

**RANCANG BANGUN EKSTENSI VISUAL STUDIO CODE
UNTUK DETEKSI DINI HARCODED CREDENTIAL
MENGUNAKAN METODE PATTERN MATCHING (REGEX)
DAN SHANNON ENTROPY**



DISUSUN OLEH:

ASWANGGA SUJATI

NIM. 22106050053

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

**PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
YOGYAKARTA**

2026

LEMBAR PENGESAHAN



KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Marsda Adisucipto Telp. (0274) 540971 Fax. (0274) 519739 Yogyakarta 55281

PENGESAHAN TUGAS AKHIR

Nomor : B-2094/Un.02/DST/PP.00.9/08/2026

Tugas Akhir dengan judul : Rancang Bangun Ekstensi Visual Studio Code untuk Deteksi Dini Hardcoded Credential Menggunakan Metode Pattern Matching (Regex) dan Shannon Entropy

yang dipersiapkan dan disusun oleh:

Nama : ASWANGGA SUJATI
Nomor Induk Mahasiswa : 22106050053
Telah diujikan pada : Rabu, 19 Agustus 2026
Nilai ujian Tugas Akhir : A-

dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

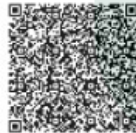
TIM UJIAN TUGAS AKHIR



Ketua Sidang

Ir. Maria Ulfah Siregar, S.Kom., MIT., Ph.D.
SIGNED

Valid ID: 6a89c9b1beb47



Penguji I

Dr. Fitri Wulandari, S.Si., M.Kom.
SIGNED

Valid ID: 6a891c36e2c90



Penguji II

Eko Hadi Gunawan, M.Eng.
SIGNED

Valid ID: 6a898dc9df40a



Yogyakarta, 19 Agustus 2026

UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

Prof. Dr. Dra. Hj. Khurul Wardati, M.Si.
SIGNED

Valid ID: 6a8bb736906c5

LEMBAR PERNYATAAN KEASLIAN**SURAT PERNYATAAN KEASLIAN**

Yang bertanda tangan dibawah ini:

Nama : Aswangga Sujati
NIM : 22106050053
Program Studi : Informatika
Fakultas : Sains dan Teknologi

Dengan ini menyatakan bahwa isi skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar sarjana di suatu Perguruan Tinggi dan sesungguhnya skripsi ini merupakan hasil pekerjaan penulis sendiri sepanjang pengetahuan penulis, bukan duplikasi atau saduran dari karya orang lain kecuali bagian tertentu yang penulis ambil sebagai bahan acuan. Apabila terbukti pernyataan ini tidak benar, sepenuhnya menjadi tanggung jawab penulis.

Yogyakarta,



Aswangga Sujati

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

LEMBAR PERSETUJUAN

LEMBAR PERSETUJUAN

Hal : Persetujuan Skripsi / Tugas Akhir

Lamp :

Kepada

Yth. Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
di Yogyakarta

Assalamu'alaikum wr. wb.

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka kami selaku pembimbing berpendapat bahwa skripsi Saudara:

Nama : Aswangga Sujati

NIM : 22106050053

Judul Skripsi : RANCANG BANGUN EKSTENSI VISUAL STUDIO CODE
UNTUK DETEKSI DINI HARCORED CREDENTIAL MENGGUNAKAN
METODE PATTERN MACHING (REGEX) DAN SHANNON ENTROPY

sudah dapat diajukan kembali kepada Program Studi Matematika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Program Studi Matematika.

Dengan ini kami berharap agar skripsi/tugas akhir Saudara tersebut di atas dapat segera dimunaqasyahkan. Atas perhatiannya kami ucapkan terima kasih.

Wassalamu'alaikum wr. wb.

Yogyakarta,
Pembimbing

Ir. Maria Ulfah Siregar, S.Kom., MIT.,
Ph.D.

