GuardSurfing : Ekstensi *Browser* sebagai Alat Bantu Deteksi *Website Phishing* dengan Metode Klasifikasi XGBoost untuk Deteksi URL *Phishing* Berbasis Flask *Framework*

Hanif Abdul Karim Afandi[1,*)], M. Lazaro Fa. Al-Dzaki[2)], Nurul Qomariasih[3)], Reza Aulia Wildana[4)]

1) Program Studi Rekayasa Kriptografi, Politeknik Siber dan Sandi Negara hanif.abdul@student.poltekssn.ac.id
2) Program Studi Rekayasa Kriptografi, Politeknik Siber dan Sandi Negara, m.lazaro@student.poltekssn.ac.id
3) Badan Siber dan Sandi Negara, nurul.qomariasih@bssn.go.id
4) Badan Siber dan Sandi Negara, reza.aulia@bssn.go.id

Riwayat Artikel

Dikirim 26 Mei 2025
Diterima 22 Agu 2025
Diterbitkan 31 Agu 2025

Kata kunci:

Deteksi *phishing*
XGBoost
Ekstensi *browser*
URL *phishing*

Abstrak

Dengan meningkatnya aktivitas di dunia maya seperti jejaring sosial, perbankan elektronik, dan *e-commerce*, ancaman URL *phishing* semakin sulit diidentifikasi oleh pengguna umum. Penelitian ini memperkenalkan GuardSurfing: alat deteksi URL *phishing* berbasis XGBoost yang diimplementasikan sebagai ekstensi *browser* dengan *backend* ringan untuk inferensi *real-time*. Pada data uji, sistem mencapai akurasi 0.970, presisi 0.973, recall 0.993, dan F1-score 0.984. Desain *privacy-by-design* (hanya string URL, tanpa mengambil konten halaman) memungkinkan latensi rendah dan jejak komputasi minimal. Secara empiris, dibandingkan *baseline Logistic Regression*, SVM, dan *Random Forest* pada protokol evaluasi identik, XGBoost menunjukkan keseimbangan terbaik antara sensitivitas (*recall*) dan stabilitas (F1), didukung penanganan ketidakseimbangan kelas (scale_pos_weight) dan penalaan *threshold* berorientasi F1/*Recall*. Sistem ini efektif membantu melindungi pengguna dari ancaman *phishing* dengan pendekatan yang efisien, mudah digunakan, dan dapat diperbarui berkala untuk menghadapi pola serangan terbaru.

Keywords:

Phishing detection
XGBoost
Browser extensions
Phishing URL

Abstract

With the increasing prevalence of online activities such as social networks, e-banking, and e-commerce, phishing URLs have become increasingly difficult for general users to identify. This work presents GuardSurfing: an XGBoost-based phishing URL detector implemented as a real-time browser extension with a lightweight backend. On the test set, the system achieves 0.970 accuracy, 0.973 precision, 0.993 recall, and a 0.984 F1-score. A privacy-by-design approach (URL-string only, no page content) enables low latency and a minimal computational footprint. Empirically, under an identical evaluation protocol, XGBoost outperforms Logistic Regression, SVM, and Random Forest by providing the best balance between sensitivity (recall) and stability (F1), supported by class-imbalance handling (scale_pos_weight) and F1/recall-oriented threshold tuning. The system effectively protects users from phishing threats and can be periodically updated to address emerging attack patterns.

1. PENDAHULUAN

Seiring dengan tumbuhnya penggunaan internet yang signifikan, semakin banyak pula orang-orang yang membagikan informasi data pribadi mereka secara daring. Hal ini akan berdampak pada sejumlah besar informasi pribadi dan transaksi keuangan menjadi rentan dengan adanya penjahat dunia maya. *Phishing* adalah salah satu bentuk kejahatan dunia maya yang terus berkembang dan menjadi ancaman serius sejak pertama kali dilaporkan pada tahun 1990. Teknik *phishing* yang semakin canggih memungkinkan penjahat menipu pengguna yang kemudian dapat menyebabkan kerugian besar, termasuk pencurian identitas, kompromi data perusahaan, dan kebocoran informasi rahasia pemerintah [1]. Penipuan ini dilakukan dimana pelaku mengirimkan pesan palsu agar nantinya korban akan memberikan kredensial milik mereka. Selain itu, *phishing* juga dapat melibatkan penyebaran perangkat lunak berbahaya yang meminta tebusan atau merusak sistem komputer seseorang, termasuk serangan massal yang menargetkan sekelompok orang dengan tujuan memanfaatkan individu yang lebih rentan [2].

URL merupakan elemen krusial dalam dunia digital yang dapat menjadi sasaran utama dalam serangan *phishing*. Dengan semakin meningkatnya penggunaan internet untuk berbagai aktivitas, para penjahat dengan mudah dapat memanfaatkan URL sebagai alat untuk menipu pengguna. *Phishing* berbasis URL melibatkan pembuatan tautan palsu yang menyerupai situs *web* asli, dengan tujuan mengelabui korban agar memasukkan informasi sensitif mereka. Keberadaan situs *web* palsu ini dapat berdampak besar terhadap keamanan data pengguna, terutama jika mereka tidak memiliki kewaspadaan yang cukup dalam mengidentifikasi tautan berbahaya [3]. Berbagai metode deteksi phishing tradisional, termasuk pendekatan heuristik, daftar hitam, dan sistem berbasis aturan, telah banyak diterapkan untuk menanggulangi permasalahan ini [4]. Namun, teknik-teknik tersebut seringkali kesulitan untuk mengimbangi taktik penjahat siber yang terus berkembang, sehingga menyebabkan tingginya tingkat positif palsu dan lambatnya adaptasi terhadap strategi *phishing* baru. Menanggapi keterbatasan tersebut, pendekatan pembelajaran mesin mulai banyak digunakan karena kemampuannya untuk beradaptasi dan belajar dari data, sehingga menawarkan solusi yang lebih tangguh dalam mendeteksi situs *web phishing* [5].

