BAB II

TINJAUAN PUSTAKA

2.1 Justifikasi Solusi

Penulis menemukan beberapa penelitian terdahulu yang berkaitan dengan klasifikasi URL berbahaya dan sistem pemfilteran web menggunakan berbagai pendekatan, seperti *machine learning*, deep learning, serta implementasi model deteksi pada sistem berbasis web dan *browser*. Penelitian-penelitian tersebut menunjukkan bahwa pengembangan sistem klasifikasi dan pemblokiran URL berbahaya dapat dilakukan dengan metode yang beragam serta diintegrasikan ke dalam aplikasi atau ekstensi *browser* untuk meningkatkan keamanan akses web secara *real-time*.

2.1.1 *Classification of Malicious URLs Using Machine Learning* [6]

Penelitian yang dilakukan oleh Abad, Gholamy, dan Aslani berjudul "Classification of Malicious URLs Using Machine Learning" berfokus pada pengembangan model *machine learning* untuk mengidentifikasi dan mengklasifikasikan URL berbahaya. Penelitian ini dilatarbelakangi oleh meningkatnya jumlah situs web baru setiap hari, sehingga proses identifikasi situs aman dan berbahaya menjadi semakin sulit apabila hanya mengandalkan pemeriksaan manual atau daftar *Blacklist*. URL berbahaya digunakan dalam berbagai ancaman siber, seperti *phishing*, *malware*, pencurian data, dan serangan terhadap pengguna internet.

Metode yang digunakan dalam penelitian tersebut melibatkan beberapa algoritma *machine learning*, seperti *Support Vector Machine, Random Forest, Decision Tree,* dan *K-Nearest Neighbor*. Selain itu, penelitian tersebut juga menggunakan pendekatan optimasi dan instance selection untuk meningkatkan efisiensi komputasi dalam proses klasifikasi. Fitur yang digunakan berkaitan dengan karakteristik URL dan informasi yang dapat membantu model dalam membedakan URL aman dan URL berbahaya.

Hasil penelitian menunjukkan bahwa *machine learning* dapat digunakan untuk mengklasifikasikan URL berbahaya dengan performa yang baik. Random Forest memberikan hasil yang kuat pada metrik presisi, *recall*, dan *F1-score*, sedangkan SVM juga menunjukkan performa kompetitif meskipun membutuhkan waktu pelatihan lebih tinggi. Penelitian ini menunjukkan bahwa proses pemilihan data dan teknik *preprocessing* memiliki pengaruh penting terhadap performa model klasifikasi URL.

Beberapa poin penting yang dapat diambil oleh penulis adalah sebagai berikut:

- *Machine learning* dapat digunakan untuk mendeteksi dan mengklasifikasikan URL berbahaya berdasarkan pola yang terdapat pada data URL.
- Pendekatan berbasis fitur URL dapat menjadi alternatif terhadap metode *Blacklist* statis.
- Algoritma seperti Random Forest dan SVM dapat dijadikan pembanding dalam evaluasi model klasifikasi URL.
- *Preprocessing* dan pemilihan data berpengaruh terhadap performa sistem klasifikasi.

2.1.2 *Advancing Malicious Website Identification: A Machine Learning Approach Using Granular Feature Analysis* [7]

Penelitian yang dilakukan oleh Tran dan Sovilj ini membahas klasifikasi situs web berbahaya dengan menggunakan analisis fitur yang lebih granular. Penelitian ini tidak hanya berfokus pada klasifikasi biner antara *benign* dan malicious, tetapi juga melakukan klasifikasi ke beberapa kategori situs berbahaya. Hal ini relevan dengan penelitian penulis karena sistem yang dikembangkan juga menggunakan pendekatan klasifikasi multikelas.

Penelitian tersebut menggunakan berbagai fitur yang berasal dari karakteristik URL, domain, konten, host, dan informasi teknis lain yang dapat membantu model mengenali perbedaan antar jenis situs. Model dilatih menggunakan *dataset* berskala besar dengan beberapa kategori klasifikasi. Hasil penelitian

menunjukkan bahwa kombinasi fitur yang lebih lengkap dapat meningkatkan performa klasifikasi situs web berbahaya. Penelitian tersebut juga menunjukkan bahwa beberapa fitur URL dan fitur berbasis konten memiliki kontribusi penting terhadap hasil klasifikasi.

Beberapa poin penting yang dapat diambil penulis adalah sebagai berikut:

- Klasifikasi situs web berbahaya dapat dikembangkan dalam bentuk multikelas, bukan hanya *benign* dan *malicious*.
- Fitur URL memiliki peran penting dalam proses klasifikasi situs web berbahaya.
- Analisis fitur membantu memahami atribut apa saja yang paling berpengaruh terhadap keputusan model.
- Pendekatan multikelas relevan dengan penelitian penulis yang mengklasifikasikan URL ke dalam kategori *benign*, *phish*, *malware*, dan *adult*.

2.1.3 *Lightweight Malicious URL Detection Using Deep Learning and Large Language Models* [8]

Penelitian ini membahas deteksi URL berbahaya menggunakan pendekatan deep learning dan large language models. Penelitian ini dilatarbelakangi oleh keterbatasan metode deteksi tradisional yang masih bergantung pada *Blacklist* dan feature engineering manual. Dalam penelitian tersebut, URL direpresentasikan menggunakan embedding dari model seperti BERT, GPT-2, TinyLLaMA, dan T5, kemudian diklasifikasikan menggunakan model deep learning ringan.

Penelitian ini menggunakan *dataset* multikelas yang terdiri dari beberapa kategori URL, seperti *phish*, *benign*, *malware*, dan *defacement*. Hasil penelitian menunjukkan bahwa representasi berbasis language model dapat menangkap pola leksikal dan kontekstual pada URL secara lebih mendalam dibandingkan pendekatan manual. Selain itu, penggunaan model lightweight menunjukkan bahwa pendekatan deep learning dapat diarahkan agar tetap efisien untuk kebutuhan deteksi *real-time*.

Beberapa poin penting yang dapat diambil oleh penulis adalah sebagai berikut:

- Klasifikasi URL berbahaya dapat dilakukan dalam bentuk multikelas.
- Pola karakter dan struktur URL dapat dipelajari oleh model untuk membedakan kategori URL.
- Pendekatan deep learning dan language model dapat meningkatkan kemampuan model dalam memahami pola URL.
- Konsep model ringan mendukung kebutuhan sistem deteksi yang dapat digunakan secara *real-time*.

2.1.4 *GuardSurfing:* Ekstensi *Browser* sebagai Alat Bantu Deteksi Website Phishing dengan Metode Klasifikasi XGBoost [10]

Penelitian dengan judul "GuardSurfing" yang dilakukan oleh Afandi, Al-Dzaki, Qomariasih, dan Wildana berfokus pada pengembangan alat bantu deteksi URL *phishing* berbasis *browser extension*. Penelitian ini menggunakan XGBoost sebagai model klasifikasi dan Flask sebagai backend untuk memproses permintaan deteksi dari *extension*. Penelitian ini dilatarbelakangi oleh meningkatnya serangan *phishing* yang sulit dikenali oleh pengguna umum, terutama pada aktivitas digital seperti e-commerce, media sosial, dan perbankan digital.

Metode yang digunakan dalam penelitian ini adalah klasifikasi *phishing* berdasarkan string URL. Pendekatan tersebut mendukung prinsip privacy-by-design karena sistem tidak perlu mengambil seluruh isi halaman web untuk melakukan analisis. *Browser extension* digunakan sebagai komponen antarmuka pada sisi pengguna, sedangkan backend digunakan untuk menjalankan model klasifikasi. Ketika pengguna mengakses suatu situs, *extension* dapat mengirimkan URL ke backend dan menampilkan hasil analisis kepada pengguna.

Hasil penelitian menunjukkan bahwa model XGBoost mampu memberikan performa klasifikasi yang tinggi untuk deteksi *phishing*. Penelitian ini juga menunjukkan bahwa *browser extension* dapat digunakan sebagai media implementasi model *machine learning* agar sistem deteksi dapat bekerja secara lebih praktis dan dekat dengan aktivitas browsing pengguna.

Beberapa poin penting yang dapat diambil oleh penulis adalah sebagai berikut:

- Integrasi *machine learning* dengan *browser extension* memungkinkan deteksi URL secara *real-time* pada sisi pengguna.
- Pendekatan berbasis string URL dapat menjaga privasi pengguna karena tidak harus mengambil seluruh konten halaman.
- XGBoost dapat digunakan sebagai algoritma klasifikasi yang kuat untuk deteksi URL *phishing*.
- Konsep backend ringan dan *browser extension* menjadi acuan dalam pengembangan sistem penulis.