NIP. 19780106 200212 2 001

INTISARI

Kebocoran *hardcoded credential* pada kode sumber dapat menimbulkan risiko keamanan karena kredensial yang tersimpan secara langsung dapat terekspos sebelum diketahui oleh pengembang. Penelitian ini bertujuan untuk merancang dan membangun ekstensi Visual Studio Code bernama CredGuard yang menerapkan konsep *Shift-Left Security* untuk mendeteksi *hardcoded credential* secara otomatis dan *real-time* selama proses penulisan kode. Penelitian ini menggunakan metode *Extreme Programming* (XP) yang meliputi tahapan *planning*, *design*, *coding*, dan *testing* yang dilakukan secara iteratif. CredGuard mengombinasikan metode *Pattern Matching* berbasis *Regular Expression* (Regex) untuk mendeteksi kredensial dengan pola tertentu dan Shannon Entropy untuk menganalisis string dengan tingkat keacakan tinggi. Pengujian dilakukan menggunakan *Black-box Testing* untuk memverifikasi fungsi sistem dan pengujian akurasi menggunakan *confusion matrix* terhadap 300 sampel yang terdiri atas 150 *Secret* dan 150 *Non-Secret*. Hasil pengujian akhir menunjukkan bahwa CredGuard memperoleh *Precision* sebesar 80,12%, *Recall* sebesar 86,00%, dan *F1-Score* sebesar 82,96%. Hasil tersebut menunjukkan bahwa CredGuard mampu mendeteksi *hardcoded credential* pada kode sumber pada lima jenis berkas, yaitu JavaScript, TypeScript, Python, PHP, dan .env, dengan kemampuan deteksi yang meningkat setelah dilakukan penyempurnaan pada iterasi ketiga. Berdasarkan hasil penelitian, CredGuard dapat digunakan sebagai mekanisme deteksi dini *hardcoded credential* pada lingkungan pengembangan Visual Studio Code dengan mengombinasikan pendekatan berbasis pola dan karakteristik keacakan string.

Kata Kunci: CredGuard, *Hardcoded Credential*, Visual Studio Code, *Shift-Left Security*, *Pattern Matching*, Regex, Shannon Entropy.

ABSTRACT

Hardcoded credentials embedded in source code can pose security risks because credentials stored directly in source code may be exposed before they are identified by developers. This research aims to design and develop a Visual Studio Code extension named CredGuard that applies the Shift-Left Security concept to detect hardcoded credentials automatically and in real time during the coding process. The research employs the Extreme Programming (XP) methodology, consisting of iterative planning, design, coding, and testing activities. CredGuard combines Pattern Matching based on Regular Expression (Regex) to detect credentials with specific patterns and Shannon Entropy to analyze strings with high levels of randomness. Testing was conducted using Black-box Testing to verify system functionality and accuracy evaluation using a confusion matrix on 300 samples consisting of 150 Secret and 150 Non-Secret samples. The final evaluation resulted in a Precision of 80.12%, a Recall of 86.00%, and an F1-Score of 82.96%. These results indicate that CredGuard is capable of detecting hardcoded credentials in five supported file types, namely JavaScript, TypeScript, Python, PHP, and .env, with improved detection performance after refinement in the third iteration. Based on the results, CredGuard can be used as an early hardcoded credential detection mechanism within the Visual Studio Code development environment by combining pattern-based detection and string randomness analysis.

Keywords: CredGuard, Hardcoded Credential, Visual Studio Code, Shift-Left Security, Pattern Matching, Regex, Shannon Entropy.

MOTTO

“Barangkali aku belum menemukan arah, tetapi setidaknya aku telah belajar untuk tidak berhenti berjalan.”

"Skripsi tidak menuntutmu menjadi sempurna. Skripsi hanya menuntutmu untuk menyelesaikannya. Maka berhentilah berunding dengan kesempurnaan, dan mulailah menulis."

"Fa inna ma'al 'usri yusra, inna ma'al 'usri yusra"

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

HALAMAN PERSEMBAHAN

Bismillahirrahmanirrahim.

Dengan menyebut nama Allah Yang Maha Pengasih lagi Maha Penyayang.

Dengan ini, dengan penuh perasaan, mempersembahkan skripsi ini dari penulis

kepada:

Kedua orang tua, Ibu dan Bapak.



STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

KATA PENGANTAR

Assalamu'alaikum warahmatullahi wabarakatuh.

Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat, taufik, dan hidayah-Nya, sehingga penulis dapat menyelesaikan penyusunan skripsi ini dengan baik. Sholawat serta salam senantiasa tercurah kepada Nabi Muhammad SAW yang telah membawa umat manusia dari zaman kegelapan menuju cahaya kebenaran.

Skripsi yang berjudul “Rancang Bangun Ekstensi Visual Studio Code untuk Deteksi Dini Hardcoded Credential Menggunakan Metode Pattern Matching (Regex) dan Shannon Entropy” ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Informatika, Fakultas Sains dan Teknologi, UIN Sunan Kalijaga Yogyakarta.

Penyusunan skripsi ini tentu tidak terlepas dari dukungan, bantuan, dan doa dari berbagai pihak. Oleh karena itu, penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Bapak Prof. Noorhaidi Hasan, S.Ag., M.A., M.Phil., Ph.D. selaku Rektor Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
2. Ibu Prof. Dr. Dra. Hj. Khurul Wardati, M.Si. selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
3. Bapak Dr. Muhammad Mustakim, S.T., M.T. selaku Ketua Program Studi Informatika yang telah memberikan dukungan selama masa perkuliahan.
4. Bapak Muhammad Didik Rohmad Wahyudi, S.T., M.T. selaku dosen pembimbing akademik yang telah membimbing, mengarahkan, dan mendampingi penulis selama menjalani proses perkuliahan.
5. Ibu Ir. Maria Ulfah Siregar, S.Kom., MIT., Ph.D. selaku dosen pembimbing skripsi yang telah meluangkan waktu, memberikan bimbingan, arahan, koreksi, serta motivasi kepada penulis selama proses penyusunan hingga penyelesaian skripsi ini.

6. Kedua orang tua tercinta, Ibu dan Bapak, yang selalu menjadi sumber kekuatan dan alasan penulis untuk terus melangkah. Terima kasih atas dukungan penuh berupa materi, kasih sayang yang tak terhitung, serta doa-doa tulus yang tidak pernah putus untuk keberhasilan dan kelancaran penulis dalam menempuh pendidikan ini.
7. Untuk seseorang, barangkali engkau tidak pernah datang untuk menjadi bagian dari akhir cerita, melainkan hanya untuk membuat penulis berani menulis halaman-halaman berikutnya. .
8. Sahabat “RPL Infomaret”, Akbel, Afrizal dan Fakhri yang telah kebersamai penulis dalam berbagai kegiatan, baik di dalam maupun di luar kampus. Terima kasih telah menjadi rekan diskusi dan selalu membantu penulis dalam menghadapi berbagai tantangan serta kerumitan selama masa perkuliahan.
9. Seluruh pihak yang tidak dapat penulis sebutkan satu per satu, terima kasih atas segala bentuk bantuan, doa, dan dukungan yang telah diberikan selama proses penyusunan skripsi ini.
10. Terima kasih sudah bertahan sampai di titik ini. Meskipun selama perjalanan ini tidak sedikit waktu yang terbuang, rencana yang berantakan, rasa malas yang sulit dilawan, dan hari-hari ketika rasanya ingin menyerah saja. Aku mungkin belum menjadi versi terbaik dari diriku, bahkan masih banyak hal yang harus diperbaiki setelah halaman ini selesai, tetapi setidaknya hari ini aku bisa mengatakan bahwa aku tidak berhenti di tengah jalan. Untuk itu, terima kasih.

Akhir kata, penulis berharap skripsi ini dapat memberikan manfaat, baik bagi pembaca maupun bagi pengembangan penelitian di bidang keamanan perangkat lunak, khususnya dalam upaya mendeteksi kebocoran credential pada kode sumber sejak tahap pengembangan. Penulis juga menyadari bahwa perjalanan belajar tidak berhenti setelah skripsi ini selesai. Masih banyak hal yang perlu dipelajari, diperbaiki, dan dikembangkan di masa mendatang.

