat efektivitas *OWASP ModSecurity Firewall* mencapai 66% dan melindungi dua celah keamanan tingkat risiko tinggi. | **Kelebihan:**<br>Mengukur efektivitas *OWASP ModSecurity* untuk mengatasi *brute-force*<br><br>**Kelemahan:**<br>a) Masih terdapat 44% celah keamanan yang belum dapat diatasi;<br>b) Lebih fokus pada serangan SQL *Injection* dan *Session Hijacking* |
| 5 | Pavol Zavarsky dan Sergey Butakov [22] | **Judul:**<br>*Experimental Study of Mod Security Web Application Firewalls*<br><br>**Tahun Publikasi:** 2020 | **Metode:**<br>Pengujian efektivitas *OWASP ModSecurity Firewall* dilakukan melalui kegiatan *planning*, *execution*, dan *analysis* menggunakan tiga aplikasi, yaitu Metasploit, NMAP, THC Hydra, Wee-vely, dan Siege pada Kali Linux. | **Kelebihan:**<br>OWASP ModSecurity Firewall dapat mengatasi 8/9 teknik serangan dan mampu mendeteksi serangan DDoS<br><br>**Kelemahan:**<br>a) Metode yang diusulkan gagal dalam |

| No | Peneliti | Judul dan Tahun Penelitian | Metode dan Hasil Penelitian | Kelebihan dan Kelemahan |
|---|---|---|---|---|
| | | | **Hasil:**<br>OWASP ModSecurity Firewall dapat mengatasi 8 dari 9 serangan. Namun, pada serangan DDoS, performa server semakin melambat dan *semi-down*. | mengatasi serangan *stored* XSS;<br>b) Peneliti menyatakan bahwa masih ada beberapa celah keamanan yang dapat dieksploitasi oleh penyerang; |
| 6 | Akbar et al [11] | **Judul:**<br>*SQL Injection and Cross Site Scripting Prevention Using OWASP Web Application Firewall*<br><br>**Tahun Publikasi:** 2018 | **Metode:**<br>Peneliti mengukur efektivitas penggunaan OWASP ModSecurity Firewall untuk mengatasi serangan SQL *Injection* dan XSS. Pengujian dilakukan dengan aplikasi SQLMap, BeEF, dan XSSer pada 3 sistem operasi.<br><br>**Hasil:**<br>OWASP ModSecurity berhasil mengatasi 100% serangan SQL *Injection*, namun gagal mengatasi *stored XSS*. | **Kelebihan:**<br>Penelitian ini menggunakan tiga aplikasi dan diujikan pada tiga sistem operasi untuk memastikan OWASP ModSecurity Firewall dapat mengatasi serangan XSS dan SQL *Injection*. Dari pengujian tersebut, metode yang diusulkan berhasil mengatasi 100% serangan SQL *Injection*.<br><br>**Kelemahan:**<br>a) Metode yang diusulkan gagal dalam mengatasi serangan *stored* XSS; |

| No | Peneliti | Judul dan Tahun Penelitian | Metode dan Hasil Penelitian | Kelebihan dan Kelemahan |
|---|---|---|---|---|
| | | | | b) Komunitas peretas pada situs https://waf-bypass.com/ membuktikan bahwa SQL *Injection* dapat dilakukan pada server yang menggunakan OWASP *ModSecurity Firewall*;<br>c) Hanya fokus ke teknik serangan XSS dan *SQL Injection* |
| 7 | Grunwaldt [23] | **Judul:**<br>*A Comparison of Modern Backend Frameworks Protections against Common Web Vulnerabilities*<br><br>**Tahun Publikasi:** 2019 | **Metode:**<br>Peneliti menggunakan metode dokumentasi untuk mengulas fitur atau tingkat keamanan *web framework*, yaitu Django, Spring, Laravel, dan Express. Peneliti mengkaji beberapa komponen atau fitur-fitur keamanan seperti injeksi, autentifikasi, XSS, *Sensitive Data Exposure*, dan seterusnya. | **Kelebihan:**<br>Kajian disinkronisasikan dengan celah keamanan yang biasa ditemukan pada aplikasi berbasis *website*.<br><br>**Kelemahan:**<br>Belum dilakukan eksperimen lebih jauh dan mendalam terkait tingkat keamanan. |

| No | Peneliti | Judul dan Tahun Penelitian | Metode dan Hasil Penelitian | Kelebihan dan Kelemahan |
|---|---|---|---|---|
| | | | **Hasil:**<br>Menurut komparasi tersebut, Django dan Laravel memiliki proteksi terbaik sehingga memiliki potensi lebih besar dalam mengatasi celah keamanan yang dapat dieksploitasi penyerang. | |
| 8 | Kaluza *et al* [24] | **Judul:**<br>*A Comparison of Backend Framework for Web Application Development*<br><br>**Tahun Publikasi:** 2019 | **Metode:**<br>Peneliti menggunakan metode komparasi untuk mengukur dan melihat bagaimana fitur dan tingkat keamanan pada masing-masing *web framework*.<br><br>**Hasil:**<br>Peneliti merekomendasikan penggunaan beberapa teknologi *web framework* karena terbilang memiliki tingkat keamanan yang baik, yaitu Django, Laravel, Ruby On Rails, dan Spring | **Kelebihan:**<br>Peneliti menggunakan metode komparasi dan analisis yang cukup komprehensif untuk membandingkan kelebihan *web framework*.<br><br>**Kelemahan:**<br>Belum dilakukan eksperimen lebih jauh dan mendalam terkait tingkat keamanan. |

Melanjutkan penjelasan terkait penelitian sebelumnya, yang fokus pada bidang *cyber attack and security*, dalam kaitannya dengan penelitian ini, berikut ini disajikan agregrasi penelitian terkait teknik/metode serangan, metode deteksi/mitigasi, dataset, dan tahun penelitian.

**Tabel 2.2 Agregrasi Metode, Dataset, dan Penelitian *Cross Site Scripting* (XSS)**

| No | Peneliti dan Tahun | Judul | Metode | Data | Hasil |
|---|---|---|---|---|---|
| 1 | Mokbal *et al* [25] Tahun 2019 | *MLPXSS: An Integrated XSS-Based Attack Detection Scheme in Web Applications Using Multilayer Perceptron Technique* | ANN technique (MLPXSS) | *Raw data from XSSed and Open Bug Bount with 38,569 malicious samples* | Skema atau metode yang diusulkan mengungguli teknik deteksi yang sebelumnya |
| 2 | Prasetio *et al* [16] Tahun 2021 | *Cross-site Scripting Attack Detection Using Machine Learning With Hybrid Features* | *Hybrid features, Logistic Regression* | Kaggle and Github | Deteksi 99.83% menggu-nakan model trigram 99.87% menggunakan *hybrid* |
| 3 | Habibi *et al* Tahun 2020 | *XSS Attack Detection With Machine Learning and n-Gram Methods* | *SVM, KNN, and Naïve Bayes (NB)* | Kaggle | Tingkat akurasi deteksi sera-ngan XSS mencapai 98% |
| 4 | Buz *et al* [26] Tahun 2021 | *A Hybrid Machine Learning Model to Detect Reflected XSS Attack* | *N-Gram, Hybrid, Convolutional Neural Network* | Kaggle | Tingkat akurasi deteksi sera-ngan XSS mencapai 99% |

| No | Peneliti dan Tahun | Judul | Metode | Data | Hasil |
|---|---|---|---|---|---|
| 5 | Yulianingsih [13] Tahun 2017 | Melindungi Aplikasi dari Serangan CrossSite Scripting (XSS) Dengan Metode Metacharacter | Metacharacter | Aplikasi | Metode *metacharacter* mampu menutup serangan XSS |
| 6 | Grunwaldt et al [24] Tahun 2019 | *A Comparison of Modern Backend Frameworks Protections against Common Web Vulnerabilities* | *Comparison* (Django and Laravel) | - | *Django is better* |
| 7 | Marashdih [27] Tahun 2019 | *Cross Site Scripting: Removing Approaches in Web Application* | OWASP-ESAPI | - | *Python and PHP* |
| 8 | Gupta [28] Tahun 2019 | *Cross-Site Scripting (XSS) Abuse and Defense: Exploitation on Several Testing Bed Environments and Its Defense* | *Filtering and Sanitation* | *XSS cheat sheet website* | Metode *filtering* dan *sanitation* dapat mengatasi serangan XSS secara efektif. |
| 9 | Putra *et al* [12] Tahun 2019 | Meningkatkan Keamanan Web Menggunakan Algoritma *Advanced Encryption Standard* (AES) terhadap Seragan *Cross Site Scripting* | Algoritma AES | *Doing data mining to E-learning website* | Sebanyak dua dari dua celah keamanan XSS dapat teratasi |

| No | Peneliti dan Tahun | Judul | Metode | Data | Hasil |
|---|---|---|---|---|---|
| 10 | Kurniawan [14]<br>Tahun 2020 | Optimasi Metode Otomatisasi Penghilangan Kerentanan Terhadap Serangan XSS Pada Aplikasi Web | OWASP-ESAPI | *Data* bentuk *Google Application Security* | Terdapat peningkatan efisiensi deteksi 61,90% dari penelitian sebelumnya |

**Tabel 2.3 Agregrasi Metode, Dataset, dan Penelitian *SQL Injection (SQLi)***

| No | Peneliti dan Tahun | Judul | Metode | Data | Hasil |
|---|---|---|---|---|---|
| 1 | Jemal [20]<br>Tahun 2020 | *SQL Injection Attack Detection and Prevention Techniques Using Machine Learning (Review)* | ▪ *Naïve Bayes*<br>▪ *Backpropagation NN*<br>▪ *Neural Network Based Model*<br>▪ *Decision Tree*<br>▪ *SVM*<br>▪ *KNN* | *Various Datset* | ▪ *Accuracy = 97,6%*<br>▪ *Accuracy = 96.8%*<br>▪ *Accuracy = 95%*<br>▪ *Accuracy = 83.7%*<br>▪ *Accuracy = 96.23%*<br>▪ *Accuracy = 97%* |
| 2 | Hasan *et al* [29]<br>Tahun *2019* | *Detection of SQL Injection Attacks: A Machine Learning Approach* | *Heuristic Algorithm* | Kaggle | Tingkat akurasi deteksi mencapai 93.8% |
| 3 | Kranthikumar [30]<br>Tahun 2020 | *SQL injection detection using REGEX classifier* | *REGEX Classifier (SVM, Naïve Bayes, Gradient Boosting)* | Github | Tingkat akurasi deteksi mencapai 97% |

| No | Peneliti dan Tahun | Judul | Metode | Data | Hasil |
|---|---|---|---|---|---|
| 4 | Akbar *et al* [11] Tahun 2018 | *SQL Injection and Cross Site Scripting Prevention Using OWASP Web Application Firewall* | OWASP WAF | Apache Web-server on CPanel | Melakukan deteksi dan pertahanan 100% dari serangan SQL *Injection* |
| 5 | Singh *et al* [31] Tahun 2015 | *Detection of SQL Injection and XSS Vulnerability in Web Application* | *Crawling and analizer, in form of tool or desktop application* | *Crawling* pada Aplikasi | Mampu melakukan deteksi serangan SQLi secara efektif |