2.1.5 *A Comparative Analysis of Machine Learning Models for URL-Based Phishing Detection [9]*

Penelitian Rafi et al. berjudul "*A Comparative Analysis of Machine Learning Models for URL-Based Phishing Detection*" membandingkan lima algoritma, yaitu *Decision Tree*, *Random Forest*, *Support Vector Machine*, XGBoost, dan *Stacking Classifier* untuk mendeteksi URL *phishing*. Penelitian tersebut menggunakan 11.055 URL dengan 30 fitur yang mencakup karakteristik leksikal, *host-based*, isi halaman, dan informasi eksternal.

Hasil pengujian menunjukkan bahwa XGBoost memperoleh performa tertinggi dengan akurasi sekitar 97,4% dan ROC-AUC sebesar 0,9730. Temuan tersebut menjadi salah satu dasar pemilihan XGBoost dalam penelitian ini karena menunjukkan kemampuan XGBoost dalam menangani hubungan nonlinier, menggabungkan berbagai jenis fitur, serta menghasilkan probabilitas prediksi yang dapat digunakan sebagai nilai tingkat kepercayaan.

Beberapa poin penting yang dapat diambil oleh penulis adalah sebagai berikut:

- XGBoost memperoleh performa tertinggi dibandingkan empat model pembanding.
- Perbandingan model pada dataset yang sama memberikan dasar empiris dalam pemilihan algoritma.
- XGBoost sesuai untuk menangani fitur URL yang beragam dan hubungan yang kompleks.

- *Hyperparameter tuning* diperlukan agar model sesuai dengan karakteristik dataset.

2.1.6 *NoPhish: Efficient Chrome Extension for Phishing Detection Using Machine Learning Techniques* [13]

Penelitian ini mengembangkan *extension* Chrome bernama "NoPhish" untuk mendeteksi halaman *phishing* menggunakan teknik *machine learning*. Penelitian ini menyatakan bahwa *browser* merupakan salah satu media utama yang sering menjadi target serangan *phishing* karena pengguna banyak melakukan aktivitas digital melalui *browser*, termasuk login akun, transaksi, dan komunikasi.

NoPhish dirancang sebagai *middleware* antara pengguna dan situs *phishing*. Sistem menggunakan beberapa algoritma *machine learning*, yaitu Random Forest, Support Vector Machine, dan K-Nearest Neighbor. *Dataset* yang digunakan berasal dari PhishTank dengan ekstraksi 22 fitur utama yang berkaitan dengan karakteristik URL dan indikator halaman web. *Browser extension* bertugas menangkap URL aktif, kemudian sistem melakukan klasifikasi untuk menentukan apakah halaman tersebut berbahaya atau tidak.

Hasil penelitian menunjukkan bahwa Random Forest memberikan performa presisi terbaik dibandingkan algoritma lain yang diuji. Penelitian ini memperkuat bahwa *browser extension* dapat digunakan sebagai lapisan deteksi langsung pada sisi pengguna, sedangkan *machine learning* digunakan sebagai dasar pengambilan keputusan.

Beberapa poin penting yang dapat diambil oleh penulis adalah sebagai berikut:

- *Extension* Chrome dapat digunakan sebagai media proteksi sisi klien terhadap *phishing*.
- Integrasi antara *extension* dan model *machine learning* memungkinkan deteksi URL secara langsung ketika pengguna mengakses situs.
- Fitur URL dan karakteristik halaman web dapat digunakan untuk meningkatkan akurasi klasifikasi.
- Konsep *middleware* antara pengguna dan situs web menjadi dasar bagi

rancangan *browser extension* pada penelitian penulis

2.1.7 *Security Advice on Content Filtering and Circumvention for Parents and Children as Found on YouTube and TikTok* [14]

Penelitian yang dilakukan oleh Elgedawy et al. berjudul "Security Advice on Content Filtering and Circumvention for Parents and Children as Found on YouTube and TikTok" membahas bagaimana orang tua atau administrator dan anak memperoleh informasi mengenai pemfilteran konten dan cara melewati sistem pembatasan konten. Penelitian ini menganalisis video dari platform YouTube dan TikTok yang berkaitan dengan topik pemfilteran konten, kontrol orang tua atau administrator, dan circumvention.

Hasil penelitian menunjukkan bahwa terdapat banyak konten yang memberikan informasi mengenai penggunaan atau pelewatan sistem pemfilteran. Penelitian ini juga menyoroti bahwa anak dapat menemukan panduan yang bersifat actionable untuk melewati pembatasan tertentu, sementara pembahasan mengenai risiko dan etika dari tindakan tersebut tidak selalu disampaikan secara seimbang. Hal ini menunjukkan bahwa sistem kontrol orang tua atau administrator tidak hanya perlu berfokus pada kemampuan memblokir konten, tetapi juga perlu mempertimbangkan potensi *bypass* yang dapat dilakukan oleh pengguna.

Beberapa poin penting yang dapat diambil oleh penulis adalah sebagai berikut:

- Pemfilteran konten dan kontrol orang tua atau administrator dapat dilewati apabila tidak dirancang dengan mekanisme proteksi tambahan.
- VPN, *proxy*, dan perubahan konfigurasi perangkat dapat menjadi bagian dari skenario *bypass* yang perlu dipertimbangkan.
- Sistem kontrol orang tua atau administrator perlu memiliki pendekatan berlapis agar tidak hanya bergantung pada satu mekanisme pemblokiran.
- Penelitian penulis menggunakan temuan ini sebagai dasar untuk menambahkan perangkat lunak, *browser policy*, dan *application control* sebagai penguatan proteksi.

2.1.8 *Browser Policy* dan Manajemen *Extension* pada Chrome dan Microsoft Edge [11], [12], [15]

Selain penelitian ilmiah, dokumentasi teknis resmi dari Google Chrome Enterprise dan Microsoft Edge digunakan sebagai landasan implementasi perangkat lunak *desktop*. Dokumentasi Chrome Enterprise menjelaskan bahwa *extension* dapat diatur melalui kebijakan seperti ExtensionInstallForcelist, yaitu kebijakan untuk memasang *extension* secara otomatis tanpa interaksi pengguna dan mencegah pengguna menghapus atau menonaktifkan *extension* tersebut. Google juga menyediakan mekanisme pengelolaan *extension* melalui kebijakan, seperti mengizinkan, memblokir, atau memasang *extension* secara otomatis pada perangkat.

Microsoft Edge juga mendukung pengelolaan *extension* melalui kebijakan, seperti ExtensionInstallForcelist, ExtensionInstallBlocklist, dan ExtensionInstallAllowlist. Kebijakan tersebut dapat digunakan untuk menentukan *extension* yang dipasang secara otomatis, *extension* yang tidak boleh dipasang, dan *extension* yang boleh digunakan. Dalam konteks kontrol orang tua atau administrator, kebijakan ini dapat dimanfaatkan untuk memperkuat *browser extension* agar tidak mudah dihapus oleh pengguna non-administrator dan untuk membatasi pemasangan *extension* yang dapat digunakan sebagai *bypass*, seperti VPN atau *proxy extension*.

Beberapa poin penting yang dapat diambil oleh penulis adalah sebagai berikut:

- *Browser policy* dapat digunakan untuk mengatur pemasangan, pembatasan, dan pengelolaan *extension* pada *browser* berbasis Chromium.
- ExtensionInstallForcelist dapat digunakan untuk memasang *extension* secara otomatis dan mencegah pengguna menghapus atau menonaktifkannya.
- ExtensionInstallBlocklist dan ExtensionInstallAllowlist dapat digunakan untuk membatasi *extension* yang berpotensi menjadi celah

bypass.

- Perangkat lunak *desktop* dapat dikembangkan untuk membantu orang tua atau administrator menerapkan kebijakan tanpa harus melakukan konfigurasi manual yang kompleks.

2.1.9 *Application Control* dan *AppLocker* pada Sistem Operasi Windows [16]

Dokumentasi Microsoft mengenai AppLocker menjelaskan bahwa AppLocker dapat digunakan untuk mengontrol aplikasi dan file apa saja yang boleh dijalankan oleh pengguna pada sistem operasi Windows. Jenis berkas yang dapat dikontrol mencakup berkas eksekusi, skrip, berkas Windows Installer, pustaka tautan dinamis, aplikasi terpaket, dan pemasang aplikasi terpaket. Dengan mekanisme ini, administrator dapat membuat aturan untuk membatasi eksekusi aplikasi tertentu pada perangkat.