DAFTAR ISI

LEMBAR PENGESAHAN.....	2
LEMBAR PERNYATAAN KEASLIAN.....	3
LEMBAR PERSETUJUAN.....	4
INTISARI.....	5
ABSTRACT.....	6
MOTTO.....	7
HALAMAN PERSEMBAHAN.....	8
KATA PENGANTAR.....	9
DAFTAR ISI.....	11
DAFTAR TABEL.....	15
DAFTAR GAMBAR.....	16
BAB I.....	18
PENDAHULUAN.....	18
1.1 Latar Belakang.....	18
1.2 Rumusan Masalah.....	24
1.3 Tujuan.....	24
1.4 Manfaat.....	25
1.4.1 Manfaat Akademis.....	25
1.4.2 Manfaat Praktis.....	25
1.5 Batasan Masalah.....	25
BAB II.....	28
TINJAUAN PUSTAKA.....	28
2.1 Penelitian Terdahulu.....	28

BAB III.....	36
METODE PENGEMBANGAN SISTEM.....	36
3.1 Lokasi dan Waktu.....	36
3.2 Alat.....	36
3.3 Bahan.....	37
3.4 Metode Pengembangan Sistem.....	38
3.4.1 Planning.....	39
3.4.2 Design.....	40
3.4.3 Coding.....	40
3.4.4 Testing.....	40
3.5 Metode Pengujian.....	40
3.5.1 Black-box Testing.....	40
3.5.2 Pengujian Akurasi.....	41
BAB IV.....	44
HASIL DAN PEMBAHASAN.....	44
4.1 Iterasi Pertama.....	44
4.1.1 Planning.....	44
4.1.2 Design.....	46
a. Use Case Diagram.....	46
b. Activity Diagram.....	48
4.1.3 Coding.....	49
4.1.4 Testing.....	51
4.2 Iterasi Kedua.....	53
4.2.1 Planning.....	53
4.2.2 Design.....	55

4.2.2.1 Sequence Diagram.....	56
4.2.2.2 Component Diagram.....	57
4.2.3 Coding.....	57
4.2.3.1 Implementasi Pattern Matching (Regex).....	57
4.2.3.2 Implementasi Shannon Entropy.....	59
4.2.4 Testing.....	61
4.2.4.1 Pengujian Deteksi AWS Acces Key (CG001).....	63
4.2.4.2 Pengujian Deteksi Google Cloud API Key (CG002).....	64
4.2.4.3 Pengujian Deteksi Stripe Secret Key (CG003).....	64
4.2.4.4 Pengujian Deteksi GitHub Personal Access Token (CG004).....	65
4.2.4.5 Pengujian Deteksi JSON Web Token (CG005).....	65
4.2.4.6 Pengujian Deteksi Private Key (CG006).....	66
4.2.4.7 Pengujian Deteksi Password Hardcoded (CG007).....	66
4.2.4.8 Pengujian Deteksi Slack Bot Token (CG008).....	67
4.2.4.9 Pengujian Deteksi Shannon Entropy (CG009).....	67
4.2.5 Pengujian Akurasi Awal.....	68
4.3 Iterasi Ketiga.....	69
4.3.1 Planning.....	69
4.3.2 Design.....	71
4.3.3 Coding.....	72
4.3.3.1 Penyempurnaan Regex AWS Access Key (CG001).....	72
4.3.3.2 Penyempurnaan Regex Google Cloud API Key (CG002).....	73
4.3.3.3 Penyempurnaan Regex GitHub Personal Access Token (CG004).....	73
4.3.3.4 Penyempurnaan Deteksi Password Hardcoded (CG007).....	74
4.3.3.5 Penyempurnaan Parser TypeScript.....	74