**Tabel 2.4 Agregrasi Metode, Dataset, dan Penelitian *Remote Code Execution (RCE)***

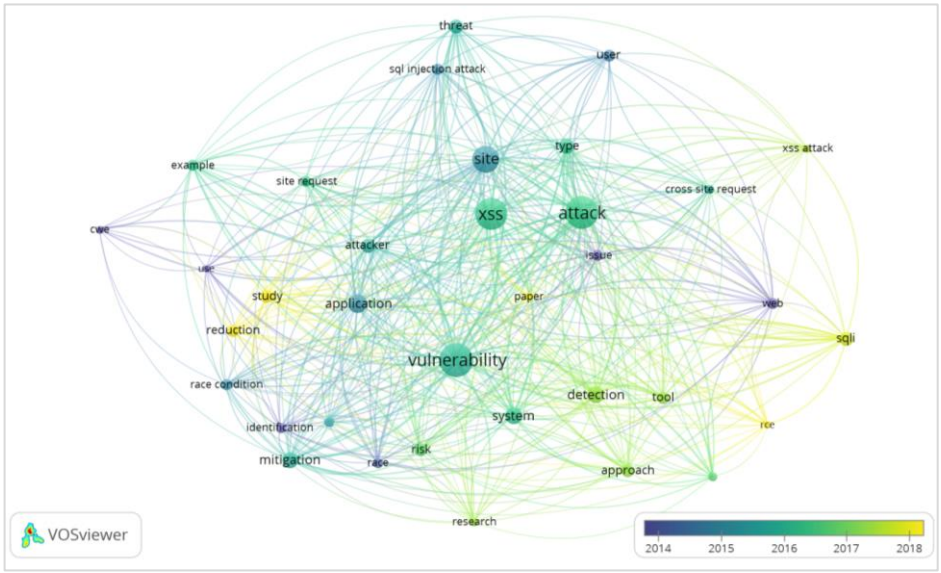
| No | Peneliti dan Tahun | Judul | Metode | Data | Hasil |
|---|---|---|---|---|---|
| 1 | Zhao *et al* [32] Tahun 2021 | *Cefuzz: An Directed Fuzzing Framework for PHP RCE Vulnerability* | *Directed Fuzzing Framework* | 10 CMS web yang popular atau terkenal | Hasil eksperimen menunjukkan bahwa verifikasi Cefuzz pada kerentanan RCE lebih baik dari teknologi deteksi kerentanan aplikasi web saat ini, dan 13 kerentanan yang tidak diketahui ditemukan di 10 CMS web popular. |
| 2 | Triwibowo [33] Tahun 2016 | Deteksi dan Pencegahan Serangan Remote Code Execution Terhadap | Menggunakan *tools snort* | *Wing FTP Web Serve* | Mampu mendeteksi dan mencegah terjadinya serangan *Remote Code Execution* |

| No | Peneliti dan Tahun | Judul | Metode | Data | Hasil |
|---|---|---|---|---|---|
| | | WING FTP Web Server Menggunakan SNORT | | | |
| 3 | Biswas *et al* [34] Tahun 2018 | *A Study on Remote Code Execution Vulnerability in Web Applications* | *Review and Examining* | Lima tipe aplikasi berbasis web | Kebanyakan serangan RCE dikirimkan melalui metode POST dan GET |
| 4 | Mishra [35] Tahun 2019 | *SQL Injection Detection Using Machine Learning* | *Naïve Bayes vs Gradient Boosting* | Github | Naïve Bayes = 92,8% Gradient Boosting = 97.4% |
| 5 | Sagane [36] | *Malicious Code Detection Using Naïve Bayes Classifier* | *Naïve Bayes* | Knowledge Discovery Dataset | Tingkat akurasi deteksi menggunakan Naïve Bayes mencapai 97% |

Sesuai dengan data agregrasi pada tabel 2.2 s.d 2.4, kebanyakan metode atau algoritma deteksi yang digunakan untuk mendeteksi serangan siber dalam bentuk XSS, SQLi, dan RCE adalah Naïve Bayes, *Logistic Classifier*, *Gradient Boosting*, *Support Vector Machine*, dan K-Nearest Neighbor. Oleh karena itu, peneliti berfokus pada penggunaan kelima algoritma tersebut. Selain berfokus pada tingkat akurasi dan presisi, efektivitas implementasi deteksi juga mempertimbangkan lama ToP saat melakukan proses. Metode deteksi yang baik adalah yang akurat dan menggunakan *ToP* secara efisien. Selain metode deteksi, metode mitigasi yang kebanyakan digunakan peneliti sebelumnya adalah *tools* yang dikembangkan oleh OWASP, mulai dari OWASP ESAPI dan OWASP WAF. Pada penelitian terlampir pada tabel 2.2 s.d 2.4 juga disebutkan bahwa *filtering* juga dapat efektif menangani masalah serangan siber.

### 2.1.1. *Research Trend* Terkait Deteksi dan Mitigasi *Cyber Attack*

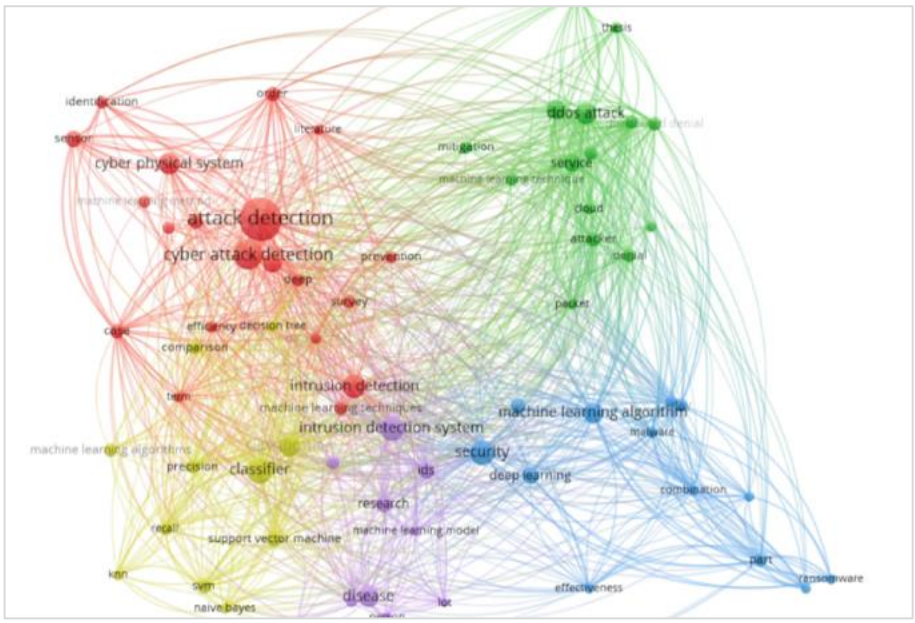
Selain melakukan telaah pada penelitian sebelumnya, seperti yang telah diungkapkan pada tabel 2.1, analisis *research trend* terkait deteksi dan mitigasi *cyber-attack* juga dilakukan. Analisis ini dilakukan untuk menelusuri beberapa hal, seperti metode dan teknik yang biasa digunakan, celah keamanan yang telah diteliti, dan hubungan antar sub-elemen *research trend*. Dengan demikian, penelitian ini diharapkan dapat melengkapi, selaras, dan berkontribusi pada ilmu pengetahuan, terutama pada bidang *cyber attack detection and mitigation*. Hasil visualisasi *research trend* menggunakan *VOS Viewer*, terkait deteksi dan mitigasi *cyber attack* dapat dilihat pada gambar 2.1 dan 2.2 di bawah ini.




**Gambar 2.1 Visualisasi *Related Research* Kata Kunci *Cyber Attack Mitigation***

Seperti yang terlihat pada gambar 2.1 di atas, celah keamanan *Cross Site Scripting* (XSS) sepertinya menjadi celah keamanan yang cukup sering diteliti pada bidang *cyber attack*. Celah keamanan berikutnya adalah *SQL Injection*, *Cross Site Requests Forgery*, dan seterusnya. Selain itu, objek-objek yang pernah diteliti pada bidang keamanan siber berdasarkan gambar 2.1 di atas adalah (1) upaya mitigasi;

(2) tindakan reduksi; dan (3) identifikasi. Selain itu, terdapat juga beberapa riset keamanan siber seperti pendekatan pada keamanan siber, manajemen risiko serangan siber, dan penelitian terkait peralatan (*tool*) keamanan siber.




***Gambar 2.2 Visualisasi Related Research Kata Kunci Cyber Attack Detection***

Selain terkait mitigasi *cyber attack*, analisis juga dilakukan untuk melihat perkembangan riset terkait deteksi *cyber attack*. Sedikit berbeda dengan mitigasi, riset terkait deteksi memiliki jumlah yang lebih banyak dari mitigasi. Berdasarkan pada gambar 2.2 di atas, riset terkait deteksi serangan siber kebanyakan berhubungan dengan pemanfaatan *machine learning* untuk mendeteksi berbagai jenis serangan siber. *Classifier* yang digunakan antara lain *Support Vector Machine (SVM)*, *K Nearest Neighbors (KNN)*, *Deep Learning*, dan *Naïve Bayes*.

### 2.1.2. Formulasi, Urgensi, dan Kebaruan Penelitian

Berdasarkan analisis berbagai penggunaan metode dan hasil penelitian sebelumnya pada tabel 2.1, secara garis besar penelitian terkait metode deteksi maupun mitigasi, masih dilakukan pada celah keamanan tunggal. Metode *machine learning*

yang digunakan cukup bervariasi, namun metode mitigasi yang digunakan kebanyakan berbasis *single layer* atau lapisan tunggal. Penggunaan lapisan tunggal tersebut masih menyebabkan munculnya berbagai serangan. Secara rinci, kebaruan dan perbedaan penelitian ini dengan sebelumnya adalah sebagai berikut.

1) penelitian ini tidak hanya mengusulkan metode deteksi serangan siber, tetapi juga metode mitigasi terhadap serangan tersebut. Sementara itu, penelitian sebelumnya hanya melakukan salah satunya;
2) penelitian ini mengkaji beberapa teknik pada serangan siber yaitu *Cross Site Scripting* (XSS), SQL *Injection (SQLi)*, dan *Remote Code Execution* (RCE). Sementara itu, penelitian sebelumnya hanya salah satunya;
3) penelitian ini mengukur tingkat efektivitas mitigasi pada *multi-websites* atau banyak *website*. Sementara itu, penelitian sebelumnya hanya *single website*;
4) penelitian ini menggunakan metode mitigasi yang terdiri dari lima lapisan. Sementara itu, penelitian sebelumnya hanya menggunakan *single-layer*;
5) penelitian ini menggunakan beberapa metode deteksi untuk mendeteksi gejala serangan siber dengan teknik yang berbeda-beda;
6) penelitian ini menggunakan dua *attack simulator software* untuk mengukur akurasi deteksi dan efektivitas mitigasi, sementara penelitian yang menggunakan *software* untuk menguji akurasi dan efektivitas masih terbatas; dan
7) penelitian ini menggunakan dataset yang lebih banyak daripada beberapa penelitian sebelumnya yang masih menggunakan data terbatas.