Berangkat dari tantangan tersebut, penelitian ini tidak hanya mengandalkan model pembelajaran mesin untuk klasifikasi URL, tetapi juga memperkenalkan serangkaian kebaruan pada sisi akuisisi data di *browser*, penjelasan keputusan (*explainability*), mekanisme *human-in-the-loop*, serta pemastian keamanan dan privasi pengguna. Sistem yang dikembangkan diimplementasikan sebagai ekstensi *browser* dengan kemampuan ekstraksi tautan halaman riil yang komprehensif (termasuk *iframe, event-driven link,* dan *shortener*), antarmuka penjelasan serta intersepsi form sensitif, serta kalibrasi *threshold* dan *heuristik* IDN/*homoglyph* untuk meningkatkan ketahanan terhadap serangan baru. Selain itu, arsitektur *hybrid* yang digunakan memudahkan adopsi di lingkungan terkelola (*container*/CI/CD) tanpa mengorbankan privasi pengguna.

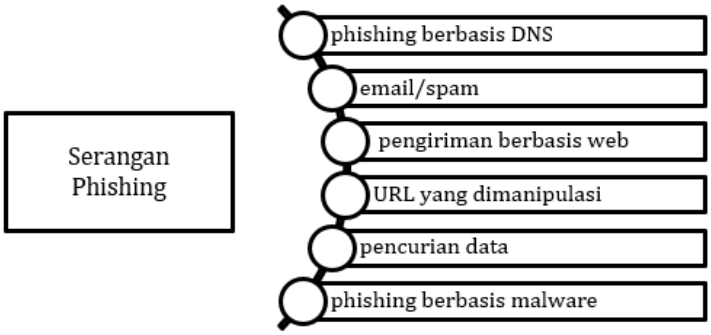
Maka dalam penelitian ini dirancanglah sebuah alat bantu untuk pengguna berupa ekstensi *browser* yang mengimplementasikan model pembelajaran mesin dalam melakukan deteksi URL *phishing*. Dengan solusi ini diharapkan dapat membantu pengguna mengenali dan mencegah *phishing* melalui URL, menciptakan lingkungan daring yang lebih aman, serta mengurangi risiko pengguna menjadi korban serangan *phishing* berbasis *web* [6].

2. LANDASAN TEORI

2.1. Phishing

Istilah "*Phishing*" pertama kali muncul pada tahun 1996 dalam grup berita Usenet bernama AOHell yang digunakan untuk menggambarkan terjadinya pencurian kredensial pengguna di platform *America Online* (AOL). Sejak saat itu, teknik *phishing* berkembang pesat dari metode sederhana hingga serangan yang lebih kompleks [7]. *Phishing* termasuk salah satu dari teknik rekayasa sosial yang teridentifikasi sebagai metode paling dominan digunakan para penjahat dunia maya untuk mendapatkan akses ke informasi pribadi pengguna internet [8]. Ada berbagai jenis

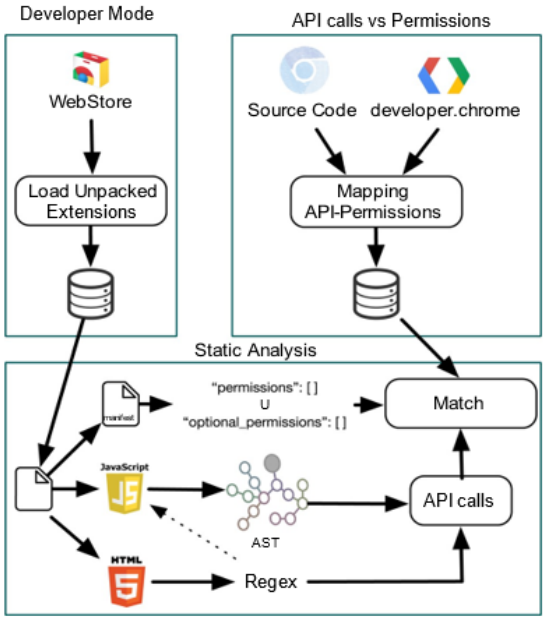
serangan *phishing*, tidak hanya sebatas *spoofing* sebagaimana dikenal banyak orang, terdapat *phishing* berbasis *malware*, *phishing* berbasis DNS, pencurian data, email/spam, pengiriman berbasis *web*, dan URL yang dimanipulasi, sebagaimana ditunjukkan pada Gambar 1 [9]. Target serangan *phishing* sudah merambah ke platform populer seperti PayPal, Facebook, Microsoft, Netflix, dan WhatsApp. Hal ini menunjukkan bahwa pelaku *phishing* berusaha mencuri data terkait kehidupan sosial dan keuangan pengguna. Untuk mengatasi ancaman ini, berbagai pihak, termasuk industri dan akademisi, terus mencari solusi efektif [10].



Gambar 1. Jenis-jenis serangan *phishing*

2.2. Ekstensi *Browser*

Ekstensi *browser* menjadi platform yang dapat dimanfaatkan untuk memperkaya pengalaman pengguna dalam menjelajah di dunia maya [11]. Pengertian dari ekstensi *browser* merupakan perangkat lunak dengan sumber daya yang ringan untuk meningkatkan fungsionalitas peramban *web* sehingga memungkinkan pengguna untuk menyesuaikan pengalaman *browsing* mereka dengan fitur tambahan seperti pemblokiran iklan, pengelolaan kata sandi, integrasi layanan pihak ketiga, atau pengoptimalan tampilan halaman *web* [12].