4.3.3.6 Penambahan Filtering Shannon Entropy.....	75
4.3.3.7 Evaluasi Threshold Shannon Entropy.....	77
4.3.4 Testing.....	79
4.4 Hasil Pengujian.....	82
4.4.1 Skenario Pengujian.....	82
4.4.2 Rekapitulasi Confusion Matrix.....	83
4.4.3 Perhitungan <i>Precision</i> , <i>Recall</i> , dan <i>F1-Score</i>	84
4.4.3.1 Precision.....	84
4.4.3.2 Recall.....	84
4.4.3.3 F1-Score.....	84
4.4.4 Analisi Hasil Pengujian.....	85
BAB V.....	88
KESIMPULAN DAN SARAN.....	88
5.1 Kesimpulan.....	88
5.2 Saran.....	90
DAFTAR PUSTAKA.....	91
LAMPIRAN.....	94
Lampiran 1. Dataset.....	94
Lampiran 2. Source Code Utama CredGuard.....	102
Lampiran 3. Repository GitHub.....	111
Lampiran 4. File VSIX.....	112

DAFTAR TABEL

Tabel 2. 1 Ringkasan Tinjauan Pustaka.....	32
Tabel 3. 1 Spesifikasi Perangkat Keras.....	36
Tabel 3. 2 Perangkat Lunak yang Digunakan.....	36
Tabel 3. 3 Hubungan antar label.....	42
Tabel 4. 1 User Story Iterasi 1.....	45
Tabel 4. 2 Kebutuhan Fungsional Iterasi 1.....	45
Tabel 4. 3 Kebutuhan Non-fungsional Iterasi 1.....	46
Tabel 4. 4 BlackBox Testing Iterasi 1.....	51
Tabel 4. 5 User Story Iterasi 2.....	54
Tabel 4. 6 Kebutuhan Fungsional Iterasi 2.....	55
Tabel 4. 7 BlackBox Testing Iterasi 2.....	62
Tabel 4. 8 Pengujian Akurasi Awal Setelah Iterasi 2.....	68
Tabel 4. 9 User Story Iterasi 3.....	70
Tabel 4. 10 Kebutuhan Fungsional Iterasi 3.....	71
Tabel 4. 11 Evaluasi Nilai Entropy.....	78
Tabel 4. 12 BlackBox Testing Iterasi 3.....	79
Tabel 4. 13 Kategori Hasil Pengujian.....	83
Tabel 4. 14 Rekapitulasi Hasil Pengujian Akurasi Akhir.....	83
Tabel 4. 15 Hasil Precision, Recall, dan F1-Score.....	85
Tabel 4. 16 Perbandingan Hasil Pengujian Sebelum dan Setelah Iterasi Ketiga...	86

DAFTAR GAMBAR

Gambar 4. 1 Use Case.....	46
Gambar 4. 2 Activity Diagram.....	48
Gambar 4. 3 Implementasi fungsi activate().....	49
Gambar 4. 4 Implementasi fungsi scanDocument().....	50
Gambar 4. 5 Struktur direktori CredGuard.....	51
Gambar 4. 6 Aktivasi ekstensi pada Debug Console.....	51
Gambar 4. 7 Diagnostic Warning pada Visual Studio Code.....	52
Gambar 4. 8 Sequence Diagram.....	56
Gambar 4. 9 Component Diagram.....	57
Gambar 4. 10 Pattern Dictionary.....	58
Gambar 4. 11 Implementasi fungsi detectWithRegex().....	59
Gambar 4. 12 Implementasi Config dan calculateShannonEntropy.....	60
Gambar 4. 13 Implementasi fungsi detectWithEntropy().....	61
Gambar 4. 14 Hasil Pengujian CG001.....	63
Gambar 4. 15 Hasil Pengujian CG002.....	64
Gambar 4. 16 Hasil Pengujian CG003.....	64
Gambar 4. 17 Hasil Pengujian CG004.....	65
Gambar 4. 18 Hasil Pengujian CG005.....	65
Gambar 4. 19 Hasil Pengujian CG006.....	66
Gambar 4. 20 Hasil Pengujian CG007.....	66
Gambar 4. 21 Hasil Pengujian CG008.....	67
Gambar 4. 22 Hasil Pengujian CG009.....	67
Gambar 4. 23 Penyempurnaan Regular Expression C001.....	73
Gambar 4. 24 Penyempurnaan Regular Expression CG002.....	73
Gambar 4. 25 Penyempurnaan Regular Expression CG004.....	74
Gambar 4. 26 Penyempurnaan Regular Expression CG007.....	74
Gambar 4. 27 Perubahan Assignment Pattern pada modul entropyDetector.ts.....	75
Gambar 4. 28 Penyempurnaan filtering SECRET_CONTEXT.....	76
Gambar 4. 29 Penyempurnaan filtering HIGH_ENTROPY_CHARSETS.....	76
Gambar 4. 30 Penyempurnaan filtering SKIP_KEYWORDS.....	77