Merujuk pada kebaruan dan perbedaan di atas, penelitian ini juga mengembangkan atau memperbaiki kelemahan yang masih ditemukan pada penelitian sebe-

lumnya, terutama pada lapisan pertama, yaitu OWASP *ModSecurity Firewall*. Menurut penelitian sebelumnya, lapisan ini masih belum mampu mengatasi teknik seraangan siber seperti *stored* XSS, SQL *Injection*, dan RCE. Hal tersebut telah diungkapkan pada penelitian Ayunda *et al* [21], Zavarsky [22], dan Akbar et al [11]. Berdasarkan kelemahan-kelemahan tersebut, peneliti mengusulkan empat lapisan tambahan untuk memaksimalkan upaya mitigasi, yaitu (1) *Framework/CMS built in security;* (2) *HTTP Middleware*, (3) *Template Engine*; dan (4) *Data Sanitizer*. Melalui kelima lapisan tersebut, upaya mitigasi diharapkan dapat lebih baik.

### 2.2. Keamanan Siber (*Cyber Security)*

Berdasarkan ITU-T (*International Telecommunication Union Telecommunication Standardization Sector)*, keamanan siber adalah seperangkat alat, kebijakan, konsep keamanan, perlindungan keamanan, pedoman, pendekatan dan manajemen risiko, tindakan, pelatihan, praktik baik, jaminan, dan teknologi yang digunakan untuk melindungi lingkungan siber, organsasi, dan aset-aset pengguna [37].

### 2.3. Ancaman Siber (*Cyber Threats)*

Berdasarkan pengertian para ahli di atas, ancaman siber dapat didefinisikan sebagai upaya atau tindakan yang mengancam keamanan suatu sistem, infrastruktur, atau media lainnya, yang dilakukan pada suatu ekosistem teknologi, baik secara digital maupun fisik. Meskipun dalam bentuk ancaman atau tindakan secara fisik, tujuan utama ancaman siber tersebut tetaplah teknologi basis digital seperti sistem, aplikasi, atau database yang terhubung dengan perangkat fisik tersebut. Namun, sesuai dengan pembatasan masalah, penelitian ini berfokus pada tindakan ancaman siber yang dilakukan melalui dan/atau pada aplikasi berbasis *website*.

### 2.3.1. Jenis-Jenis Ancaman Siber

Maurya dan Sunghrova membagi ancaman siber menjadi lima tingkatan yang disesuaikan pada tingkatan ancamannya, yaitu: (1) vandalisme siber; (2) kejahatan siber; (3) serangan siber; (4) sabotase siber atau spionase; dan (5) perang siber [38]. Sementara itu, Donaldson membagi ancaman siber berdasarkan tipe penyerangnya, yaitu sebagai berikut: (1) pengancam komoditi; (2) aktivis peretasan; (3) organisasi kriminal; (4) kelompok mata-mata; dan (5) pelaku perang siber.

Merujuk pada pendapat para ahli tersebut, pembagian jenis ancaman siber dapat berbeda-beda, tergantung pada referensi yang dipergunakan. Namun, secara umum ancaman siber dapat dibagi menjadi tiga jenis ancaman, yaitu: (1) serangan siber; (2) kejahatan siber; dan (3) terorisme siber. Jika dikaitkan dengan pendapat sebelumnya, terorisme siber dapat diartikan sebagai vandalisme, tindakan sabotase atau spionase, dan konflik siber. Dalam hal ini, meskipun pola pembagiannya berbeda-beda, secara umum definisi dan konsep masing-masing ancaman masih saling beririsan dan terkait satu sama lain. Berikut ini adalah definisi dan penjelasan masing-masing jenis ancaman siber.

#### 2.3.1.1. Serangan Siber *(Cyber Attack)*

Jenis ancaman siber yang pertama adalah serangan siber. Ancaman ini berada pada posisi pertama, karena ancaman ini adalah tindakan awal yang pasti dilakukan sebelum memulai jenis ancaman siber lainnya. Dengan kata lain, serangan siber adalah pembuka bagi ancaman lain yang lebih berbahaya. Donaldson menyatakan bahwa serangan siber adalah serangan yang berkaitan dengan pencurian data, modi-fikasi data, dan mengganggu akses pada data, layanan, atau sistem [39].

Merujuk pada pengertian para ahli, serangan siber merupakan salah satu jenis ancaman siber yang memiliki tujuan destruktif atau melumpuhkan, seperti menginterferensi proses bisnis aplikasi, memperlambat akses aplikasi, atau bahkan meniadakan keberadaan suatu *website*. Oleh karena itu, dalam kaitannya dengan pelaksanaan penelitian ini, ancaman siber ini dipilih untuk diteliti, karena lebih sering terjadi dan merupakan pembuka ancaman-ancaman selanjutnya.

#### 2.3.1.2. Kejahatan Siber (*Cyber Crime*)

Kejahatan siber merupakan tindakan lanjutan, yang biasanya setelah melakukan serangan siber. Menyimpulkan pendapat para ahli sebelumnya, kejahatan siber dapat diartikan sebagai salah satu jenis ancaman siber yang tujuannya lebih pada tujuan-tujuan profit atau menguntungkan penyerang seperti pencurian akun pengguna yang memiliki akses terhadap uang, pencurian rincian kartu kredit secara *online*, dan seterusnya [40].

#### 2.3.1.3. Terorisme Siber (*Cyber Terrorism*)

Tujuan utama dari terorisme siber adalah untuk membuat kepanikan massal. Semakin banyak kepanikan yang ditimbulkan, semakin baik. Selain itu, terorisme siber juga dilakukan untuk melakukan teror pada pihak tertentu. Tindakan yang dilakukan dapat bervariasi, mulai dari pencurian data, perusakan inftrastruktur, dan seterusnya [41]. Istilah ini muncul pertama kali pada pertengahan tahun 80-an [40]. Secara teknis, Lewis mengatakan bahwa terorisme siber adalah penggunaan perangkat jaringan komputer untuk mematikan infrastruktur nasional, terutama yang vital atau untuk mengitimidasi pemerintah dan populasi sipil [40]. Terorisme siber dianggap sangat dekat dengan kemungkinan terjadinya perang siber [41].

### 2.3.2. Teknik Serangan Siber

Ancaman sibernya akan erat kaitannya dengan data. Pada era ini, aksesibilitas data dan keterbukaan informasi terjadi secara masif, sehingga pengguna internet sebaiknya berhati-hati dengan privasi [42]. Terkait dengan privasi tersebut, jenis data sensitif yang tersimpan di suatu sistem via internet meliputi (1) sidik jari; (2) data wajah (*facial recognition*); (3) catatan dan informasi medis pasien; (4) sekuensi genetika (*genetic sequence);* dan (5) kredensial [42]. Selain data, serangan siber juga akan berkaitan dengan implementasi teknik serangan.

Teknologi yang digunakan pada serangan siber saat ini berkembang pesat, sehingga metode dan teknik serangan juga semakin berkembang. Sebagai contoh, serangan siber dapat dilakukan secara otomatis menggunakan bantuan Botnets. Dengan Botnets, berbagai rangkaian serangan dapat dilakukan terautomasi [43], sehingga interfensi penyerang sangat sedikit dan prosesnya dapat dilakukan secara cepat. Aplikasi untuk menginspeksi masalah dan tingkat keamanan *website* juga dapat ditemukan secara mudah, sehingga celah keamanan pada suatu *website* dapat diidentifikasi secara rinci. Dengan kat alain, setiap *website* memiliki kemungkinan sebagai korban peretasan yang cukup tinggi. Berikut adalah penjelasan masing-masing teknik serangan yang kebanyakan digunakan, dan dalam kaitannya dengan OWASP *Top-10 Common Web Vulnerabilities.*

#### 2.3.2.1. *Cross*-Site *Scripting* (XSS)

Marashdih mendefinisikan *Cross-Site Scripting* (XSS) sebagai suatu serangan yang memanfaatkan kode-kode Javascript untuk mengeksekusi perintah tertentu, seperti mencuri akun pengguna dan *cookie*, mengirimkan data privasi kepada pe-

nyerang, dan sebagainya [27]. Serangan ini dieksekusi *browser* dan terjadi ketika korban mengakses laman yang terinfeksi. Serangan XSS dibagi menjadi tiga jenis, yaitu (1) *stored* XSS (XSS disimpan ke dalam database); (2) *reflected* XSS (kode XSS direfleksikan langsung oleh *browser*); dan (3) DOM-based XSS (eksekusi melalui *Document Object Manipulation*). Celah keamanan ini telah muncul beberapa kali dalam daftar *OWASP Top-10 Common Web Vulnerabilities*.

#### 2.3.2.2. *SQL Injection*

Cherry menyatakan bahwa *SQL Injection* adalah serangan yang dilakukan melalui *query SQL* yang dimodifikasi dan dikirimkan melalui permintaan *browser*. Jenis serangan ini dapat terjadi pada aplikasi berbasis *website* yang menggunakan *Active Server Page* (ASP) atau *Hypertext Preprocessor* (PHP) dan menggunakan data berbasis *SQL* [44]. Pada serangan ini, penyerang mengirimkan perintah-perintah SQL untuk mengambil atau mengakses data langsung ke *database server*. Serangan ini biasanya dilakukan untuk mengambil atau mengakses data pengguna (*users*) yang tersimpan di database.

#### 2.3.2.3. *Remote Code Execution* (RCE)

Sen et al menyatakan bahwa serangan *Remote Code Execution* (RCE) adalah serangan yang dieksekusi pada sisi *server* [45]. Senada dengan pendapat tersebut, Biswas et al mngatakan bahwa RCE adalah ancaman siber yang mengeksploitasi fungsional *web server* melalui skrip atau berkas [34]. Serangan RCE dapat dibagi menjadi dua jenis, yaitu RCE berbasis *website* dan RCE berbasis sistem. Melalui serangan ini, penyerang dapat mengeksekusi dan mengendalikan *server*. Merujuk

pada OWASP Top-10, serangan ini masuk dalam kategori celah keamanan *injection*. Serangan ini dapat terjadi pada banyak platform dan bahasa pemrograman.

## 2.4. Celah Keamanan Berdasarkan OWASP

OWASP adalah yayasan nirlaba skala internasional yang secara konsisten melakukan publikasi jenis-jenis celah keamanan yang paling umum ditemukan. Publikasi dilakukan secara berkala atau tiap tahun. Jika merujuk pada daftar celah keamanan yang telah dipublikasikan tersebut, terdapat beberapa celah keamanan yang bertahan selama beberapa kali ke dalam daftar OWASP Top-10 [43]. Fakta tersebut menunjukan bahwa setiap celah keamanan memiliki karakteristik tersendiri.