Dalam konteks penelitian penulis, *application control* menjadi relevan karena sistem kontrol orang tua atau administrator tidak hanya menghadapi risiko akses URL berbahaya, tetapi juga potensi penggunaan aplikasi yang dapat melewati proteksi. Contohnya adalah aplikasi VPN, *proxy*, *browser* alternatif, atau aplikasi lain yang digunakan untuk menghindari pemblokiran pada *browser* utama. Oleh karena itu, *application control* dapat digunakan sebagai lapisan tambahan untuk membatasi aplikasi yang berpotensi melemahkan proteksi.

Untuk penggunaan *application control* dalam penelitian ini diposisikan sebagai komponen pendukung, bukan fokus utama klasifikasi URL. Fokus utama penelitian tetap berada pada model *machine learning*, *browser extension*, dan mekanisme pemblokiran URL pada sisi *browser*. *Application control* digunakan sebagai bagian dari rancangan perangkat lunak *desktop* untuk memperkuat proteksi dan mengurangi celah *bypass* pada lingkungan Windows.

Beberapa poin penting yang dapat diambil oleh penulis adalah sebagai berikut:

- *Application control* dapat digunakan untuk membatasi aplikasi yang dapat dijalankan oleh pengguna.
- AppLocker menyediakan mekanisme pengaturan aplikasi berbasis

kebijakan pada Windows.

- Pembatasan aplikasi dapat membantu mengurangi celah *bypass* melalui VPN, *proxy*, atau *browser* alternatif.

2.1.10 *API Extension Chrome dan Mekanisme Pemblokiran pada Manifest V3 [17]*

Dokumentasi Chrome for Developers menjelaskan bahwa *extension* pada Manifest V3 memiliki perubahan dalam mekanisme pemantauan dan pemblokiran permintaan. Pada Manifest V3, penggunaan webRequestBlocking dibatasi untuk sebagian besar *extension*, dan Chrome mendorong penggunaan declarativeNetRequest sebagai pendekatan pemblokiran berbasis aturan.

Beberapa poin penting yang dapat diambil oleh penulis adalah sebagai berikut:

- Manifest V3 memengaruhi cara *extension* melakukan pemantauan dan pemblokiran permintaan.
- declarativeNetRequest dapat digunakan untuk melakukan pemblokiran berbasis aturan.
- webRequestBlocking memiliki pembatasan pada Manifest V3, tetapi masih dapat digunakan pada *extension* yang dipasang melalui kebijakan dalam kondisi tertentu.

2.1.11 Justifikasi Pemilihan Metrik Evaluasi Klasifikasi URL

Evaluasi klasifikasi URL pada penelitian terdahulu umumnya menggunakan kombinasi akurasi, presisi, *recall*, F1-*score*, matriks konfusi, dan ROC-AUC. Penelitian yang dilakukan oleh Rafi et al. [9] menggunakan metrik tersebut untuk membandingkan *Decision Tree*, *Random Forest*, Support Vector Machine, XGBoost, dan *Stacking Classifier* pada deteksi URL *phishing*. Penggunaan beberapa metrik diperlukan karena akurasi saja hanya menunjukkan proporsi keseluruhan prediksi yang benar dan belum menggambarkan kemampuan model pada setiap kelas.

Pada penelitian ini, *macro F1-score* ditetapkan sebagai metrik utama karena klasifikasi dilakukan terhadap empat kelas dengan jumlah data yang berbeda. *Macro F1-score* memberikan bobot yang sama terhadap kelas benign, *phish, malware,* dan *adult*, sehingga performa kelas dengan jumlah sampel lebih sedikit tetap diperhitungkan. Hal tersebut juga sesuai dengan proses *hyperparameter tuning* yang menggunakan *macro F1-score* sebagai metrik evaluasi. *Weighted F1-score* digunakan sebagai metrik pendukung untuk menggambarkan performa berdasarkan distribusi jumlah data setiap kelas. Selain itu, presisi dan *recall* dianalisis pada setiap kelas. *Recall* kelas *phish, malware,* dan *adult* digunakan untuk mengetahui jumlah URL berbahaya yang berhasil dikenali, sedangkan presisi digunakan untuk mengetahui kemungkinan data dari kelas lain salah diklasifikasikan sebagai ancaman. Matriks konfusi digunakan untuk mengidentifikasi pola kesalahan antar kelas, sedangkan ROC-AUC digunakan sebagai pengukuran tambahan terhadap kemampuan model membedakan kelas berdasarkan probabilitas prediksi

2.2 Tinjauan Teori

Pada bagian ini, penulis meninjau berbagai teori dan konsep yang dapat menjadi dasar ilmiah dalam perancangan solusi terhadap permasalahan yang telah dijelaskan. Pemahaman terhadap konsep klasifikasi URL, fitur struktural URL, ekstraksi fitur, *machine learning*, klasifikasi multikelas, serta algoritma XGBoost diperlukan untuk mengembangkan model yang mampu membedakan URL ke dalam kategori *benign*, *phish*, *malware*, dan *adult*. Selain itu, pembahasan mengenai sistem pemfilteran web, *browser extension*, REST API, *browser policy*, serta mekanisme pemblokiran situs web diperlukan untuk mendukung proses integrasi model ke dalam lingkungan *browser*. Penulis juga membahas teknik *circumvention*, pendekatan *anti-bypass*, *application control*, *local agent*, pemantauan sistem, dan *persistence* sebagai dasar dalam mengembangkan proteksi berlapis yang dapat mengurangi peluang pengguna melewati mekanisme pembatasan akses.

2.2.1 Klasifikasi URL

Klasifikasi URL merupakan proses pengelompokan alamat web ke dalam kategori tertentu berdasarkan karakteristik yang terdapat pada URL. Dalam konteks keamanan siber, klasifikasi URL digunakan untuk membedakan URL yang aman dan URL yang berpotensi berbahaya, seperti *phishing*, *malware*, *spam*, *defacement*, maupun konten yang tidak sesuai dengan kebijakan akses pengguna. URL berbahaya sering digunakan sebagai pintu masuk serangan siber karena pengguna dapat diarahkan ke halaman palsu, file berbahaya, formulir pencurian kredensial, atau konten ilegal melalui tautan yang tampak menyerupai situs resmi [18].

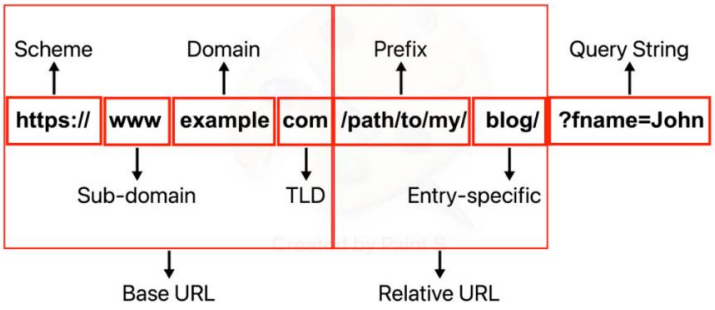
Metode klasifikasi URL telah berkembang dari pendekatan berbasis *Blacklist* menuju pendekatan berbasis data. *Blacklist* bekerja dengan mencocokkan URL atau domain yang diakses pengguna dengan daftar situs berbahaya yang telah diketahui sebelumnya. Pendekatan ini mudah diterapkan, tetapi memiliki kelemahan karena tidak mampu mengenali URL baru yang belum terdaftar. Sementara itu, pendekatan

berbasis *machine learning* memungkinkan sistem mempelajari pola dari URL yang telah diberi label sehingga mampu melakukan prediksi terhadap URL baru berdasarkan karakteristik struktural dan leksikalnya [18].

2.2.2 Fitur Struktural URL

Fitur struktural URL merupakan karakteristik yang diperoleh dari bentuk dan susunan URL. Fitur ini dapat diekstraksi langsung dari string URL tanpa harus membuka isi halaman web. Pendekatan ini penting karena proses klasifikasi dapat dilakukan lebih cepat, lebih ringan, dan lebih menjaga privasi pengguna. Beberapa penelitian menunjukkan bahwa fitur URL seperti panjang URL, jumlah karakter khusus, jumlah angka, struktur domain, jumlah *subdomain*, penggunaan HTTPS, dan *token* mencurigakan dapat membantu model dalam membedakan URL normal dan URL berbahaya [19].