Gambar 4. 31 Konfigurasi parameter Shannon Entropy pada entropyDetector.ts.78



BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam proses pengembangan perangkat lunak, pengembang sering menggunakan berbagai kredensial digital seperti API key, access token, secret key, dan password agar aplikasi dapat melakukan autentikasi dan mengakses layanan tertentu. Kredensial tersebut digunakan untuk mengakses layanan cloud, basis data, maupun layanan pihak ketiga. Oleh karena itu, pengelolaan kredensial menjadi salah satu aspek penting dalam menjaga keamanan aplikasi karena kredensial yang jatuh ke tangan pihak yang tidak berwenang dapat disalahgunakan untuk mengakses sumber daya yang dilindungi[1], [2].

Namun, dalam praktiknya, pengembang masih menyimpan kredensial secara langsung di dalam source code atau file konfigurasi, yang dikenal sebagai hardcoded credential. Praktik tersebut dapat menimbulkan risiko keamanan apabila source code dipublikasikan atau tersimpan pada repositori yang dapat diakses oleh pihak yang tidak berwenang[1]. Berbagai penelitian menunjukkan bahwa kebocoran kredensial masih ditemukan pada repositori perangkat lunak. Kondisi tersebut menunjukkan bahwa penyimpanan kredensial secara langsung di dalam source code masih menjadi permasalahan dalam pengembangan perangkat lunak[2], [3].

Kredensial yang terekspos dapat dimanfaatkan oleh pihak yang tidak berwenang untuk memperoleh akses terhadap layanan atau sumber daya yang dilindungi. Kondisi tersebut dapat menimbulkan berbagai risiko keamanan bagi pengembang maupun organisasi. Oleh karena itu, perlindungan terhadap kredensial perlu menjadi bagian dari proses pengembangan perangkat lunak[4].

Salah satu contoh kasus kebocoran kredensial terjadi pada Mercedes-Benz pada tahun 2024. Sebuah *authentication token* yang tersimpan pada repositori GitHub menyebabkan pihak yang tidak berwenang memperoleh akses terhadap platform internal perusahaan. Meskipun token tersebut kemudian dicabut, kasus tersebut menunjukkan bahwa satu kredensial yang tersimpan secara tidak aman

dapat membuka akses terhadap sumber daya internal organisasi[5], [6]. Kasus tersebut menunjukkan pentingnya mendeteksi kredensial sebelum kode dipublikasikan ke repositori. Oleh karena itu, diperlukan mekanisme yang dapat membantu pengembang menemukan *hardcoded credential* sedini mungkin selama proses pengembangan[7].

Permasalahan kebocoran kredensial telah mendorong pengembangan berbagai alat untuk mendeteksi secret pada kode sumber. Alat seperti TruffleHog dan Gitleaks digunakan untuk memindai kode dan repositori guna menemukan kredensial yang terindikasi terekspos. Selain itu, GitHub menyediakan fitur *secret scanning* untuk mendeteksi *secret* yang terdapat dalam repositori. Keberadaan berbagai alat tersebut menunjukkan bahwa deteksi otomatis telah menjadi salah satu upaya untuk mengurangi risiko kebocoran kredensial[3], [8].