### 2.4.1. *Broken Access Control*

Celah keamanan ini didefinisikan sebagai pembatasan atau pengendalian tindakan yang dapat dilakukan oleh pengguna [46]. Pada celah keamanan ini, penyerang biasanya melakukan eksploitasi ketika sistem kontrol tersebut tidak diimplementasikan secara tepat. Misalnya, pengguna X dapat mengakses atau bahkan menyimpan data milik pengguna Y dan Z, sementara harusnya tidak bisa dilakukan. Contoh teknik serangan yang dapat diimplementasikan untuk mengeksploitasi celah keamanan ini adalah IDOR.

### 2.4.2. *Crytopgraphic Failures*

Sebelum bernama *Cyptographic Failures*, pada OWASP Top-10 2017 celah keamanan ini bernama *Sensitive Data Exposure*. Secara sederhana, celah keamanan ini terjadi ketika data sensitif disimpan dan/atau dikirimkan dalam bentuk *plain text* [47]. Dampaknya, penyerang dapat melakukan eksploitasi pada celah keamanan ini.

Teknik serangan untuk mengimplementasikan serangan ini beragam, seperti SQL *Injection*, XSS, dan lain-lain. Peningkatan drastis nilai Bitcoin juga membuat para pelaku krimimal siber mulai melakukan invasi pada dunia kriptografi [43].

### 2.4.3. *Injection*

Celah keamanan ini berhubungan dengan lemahnya validasi dan sanitasi data yang dikirimkan pengguna. Dampaknya, penyerang dapat menginjeksikan kode-kode tertentu untuk melumpuhkan sistem web [31]. Contoh celah keamanan yang masuk pada kategori ini adalah SQL/NoSQL *Injection* dan XSS [46]. Terkait dengan hal tersebut, laporan Symantec tahun 2019 mempublikasikan serangan yang disebut *Formjacking* yang secara esensial adalah serangan XSS yang lebih ditujukan pada pencurian nomor kartu kredit secara daring [43].

### 2.4.4. Insecure Design

*Insecure design* atau desain yang tidak aman kebanyakan terjadi karena desain aplikasi atau *website* yang kurang memperhatikan aspek atau tingkat keamanan [48]. Celah keamanan ini berhubungan dengan serangan siber yang mengeksploitasi *business logic error*, yaitu celah keamanan yang memanfaatkan kesalahan logika pada alur bisnis di dalam aplikasi.

### 2.4.5. *Security Misconfiguration*

Celah keamanan ini paling sering ditemukan pada aristektur *server* berbasis *cloud*. Laporan keamanan *cloud* Palo Alto tahun 2018, sebanyak 72% ancaman terbesar pada keamanan *server* berbasis *cloud* adalah *security misconfiguration* [49]. *Security misconfiguration* dapat mempengaruhi tingkat keamanan *server* [43]. Pada

celah keamanan ini, konfigurasi meliputi keamanan *login,* manajemen akun, kebijakan kata sandi, protokol, dan *firewall*.

### 2.4.6. *Vulnerable and Outdated Components*

Pada dasarnya, *cyber-security* dan *cyber-attack* adalah dua hal yang saling berkejaran. Penyerang dapat memanfaatkan komponen-komponen yang telah kadaluarsa atau memiliki celah keamanan [50]. Sebelum *web administrator* melakukan *patch* atau memperbarui komponen, penyerang akan memanfaatkan peluang tersebut. Eksploitasi terhadap celah keamanan ini terbilang cukup mudah, karena bahkan dapat dilakukan oleh pemula. Penyerang dapat memanfaatkan informasi celah keamanan yang biasanya akan dipublikasi sesuai dengan versi yang dipergunakan.

### 2.4.7. *Identification and Authentication Failures*

Kebanyakan kasus celah keamanan ini terjadi karena sistem tidak menggunakan *two-factor authentication* seperti pengenalan wajah, sidik jari, *smart card*, pesan rahasia, dan OTP (*one-time password*) [42], [51]. Kegagalan dalam mengautentifikasi dan mengidentifikasi dapat memberikan penyerang peluang lebih besar untuk melumpuhkan sistem keamanan *website*. Misalnya, terdapat beberapa *route* yang memerlukan autentifikasi namun tidak diproteksi. Akibatnya, pengguna yang tidak *login* dapat mengakses laman tersebut.

### 2.4.8. *Software and Data Integrity Failures*

*Software and data integrity failures* terjadi ketika kode dan infrastruktur yang tidak saling melindungi. Saat ini, banyak aplikasi yang menyisipkan fitur pembaruan secara otomatis. Apabila proses unduh pembaruan tidak memverifikasi integri-

tas, penyerang dapat mengunggah konten berbahaya pada pembaruan tersebut [52].

### 2.4.9. *Security Logging and Monitoring Failures*

Celah keamanan ini terkait dengan ekosistem *website* yang memiliki banyak *route* dan *controller*, sehingga beberapa *route* dan *controller* tersebut tidak membe-namkan validasi dan verifikasi secara tepat. Misalnya, aplikasi atau *website* tersebut tidak dapat mendeteksi adanya serangan [53]. Untuk mengeksploitasi celah keama-nan ini, penyerang dapat menggunakan teknik serangan yang tidak/belum terdeteksi oleh aplikasi. Karena terjadi kegagalan pada *monitoring* dan *logging*, penggunaan Botnet sangat mungkin terjadi pada celah keamanan ini.

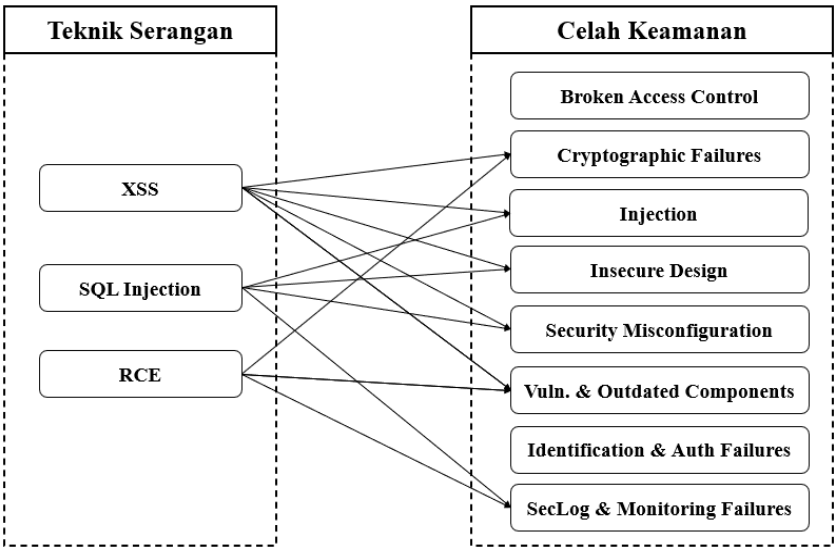
### 2.4.10. *Server-Side Request Forgery* (SSRF)

Sedikit memiliki kesamaan dengan celah keamanan CSRF, celah keamanan SSRF atau *Server-Side Request Forgery* juga melakukan eksploitasi pada proses *redirection*/ Penyerang dapat mengakses jaringan internal dan layanan lokal [54]. Penyerang juga dapat mengeksekusi perintah pada tingkatan sistem operasi, terutama pada *server* yang memiliki celah keamanan. Kasus terkenal yang melibat-kan celah keamanan ini pernah terjadi pada tahun 2020 ketika penyerang berhasil mencuri 100 juta kartu kredit pengguna layanan perusahaan Capital One.

## 2.5. Pemetaan Serangan dan Celah Kemananan OWASP Top-10

Seperti yang telah dijelaskan pada poin 2.2.2 dan 2.3 bahwa penelitian ini akan fokus pada celah keamanan yang paling sering terjadi, sesuai dengan OWASP *Top-10 Common Web Vulnerabilities* Tahun 2021. Eksploitasi pada celah keama-nan tersebut dapat menggunakan satu atau beberapa metode serangan relevan,

seperti yang telah disebutkan pada poin 2.2.2. Oleh karena itu, untuk melihat bagaimana hubungan atau *link* antara teknik serangan dan celah keamanan OWASP Top-10, berikut ini disampaikan pemetaannya pada gambar 2.3 ini.




**Gambar 2.3 Pemetaan antara Teknik Serangan dan Celah Keamanan**

Berdasarkan gambar 2.3, satu celah keamanan dapat dieksploitasi oleh satu atau beberapa teknik serangan. Hal ini menunjukan bahwa penyerang atau peretas memiliki cukup banyak alternatif teknik serangan untuk diujikan, sehingga celah keamanan tersebut harus di-*patch* secara komprehensif dan menyeluruh. Jika tidak, penyerang dapat menguji-coba tiap teknik serangan pada tiap celah keamanan. Dengan keadaan seperti itu, data-data penting menjadi terancam keberadaanya. Rincian pemetaan teknik serangan yang dapat digunakan untuk mengeksploitasi celah keamanan tersebut juga dapat dilihat melalui tabel 2.2 di bawah ini.

**Tabel 2.5 Pemetaan Teknik Serangan dan Celah Keamanan**

| No | Teknik Serangan | Celah Keamanan |
|---|---|---|
| 1 | *Cross Site Scripting (XSS)* | *a) Cryptographic Failures*<br>*b) Injection*<br>*c) Insecure Design* |

| No | Teknik Serangan | Celah Keamanan |
|---|---|---|
| | | *d) Security Misconfiguration*<br>*e) Vulnerable and Outdated Components* |
| 2 | *SQL Injection* | *a) Injection*<br>*b) Insecure Design*<br>*c) Security Misconfiguration*<br>*d) Security Log and Monitoring Failures* |
| 3 | *Remote Execution Code (RCE)* | *a) Broken Access Control*<br>*b) Cryptographic Failures*<br>*c) Security Log and Monitoring Failures* |

### 2.6. Deteksi Serangan Siber

Implementasi deteksi terhadap serangan siber dilakukan berdasarkan komponen yang ingin diproteksi. Prosedur deteksi serangan juga dapat menggunakan berbagai cara dan metode. Pada dasarnya, deteksi serangan siber adalah tindakan atau metode untuk mengenali gejala-gejala atau kemunculan serangan siber. Penempatan *detection engine* disesuaikan dengan jenis serangan yang ingin dideteksi.

Singh dan Govindarasu mengembangkan metode deteksi anomali pada lapisan jaringan menggunakan *Machine Learning* [55]. Selain itu, deteksi serangan siber menggunakan *unsupervised machine learning* juga pernah dilakukan oleh Karimipour et al [56] dan Dutta et al yang menggunakan *deep Learning* [57]. Selain menggunakan metode tersebut, deteksi serangan juga dapat menggunakan bantuan perangkat lunak seperti IDS atau *Intrusion Detection System*, yang dapat digunakan menga-wasi jaringan/sistem dari aktivitas ilegal.

Salah satu deteksi yang sulit dilakukan adalah mendeteksi *insider attacks* atau serangan dari dalam, karena serangan biasanya akan terdeteksi setelah serangan dilakukan [43]. Berdasarkan penelitian, 70% pencurian dilakukan dari dalam diban-

dingkan dari luar, namun 90% kendali dan pengawasan keamanan difokuskan pada ancaman eksternal [51]. Untuk melakukan deteksi anomali, sistem berbasis pembelajaran mesin dapat dipergunakan. Sistem deteksi akan memicu alarm ketika terdapat objek yang berprilaku berbeda dari pola normal yang telah ditentukan [58]. Oleh karena itu, penggunaan *machine learning* sangat direkomendasikan.