Gambar 2. Analisis proses langkah ekstensi

Diagram pada Gambar 2 merupakan visualisasi alur data mulai dari pengunggahan paket ekstensi yang telah disiapkan hingga pemetaan API dan analisis statis [13]. Sehingga ekstensi *browser*

sendiri bekerja dengan memanfaatkan berbagai API yang disediakan oleh *browser* untuk berinteraksi dengan halaman *web* dan sistem pengguna. Beberapa komponen utama dalam ekstensi *browser* meliputi:

a. Model Izin (*Permission Models*)

Setiap ekstensi memerlukan *file manifest* (manifest.json) yang menentukan izin yang diperlukan untuk operasinya. Izin ini terbagi menjadi dua kategori utama:

- *Host Permissions*: Menentukan situs *web* mana yang dapat diakses oleh ekstensi.
- API *Permissions*: Memungkinkan ekstensi menggunakan fitur tertentu dari *WebExtension* APIs, seperti *browser*.storage atau *browser*.cookies.

b. Script Konten (*Content Scripts*)

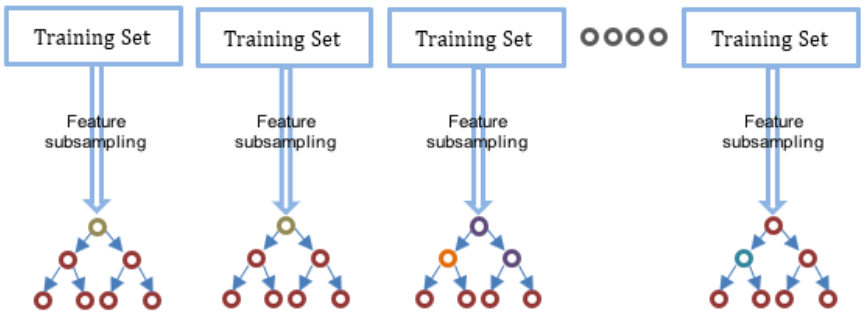
Ini adalah skrip JavaScript statis yang secara otomatis dimuat bersama dengan halaman web tertentu. *Content scripts* berfungsi sebagai ekstensi dari struktur DOM halaman web dan memungkinkan modifikasi tampilan atau perilaku halaman.

c. Halaman Latar Belakang (*Background Pages / Service Workers*)

Tidak seperti content *scripts*, halaman latar belakang tidak dimuat dalam setiap halaman web. Mereka bekerja di belakang layar untuk merespons peristiwa *browser* dan menjalankan API WebExtension. Untuk memperluas akses API, *content scripts* sering kali berkomunikasi dengan *background pages* menggunakan teknik *message passing* [14].

2.3. XGBoost

XGBoost merupakan algoritma pembelajaran mesin berbasis *gradient boosting* yang diperkenalkan oleh Friedman guna meningkatkan performa dalam analisis data kompleks. Algoritma ini bekerja dengan menggabungkan fungsi dasar (*basis functions*) dan bobot, sehingga mampu menghasilkan model dengan tingkat kecocokan tinggi terhadap data[15]. XGBoost menggunakan pendekatan *ensemble learning*, di mana beberapa pohon keputusan dibangun secara berurutan. Setiap pohon yang dibuat mempertimbangkan kesalahan prediksi dari pohon sebelumnya, lalu memberikan bobot lebih besar pada variabel yang sulit diprediksi sebagaimana diilustrasikan pada Gambar 3. Regularisasi dan pemetaan nilai kontinu ke dalam probabilitas menjadikannya metode yang sangat andal dalam berbagai tugas pembelajaran mesin, termasuk klasifikasi dan regresi [16].



Gambar 3. Representasi Klasifikasi XGBoost

Algoritma XGBoost dipilih karena mampu menunjukkan keunggulan yang signifikan dalam mendeteksi dibandingkan model lain, terutama dalam hal akurasi dan keseimbangan metriks evaluasi. Akurasi dari pembelajaran mesin didapat dari jumlah nilai *true positives* dan *true negative* dibagi dengan total *instance* dalam dataset sebagaimana ditunjukkan pada Gambar 4. Tingkat akurasi yang tinggi menunjukkan bahwa model bekerja dengan sangat baik dalam membedakan antara URL *phishing* dan bukan. Untuk presisi, model bertujuan untuk mengukur proporsi URL

phishing yang teridentifikasi dengan benar seperti ditunjukkan oleh Gambar 5.

Recall yang dikenal sebagai sensitivitas atau *true positivity*, mengukur proporsi URL *phishing* yang teridentifikasi dengan benar dari contoh-contoh URL *phishing* yang ada di dataset. Apabila rentang *recall* 0,97%-0,99 menunjukkan bahwa model menangkap 97-99% URL phishing seperti yang ditunjukkan Gambar 6. Kemudian, terdapat F1-*score* yang dikenal sebagai rata-rata harmonis dari *recall* dan presisi. Skor ini akan memberikan ukuran yang sangat seimbang dari kinerja model, perhitungannya sesuai dengan Gambar 7 [17].

$$Akurasi = \frac{TN + TP}{TN + FP + TP + FN}$$

Gambar 4. Rumus perhitungan akurasi model

$$Presisi = \frac{TP}{TP + FP}$$

Gambar 5. Rumus perhitungan presisi model

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

Gambar 6. Rumus perhitungan recall model

$$F1 - score = 2\,x\,\frac{Presisi\ x\ Recall}{Presisi + Recall}$$

Gambar 7. Rumus perhitungan F1-skor model

2.4. Metodologi Pengembangan

Dalam mengembangkan perangkat lunak berupa ekstensi *browser* ini, perlu adanya metode yang digunakan agar hasil yang diperoleh sesuai dengan apa yang diharapkan. Salah satu metodologi pengembangannya ialah *Agile Method* yang mencakup filosofi *agile*, nilai-nilai *agile*, dan prinsip-rinsip *agile*. Kerangka konseptual yang ditawarkan *agile* dalam rekayasa perangkat lunak yang dimulai dari fase perencanaan hingga sampai ke fase *deployment* sepanjang siklus hidup proyek akan sangat bermanfaat dalam penelitian [18]. *Agile* juga mampu dikolaborasikan dengan DRM yang menjadi suatu pendekatan sistematis untuk memandu dan menyusun penelitian dengan model desain [19].

Dengan DRM, kualitas dan validitas hasil penelitian dapat dikembangkan untuk meningkatkan dengan menyediakan suatu kerangka kerja yang jelas untuk merancang, melakukan, dan mengevaluasi penelitian. Bentuk langkah dalam teknologi pengembangan ini adalah *System Integration Testing* (SIT) yang bertujuan untuk memverifikasi interaksi antar komponen, dan kriteria apakah komponen dapat digunakan untuk mengorganisasikan interaksi yang dihasilkan dan dapat diperkenalkan ke dalam sistem setelah melalui serangkaian pengujian [20].