Metode deteksi secret terus berkembang dan dapat dibedakan menjadi pendekatan berbasis aturan (*rule-based*) dan pendekatan berbasis pembelajaran (*learning-based*). Pendekatan berbasis aturan menggunakan pola yang telah ditentukan, seperti *Regular Expression* (Regex), serta karakteristik statistik seperti nilai entropi untuk mengenali string yang berpotensi sebagai kredensial. Sementara itu, pendekatan berbasis pembelajaran memanfaatkan model pembelajaran mesin atau *Large Language Model* (LLM) untuk menganalisis karakteristik maupun konteks kode dalam proses deteksi[9], [10].

Pendekatan berbasis aturan masih digunakan dalam berbagai alat deteksi secret karena dapat mengenali kredensial berdasarkan pola maupun karakteristik string. Regex digunakan untuk mendeteksi kredensial yang memiliki format tertentu, sedangkan entropi digunakan untuk mengidentifikasi string dengan tingkat keacakan tinggi. Namun, metode tersebut memiliki keterbatasan dalam memahami konteks kode sehingga string yang menyerupai kredensial, seperti data contoh atau *placeholder*, dapat ikut terdeteksi. Kondisi ini dapat menghasilkan *false positive* dan memengaruhi kepercayaan pengguna terhadap hasil deteksi[4], [8].

Keterbatasan tersebut mendorong pengembangan pendekatan berbasis pembelajaran dalam proses deteksi informasi sensitif. Salah satu penelitian menggunakan *Convolutional Neural Network* (CNN) dengan *Fuzzy Edge*

Detection, sedangkan penelitian lain memanfaatkan LLM untuk menganalisis konteks kode dalam proses deteksi *secret*. Perkembangan tersebut menunjukkan adanya upaya untuk melengkapi keterbatasan pendekatan berbasis aturan melalui analisis karakteristik dan konteks yang lebih kompleks[10], [11].

Meskipun metode deteksi terus berkembang, solusi yang dibahas dalam penelitian terdahulu banyak diterapkan pada repositori, sistem kontrol versi, atau kode sumber yang telah tersedia sebagai masukan bagi alat deteksi. Pada kondisi tersebut, kredensial dapat terlebih dahulu ditulis dan berpotensi masuk ke repositori sebelum ditemukan oleh mekanisme pemindaian. Hal ini menunjukkan bahwa perkembangan metode deteksi masih perlu disertai dengan upaya untuk menemukan *hardcoded credential* lebih awal selama proses pengembangan[3], [8].

Berdasarkan kondisi tersebut, proses deteksi perlu ditempatkan lebih dekat dengan tahap munculnya kesalahan, yaitu ketika pengembang masih menulis kode. Pendekatan ini sejalan dengan konsep Shift-Left Security yang mendorong aktivitas keamanan dilakukan lebih awal dalam siklus pengembangan perangkat lunak. Dengan mendeteksi masalah pada tahap pengembangan, pengembang memiliki kesempatan untuk melakukan perbaikan sebelum kode dikirim ke repositori atau memasuki tahap pengembangan berikutnya[7].

Pemilihan sisi pengembang sebagai tempat penerapan deteksi didasarkan pada posisi pengembang sebagai pihak yang menulis dan memasukkan kredensial ke dalam kode program. Apabila deteksi baru dilakukan setelah kode dikirim ke repositori, kredensial telah berpotensi tersimpan dalam riwayat perubahan kode. Oleh karena itu, deteksi selama proses penulisan kode memberikan kesempatan kepada pengembang untuk memperbaiki kesalahan sebelum kredensial masuk ke repositori[3], [7].

Integrated Development Environment (IDE) dipilih sebagai tempat penerapan deteksi karena menjadi lingkungan yang digunakan pengembang untuk menulis dan mengubah kode program. Integrasi mekanisme keamanan ke dalam IDE memungkinkan pemeriksaan dilakukan ketika proses pengembangan masih berlangsung dan peringatan diberikan langsung pada bagian kode yang terindikasi

mengandung masalah. Dengan demikian, pengembang dapat melakukan pemeriksaan tanpa menunggu kode dikirim ke repositori[12], [13].