## 2.7. Pembelajaran Mesin (*Machine Learning*)

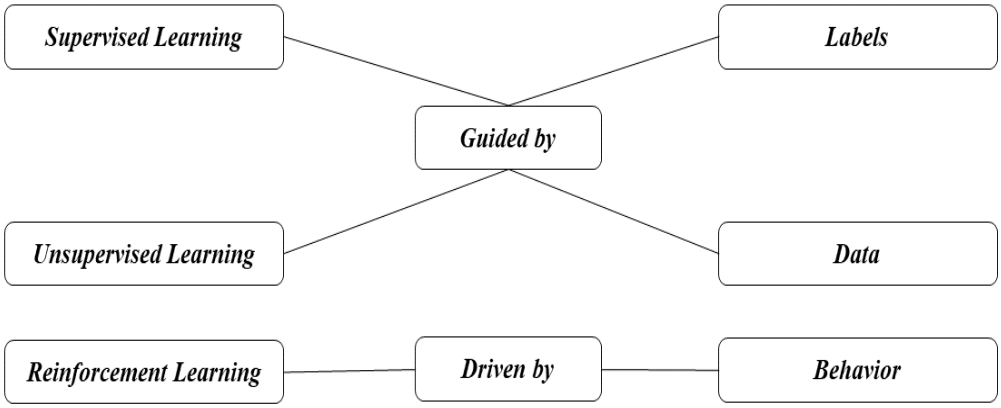
Dua mendefinisikan *machine learning* sebagai proses komputasi yang dapat menyimpulkan dan menggeneralisasikan model pembelajaran dari dataset atau sampel yang diberikan [58]. Lebih rinci dari definisi yang diberikan Dua, Herron menyatakan bahwa *machine learning* adalah salah satu bidang komputasi yang melibatkan mesin untuk mencari, mempelajari, dan memahami pola pada kumpulan data, dan menggunakan data tersebut untuk menghasilkan keputusan tanpa melibatkan pengguna secara eksplisit [59].

Bell mendefinisikan *machine learning* sebagai cabang kecerdasan buatan, yang memungkinkan sistem dapat belajar dari data yang digunakan untuk latihan. Sistem tersebut kemudian belajar dan mengembangkan pengalaman seiring waktu, sehingga sistem tersebut dapat melakukan prediksi suatu luaran pertanyaan berdasarkan pada pembelajaran yang telah dilakukan sebelumnya [60]. Pembelajaran mesin adalah metode komputasi untuk membuat suatu mesin dapat belajar dan membuat keputusannya sendiri [61].

### 2.7.1. Jenis-Jenis Pembelajaran Mesin

*Machine learning* atau algoritmanya secara umum dapat diklasifikasikan ke

dalam tiga kategori, yaitu *supervised learning*, *unsupervised learning,* dan *reinforcement learning* [59]. Perbedaan mendasar dari ketiga jenis pembelajaran mesin adalah pada karakteristik data dan masalah atau model yang ingin dihasilkan. Oleh karena itu, sebelum memilih jenis pembelajaran, karakteristik data diidentifikasi secara komprehensif terlebih dahulu. Konsep pembelajaran mesin tersebut dapat diungkapkan melalui gambar 2.4 di bawah ini.




**Gambar 2.4 Jenis-Jenis Pembelajaran Mesin Berdasarkan Media Belajar** *[59]*

### 2.7.1.1. *Supervised Learning*

Pada *machine learning* ini, mesin belajar dengan bantuan atau interferensi pemrogramnya. Data diberi label, sehingga mesin dapat mengenali kategori data tersebut. Pembelajaran ini dapat digunakan untuk melakukan prediksi. Semakin banyak *data training* yang dimiliki, algoritma akan memiliki predkisi yang lebih baik. Contoh penerapan *supervised* learning adalah penyaring SPAM dan preferensi iklan berdasarkan pada penggunanya.

### 2.7.1.2. *Unsupervised Learning*

Pada pembelajaran ini, data yang ingin diproses cukup banyak dan algoritma harus mempelajari properti dan mencari fitur yang unik. Setelah diproses, algoritma

dapat membuat klusterisasi yang unik sehingga setiap data akan berada pada kelompoknya masing-masing. Contoh implementasi *unsupervised learning* adalah rekomendasi video, pola pembelian, dan layanan FAQ (*Frequenly Asked Question*).

#### 2.7.1.3. *Reinforcement Learning*

Selain *supervised* dan *unsupervised learning*, terdapat pembelajaran yang disebut dengan *reinforcement learning* atau pembelajaran penguatan. Pada pembelajaran ini, mesin diminta untuk belajar dari kesalahan. Setiap kesalahan dan kebenaran diberikan nilai tertentu, sehingga pembelajaran ini seperti membentuk suatu kebiasan pada mesin. Contoh implementasi *reinforcement learning* adalah *gaming*, pengelolaan sumber daya, dan lain-lain.

### 2.7.2. Pembelajaran Mesin bagi Keamanan Siber

*Machine learning* dan keamanan siber memiliki ruang lingkup dan kesamaan yang cukup signifikan, karena keduanya adalah bagian dari ilmu komputer [43]. Pembelajaran mesin kini banyak digunakan pada bidang keamanan siber, terutama untuk mendeteksi terjadinya serangan siber. Terdapat banyak algoritma yang dapat digu-nakan untuk mengimplementasikan deteksi tersebut [62]. Deteksi serangan tersebut berbasis prediksi, sehingga seorang administrator dapat mengambil langkah yang cepat guna mengantisipasi serangan tersebut [18].

Sistem deteksi berbasis *hybrid* dapat semakin meningkatkan deteksi serangan. Sistem ini memadukan antara *machine learning* dan pengguna atau manusia [43]. Secara khusus, Tsukerman menjelaskan bahwa pada bidang keamanan siber, *machine learning* dapat dipergunakan untuk banyak keperluan, yaitu sebagai beri-

kut: (1) mempelajari karakteristik dan mendeteksi keberadaan *malware*; (2) membaca kebiasan untuk *social enginering*; (3) *penetration testing* (misalnya untuk memindai celah keamanan *web server* dan mendeteksi URL atau tautan yang berbahaya); (4) *automatic intrusion detection;* dan (5) mengamankan data [63].

Selain pemaparan dari para ahli di atas, beberapa penelitian dan pengembangan deteksi serangan siber telah dikembangkan menggunakan bantuan *machine learning* [55]–[57]. Pola belajar yang peneliti sebelumnya gunakan berbeda-beda, tergantung pada teknik serangan yang dideteksi. Namun, terlepas dari perbedaan tersebut, penggunaan *machine learning* pada berbagai penelitian keamanan siber menunjukan bahwa *machine learning* adalah metode efektif yang dapat digunakan untuk mendeteksi serangan siber secara cepat dan akurat. Pendapat ahli dan penelitian tersebut menunjukan bahwa pembelajaran mesin adalah metode yang potensial untuk digunakan dalam bidang keamanan siber. Oleh karena itu, penelitian ini juga akan menggunakan *machine learning* dalam mendeteksi serangan.

### 2.7.3. Pemrosesan dan Pengenalan Teks untuk Deteksi Serangan

Pemrosesan dan pengenalan teks (*text processing and recognition*) dapat digunakan untuk mengklasifikasikan atau mengklusterisasikan perintah atau kode-kode serangan. Pada tahapan ini, mesin dilatih, sehingga dapat mengklasifikasikan mana teks atau kode yang berupa serangan dan bukan. Pola *supervised learning* digunakan karena beberapa pola atau bentuk perintah serangan dapat diidentifikasi. Untuk melakukan pemrosesan dan pengenalan teks secara efektif, peneliti menggunakan bantuan beberapa aplikasi atau *library* pendukung berbasis bahasa pemrograman Python yaitu: (1) NLTK; (2) Scikit-Learn, dan (3) lain-lain.

Analisis dan pengenalan teks digunakan, karena kelima teknik serangan yang dipilih dan diujikan dengan metode deteksi adalah serangan yang memanfaatkan perintah berbasis teks, bukan numerik. Berdasarkan fitur dan tahapan implementasinya, analisis dan pengenalan teks dapat dilakukan melalui beberapa cara yaitu melalui (1) pemilihan fitur berdasarkan karakter; (2) pemilihan fitur berdasarkan kata; (3) kata-kata fungsional; (4) fitur struktural; dan (5) *bag of words* [38]. Berikut ini adalah penjelasan dari *tools* untuk memproses dan mengenali teks.

#### 2.7.3.1. Natural Language Processing Kit (NLTK)

Bird mendefinisikan bahwa *Natural Language Processing Kit* (NLTK) adalah perangkat lunak berbasis Python yang digunakan untuk melakukan *natural language processing* atau pemrosesan bahasa alami. NLTK memiliki cukup banyak fitur mulai dari *part-of-speech tagging*, *syntatic parsing*, dan klasifikasi teks [64]. Pustaka ini dapat digunakan untuk memproses serangan berbasis teks.

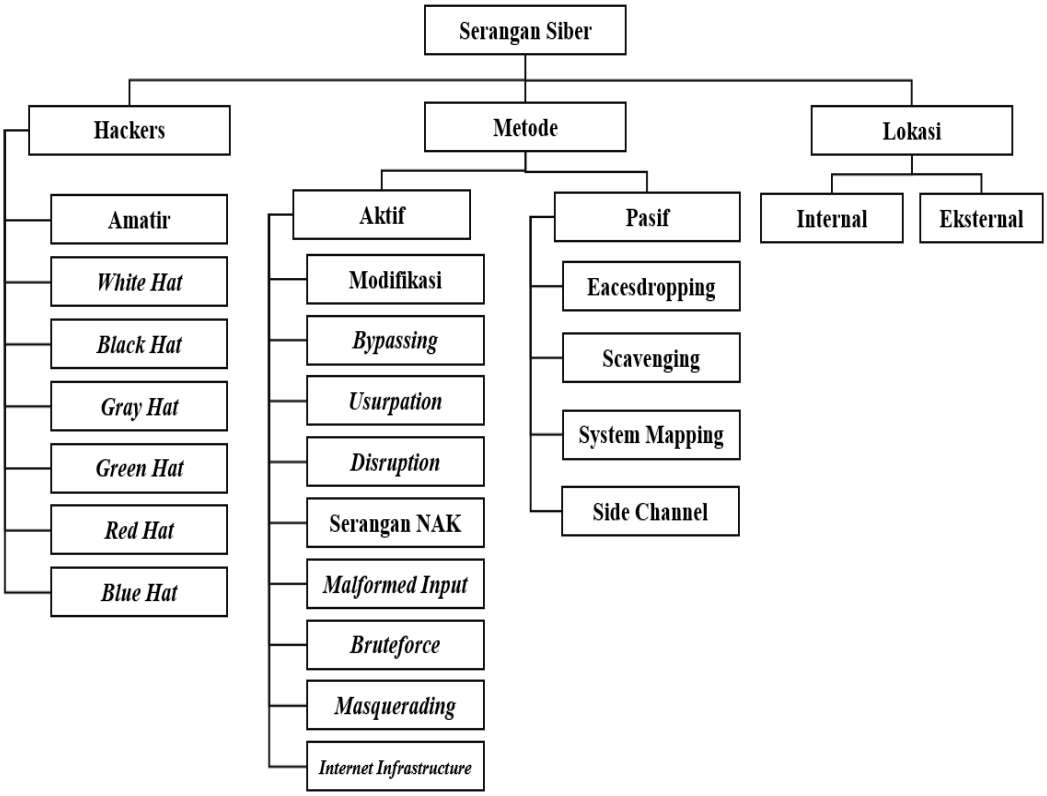
#### 2.7.3.2. *Scikit-Learn*

Selain Python NLTK, pustaka berbasis Python digunakan pada penelitian ini adalah Scikit-Learn. Pustaka ini dapat membantu peneliti dalam melakukan berbagai penghitungan statistik. Oleh karena itu, pustaka ini digunakan untuk melakukan penghitungan algoritma *Naïve Bayes, Logistic Regression, Gradient Boosting, Support Vector Machine,* dan *K-Nearest Neighbor*.