Secara umum, fitur dalam deteksi URL berbahaya dapat dibagi menjadi beberapa jenis, yaitu fitur leksikal, fitur berbasis host, fitur berbasis jaringan, dan fitur berbasis konten. Fitur leksikal berasal dari struktur teks URL, seperti panjang URL, jumlah titik, jumlah garis miring, jumlah tanda tanya, jumlah tanda sama dengan, dan keberadaan kata tertentu seperti yang ditunjukkan pada Gambar 2.1. Host-based features berkaitan dengan informasi host atau domain, seperti usia domain, reputasi domain, atau informasi WHOIS. Network-based features berkaitan dengan informasi jaringan, sedangkan berbasis konten features diperoleh dari isi halaman web [19].



Gambar 2.1 Struktur URL [41]

2.2.3 Ekstraksi Fitur URL

Ekstraksi fitur URL merupakan proses mengubah URL yang berbentuk teks menjadi representasi numerik agar dapat diproses oleh algoritma *machine learning*. URL pada dasarnya merupakan data tidak terstruktur, sehingga model tidak dapat langsung memproses URL mentah tanpa proses transformasi. Oleh karena itu, fitur-fitur penting dari URL perlu diekstraksi agar model dapat mengenali pola yang membedakan URL *benign* dan URL berbahaya [20].

Fitur yang dapat diekstraksi dari URL meliputi panjang URL, panjang domain, jumlah *subdomain*, jumlah karakter khusus, jumlah digit, jumlah huruf, rasio digit terhadap panjang URL, jumlah parameter, panjang *path*, panjang *query*, penggunaan HTTPS, keberadaan alamat IP, jumlah titik, jumlah tanda hubung, jumlah simbol @, serta keberadaan kata kunci mencurigakan. Selain itu, fitur seperti entropy dan randomness score juga dapat digunakan untuk mengukur tingkat keacakan URL. URL berbahaya sering kali memiliki pola yang tidak wajar, seperti domain acak, *subdomain* berlapis, atau penggunaan kata-kata yang meniru merek tertentu [18], [20].

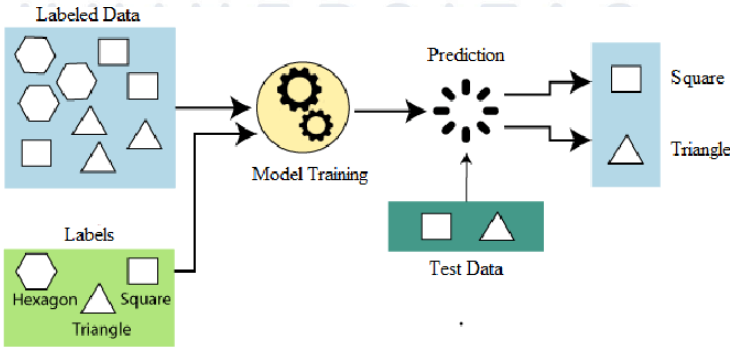
2.2.4 *Machine Learning*

Machine learning merupakan cabang dari kecerdasan buatan yang memungkinkan sistem komputer mempelajari pola dari data dan membuat prediksi terhadap data baru. Dalam *machine learning*, sistem tidak diprogram secara eksplisit untuk setiap kondisi, tetapi dilatih menggunakan data historis agar mampu mengenali pola tertentu. Pendekatan ini banyak digunakan dalam klasifikasi teks, pengenalan Gambar, deteksi penipuan, rekomendasi, dan keamanan siber [22]. Dalam keamanan siber, *machine learning* banyak diterapkan untuk mendeteksi ancaman yang memiliki pola kompleks, seperti *spam*, *phishing*, *malware*, intrusi jaringan, dan URL berbahaya. Hal ini disebabkan karena ancaman siber terus berubah dan tidak selalu dapat ditangani menggunakan aturan

statis. Dengan *machine learning*, sistem dapat mempelajari pola dari data berlabel dan melakukan generalisasi terhadap data baru yang belum pernah dilihat sebelumnya [18].

2.2.4.1 *Supervised Learning*

Supervised learning merupakan pendekatan *machine learning* yang menggunakan data berlabel dalam proses pelatihan model seperti yang ditunjukkan pada Gambar 2.2. Setiap data pelatihan terdiri dari masukan dan label keluaran yang benar. Model kemudian mempelajari hubungan antara masukan dan label tersebut agar dapat melakukan prediksi terhadap data baru [22]. *Supervised learning* banyak digunakan dalam permasalahan klasifikasi dan regresi. Dalam klasifikasi URL, masukan berupa fitur-fitur URL, sedangkan label berupa kategori URL seperti *benign*, *phish*, *malware*, atau *adult*. Model dilatih menggunakan data yang sudah memiliki label sehingga dapat mempelajari pola dari masing-masing kelas. Setelah proses pelatihan selesai, model dapat digunakan untuk memprediksi kelas dari URL baru yang belum pernah dilihat sebelumnya. *Supervised learning* sesuai digunakan dalam penelitian ini karena *dataset* yang digunakan memiliki label kelas. Dengan adanya label tersebut, model dapat dievaluasi secara objektif menggunakan metrik klasifikasi. Evaluasi dilakukan dengan membandingkan hasil prediksi model dengan label sebenarnya pada data uji.



Gambar 2.2 Metode *Supervised Learning* [42]

2.2.4.2 Klasifikasi Multikelas

Klasifikasi multikelas atau *multiclass classification* merupakan metode klasifikasi yang digunakan ketika data memiliki lebih dari dua kelas. Berbeda dengan klasifikasi biner yang hanya membedakan dua kategori, klasifikasi multikelas memungkinkan model memprediksi satu kelas dari beberapa kelas yang tersedia. Dalam konteks keamanan siber, klasifikasi multikelas memungkinkan sistem membedakan beberapa jenis ancaman secara lebih spesifik, seperti *phishing*, *malware*, *defacement*, *spam*, dan *benign* [21].

Pada penelitian ini, klasifikasi multikelas digunakan karena sistem tidak hanya membedakan URL aman dan URL berbahaya. Sistem mengklasifikasikan URL ke dalam empat kelas, yaitu *benign*, *phish*, *malware*, dan *adult*. Pendekatan ini memberikan informasi yang lebih rinci karena sistem dapat mengetahui jenis ancaman yang terdeteksi. Informasi tersebut kemudian dapat ditampilkan pada halaman pemblokiran agar pengguna atau orang tua atau administrator memahami alasan pemblokiran.

Klasifikasi multikelas juga berpengaruh pada pemilihan model dan metrik evaluasi. Model harus mampu membedakan lebih dari dua kelas, sedangkan evaluasi tidak cukup hanya menggunakan akurasi. Metrik seperti presisi, *recall*, *F1-score*, dan matriks konfusi diperlukan untuk melihat performa model pada masing-masing kelas [24].

2.2.5 XGBoost

XGBoost (eXtreme Gradient Boosting) merupakan algoritma *machine learning* berbasis *gradient boosting decision tree* yang dirancang untuk memberikan performa klasifikasi dan prediksi yang tinggi dengan efisiensi komputasi yang baik. Algoritma ini dikembangkan sebagai pengembangan dari metode gradient boosting tradisional dengan menambahkan optimisasi sistem, regularisasi, serta kemampuan

pemrosesan paralel sehingga mampu menangani *dataset* berskala besar secara lebih cepat dan akurat [23]. Secara umum, XGBoost bekerja dengan membangun sekumpulan *decision tree* secara bertahap (sequential boosting), di mana setiap pohon baru dibangun untuk memperbaiki kesalahan prediksi yang dihasilkan oleh pohon sebelumnya. Pendekatan ini memungkinkan model mempelajari pola data secara lebih efektif dan menghasilkan performa yang lebih baik dibandingkan metode single decision tree [23]. Selain itu, XGBoost menggunakan fungsi objektif yang terdiri dari fungsi loss dan regularisasi untuk mengurangi risiko *overfitting* sekaligus meningkatkan kemampuan generalisasi model terhadap data baru [23]. Fungsi objektif pada XGBoost dapat dirumuskan sebagai berikut:

$$\mathrm{Obj} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Pada persamaan tersebut, $l(y_i, \hat{y}_i)$ menyatakan fungsi loss antara nilai aktual y_i dan hasil prediksi $\hat{y}_i$, sedangkan $\Omega(f_k)$ menyatakan fungsi regularisasi pada setiap pohon keputusan f_k. XGBoost memiliki beberapa keunggulan dibandingkan algoritma *machine learning* lainnya, seperti kemampuan menangani data berdimensi besar, mendukung parallel processing, memiliki mekanisme regularisasi untuk mengurangi *overfitting*, serta mendukung penanganan missing value dan data sparse secara efisien [24].