3. METODE PENELITIAN

Penelitian ini menggunakan metode metode DRM (*Design Research Methodology)* yang terdiri dari empat tahapan yaitu *Research Clarification* (RC), *Descriptive Study* I (DS I), *Prescriptive Study* (PS), dan *Descriptive Study* II (DS II) [18]. Kemudian ekstensi akan dikembangkan menggunakan metode *agile.*

3.1. *Research Clarification* (RC)

Pada tahap ini dilakukan pengumpulan data melalui ekstraksi fitur dari situs *web phishing* dan *web* yang *legitimate* atau sah. Tujuan dari tahapan ini untuk membantu peneliti dalam memperdalam

pemahaman dan tujuan umum yang akan dicapai selama penelitian.

3.2. *Descriptive Study* I (DS I)

Pada tahap ini dilakukan identifikasi dan klarifikasi lebih lanjut terkait faktor-faktor yang akan memengaruhi keberhasilan penelitian ini. Tujuannya untuk memahami lebih dalam kondisi yang nanti akan terjadi selama penelitian dilakukan. Tahap ini akan menententukan juga faktor keberhasilan dan faktor kunci untuk membuat model acuan yang akan digunakan.

3.3. *Prescriptive Study* (PS)

Pada tahap ini dilakukan penyusunan faktor-faktor keberhasilan untuk meningkatkan, menghilangkan, atau mengurangi pengaruh faktor-faktor kunci yang telah ditemukan pada tahap DS I. Dalam penelitian ini, pengembangan ekstensi GuardSurfing akan dibagi menjadi dua tahapan, pertama tahap pengolahan dataset dan kedua tahap pengembangan ekstensi *browser*. Tahap pengolahan dataset merupakan proses lanjutan setelah data hasil ekstraksi didapatkan dan dilakukan sinkronisasi dengan model pembelajaran mesin. Selanjutnya pada tahap pengembangan ekstensi dilakukan proses pembuatan ekstensi yang sudah memiliki fungsi deteksi URL *phishing*.

Pada tahap pengembangan ekstensi *browser* akan digunakan metode *Agile* yang melalui lima proses utama, yaitu *Planning, Design, Implementation, Testing* dan *Deployment*. Pada metode *Agile*, proses pengembangan ekstensi dapat dilakukan secara dinamis, sehingga memungkinkan apabila terdapat ditengah jalan meskipun ekstensi telah melalui proses *testing* dan *deployment* [21].

3.4. *Descriptive Study* II (DS II)

Pada tahap terakhir ini, dilakukan evaluasi berupa pengujian model XGBoost. Pada tahap ini akan dilakukan lima jenis pengujian evaluasi matriks, yaitu akurasi, presisi, *recall*, F1-*score*, dan *confusion matrix*. Pengujian tersebut dilakukan untuk melihat hasil performa model XGBoost antara data latih dan data uji. Hasil yang diperoleh akan menjadi evaluasi dan bahan pertimbangan apakah pemilihan model XGBoost dalam melakukan deteksi URL *phishing* merupakan pilihan yang tepat dibandingkan menggunakan algoritma pembelajaran mesin lainnya seperti *Adaboost*, *Gradient Boosting, Gaussian NB, Decision Trees, Random Forests, Logistic Regression*, dan masih banyak lagi [22].

4. HASIL DAN PEMBAHASAN

4.1. Pengolahan Dataset

a. Pengumpulan Data

Dataset dikumpulkan dari berbagai sumber untuk memastikan akurasi dan kualitas data. Berikut rincian data yang diperoleh:

1. Total entri: 11.054
2. Pemetaan label: -1 = *phishing*, 1 = *non-phishing*
3. Distribusi kelas:
   - *Phishing*: 4.897 (44,3%)
   - *Non-phishing*: 6.157 (55,7%)
4. Tipe fitur: indikator leksikal/host-based URL (mis. UsingIP, LongURL, ShortURL, Symbol@)
5. Nilai hilang: ~0% (tidak ada *missing* signifikan pada fitur yang dimuat)
6. Sumber dataset: Kaggle, Mendeley, dan Github [23][24][25]

b. *Data Pre-processing*

Proses ini mencakup pembersihan dan pengorganisasian data agar sesuai untuk digunakan

oleh algoritma pembelajaran mesin. Tahapan yang dilakukan meliputi penanganan nilai yang hilang (*missing values*), penskalaan fitur numerik ke dalam rentang standar (*scaling*), dan *encoding* variabel kategorikal. Tujuan utama dari pra-pemrosesan ini adalah untuk menciptakan dataset yang terstruktur dan seragam sehingga dapat digunakan secara efektif dalam pelatihan dan pengujian model pembelajaran mesin.

c. Pembagian Data

Dataset yang telah dikumpulkan dibagi menjadi beberapa subset untuk pelatihan validasi dan pengujian model sehingga kinerja model berjalan secara objektif serta mencegah *overfitting*. Pembagian data dilakukan sebelum proses pelatihan model agar model dapat melakukan generalisasi akan menghasilkan luaran yang baik terhadap data yang belum pernah dilihat sebelumnya. Pada penelitian ini, data akan dibagi menjadi 70% untuk data pelatihan dan 30% untuk data pengujian. Pendekatan ini tidak hanya mendukung validitas statistik dalam menilai kinerja model, tetapi juga membantu mengurangi risiko *overfitting* yang dapat terjadi ketika model terlalu spesifik terhadap data pelatihan.

d. Ekstraksi Fitur

Karakteristik penting dari URL diekstraksi untuk membantu membedakan situs *phishing* dari situs yang sah. Fitur-fitur ini dapat mencakup panjang URL, keberadaan kata kunci mencurigakan, penggunaan HTTPS, jumlah subdomain, dan elemen lain yang sering digunakan dalam serangan *phishing*. Pemilihan fitur yang tepat sangat berpengaruh terhadap kinerja model dalam mendeteksi URL *phishing* secara akurat.