## 2.8. Mitigasi Serangan Siber

Merujuk pada definisi Rains dan Safari, mitigasi serangan siber didefinisikan sebagai serangkaian cara, metode, atau langkah untuk mengatasi atau meminimali-

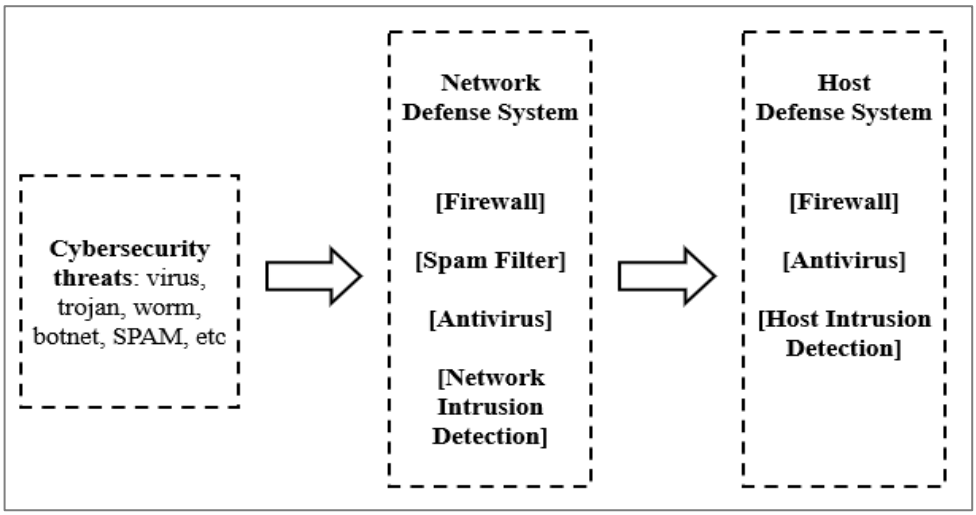
sasi serangan siber pada suatu sistem atau sub-sistem [66]. Donaldson juga mengatakan bahwa mitigasi adalah ketika kendali keamanan digunakan untuk mengurangi atau meminimalisasi dampak dari risiko yang terjadi. Ambalavanan dan Bala dalam Ganapathi menyatakan bahwa mitigasi serangan dapat dilakukan pada tiga lingkup serangan siber, yaitu (1) mitigasi berdasarkan *hacker*; (2) mitigasi berdasarkan metodenya; dan (3) mitigasi berdasarkan lokasi [67]. Klasifikasi serangan siber menurut Ambalavanan dan Bala tertuang pada gambar 2.5 berikut ini.




**Gambar 2.5 Klasifikasi Serangan Siber dalam Konteks Mitigasinya** *[67]*

Berdasarkan definisi para ahli, mitigasi serangan siber dapat diartikan sebagai serangkaian upaya untuk mengatasi atau meminimalisasi serangan siber. Dalam hal ini, Bentuk atau upaya mitigasi dapat berbeda-beda tergantung pada infrastruktur yang digunakan. Sebagai gambarannya, merujuk pada beberapa kasus yang pernah terjadi di *cloud-based server*, upaya mitigasi dilakukan hanya dengan *public-key*

*infrastructure* [38], karena kunci publik tersebut adalah ID server, sehingga *backup* dan *restore* dapat dilakukan secara mudah. Gambaran tersebut menunjukan bahwa mitigasi disesuaikan dengan karakteristik dan kebutuhan infrastruktur.




**Gambar 2.6 Model Mitigasi Sistem Keamanan Siber Konvensional** *[58]*

Gambar 2.6 adalah model mitigasi sistem keamanan siber konvensional. Pada model konvensional, mitigasi kebanyakan dilakukan pada *layer* atau lapisan jaringan atau bahkan perangkat fisik. Berbeda dengan model mitigasi konvensional tersebut, penelitian ini menawarkan model mitigasi pada *layer* aplikasi. Mitigasi serangan lapisan aplikasi harus dilakukan lebih komprehensif dan menyeluruh, karena kebanyakan peretasan dan *data breach* terjadi pada lapisan aplikasi [68]. Weidman menyatakan bahwa satu teknik mitigasi tidak cukup untuk mengatasi serangan, apalagi dengan metode serangan yang terbaru [69]. Oleh karena itu, metode mitigasi yang pada penelitian ini menggunakan keamanan berlapis.

Dalam melakukan mitigasi atau membangun sistem keamanan, Diogenes mengajukan dua strategi, yaitu (1) berpikir dan bertindak seolah seperti penyerang dan (2) mengoptimalisasi pertahanan yang sudah ada secara mendalam [49]. Pada strategi pertama, terdapat empat pengujian keamanan yang sebaiknya dilakukan

yaitu pengujian eksternal, internal, *blind*, dan bertarget. Sub-strategi yang dilakukan pada strategi kedua adalah bertahan secara mendalam (*defense in depth*) dan bertahan secara menyeluruh (*defense in breadth*).

Terkait dengan pelaksanaan penelitian ini, peneliti menggunakan kedua konsep strategi keamanan tersebut. Penggunaan aplikasi ZAP dan Arachni untuk menguji tingkat keamanan metode yang diusulkan sesuai dengan strategi pertama. Sementara itu, penggunaan *multi-layer security* sesuai strategi kedua [49]. ZAP dan Arachni adalah aplikasi yang memiliki fitur "penyerangan" secara terotomatis dan *recursion-deep*. Dengan kedua aplikasi ini, peneliti dapat mengukur bagaimana tingkat akurasi dan efektivitas metode yang diusulkan. Pengujian menggunakan kedua aplikasi tersebut untuk semakin merepresentatifkan hasil penelitian, yaitu bahwa kedua metode telah diujikan dengan aplikasi yang memadai.

### 2.8.1. Zed Attack Proxy (ZAP)

ZAP atau Zed Attack Proxy adalah aplikasi yang biasa digunakan untuk menginspeksi celah keamanan pada suatu sistem atau *website*. ZAP memiliki fitur intercept berbasis proxy, mirip dengan aplikasi *BurpSuite*. Ketika proxy diaktifkan, ZAP dapat menangkap lalu lintas jaringan atau data, sehingga inspektor keamanan web dapat melihat data apa saja yang dikirimkan dan diterima saat web diakses. Informasi tersebut bukan hanya menampilkan *body*, tetapi juga *header* dari *request* dan *response*. Oleh karena itu, ZAP terbilang aplikasi yang *powerful* dalam menginspeksi elemen *website*. Selain itu, aplikasi ini juga memiliki fitur/mode *attack*, sehingga memungkinkan implementasi simulasi serangan. Dengan mode tersbebut, ZAP dapat digunakan sebagai aplikasi untuk menyerang *website* yang diujikan.

### 2.8.2. Arachni

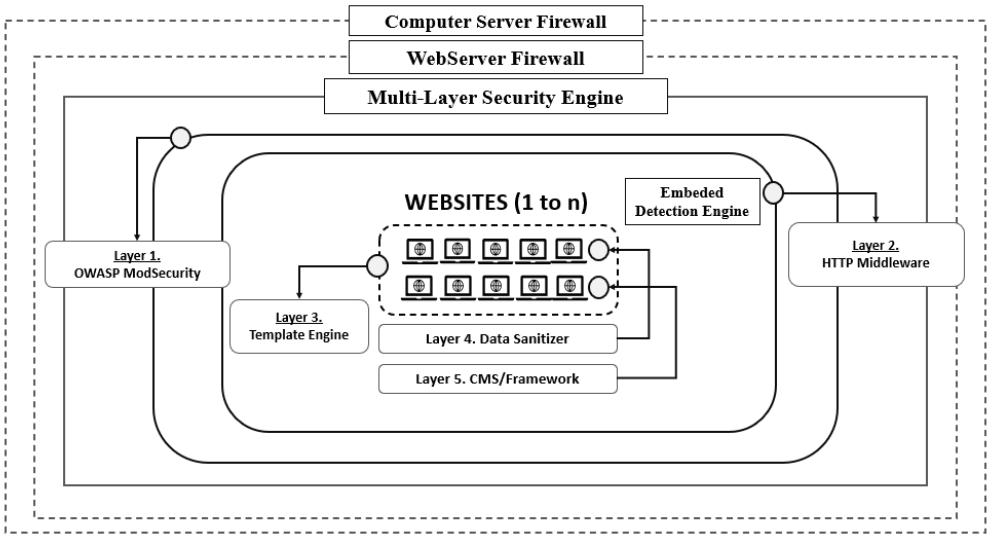
Arachni adalah aplikasi *open source* berbasis kerangka Ruby yang dikembangkan untuk membantu *penetration tester* dan administrator mengevaluasi keamanan aplikasi *website*. Aplikasi ini bersifat multi-*platform* dan berdiri secara *portable*, sehingga mudah untuk diinstalasi. Arachni mendukung beberapa teknik pemindaian celah keamanan secara otomatis yang meluputi (1) XSS; (2) SQL Injection; (3) NoSQL Injection; (4) Code Injection; (5) File Inclusion; (6) CSRF Detection; (7) OS Command Injection; (8) Remote File Inclusion; dan (9) lain-lain [70].

## 2.9. *Multi-Layer Security*

Merujuk pada pendapat Weidman yang menyatakan bahwa satu teknik mitigasi tidak cukup untuk mengatasi serangan, apalagi untuk menghadapi metode-metode serangan terbaru [69]. Bahkan, hanya untuk melakukan mitigasi serangan DDoS (*Distributed Denial of Service*), Thakkar membagi tindakan mitigasi menjadi beberapa sesi/bagian seperti berdasarkan *bandwith,* sumber daya, berdasarkan aplikasi, berdasarkan praktis untuk perusahaan, dan *future trends* [71]. Penerapan metode keamanan berlapis pernah dimulai oleh beberapa peneliti, seperti penelitian Chen *et al* [72], Fan *et al* [73], dan Miller dan Zhang [74]. Gambaran metode *multi-layer security* pada penelitian ini dapat dilihat pada gambar 2.7.