XGBoost banyak digunakan untuk deteksi *phishing*, klasifikasi URL berbahaya, deteksi *malware*, dan sistem deteksi ancaman lainnya karena kemampuannya dalam mengenali pola kompleks dari fitur struktural dan leksikal data digital [24]. Penelitian terbaru menunjukkan bahwa XGBoost mampu memberikan performa klasifikasi yang tinggi bahkan pada kondisi *dataset* tidak seimbang (imbalanced *dataset*) melalui kombinasi teknik boosting dan regularisasi yang efektif [24]. Pemilihan XGBoost dalam penelitian ini juga didukung oleh penelitian Rafi et al. [9], yang membandingkan *Decision Tree*, *Random Forest*, Support Vector Machine,

XGBoost, dan *Stacking Classifier* untuk deteksi URL *phishing*. Dalam penelitian tersebut, XGBoost memberikan performa tertinggi di antara model individual yang dibandingkan. Temuan tersebut digunakan sebagai bukti pendukung bahwa XGBoost sesuai untuk mempelajari fitur struktural URL. Meskipun demikian, penelitian Rafi et al. merupakan penelitian klasifikasi biner dan berbentuk *preprint*, sehingga hasilnya tidak digunakan sebagai bukti bahwa XGBoost selalu menjadi algoritma terbaik untuk seluruh dataset dan permasalahan klasifikasi URL.

2.2.6 *Balanced & Imbalanced Dataset*

Dalam *machine learning*, distribusi data pada setiap kelas sangat memengaruhi kinerja model. *Dataset* dikatakan balanced apabila jumlah sampel pada setiap kelas relatif seimbang, sehingga model dapat mempelajari pola dari tiap kelas secara proporsional. Sebaliknya, *dataset* imbalanced terjadi ketika terdapat perbedaan signifikan jumlah data antar kelas, misalnya satu kelas jauh lebih dominan dibanding kelas lainnya. Kondisi ini sering ditemukan pada permasalahan dunia nyata seperti deteksi penipuan, deteksi penyakit langka, dan keamanan siber [25].

Ketidakseimbangan data dapat menyebabkan model cenderung bias terhadap kelas mayoritas dan mengabaikan kelas minoritas. Akibatnya, meskipun nilai akurasi terlihat tinggi, model mungkin gagal mengidentifikasi kelas minoritas dengan baik. Oleh karena itu, evaluasi model pada *dataset* imbalanced tidak cukup hanya menggunakan akurasi, tetapi juga metrik lain seperti presisi, *recall*, *F1-score*, dan matriks konfusi untuk memberikan gambaran performa yang bermakna [25].

2.2.7 Evaluasi Klasifikasi Multikelas

Dalam klasifikasi multikelas, evaluasi dilakukan terhadap setiap kelas menggunakan pendekatan *one-versus-rest*. Kelas yang sedang dievaluasi diperlakukan sebagai kelas positif, sedangkan kelas lainnya diperlakukan sebagai kelas negatif. Berdasarkan pendekatan tersebut, nilai *true positive*, *false positive*, *false negative*, dan *true negative* dapat dihitung untuk setiap kelas. Pada konteks klasifikasi URL, *false negative*

pada kelas berbahaya menunjukkan URL *phish*, *malware*, atau *adult* yang tidak berhasil dikenali sebagai kelas sebenarnya. Sementara itu, *false positive* menunjukkan URL dari kelas lain yang salah diprediksi sebagai suatu kelas ancaman. Oleh karena itu, evaluasi perlu memperhatikan presisi dan *recall* setiap kelas, bukan hanya nilai akurasi keseluruhan.

2.2.8 Sistem Pemfilteran Web

Sistem pemfilteran web merupakan mekanisme yang digunakan untuk menyaring dan membatasi akses terhadap konten internet yang dianggap tidak pantas, berbahaya, atau ilegal. Sistem ini bekerja dengan menganalisis konten web, email, maupun URL untuk menentukan apakah suatu halaman diperbolehkan atau harus diblokir. Secara umum, pemfilteran web berfungsi sebagai lapisan keamanan yang bertindak seperti *firewall* guna mencegah pengguna mengakses situs berisiko serta melindungi jaringan dari konten berbahaya [30].

Implementasi pemfilteran web dapat dilakukan melalui berbagai pendekatan, seperti pemfilteran berbasis *browser*, pemfilteran sisi klien, pemfilteran berbasis jaringan, hingga pemfilteran pada level ISP. Pemfilteran berbasis *browser* biasanya diimplementasikan melalui ekstensi pihak ketiga pada *browser*, sedangkan pemfilteran berbasis jaringan diterapkan pada tingkat jaringan menggunakan *proxy* atau gateway untuk mengontrol akses seluruh pengguna dalam suatu organisasi [30].

Selain itu, pemfilteran web modern tidak hanya memeriksa domain, tetapi juga menganalisis struktur HTML, teks, Gambar, serta reputasi URL untuk meningkatkan akurasi deteksi konten berbahaya. Teknik seperti kata kunci matching, URL reputation analysis, dan klasifikasi berbasis *machine learning* sering digunakan untuk menentukan apakah suatu halaman termasuk kategori aman atau terlarang. Beberapa sistem bahkan menggunakan algoritma keputusan awal (early decision algorithm) yang dapat mengklasifikasikan halaman berdasarkan sebagian konten awal tanpa perlu memindai seluruh halaman, sehingga

meningkatkan efisiensi proses pemfilteran [30].

2.2.9 *Browser Extension*

Browser extension merupakan perangkat lunak tambahan yang dipasang pada web *browser* untuk memperluas fungsi dan kemampuan *browser* tanpa perlu memodifikasi inti sistem *browser* itu sendiri. Ekstensi ini memungkinkan pengembang menambahkan berbagai fitur seperti pemblokiran iklan, pengelolaan kata sandi, penerjemah bahasa, hingga mekanisme keamanan web secara langsung pada sisi pengguna (sisi klien). *Browser extension* bekerja dengan memanfaatkan Application Programming Interface (API) yang disediakan oleh *browser* untuk mengakses tab, memonitor aktivitas browsing, memodifikasi halaman web, serta berinteraksi dengan konten yang sedang diakses pengguna [31].

Pada *browser* modern seperti Google Chrome dan Microsoft Edge, pengembangan *extension* umumnya menggunakan standar *Manifest V3* yang memperkenalkan arsitektur berbasis *service worker* sebagai pengganti halaman *background*. Pendekatan ini dirancang untuk meningkatkan keamanan, efisiensi penggunaan sumber daya, serta membatasi akses ekstensi terhadap proses *browser* secara langsung [32]. Dalam implementasinya, *browser extension* dapat menggunakan beberapa komponen utama seperti content script, *service worker*, dan *browser* API untuk melakukan komunikasi dengan halaman web maupun layanan backend eksternal [29]. Salah satu kemampuan utama *browser extension* adalah melakukan pemantauan URL dan navigasi halaman web secara *real-time* melalui Tabs API dan Web Navigation API. Dengan kemampuan tersebut, *extension* dapat mendeteksi URL yang diakses pengguna, melakukan analisis terhadap URL tersebut, dan mengambil tindakan tertentu seperti menampilkan peringatan atau memblokir akses ke halaman berbahaya [31]. Pendekatan ini banyak digunakan pada sistem keamanan berbasis *browser* karena memungkinkan proteksi dilakukan langsung pada sisi pengguna tanpa memerlukan kontrol pada tingkat jaringan maupun sistem operasi [33].

2.2.9.1 Manifest V3

Manifest V3 merupakan standar terbaru dalam pengembangan *browser extension* pada Google Chrome yang digunakan untuk meningkatkan keamanan, privasi, dan efisiensi sistem dibandingkan Manifest V2 [29]. Manifest V3 menggunakan berkas manifest.json sebagai konfigurasi utama *extension* yang berisi informasi seperti *permissions*, *service worker*, *content script*, dan pengaturan API *browser* [29]. Salah satu perubahan utama pada Manifest V3 adalah penggunaan *service worker* sebagai pengganti halaman *background*. Pendekatan ini memungkinkan *extension* berjalan secara *event-driven* sehingga lebih ringan dan efisien dalam penggunaan sumber daya *browser* [29]. Manifest V3 juga menyediakan berbagai API seperti Tabs API dan Web Navigation API yang memungkinkan *extension* memonitor URL, mendeteksi navigasi halaman, dan melakukan pemblokiran situs web secara *real-time* [29].