Untuk mendukung proses ini, kami mengembangkan content script yang berjalan pada seluruh *frame* (*all_frames*) dan mendukung pemantauan DOM dinamis melalui *MutationObserver*. Skrip ini secara komprehensif menyapu berbagai sumber tautan, meliputi:

1. Elemen jangkar (<a href=...>), area pada *image map*, serta *iframe* (termasuk *nested frames*).
2. Elemen atau *banner* dengan *event handler onclick* maupun atribut data-* yang memicu navigasi.
3. *Domain shortener* umum (misalnya t.ly, *rebrand*.ly) yang ditangani melalui strategi perluasan aman, dengan memanfaatkan permintaan *HEAD/resolve* terkontrol, *local cache*, serta integrasi daftar putih dan daftar hitam.

Proses ekstraksi dilakukan secara kombinasi *periodic scanning* dan *event-driven* untuk meminimalkan *overhead* sekaligus tetap mampu menangkap tautan baru yang diinjeksikan oleh *JavaScript* pihak ketiga. Pendekatan ini memastikan cakupan data URL yang komprehensif dan relevan bagi sistem klasifikasi.

e. Pelatihan dan Evaluasi Model

Proses pelatihan model dimulai dengan mengonfigurasi parameter yang telah ditentukan sesuai karakteristik masing-masing algoritma. Model kemudian dilatih menggunakan dataset latih hingga mampu mengklasifikasikan URL apakah sah atau *phishing*. Selanjutnya, model dievaluasi menggunakan dataset uji dengan berbagai metrik.

Tabel 1. Hasil evaluasi performa model

Model	*Accuracy*	*Precision*	*Recall*	F1	ROC-AUC	PR-AUC	threshold_ mean
XGB	0.973946	0.976460	0.964675	0.970431	0.996757	0.996345	0.500033
RF	0.973223	0.969688	0.970191	0.969796	0.996705	0.996233	0.428149
SVM-RBF	0.960738	0.951943	0.959977	0.955870	0.990008	0.987967	0.369235
SVM-*Linear*	0.930070	0.930627	0.910154	0.920162	0.978747	0.976388	0.612405
LR	0.929980	0.931042	0.909536	0.920090	0.978878	0.976640	0.507186

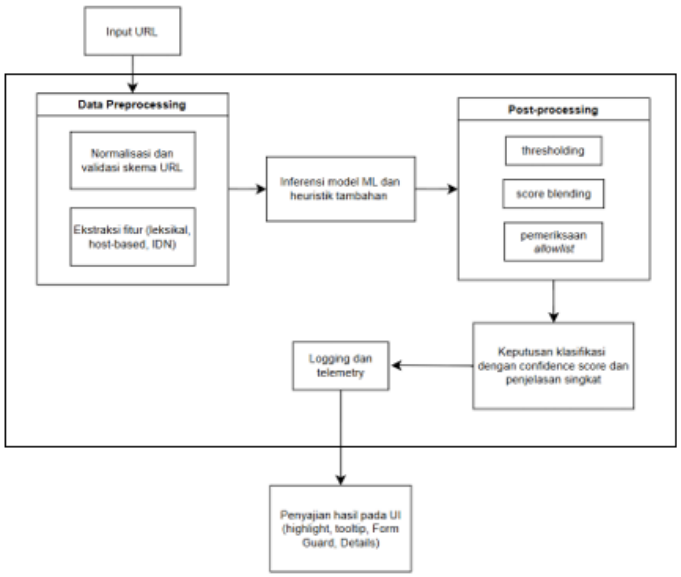
Hasil evaluasi yang ditunjukkan pada Tabel 1 memperlihatkan bahwa seluruh model mampu memberikan performa yang baik, dengan nilai ROC-AUC dan PR-AUC di atas 0.97. Model XGBoost menempati posisi terbaik dengan *Accuracy* (97.39%), F1-*Score* (0.9704), dan PR-AUC (0.9963), mengungguli model lain terutama pada kemampuan mendeteksi phishing (*recall* tinggi) serta menjaga keseimbangan *precision* dan *recall*.
Sementara itu, Tabel 2 memberikan gambaran kelebihan dan *trade-off* dari setiap algoritma. XGBoost, meskipun membutuhkan *tuning* parameter untuk menghindari risiko *overfitting*, menawarkan keseimbangan terbaik antara akurasi, kecepatan, dan interpretabilitas melalui fitur SHAP. Dengan demikian, model ini dipilih sebagai model utama untuk implementasi sistem deteksi URL phishing berbasis *browser extension*.

Tabel 2. Analisis kelebihan dan kekurangan masing-masing model

Model	Kelebihan di proyek	Kekurangan/*Trade-off*
XGBoost	Memiliki nilai F1 dan PR-AUC tertinggi; mampu menangkap pola non-linear; komputasi relatif cepat di CPU; mendukung interpretabilitas melalui SHAP.	Memiliki nilai F1 dan PR-AUC tertinggi; mampu menangkap pola non-linear; komputasi relatif cepat di CPU; mendukung interpretabilitas melalui SHAP.
Random Forest	Stabil pada berbagai kondisi; membutuhkan sedikit tuning; relatif robust terhadap noise.	Stabil pada berbagai kondisi; membutuhkan sedikit tuning; relatif robust terhadap noise.
SVM-RBF	Kuat pada pemisahan data dengan margin yang kompleks; memberikan performa baik pada dataset berukuran kecil.	Kuat pada pemisahan data dengan margin yang kompleks; memberikan performa baik pada dataset berukuran kecil.
SVM-Linear	Sederhana dan cepat; koefisien model dapat dijelaskan secara langsung sehingga meningkatkan interpretabilitas.	Sederhana dan cepat; koefisien model dapat dijelaskan secara langsung sehingga meningkatkan interpretabilitas.
Logistic Regression	Menyediakan baseline yang kuat; interpretabel; komputasi cepat.	Menyediakan baseline yang kuat; interpretabel; komputasi cepat.

4.2. Pengembangan Ekstensi *Browser*

Setelah diperoleh model XGBoost dengan performa yang sesuai, tahap berikutnya adalah mengimplementasikan model tersebut ke dalam perangkat lunak berbasis ekstensi *browser*. Ekstensi yang dikembangkan dinamakan "GuardSurfing" [26], yang berfungsi memberikan peringatan secara *real-time* kepada pengguna saat mengakses halaman *web* yang berpotensi *phishing*. Alur proses GuardSurfing, mulai dari pengolahan data hingga menghasilkan ekstensi yang mampu mengklasifikasikan suatu situs *web* sebagai sah atau *phishing*, ditunjukkan pada Gambar 8.