Metode keamanan berlapis digunakan untuk mengatasi beragam teknik serangan siber yang terus berkembang. Teknik serangan tersebut semakin bervariatif, sehingga memerlukan penanganan yang berbeda-beda atau secara tersendiri. Metode keamanan berlapis ini menawarkan keamanan yang inklusif dan *multi-websites*, sehingga cocok diterapkan pada berbagai *website*, yang berada di satu lingkungan

*server*. Metode ini berada di lapisan aplikasi, sehingga aplikasi yang berada di dalam-nya harus dibangun dengan mempertimbangkan ketentuan yang disyaratkan metode ini. Namun demikian, metode ini menawarkan pengembangan yang fleksi-bel, karena dapat disesuaikan dengan karakteristik aplikasi yang digunakan.




**Gambar 2.7 Mekanisme Penerapan *Multi-Layer Security* pada *Multi-Websites***

Metode *multi-layer security* terdiri dari lima lapisan yaitu (1) OWASP Mod *Security Firewall* atau sering disebut juga sebagai OWASP ModSecurity *Web App-lication Firewall*; (2) HTTP Middleware; (3) *Template Engine;* (4) *Data Sanitizer;* dan (5) Framework/CMS *Default Security Features*. Karena metode ini ditujukan untuk memberikan perlindungan pada *multi websites* yang tentunya memiliki ting-kat dan celah keamanan serta teknologi yang berbeda, metode keamanan berlapis ini memiliki karakteristik, yaitu sebagai berikut.

a) Inklusif, artinya dapat digunakan oleh berbagai *website* yang memiliki tingkat keamanan dan teknologi yang berbeda-beda;
b) Fleksibel, artinya dapat disesuaikan dengan kebutuhan tiap *website*; dan
c) Independen, artinya metode ini dapat berdiri sendiri atau diintegrasikan

dengan teknologi *website* yang digunakan;

Berdasarkan karakteristik tersebut, metode pada penelitian ini menawarkan ber-bagai kelebihan. Tidak hanya bagi *web developer* dan *penetration tester*, tetapi juga bagi *web administrator*. Selain itu, metode berlapis ini merupakan kombinasi berbagai teknik dan stra-tegi keamanan pada *layer* aplikasi yang telah diuji.

Dalam hal ini, beberapa lapisan di dalamnya telah dikaji dan diuji secara parsial oleh peneliti sebelumnya. Peneliti menyadari bahwa penelitian sebelumnya tersebut masih memiliki kekurangan, sehingga tiap lapisan pada metode ini telah dioptimalisasi atau *patch*. Dengan kata lain, penelitian ini mengambil kelebihan-kelebihan pada tiap lapisan, lalu mengatasi kelemahan di dalamnya. Penjelasan ter-kait optimalisasi tersebut diungkapkan tersendiri pada poin **2.8.1**—**2.8.5**.

### 2.9.1. OWASP Mod Security Firewall

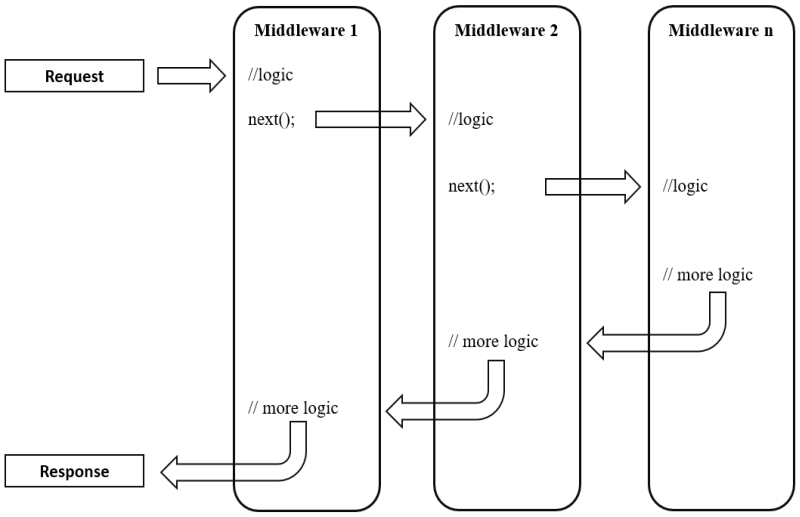
OWASP ModSecurity Firewall (WAF) adalah aplikasi yang dikembangkan OWASP untuk mengatasi celah keamanan *website*. Penanganan celah keamanan aplikasi ini didasarkan pada daftar Top-10 *Common Web Application Vulnerabili-ties*. Aplikasi ini berperan sscholebagai *service* pada lalu lintas HTTP dan diinsta-lasi sebagai modul *web server* seperti Apache2 dan Nginx. Ketika bekerja, aplikasi ini dapat melakukan penyaringan, pengawasan, dan pemblokiran lalu lintas HTTP, berdasarkan *rules* atau aturan yang diberikan. Aplikasi ini bekerja seperti *firewall* sehingga berpengaruh pada semua aplikasi yang berada pada *web server* sama.

Pengujian terhadap aplikasi ini telah dilakukan oleh Akbar *et al* [11], Betarte *et al* [75], Sobola *et al* [22], dan Prabudhesai [76]. Berdasarkan penelitian yang te-lah dilakukan, OWASP ModSecurity berhasil mengatasi cukup banyak jenis sera-

ngan siber yang diujikan, seperti XSS, SQL, dan RCE. Namun, mengacu pada data dari *Web Application Firewalls Bypasses Collection* [77], terdapat beberapa serangan yang masih tidak terdeteksi dan teratasi oleh OWASP ModSecurity. Oleh karena itu, untuk mengatasi masalah tersebut, peneliti telah melakukan beberapa tindakan optimalisasi dan konfigurasi sebagai berikut:

a) memperbarui OWASP Mod Security ke versi yang paling terbaru;
b) menggunakan *template engine* atau *data sanitizer* untuk mengatasi serangan *stored*-XSS karena tidak terdeteksi dan teratasi; dan
c) menggunakan layer 2, CMS/Framework *default security* untuk mengatasi beberapa pola SQL injection yang masih bisa di *ByPass*

### 2.9.2. HTTP *Middleware*




**Gambar 2.8 Konsep HTTP Middleware Berdasarkan Microsoft**

HTTP *Middleware* adalah suatu pustaka, *controller*, atau fungsi di dalam *web framework* atau CMS yang berperan sebagai pengendali lalu lintas HTTP. Dengan kata lain, setiap *request* dan *response* dapat diatur, disaring, divalidasi, dan diawasi melalui *Middleware* [85], [86]. Kebanyakan teknologi *website* yang dikembangkan

dengan prinsip MVC (*Model, View,* dan *Controller*) atau MTV (*Model, Template,* dan *View)* dan berbasis *routing* memiliki *Middleware*.

Melalui lapisan ini, konten lalu lintas yang mengandung *malicious codes* dapat disaring atau bahkan ditolak berdasarkan aturan yang telah diatur. Pada dasarnya, *Middleware* bekerja melalui dua lingkup, yaitu (1) sebelum permintaan (*request*) diproses dan (2) sebelum respon ditampilkan atau diproses. Dengan dua lingkup ini, implementasi aturan juga dapat dilakukan lebih terfokus. Dengan konfigurasi yang tepat, lapisan ini dapat mengatasi berbagai jenis serangan siber.

### 2.9.3. *Template Engine*

*Template engine* atau prosesor tata letak adalah perangkat lunak yang dikembangkan untuk membuat tata letak dinamis, menggunakan berkas konfigurasi statis. Dengan *template engine*, pengembang dapat membangun tata letak *website* secara efisien, karena kode yang diperlukan lebih sedikit dan terstruktur. Prosesor tata letak juga mendukung penyusunan tata letak secara *inheritance*, sehingga pengelolaan *child template* berdasarkan *parent* nya dapat dilakukan dengan lebih terkendali. Pada tingkatan yang lebih *advanced*, fleksibilitas *coding* juga tidak terpengaruh karena dapat menyisipkan *native code* ke *template engine*.

Berdasarkan fungsi-fungsinya tersebut, *template engine* sebenarnya bukan perangkat lunak yang dikembangkan untuk keperluan keamanan *website*. Namun, dengan konfigurasi dan penerapan yang tepat, *template engine* dapat dipergunakan untuk mengatasi beberapa teknik serangan siber seperti XSS, IDOR, dan CSRF. Hal tersebut juga telah disadari oleh para pengembang *template engine*, sehingga dokumentasi terkait *template engine* biasanya dilengkapi dengan sesi atau pemba-

hasan terkait keamanan. Oleh karena itu, *template engine* dipergunakan sebagai lapisan ketiga dalam metode *multi-layer security* ini.

Karena jumlah *template engine* yang sangat bervariasi, terdapat persyaratan yang harus dipenuhi oleh *tem-plate engine* tersebut agar dapat digunakan pada lapisan keempat. Persyaratan terse-but adalah setiap *output* harus disaring sebelum ditampilkan. Dengan kata lain, vari-abel yang mengandung kode-kode yang dapat dieksekusi *browser* harus dikonversi ke teks. Berikut ini adalah beberapa *template engine* yang dapat dipergunakan.

**Tabel 2.6 Daftar Template Engine yang Paling Sering Digunakan**

| No | Nama *Template Engine* | Bahasa Pemrograman | Laman Resmi |
|---|---|---|---|
| 1 | Jinja | Python | https://jinja.palletsprojects.com |
| 2 | Django | Python | https://www.djangoproject.com/ |
| 3 | Laravel Blade | PHP | https://laravel.com/ |
| 4 | Smarty | PHP | https://www.smarty.net/ |
| 5 | Slim | Ruby | http://slim-lang.com/ |
| 6 | Jade | Javascript | https://jade-lang.com/ |
| 7 | Mustache | Javascript | https://mustache.github.io/ |

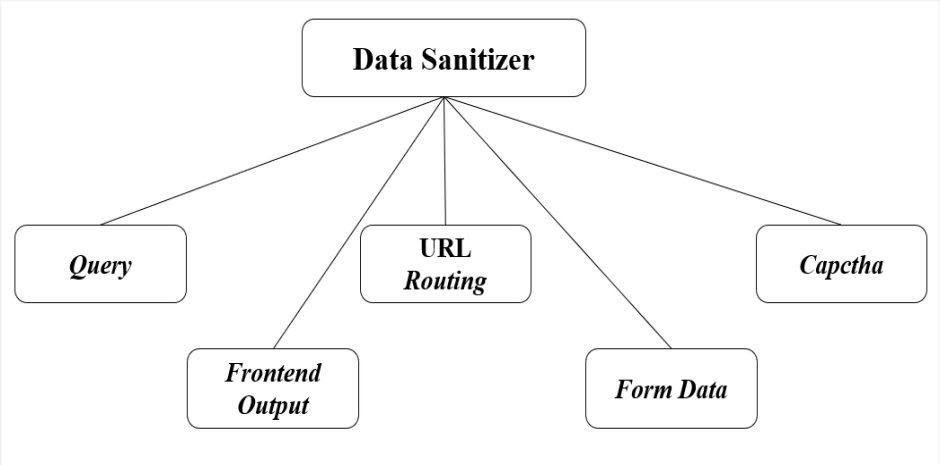
### 2.9.4. *Data Sanitizer*

Lapisan keempat adalah data *sanitizer*. Lapisan ini berperan dalam menyaring setiap masukan atau *input*, baik yang dikirimkan melalui for-mulir maupun melalui URL. Selain itu, lapisan kelima ini juga meliputi penyediaan kode tantangan atau *captcha* pada laman yang vital. Penyediaan kode tantangan tersebut dapat menghin-dari serangan *brute-force*. Untuk memaksimalkan mitigasi terhadap serangan se-perti *brute-force*, lapisan ini dapat sinergi dengan lapisan ketiga, yaitu HTTP

*Middleware*. Pada beberapa *web framework* seperti Django dan Laravel atau CMS dengan komunitas besar seperti Wordpress, Drupal, dan Joomla, *data sanitizer* sudah merupakan fungsi standar yang harus tersedia secara *default*. Dalam hal ini, secara rinci, data *sanitizer* meliputi hal-hal penyaringan sebagai berikut.

a) sanitasi terhadap kueri untuk mencegah SQL dan NoSQL *Injection*;
b) sanitasi luaran *frontend* untuk mencegah serangan XSS;
c) sanitasi terhadap URL atau *routing*;
d) sanitasi pada data yang dikirimkan melalui formulir; dan
e) sanitasi terhadap isian *captcha*.