2.2.9.2 *Service Worker*

Service Worker merupakan komponen pada Manifest V3 yang digunakan sebagai proses latar belakang (*background process*) pada *browser extension*. *Service worker* bekerja secara *event-driven*, yaitu hanya aktif ketika terdapat *event* tertentu seperti perubahan tab, navigasi halaman, atau permintaan API sehingga lebih efisien dalam penggunaan memori dan CPU dibandingkan halaman *background* pada Manifest V2 [31]. *Service worker* digunakan untuk menjalankan logika utama *extension* seperti memonitor aktivitas *browser*, mengambil URL halaman aktif, melakukan komunikasi dengan *backend* API, serta menentukan tindakan pemblokiran terhadap situs web berbahaya [31]. Pendekatan ini memungkinkan sistem deteksi URL berjalan secara *real-time* dengan konsumsi sumber daya yang lebih ringan dan aman.

2.2.9.3 *Web Request API* dan *Declarative Net Request*

Web Request API merupakan API pada *extension* Chrome yang memungkinkan *extension* mengamati dan menganalisis permintaan jaringan yang dibuat oleh *browser*. API ini dapat digunakan untuk memantau permintaan, melihat URL tujuan, dan dalam kondisi tertentu melakukan modifikasi atau pemblokiran permintaan sebelum halaman dimuat.

DeclarativeNetRequest bekerja dengan cara mendefinisikan aturan pemblokiran atau pengalihan permintaan yang diproses oleh *browser*. Dengan pendekatan ini, *extension* tidak harus memproses setiap permintaan secara langsung, sehingga dapat meningkatkan performa dan mengurangi risiko penyalahgunaan API. Namun, mekanisme deklaratif juga memiliki keterbatasan apabila sistem membutuhkan keputusan dinamis dari model *machine learning*.

2.2.9.4 Chrome Tabs API

Chrome Tabs API merupakan API bawaan pada *browser extension* yang memungkinkan *extension* mengakses dan memonitor informasi tab *browser* seperti URL aktif, status halaman, judul halaman, serta aktivitas navigasi pengguna [35]. API ini banyak digunakan dalam pengembangan sistem keamanan berbasis *browser extension* karena memungkinkan proses deteksi URL dilakukan secara *real-time* ketika pengguna membuka halaman web tertentu. Pada penelitian ini, Chrome Tabs API digunakan untuk mengambil URL dari tab aktif dan mengirimkannya ke sistem *machine learning* untuk dilakukan proses klasifikasi. Melalui API ini, *extension* dapat mendeteksi perubahan halaman secara otomatis dan melakukan tindakan seperti menampilkan peringatan atau memblokir akses terhadap situs web berbahaya [35].

2.2.9.5 *Browser Policy*

Selain kemampuan teknis *extension* dalam memantau URL dan melakukan pemblokiran, sistem berbasis *browser extension* juga perlu mempertimbangkan aspek pengelolaan *browser*. *Extension* yang dipasang secara manual masih memiliki keterbatasan karena pengguna dapat mencoba menonaktifkan *extension*, menghapus *extension*, membuka situs melalui mode Incognito atau InPrivate, atau memasang *extension* lain yang dapat mengubah jalur akses. Kondisi tersebut menunjukkan bahwa *browser extension* perlu didukung oleh mekanisme administrasi agar proteksi tidak hanya bergantung pada pemasangan manual.

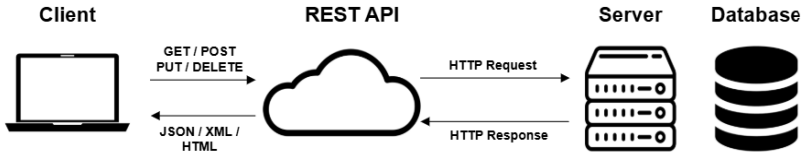
Browser policy merupakan mekanisme administrasi yang digunakan untuk mengatur perilaku *browser* dan *extension*. Pada Google Chrome dan Microsoft Edge, *browser policy* dapat digunakan untuk mengatur pemasangan *extension*, membatasi *extension* tertentu, mengelola mode Incognito atau InPrivate, serta menerapkan konfigurasi tertentu pada *browser* [32], [33]. Firefox juga menyediakan mekanisme kebijakan melalui Group Policy atau file policies.json, sehingga konfigurasi *browser* dapat dikelola secara lebih terstruktur [53].

Dalam konteks sistem pembatasan akses situs web, *browser policy* berperan sebagai lapisan penguat agar *extension* tidak hanya bergantung pada pemasangan manual. Beberapa kebijakan yang relevan antara lain ExtensionInstallForcelist, ExtensionInstallBlocklist, ExtensionInstallAllowlist, dan IncognitoModeAvailability. ExtensionInstallForcelist digunakan untuk memasang *extension* secara otomatis apabila ID *extension* resmi tersedia. ExtensionInstallBlocklist digunakan untuk membatasi *extension* tertentu, sedangkan ExtensionInstallAllowlist digunakan untuk menentukan *extension* yang diizinkan.

IncognitoModeAvailability digunakan untuk mengatur ketersediaan mode Incognito. Dengan adanya *browser policy*, sistem dapat memperkuat penerapan *extension* pada lingkungan *browser desktop*.

2.2.10 REST API

REST API (*Representational State Transfer Application Programming Interface*) merupakan arsitektur komunikasi berbasis web yang digunakan untuk pertukaran data antara client dan server melalui protokol HTTP [36]. Seperti yang ditunjukkan oleh Gambar 2.3, REST API memungkinkan aplikasi saling berkomunikasi menggunakan metode seperti GET, POST, PUT, dan DELETE dengan format pertukaran data yang umum digunakan seperti *JSON* [37]. Pendekatan ini banyak digunakan karena bersifat ringan, fleksibel, dan mudah diintegrasikan dengan berbagai platform dan layanan web. Pada penelitian ini, REST API digunakan sebagai media komunikasi antara *browser extension* dan model klasifikasi URL yang berjalan pada server *backend*. *Browser extension* mengirimkan URL yang diakses pengguna ke API untuk dilakukan proses klasifikasi, kemudian server mengembalikan hasil prediksi berupa kategori URL dan tingkat kepercayaan model. Penggunaan REST API memungkinkan proses klasifikasi dilakukan secara *real-time* tanpa menjalankan model *machine learning* secara langsung pada *browser extension*.



Gambar 2.3 Alur kerja REST API

2.2.11 Pemblokiran Situs Web

Pemblokiran situs web merupakan mekanisme pengendalian akses yang digunakan untuk membatasi atau mencegah pengguna mengakses situs web yang dianggap berbahaya, ilegal, atau melanggar kebijakan tertentu. Pada sistem keamanan modern, pemblokiran situs web tidak hanya diterapkan pada tingkat jaringan, tetapi juga dapat dilakukan pada

sisi pengguna (sisi klien) melalui *browser extension* dan aplikasi keamanan web. Pendekatan ini memungkinkan sistem melakukan pembatasan akses secara langsung ketika pengguna mencoba membuka URL yang telah teridentifikasi sebagai berbahaya, seperti *phishing*, *malware*, maupun konten illegal. Dalam implementasinya, pemblokiran situs web bekerja dengan mendeteksi URL yang diakses pengguna kemudian melakukan proses klasifikasi atau pencocokan terhadap aturan keamanan tertentu. Apabila URL terdeteksi berbahaya, sistem akan melakukan tindakan menampilkan peringatan, mengalihkan halaman, atau memblokir akses sebelum halaman web berhasil dimuat sepenuhnya pada *browser* pengguna [1]. Pendekatan berbasis *browser extension* memungkinkan proses pemblokiran dilakukan secara *real-time* tanpa memerlukan kontrol langsung pada tingkat jaringan atau ISP. Perkembangan *machine learning* juga mendorong pengembangan pemblokiran situs web yang lebih adaptif dibandingkan pendekatan *Blacklist* tradisional. Sistem berbasis *machine learning* mampu mengidentifikasi pola URL berbahaya secara otomatis sehingga dapat mendeteksi situs baru yang belum terdaftar pada daftar blokir statis [2]. Selain itu, mekanisme pemblokiran situs web berbasis *browser extension* dinilai lebih fleksibel dan mudah diimplementasikan karena dapat bekerja langsung pada *browser* pengguna serta mendukung sistem proteksi *real-time* terhadap ancaman siber.