Gambar 8. Alur proses kerja ekstensi GuardSurfing

Berikut penjelasan mengenai GuardSurfing:

a. Arsitektur Sistem

GuardSurfing dirancang dengan arsitektur berbasis *browser extension* yang terhubung dengan layanan *back-end*. Secara umum, sistem terdiri atas komponen berikut:

1. *Browser Extension* (*Manifest* V3)
   - *Content script* (page_scanner.js): melakukan pemindaian halaman aktif dan *iframe*.
   - *Popup page*, *details page*, dan *options page*: menyediakan antarmuka pengguna yang interaktif.
2. API *Gateway/Server*
   Dibangun menggunakan Flask (server.py) dengan *endpoint*:
   - /process untuk menganalisis URL tunggal.
   - /analyze_page untuk menganalisis seluruh tautan dalam halaman.
   - /config untuk pengaturan konfigurasi.
3. *Normalizer & Feature Extractor*
   - Modul FeatureExt.py mengekstraksi fitur berbasis leksikal dan *host*.
   - Mendukung deteksi *Internationalized Domain Name* (IDN) serta konversi *Punycode*.
4. *Model Service & Rule Engine*
   - Mendukung beberapa model: *XGBoost*, *RandomForest*, *Logistic Regression*.
   - Dilengkapi heuristik tambahan: deteksi *homoglyph*, *brand mismatch*, serta integrasi opsional dengan LLM untuk analisis mendalam.
5. *Calibration & Thresholding*
   - Optimasi berbasis metrik F1.
   - Mendukung *per-context mode* seperti *balanced, strict, relaxed*, dan *Kids Mode*.
6. *Caching*
   - Dilakukan pada sisi background maupun popup.
   - Mengurangi *overhead* dengan pengiriman ulang GET_LINKS bila data kosong.

b. Arsitektur Sistem

GuardSurfing dirancang dengan arsitektur berbasis *browser extension* yang terhubung dengan layanan *back-end*. Secara umum, sistem terdiri atas komponen berikut:

1. *Browser Extension* (*Manifest* V3)
   - *Content script* (page_scanner.js): melakukan pemindaian halaman aktif dan *iframe*.
   - *Popup page*, *details page*, dan *options page*: menyediakan antarmuka pengguna yang interaktif.
2. API *Gateway/Server*
   Dibangun menggunakan Flask (server.py) dengan *endpoint*:
   - /process untuk menganalisis URL tunggal.
   - /analyze_page untuk menganalisis seluruh tautan dalam halaman.
   - /config untuk pengaturan konfigurasi.
3. *Normalizer & Feature Extractor*
   - Modul FeatureExt.py mengekstraksi fitur berbasis leksikal dan *host*.
   - Mendukung deteksi *Internationalized Domain Name* (IDN) serta konversi *Punycode*.
4. *Model Service & Rule Engine*
   - Mendukung beberapa model: *XGBoost*, *RandomForest*, *Logistic Regression*.
   - Dilengkapi heuristik tambahan: deteksi *homoglyph*, *brand mismatch*, serta integrasi opsional dengan LLM untuk analisis mendalam.
5. *Calibration & Thresholding*
   - Optimasi berbasis metrik F1.
   - Mendukung *per-context mode* seperti *balanced, strict, relaxed*, dan *Kids Mode*.
6. *Caching*
   - Dilakukan pada sisi background maupun popup.

- Mengurangi overhead dengan pengiriman ulang GET_LINKS bila data kosong.

c. Penanganan Kesalahan (*Error Handling & Fallback*)

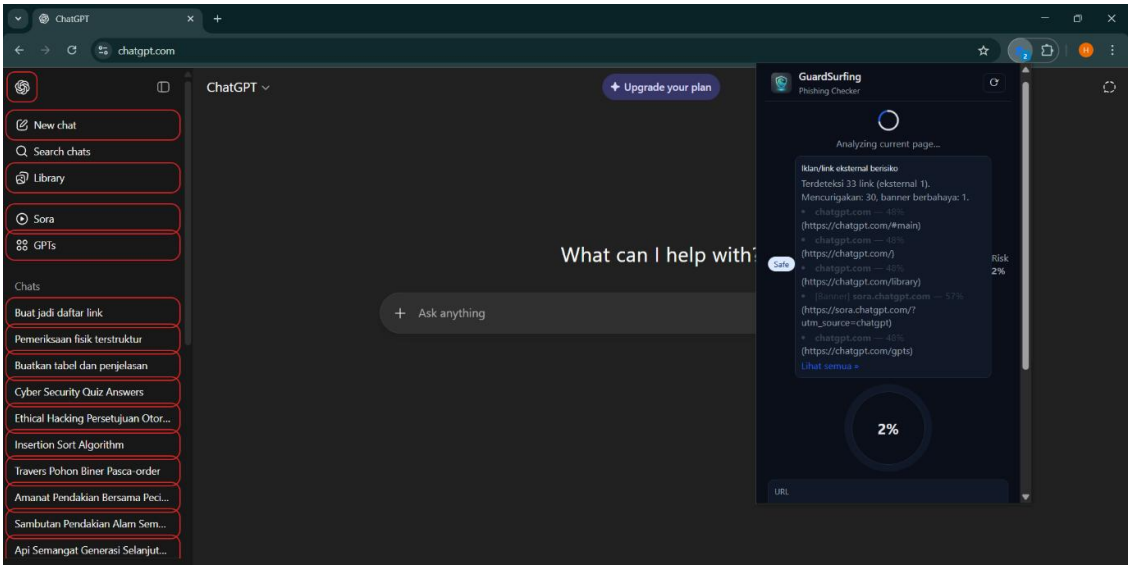
Untuk meningkatkan keandalan sistem, GuardSurfing dirancang dengan mekanisme penanganan kesalahan:

1. Backend tidak tersedia: antarmuka menampilkan pesan "*Backend* tidak tersedia" dengan opsi *retry*, sementara akses pengguna tetap aman.
2. Tautan kosong: *background service* mengirimkan perintah GET_LINKS agar *content script* melakukan pemindaian ulang (termasuk dukungan iframe).
3. *Timeout* pada LLM: jika proses melebihi LLM_TIMEOUT (default 8 detik), sistem menggunakan *fallback* skor ML.