Berdasarkan rincian tersebut, *data sanitizer* digambarkan sebagai berikut.




**Gambar 2.9 Distribusi *Data Sanitizer* pada Metode *Multi Layer Security***

### 2.9.5. *Framework/CMS Built-In Security*

*Layer* atau lapisan kelima adalah *web framework* atau CMS (*Content Management System*) yang dilengkapi oleh fitur keamanan yang meliputi *data sanitizer*, terutama pada sisi *query* atau pengambilan data dari database, *CSRF token*, dan *safe templating output*. Oleh karena itu, untuk berada pada lapisan yang kedua

ini, *web framework* atau CMS harus memenuhi setidaknya tujuh persyaratan sehingga dapat dikategorikan berteknologi cukup aman, yaitu sebagai berikut.

a) pembaruan dilakukan secara rutin atau terjadwal, terutama pembaruan terkait *security issues* atau *security patch* pada tiap versinya;
b) lalu lintas dapat dikendalikan melalui *middleware* atau sejenisnya;
c) pemanggilan data berbasis ORM (*Object Relation Mapping*) atau *Query Builder* memiliki fitur sanitasi terhadap masukan yang ketat;
d) manajemen *output* menggunakan *template engine* yang menerapkan fitur *escape* saat memanggil variabel dengan output HTML/Javascipt;
e) terdapat pemetaan *routing* berdasarkan metode HTTP yang ingin digunakan, yaitu GET, POST, PUT, PATCH, dan DELETE;
f) memiliki kendali dan manajemen autentifikasi dan otorasi yang baik;

Terdapat beberapa *web framework* atau CMS yang telah memenuhi kriteria atau persyaratan tersebut. Selain berdasarkan persyaratan minimal, pemilihan *web framework*/CMS juga didasarkan pada komparasi dan pemeringkatan terhadap *web framework*/CMS tersebut. Hasil komparasi dan pemeringkatan yang baik, terutama pada sisi keamanan dan performa, menunjukan bahwa *web framework* atau CMS tersebut memiliki sistem keamanan baik. Mengacu pada salah satu karakteristik inklusif, tidak menutup kemungkinan penggunaan berbagai *web framework/CMS*.

Terkait komparasi *web framework*, penelitian ini mengacu pada komparasi sistem keamanan *web framework* yang dilakukan Grunwald (Tufts University) [23], dan didukung juga oleh hackr.io [78] dan veracode [79]. Grunwald mendeskripsikan bagaimana fitur keamanan pada beberapa *web framework* bekerja. Sementara

itu, komparasi tingkat keamanan CMS mengacu pada komparasi yang dilakukan oleh Martinez-Caro *et al* [80], Contu *et al* [81], K.Patel [82], dan didukung juga oleh penelitian tesis Alghofaili [83]. Berdasarkan kriteria pemenuhan persyaratan minimal dan komparasi yang telah dilakukan, daftar *web framework* dan CMS yang masuk lapisan kedua adalah sebagai berikut.

**Tabel 2.7 Daftar *Web Framework* dalam Kriteria Lapisan Kedua**

| No | *Web Framework* | Bahasa Pemrograman | Laman Resmi |
|---|---|---|---|
| 1 | Django | Python | https://www.djangoproject.com/ |
| 2 | Laravel | PHP | https://laravel.com/ |
| 3 | Ruby of Rails | Ruby | https://rubyonrails.org/ |
| 4 | ASP.NET | ASP | https://dotnet.microsoft.com/apps/aspnet |
| 5 | CodeIgniter | PHP | https://codeigniter.com/ |
| 6 | Yii | PHP | https://www.yiiframework.com/ |
| 7 | Express | JavaScript | https://expressjs.com/ |

**Tabel 2.8 Daftar CMS dalam Kriteria Lapisan Kedua**

| No | CMS | Bahasa Pemograman | Persentase | Domain |
|---|---|---|---|---|
| 1 | Wordpress | PHP | 77,9% | 691,237 |
| 2 | Drupal | PHP | 5,6% | 49,834 |
| 3 | Joomla | PHP | 3,7% | 33,029 |
| 4 | Squarespace | PHP | 2.6 | 22,694 |
| 5 | Moodle | PHP | - | - |

Sumber: https://www.experte.com/ [84]

Namun, permasalahannya adalah tidak semua *web framework/CMS* dibangun dengan tingkat keamanan yang sama dan selalu stabil. Ditambah lagi apabila pihak yang melakukan *develop* kurang terampil dan tidak peduli dengan tingkat keamanan *website*. Selain itu, tidak sedikit *web developer* yang melakukan pengembangan

*website* menggunakan *native code* seperti PHP, tanpa bantuan *web framework* atau CMS. Dengan kata lain, tidak semua teknologi *website* yang digunakan telah memenuhi hal-hal yang disyaratkan oleh lapisan ini. Oleh karena itu, berdasarkan pada masalah tersebut, peneliti melakukan optimalisasi sebagai berikut:

a) menggunakan versi termuktahir lapisan pertama dan mengatur *rules* agar dapat mengatasi serangan siber secara maksimal;
b) menggunakan 3rd *template engine* jika belum tersedia *default*; dan
c) menerapkan sanitasi data yang baik, terutama sebelum menerima masukan dan menampilkan luaran.

### 2.9.6. Skenario Implementasi Berdasarkan Karakteristik *Website*

Penelitian ini menerapkan *multi-layers security* pada *multi-websites.* Seperti yang telah disinggung pada penjelasan sebelumnya, tiap lapisan dan *website* memiliki karakteristik dan kebutuhan yang berbeda-beda. Namun, apabila disimpulkan, metode ini mencoba untuk memenuhi berbagai karakteristik *website* yang memiliki tingkat keamanan yang berbeda-beda. Oleh karena itu, untuk mempermudah identifikasi dan mengefektifkan implementasi metode, peneliti membagi tingkat keamanan *website* menjadi tiga tingkatan, yaitu **tinggi**, **sedang**, dan **rendah**. Adapun dasar penentuan tingkat keamanan adalah pada tabel 2.6 sebagai berikut.

**Tabel 2.9 Kriteria Pemeringkatan Keamanan *Website***

| No | Tingkatan Keamanan | Standar Pemenuhan Kriteria |
|---|---|---|
| 1 | Tinggi | Tingkat keamanan yang tinggi terpenuhi apabila teknologi pada *website* tersebut telah: |

| No | Tingkatan Keamanan | Standar Pemenuhan Kriteria |
|---|---|---|
| | | a) memenuhi kriteria *web framework* atau CMS pada lapisan kedua, seperti disebutkan pada poin 2.8.2;<br>b) memiliki HTTP *Middleware;* dan<br>c) memiliki *Data Sanitizer* dan *Templating Engine*. |
| 2 | Sedang | Tingkat keamanan sedang terpenuhi apabila teknologi *website* tersebut memiliki HTTP *Middleware*, *Data Sanitizer,* dan *Templating Engine* secara *default*. |
| 3 | Rendah | Tingkat keamanan rendah dilabelkan pada *website* yang belum memenuhi standar pemenuhan kriteria tingkat keamanan tinggi dan sedang. |

Terkait tingkatan keamanan tersebut, skenario implementasi metode *multi-layer security* diperlukan untuk mengambarkan proses implementasi upaya mitigasi menggunakan *multi-layer security*. Adapun skenario implementasinya telah dituangkan pada tabel 2.7 di bawah ini.

**Tabel 2.10 Model Skenario Implementasi Metode *Multi-Layer Security***

| No | Lapisan dan Kemampuan *Website* | Tingkat Keamanan | Tindakan |
|---|---|---|---|
| 1 | Semua lapisan telah diaktivasi | **tinggi** | Semua serangan yang diujikan akan diatasi oleh masing-masing lapisan keamanan. |
| 2 | *Web framework* atau CMS telah dilindungi dengan HTTP *middleware, data sanitation,* dan *templating engine.* | **tinggi** | Tindakan yang diambil sama dengan tindakan pertama, namun hanya berlaku per-*website* atau tidak menyeluruh |
| 3 | *Website* telah memenuhi standar pemenuhan kriteria | **medium** | Beberapa lapisan yang tersedia akan menghadapi |

| No | Lapisan dan Kemampuan *Website* | Tingkat Keamanan | Tindakan |
|---|---|---|---|
| | pada tingkatan keamanan sedang | | teknik serangan siber yang dapat terdeteksi saja. |
| 4 | Tingkat keamanan *website* rendah, namun OWASP ModSecurity diaktifkan pada tingatan *WebServer* | **rendah** | Teknik serangan siber akan dihadapi oleh lapisan pertama, yaitu OWASP ModSecurity |
| 5 | Tingkat keamanan *website* rendah, dan OWASP ModSecurity tidak tersedia | **rendah** | tidak dapat melakukan tindakan apapun |

Dengan skenario seperti yang telah diungkapkan di atas, implementasi metode mitigasi dapat terdeskripsikan lebih rinci, karena penelitian ini tidak hanya akan fokus pada satu *website*, tetapi pada beberapa *website*. Dalam hal ini, cara kerja masing-masing lapisan mitigasi juga berbeda-beda. Lapisan pertama (OWASP ModSecurity) bekerja pada lingkup *web server*, sehingga ketika lapisan ini diimplementasikan, semua *website* yang berada di bawah *web server* tersebut langsung diproteksi oleh lapisan ini. Sedikit berbeda dengan lapisan pertama, lapisan kedua bekerja pada area *request* dan *response*, namun tidak berdiri langsung di atas *web server*, melainkan di atas aplikasi. Sementara itu, lapisan yang ketiga, keempat, dan kelima berdiri lebih parsial pada masing-masing aplikasi.