2.2.12 *Circumvention*

Setelah sistem pemfilteran diterapkan pada *browser*, celah lain yang perlu diperhatikan adalah upaya pengguna untuk melewati pembatasan tersebut. *Circumvention* dalam konteks sistem pemfilteran web merujuk pada upaya untuk menghindari, melewati, atau mengurangi efektivitas mekanisme pembatasan akses yang telah diterapkan. Upaya ini dapat dilakukan melalui berbagai cara, seperti menggunakan jalur akses yang tidak dipantau oleh sistem, mengubah konfigurasi akses, menggunakan VPN atau proxy, memanfaatkan mode private browsing, atau menggunakan aplikasi lain untuk menghindari aturan pemblokiran [14]. Penelitian tersebut menunjukkan bahwa pembahasan tentang cara

melewati *content filtering* dan *parental control* dapat ditemukan pada konten yang ditujukan kepada anak maupun orang tua atau administrator [14]. Hal ini menunjukkan bahwa sistem pemfilteran tidak hanya perlu mampu menentukan apakah suatu URL boleh diakses, tetapi juga perlu mempertimbangkan kemungkinan pengguna mencoba melewati sistem.

Beberapa teknik *circumvention* yang umum ditemukan antara lain penggunaan VPN, *proxy*, perubahan DNS, mode Incognito, pemasangan *extension* VPN atau *anonymizer*, penggunaan *browser* alternatif, penghapusan *extension* proteksi, serta penggunaan aplikasi portable dari lokasi umum seperti folder Downloads atau Temp. Penelitian mengenai konten pemfilteran dan *circumvention* pada YouTube dan TikTok menunjukkan bahwa platform video dapat menjadi sumber penyebaran panduan mengenai cara melewati sistem kontrol atau pemfilteran. Hal ini menunjukkan bahwa celah bypass perlu dipertimbangkan sejak tahap perancangan sistem, bukan hanya setelah sistem selesai dibuat [14].

VPN merupakan salah satu teknik yang dapat memengaruhi efektivitas pemfilteran berbasis jaringan. VPN memungkinkan perangkat pengguna membuat koneksi melalui *tunnel* terenkripsi menuju server perantara. Dalam kondisi tersebut, lalu lintas pengguna tidak langsung menuju situs tujuan, tetapi terlebih dahulu melewati server VPN. Hal ini dapat menyulitkan sistem pemfilteran berbasis jaringan atau DNS untuk mengenali tujuan akses asli pengguna, karena sebagian lalu lintas dan permintaan jaringan dapat berada di dalam *tunnel* VPN [38].

Selain VPN, mode Incognito dan *browser* alternatif juga dapat menjadi celah apabila sistem hanya melindungi satu lingkungan *browser*. Mode Incognito dapat mengurangi efektivitas pengawasan jika *extension* tidak diizinkan berjalan pada mode tersebut. *Browser* alternatif dapat digunakan untuk membuka situs di luar *browser* yang sudah dipasangi *extension*. *Extension* VPN atau *proxy* juga dapat mengubah jalur akses dari dalam *browser*. Berdasarkan kondisi tersebut, sistem pemfilteran tidak

cukup hanya bergantung pada satu lapisan proteksi, karena setiap lapisan memiliki kelemahan yang berbeda.

2.2.13 Pendekatan Anti-Bypass dan Proteksi Berlapis

Pembahasan mengenai teknik *circumvention* menunjukkan bahwa sistem kontrol orang tua atau administrator perlu dirancang tidak hanya untuk melakukan pemblokiran, tetapi juga untuk mengurangi kemungkinan pengguna melewati sistem pembatasan akses. Dalam penelitian ini, *anti-bypass* dipahami sebagai pendekatan desain untuk mengurangi peluang pengguna menonaktifkan, menghapus, atau melewati komponen proteksi yang telah diterapkan. Pendekatan ini didasarkan pada kebutuhan untuk menghadapi berbagai bentuk *circumvention*, seperti penggunaan VPN atau *proxy*, mode *private browsing*, penonaktifan *extension*, penggunaan *extension* lain, maupun aplikasi alternatif di luar lingkungan *browser* yang diproteksi [14].

Pendekatan *anti-bypass* tidak berarti bahwa sistem dapat mencegah seluruh bentuk bypass secara mutlak, terutama apabila pengguna memiliki hak administrator penuh pada perangkat. Namun, pendekatan ini digunakan untuk memperkuat proteksi melalui beberapa lapisan, seperti *browser policy*, pembatasan mode Incognito atau InPrivate, pengelolaan *extension*, *local agent*, *application control, monitoring,* dan *persistence*. Dengan proteksi berlapis, sistem tidak hanya bergantung pada satu komponen, tetapi memiliki beberapa mekanisme pendukung untuk menjaga agar proteksi tetap aktif dan dapat dipantau oleh orang tua atau administrator [14], [28]. Pendekatan *anti-bypass* dapat dilakukan melalui proteksi berlapis.

Lapisan pertama adalah pemfilteran pada tingkat *browser* menggunakan *extension*. Lapisan ini digunakan untuk membaca URL yang diakses pengguna, menerapkan aturan pemfilteran lokal, dan mengirim URL ke layanan klasifikasi apabila diperlukan. Lapisan kedua adalah *browser policy*, yang digunakan untuk mengatur mode Incognito atau

InPrivate, mengelola *extension* yang boleh dipasang, dan mendukung pemasangan *extension* secara paksa apabila ID *extension* resmi tersedia. Mekanisme kebijakan tersebut didukung oleh fitur pengelolaan *extension* dan kebijakan pada Google Chrome, Microsoft Edge, dan Firefox [15], [32], [33], [34].

Lapisan ketiga adalah pembatasan aplikasi yang berpotensi menjadi jalur bypass. Contohnya adalah aplikasi VPN *desktop*, *proxy tool*, *browser* alternatif, atau *executable* dari lokasi umum. Lapisan ini diperlukan karena tidak semua bypass terjadi di dalam *browser*. Pengguna dapat mencoba membuka situs dari *browser* lain atau menjalankan aplikasi yang mengubah jalur koneksi. Pembatasan aplikasi sejalan dengan konsep *application control* dan AppLocker pada Windows, yang dapat digunakan untuk mengatur aplikasi atau file yang boleh dijalankan pada perangkat [16], [40].

Lapisan keempat adalah monitoring, pencatatan event, dan notifikasi. Monitoring digunakan untuk memeriksa apakah *extension* masih aktif, apakah agen berjalan, apakah kebijakan masih diterapkan, dan apakah terdapat aktivitas yang perlu diketahui oleh orang tua atau administrator. Pendekatan anti-bypass tidak berarti sistem menjadi tidak dapat dilewati sepenuhnya. Tujuan utamanya adalah mengurangi celah yang paling umum dan membuat proteksi lebih konsisten. Dengan kombinasi *browser extension*, *browser policy*, *application control*, monitoring, dan notifikasi, sistem dapat memberikan perlindungan yang lebih kuat dibandingkan sistem yang hanya bergantung pada satu mekanisme pemblokiran [14], [15], [16], [40].

2.2.14 Aplikasi Desktop sebagai Komponen Administrasi

Agar proteksi berlapis dapat dikelola oleh orang tua atau administrator, diperlukan komponen yang berfungsi sebagai pusat pengaturan pada perangkat pengguna. Pada penelitian ini, komponen tersebut diwujudkan dalam bentuk aplikasi *desktop* pengelola proteksi. Istilah ini digunakan untuk menjelaskan peran aplikasi pada rancangan

sistem, yaitu sebagai antarmuka administrasi dan pusat konfigurasi. Aplikasi *desktop* tidak menggantikan *browser extension*, karena pemantauan dan pemblokiran URL tetap dilakukan oleh *extension*. Aplikasi *desktop* berperan untuk mengelola konfigurasi proteksi, menerapkan *browser policy*, mengatur *application control*, menampilkan status sistem, menyimpan konfigurasi, serta menyediakan pengaturan notifikasi.

Aplikasi *desktop* diperlukan karena beberapa mekanisme proteksi tidak dapat dilakukan sepenuhnya dari *browser extension*. *Extension* memiliki ruang kerja di dalam *browser*, sedangkan beberapa pengaturan seperti penerapan *browser policy*, pembatasan mode Incognito atau InPrivate, pengelolaan pemasangan *extension*, pembuatan *scheduled task*, dan pembatasan aplikasi tertentu memerlukan kontrol pada tingkat sistem operasi. *Browser policy* pada Chrome dan Edge mendukung pengelolaan *extension* serta konfigurasi *browser*, sedangkan Windows menyediakan mekanisme Application Control, AppLocker, dan Task Scheduler untuk mengatur aplikasi serta task sistem [15], [16], [34], [39], [40]. Dengan adanya aplikasi *desktop*, orang tua atau administrator dapat mengatur proteksi melalui satu antarmuka tanpa perlu melakukan konfigurasi *registry*, kebijakan, atau *task* sistem secara manual.