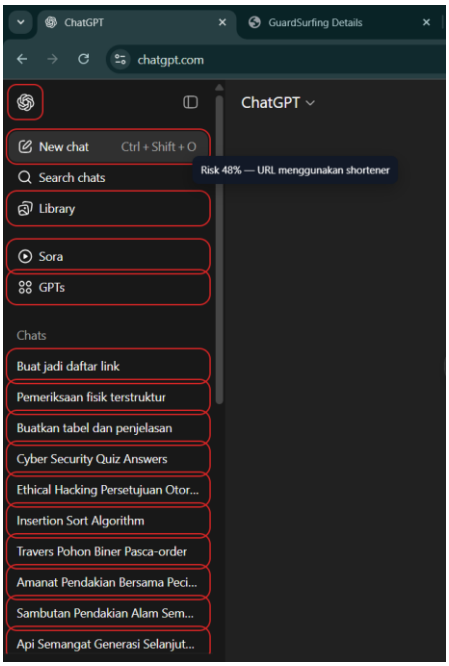
d. Implementasi Teknologi

1. *Stack* teknologi yang digunakan pada GuardSurfing meliputi:
2. *Frontend*: Chrome *Extension* (Manifest V3) berbasis JavaScript, HTML, dan CSS.
3. *Backend*: Python 3.11 dengan Flask.
4. *Machine Learning*: *scikit-learn, xgboost*, dan *statsmodels* untuk evaluasi.
5. Visualisasi: *seaborn* dan *matplotlib*.
6. Opsional:
   - confusable-homoglyphs untuk deteksi karakter Unicode berbahaya.
   - Ollama sebagai LLM lokal untuk analisis tambahan.

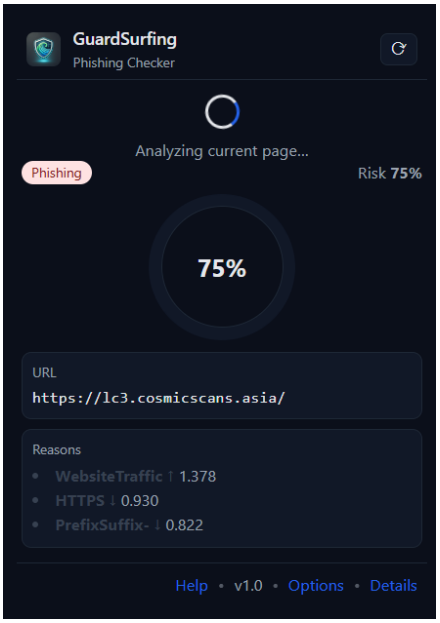
e. Tampilan Ekstensi



Gambar 9. Tampilan awal GuardSurfing ketika diaktifkan



Gambar 10. Tampilan fitur GuardSurfing untuk mengekstraksi isi dari *web* dan memberikan hasil analisis



Gambar 11. Tampilan *popup* GuardSurfing yang memberikan informasi hasil analisis pada *web* yang diakses

5. KESIMPULAN

Penelitian ini mengembangkan sistem deteksi URL phishing berbasis XGBoost yang diimplementasikan dalam bentuk ekstensi *browser*. Sistem ini mampu memberikan peringatan secara *real-time* ketika pengguna mengunjungi halaman *web* yang berpotensi berbahaya, sehingga memudahkan aksesibilitas bagi pengguna umum dan memungkinkan deteksi *phishing* otomatis tanpa mengganggu pengalaman berselancar. Selain menghasilkan performa klasifikasi yang kuat, penelitian ini juga memperkenalkan kebaruan pada sisi akuisisi *link* di klien, mekanisme penjelasan

keputusan (*explainability*), integrasi *human-in-the-loop*, serta kalibrasi/IDN yang terbukti meningkatkan kegunaan dan ketahanan sistem terhadap variasi serangan terbaru. Arsitektur *hybrid* yang mendukung operasi lokal maupun terkontainer memberikan fleksibilitas dalam adopsi, sekaligus menjaga privasi pengguna.

Meskipun demikian, masih terdapat tantangan yang perlu diatasi, seperti peningkatan adaptabilitas model terhadap serangan phishing baru melalui pendekatan *ensemble learning* atau *deep learning*, serta integrasi data dinamis yang diperbarui secara berkala agar sistem tetap relevan dengan pola serangan terkini. Arah pengembangan selanjutnya meliputi penambahan lapisan keamanan tambahan, misalnya analisis konten berbasis *Natural Language Processing* (NLP) pada halaman *web* atau integrasi dengan *multi-factor authentication* (MFA), perluasan evaluasi menggunakan dataset IDN berskala global, serta penerapan pembelajaran adaptif dari umpan balik pengguna tanpa meninggalkan perangkat. Selain itu, optimasi model ringan perlu dilakukan untuk menjaga latensi *real-time*, terutama pada perangkat dengan spesifikasi rendah. Dengan adanya sistem ini, diharapkan pengguna internet dapat lebih terlindungi dari ancaman *phishing* yang semakin kompleks, sekaligus meningkatkan kesadaran akan pentingnya keamanan siber dalam aktivitas digital sehari-hari. Pada akhirnya, penelitian ini berkontribusi dalam menciptakan lingkungan digital yang lebih aman, terpercaya, dan berorientasi pada privasi pengguna.

REFERENSI

[1] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, "Phishing Attacks: A Recent Comprehensive Study and a New Anatomy," Mar. 09, 2021, *Frontiers Media S.A.* doi: 10.3389/fcomp.2021.563060.

[2] M. F. Ansari, P. K. Sharma, and B. Dash, "Prevention of Phishing Attacks Using AI-Based Cybersecurity Awareness Training," *International Journal of Smart Sensor and Adhoc Network.*, pp. 61–72, Mar. 2022, doi: 10.47893/ijssan.2022.1221.