Dalam rancangan sistem, aplikasi *desktop* berperan sebagai pusat kendali yang menghubungkan konfigurasi orang tua atau administrator dengan komponen proteksi lain. Konfigurasi yang disimpan melalui aplikasi *desktop* dapat digunakan oleh *local agent* dan *browser extension*. Dengan memakai susunan ini, *browser extension* berperan sebagai komponen pemfilteran URL, *browser policy* berperan sebagai penguat lingkungan *browser*, *application control* berperan sebagai pembatas jalur bypass di luar *browser*, sedangkan aplikasi *desktop* berperan sebagai pusat administrasi sistem [15], [16], [39], [40].

2.2.15 *Application Control*

Application Control merupakan mekanisme untuk mengatur aplikasi, file, atau kode apa saja yang boleh dijalankan pada sistem operasi. Microsoft menjelaskan bahwa *Application Control for Windows* dapat digunakan untuk membatasi aplikasi yang dapat dijalankan pengguna dan kode yang berjalan pada sistem [40]. Salah satu teknologi yang berkaitan dengan *application control* adalah AppLocker, yang dapat digunakan untuk mengatur *executable files*, *scripts*, Windows Installer files, DLL, *packaged apps*, dan *packaged app installers* [16].

Application control dapat digunakan sebagai lapisan pendukung untuk mengurangi potensi bypass di luar *browser*. *Browser extension* bekerja pada lingkungan *browser* yang dilindungi, sedangkan pengguna masih dapat mencoba menggunakan aplikasi lain untuk menghindari proteksi. Contohnya adalah aplikasi VPN *desktop*, alat *proxy*, *browser* alternatif, atau *executable* portable yang dijalankan dari folder umum. Dengan adanya *application control*, sistem dapat membatasi aplikasi yang tidak sesuai dengan aturan orang tua atau administrator. *Application control* dapat diterapkan dengan pendekatan *allowlist* maupun blocklist. Pendekatan *allowlist* hanya mengizinkan aplikasi tertentu yang dianggap aman untuk berjalan, sedangkan pendekatan blocklist membatasi aplikasi tertentu yang dianggap berisiko. Pada sistem kontrol orang tua atau administrator, pendekatan blocklist dapat digunakan untuk membatasi aplikasi yang umum digunakan sebagai jalur bypass, seperti VPN, *proxy tool*, atau *browser* alternatif. Sementara itu, pendekatan allowlist dapat digunakan pada lingkungan yang lebih ketat, yaitu ketika hanya aplikasi tertentu yang diizinkan untuk berjalan.

Application control tidak menggantikan fungsi *browser extension*. *Extension* tetap bertugas melakukan pemfilteran URL, sedangkan *application control* berperan sebagai lapisan tambahan untuk membatasi aplikasi yang dapat melemahkan proteksi. Lapisan ini mendukung pendekatan anti-bypass karena pengguna tidak hanya dibatasi pada situs yang dibuka, tetapi juga pada aplikasi yang dapat digunakan untuk

mengubah jalur akses atau membuka situs melalui lingkungan yang tidak dilindungi [16], [40].

2.2.16 *Local agent* dan Monitoring Sistem

Sistem proteksi juga memerlukan mekanisme pemantauan agar status komponen dapat diketahui secara berkelanjutan. *Local agent* atau local *agent* dapat dipahami sebagai komponen perangkat lunak yang berjalan di latar belakang pada perangkat pengguna untuk mendukung proses pemantauan, pengelolaan konfigurasi, dan komunikasi antar komponen sistem [54]. Konsep agen seperti ini umum digunakan pada sistem manajemen *endpoint*, yaitu ketika sebuah komponen berjalan pada perangkat untuk menerima konfigurasi, menjalankan tugas tertentu, mencatat aktivitas, atau mengirimkan status perangkat ke komponen pengelola [54]. *Local agent* dapat digunakan untuk membantu memeriksa apakah komponen proteksi masih aktif, apakah aturan sistem masih berjalan, dan apakah terdapat aktivitas yang perlu diperhatikan oleh orang tua atau administrator. *Local agent* juga dapat menjadi penghubung antara perangkat lunak *desktop* dan *browser extension*, misalnya untuk sinkronisasi pengaturan, penerimaan heartbeat, pencatatan event, serta pengiriman notifikasi apabila terdapat aktivitas penting.

Monitoring sistem diperlukan karena proteksi tidak cukup hanya dirancang untuk melakukan pemblokiran, tetapi juga perlu mampu menunjukkan apakah perlindungan masih berada dalam kondisi aktif. Tanpa mekanisme monitoring, orang tua atau administrator sulit mengetahui apakah *browser extension* masih berjalan, apakah kebijakan masih diterapkan, atau apakah terdapat aplikasi yang berpotensi menjadi jalur *bypass*. Hal ini berkaitan dengan kebutuhan anti-bypass, karena sistem perlu mengurangi kemungkinan proteksi dilemahkan tanpa diketahui oleh pengelola [14].

Pada sistem yang menggunakan beberapa lapisan proteksi, monitoring juga berfungsi sebagai penghubung antara komponen yang berjalan pada *browser*, sistem operasi, dan aplikasi pengelola. Informasi

status dapat digunakan untuk membantu orang tua atau administrator memahami kondisi proteksi, mencatat aktivitas penting, serta mengambil tindakan apabila terdapat komponen yang tidak berjalan sesuai rancangan. *Local agent* dan monitoring sistem berperan sebagai lapisan pendukung yang membantu menjaga konsistensi proteksi, terutama ketika sistem melibatkan *browser extension*, *browser policy*, dan *application control* [16], [40].

2.2.17 *Persistence* dan Pemulihan Proteksi

Persistence mengacu pada kemampuan sistem untuk tetap tersedia setelah perangkat dimatikan, dihidupkan kembali, atau setelah pengguna login ulang [55]. *Persistence* penting karena proteksi tidak ideal apabila hanya aktif ketika aplikasi dibuka secara manual. Jika proteksi tidak berjalan kembali setelah perangkat *restart*, maka terdapat celah bagi pengguna untuk mengakses situs atau aplikasi tanpa pemantauan sistem.

Pada Windows, Task Scheduler dapat digunakan untuk menjalankan task secara otomatis berdasarkan *trigger* tertentu, seperti saat sistem startup atau saat pengguna login [39]. Mekanisme ini dapat dimanfaatkan untuk menjalankan kembali komponen latar belakang atau memastikan layanan tertentu tetap tersedia setelah perangkat digunakan kembali., Task Scheduler menjadi salah satu mekanisme yang relevan untuk mendukung *persistence* pada sistem yang membutuhkan proses latar belakang. *Persistence* juga berkaitan dengan pemulihan proteksi. Jika komponen tertentu berhenti berjalan atau konfigurasi berubah, sistem perlu memiliki mekanisme pemeriksaan agar kondisi tersebut dapat diketahui. Dalam sistem kontrol, *Persistence* dan pemulihan proteksi membantu menjaga agar fungsi pemantauan dan pembatasan akses tetap berjalan secara konsisten, tidak hanya pada saat aplikasi utama sedang dibuka.

2.2.18 Pengujian Fungsional Berbasis Skenario

Dalam Pengujian fungsional, pengujian yang dilakukan untuk mengevaluasi apakah suatu komponen atau sistem memenuhi persyaratan

fungsional yang telah ditentukan [59]. Pengujian dilaksanakan menggunakan *test case* yang memuat prasyarat, masukan atau tindakan pengujian, serta hasil yang diharapkan [60]. Selama pelaksanaan pengujian, hasil aktual dibandingkan dengan hasil yang diharapkan untuk menentukan kesesuaian perilaku sistem dan menetapkan status keberhasilan setiap skenario [61].

Dalam penelitian ini, pengujian fungsional digunakan untuk mengevaluasi kebutuhan anak sebagai pengguna yang dilindungi dan kebutuhan orang tua atau administrator sebagai pengelola proteksi. Kebutuhan anak dievaluasi melalui skenario akses URL aman, pemblokiran URL berbahaya, penggunaan VPN, dan variasi representasi URL. Sementara itu, kebutuhan orang tua atau administrator dievaluasi melalui fungsi PIN, pengelolaan konfigurasi, aktivasi dan deaktivasi proteksi, penerapan kebijakan, pemantauan agen, *event logging*, *persistence*, dan notifikasi. Persentase keberhasilan yang diperoleh merepresentasikan kesesuaian keluaran sistem pada skenario dan lingkungan pengujian yang telah ditentukan. Nilai tersebut tidak digunakan untuk menyatakan bahwa sistem akan selalu berhasil pada seluruh URL, perangkat, konfigurasi, dan kondisi penggunaan karena pengujian terhadap sejumlah kasus tidak dapat membuktikan tidak adanya seluruh kesalahan perangkat lunak.