[3] K. Oloyede *et al.*, "Impact Of Web (URL) Phishing and Its Detection," *International Journal of Scientific Research and Management (IJSRM)*, vol. 12, no. 04, pp. 484–493, Apr. 2024, doi: 10.18535/ijsrm/v12i04.m02.

[4] L. Tang and Q. H. Mahmoud, "A Survey of Machine Learning-Based Solutions for Phishing Website Detection," Sep. 01, 2021, *MDPI*. doi: 10.3390/make3030034.

[5] K. Omari and A. Oukhatar, "Advanced Phishing Website Detection with SMOTETomek-XGB: Addressing Class Imbalance for Optimal Results," in *Procedia Computer Science*, Elsevier B.V., 2025, pp. 289–295. doi: 10.1016/j.procs.2024.12.031.

[6] T. O. Ojewumi, G. O. Ogunleye, B. O. Oguntunde, O. Folorunsho, S. G. Fashoto, and N. Ogbu, "Performance evaluation of machine learning tools for detection of phishing attacks on web pages," *Sci Afr*, vol. 16, Jul. 2022, doi: 10.1016/j.sciaf.2022.e01165.

[7] A. A. Orunsolu, A. S. Sodiya, and A. T. Akinwale, "A predictive model for phishing detection," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 2, pp. 232–247, Feb. 2022, doi: 10.1016/j.jksuci.2019.12.005.

[8] Q. E. ul Haq, M. H. Faheem, and I. Ahmad, "Detecting Phishing URLs Based on a Deep Learning Approach to Prevent Cyber-Attacks," *Applied Sciences (Switzerland)*, vol. 14, no. 22, Nov. 2024, doi: 10.3390/app142210086.

[9] A. Safi and S. Singh, "A systematic literature review on phishing website detection techniques," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 2, pp. 590–611, Feb. 2023, doi: 10.1016/j.jksuci.2023.01.004.

[10] M. I. Alwanain, "Phishing Awareness and Elderly Users in Social Media," *International Journal of Computer Science and Network Security*, vol. 20, no. 9, pp. 114–119, Sep. 2020, doi: 10.22937/IJCSNS.2020.20.09.14.

[11] A. Sjösten, S. Van Acker, and A. Sabelfeld, "Discovering *Browser* Extensions via Web Accessible Resources," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, in CODASPY '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 329–336. doi: 10.1145/3029806.3029820.

[12] B. Jin, H. Li, and Y. Zou, "Impact of extensions on *browser* performance: An empirical study on google chrome," *Empir Softw Eng*, vol. 30, no. 4, p. 103, 2025, doi: 10.1007/s10664-025-10633-1.

[13] P. Picazo-Sanchez, L. Ortiz-Martin, G. Schneider, and A. Sabelfeld, "Are chrome extensions compliant with the spirit of least privilege?," *Int J Inf Secur*, vol. 21, no. 6, pp. 1283–1297, Dec. 2022, doi: 10.1007/s10207-022-00610-w.

[14] A. Nayak, R. Khandelwal, E. Fernandes, and K. Fawaz, "Experimental Security Analysis of Sensitive Data Access by *Browser* Extensions," in *WWW 2024 - Proceedings of the ACM Web Conference*, Association for Computing Machinery, Inc, May 2024, pp. 1283–1294. doi: 10.1145/3589334.3645683.

[15] Q. Wang *et al.*, "XGBoost algorithm assisted multi-component quantitative analysis with Raman spectroscopy," *Spectrochim Acta A Mol Biomol Spectrosc*, vol. 323, p. 124917, 2024, doi: https://doi.org/10.1016/j.saa.2024.124917.

[16] M. Bahaghighat, M. Ghasemi, and F. Ozen, "A high-accuracy phishing website detection method based on machine learning," *Journal of Information Security and Applications*, vol. 77, Sep. 2023, doi: 10.1016/j.jisa.2023.103553.

[17] J. Osamor, M. Ashawa, J. Riley, P. Owoh, A. Ajibade, and C. Iwendi, "*Real-time* Detection of Phishing Emails Using XG Boost Machine Learning Technique," 2024.

[18] S. Al-Saqqa, S. Sawalha, and H. Abdelnabi, "Agile software development: Methodologies and trends," *International Journal of Interactive Mobile Technologies*, vol. 14, no. 11, pp. 246–270, 2020, doi: 10.3991/ijim.v14i11.13269.

[19] L. T. M. Blessing and A. Chakrabarti, *DRM, a design research methodology*. Springer London, 2009. doi: 10.1007/978-1-84882-587-1.

[20] D. Hellhake, J. Bogner, T. Schmid, and S. Wagner, "Towards using coupling measures to guide black-box integration testing in component-based systems," *Software Testing Verification and Reliability*, vol. 32, no. 4, Jun. 2022, doi: 10.1002/stvr.1811.

[21] F. Rahmat Halim *et al.*, "RANCANG BANGUN SISTEM INFORMASI PENGUMUMAN KELULUSAN SISWA BERBASIS WEB MENGGUNAKAN METODE AGILE WEB-BASED STUDENT GRADUATION ANNOUNCEMENT INFORMATION SYSTEM DESIGN USING THE AGILE METHOD."

[22] V. D. Chavan, A. Gadekar, S. Bidwai, V. Gogi, and L. Kurapati, "Phishing Detection using Machine Learning and Chrome Extension," in *2nd IEEE International Conference on Advances in Information Technology, ICAIT 2024 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICAIT61638.2024.10690839.

[23] S. Kusawa, "Phishing URLs," 2025. [Online]. Available: https://www.kaggle.com/datasets/sunnykusawa/phishing-urls

[24] S. Ariyadasa, S. Fernando, and S. Fernando, "Phishing Websites Dataset," 2021. doi: 10.17632/n96ncsr5g4.1.

[25] G. Vrbancic, "Phishing-Dataset," 2019. [Online]. Available: https://github.com/GregaVrbancic/Phishing-Dataset

[26] fandcomp, "GuardSurfing-Extension-for-URL-Phishing," 2025. [Online]. Available: https://github.com/fandcomp/GuardSurfing-Extension-for-URL-Phishing