Untuk mendukung proses deteksi pada lingkungan pengembangan, penelitian ini menggunakan *Pattern Matching* berbasis *Regular Expression* (Regex) sebagai metode pertama. Regex dapat digunakan untuk mengenali kredensial berdasarkan karakteristik format yang telah ditentukan, seperti awalan, panjang, atau susunan karakter tertentu. Pendekatan tersebut sesuai untuk mendeteksi jenis kredensial yang memiliki pola yang dapat dikenali, seperti API key dan token dari layanan tertentu[8], [9].

Meskipun dapat digunakan untuk mengenali kredensial dengan pola tertentu, *Pattern Matching* memiliki keterbatasan ketika menghadapi *secret* yang tidak memiliki format baku. Beberapa kredensial dapat berupa rangkaian karakter acak tanpa awalan atau struktur khusus sehingga tidak sesuai dengan pola Regex yang telah ditentukan. Selain itu, ketergantungan terhadap aturan yang telah dibuat menyebabkan pola perlu disesuaikan ketika format kredensial berubah. Kondisi tersebut menunjukkan bahwa penggunaan *Pattern Matching* saja belum cukup untuk mencakup karakteristik *hardcoded credential* yang beragam[8], [9].

Untuk melengkapi keterbatasan tersebut, penelitian ini menggunakan *Shannon Entropy* sebagai metode kedua. *Shannon Entropy* digunakan untuk mengukur tingkat ketidakpastian berdasarkan distribusi karakter dalam suatu string. Semakin beragam dan merata distribusi karakter, semakin tinggi nilai entropi yang dihasilkan. Karakteristik tersebut dapat digunakan sebagai dasar untuk menganalisis string yang memiliki tingkat keacakan tinggi[14]. Dalam konteks deteksi *secret*, nilai entropi dapat digunakan untuk membantu menemukan string acak yang tidak dapat dikenali hanya berdasarkan pola tertentu. Pendekatan berbasis entropi juga telah digunakan dalam alat dan penelitian deteksi *secret* untuk menganalisis string yang berpotensi mengandung kredensial[8], [9].

Pattern Matching dan *Shannon Entropy* dipilih karena memiliki fungsi yang berbeda dan dapat saling melengkapi. *Pattern Matching* digunakan untuk mendeteksi kredensial yang memiliki format atau pola tertentu, sedangkan *Shannon Entropy* digunakan untuk membantu menemukan string acak yang tidak

memiliki pola baku. Dengan demikian, proses deteksi tidak hanya bergantung pada pola kredensial yang telah diketahui, tetapi juga mempertimbangkan karakteristik keacakan suatu string[8], [9].

Perkembangan penelitian *secret detection* juga telah mengarah pada penggunaan LLM untuk menganalisis konteks kode. Salah satu penelitian menggunakan Regex untuk memperoleh kandidat *secret*, kemudian memanfaatkan LLM untuk menentukan apakah kandidat tersebut merupakan *secret* sebenarnya. Penelitian lain mengembangkan pendekatan *multi-agent* untuk melakukan pemeriksaan bertingkat terhadap informasi sensitif dan menganalisis konteks yang berkaitan dengan kandidat *secret*[10], [15].

Meskipun pendekatan berbasis model menawarkan kemampuan analisis konteks yang lebih kompleks, penelitian ini tidak berfokus pada pengembangan model *machine learning*, LLM, maupun sistem *multi-agent*. Penelitian ini ditujukan sebagai mekanisme deteksi yang berjalan secara lokal pada lingkungan pengembangan dengan menggunakan *Pattern Matching* dan *Shannon Entropy*. Pemilihan tersebut disesuaikan dengan ruang lingkup penelitian, yaitu membangun ekstensi IDE yang dapat memberikan peringatan selama proses penulisan kode.

Berdasarkan penelitian terdahulu, berbagai pendekatan telah dikembangkan untuk mendeteksi *secret* pada kode sumber. Pendekatan berbasis aturan menggunakan Regex dan entropi untuk mengenali pola maupun karakteristik string, sedangkan pendekatan berbasis pembelajaran memanfaatkan CNN dan LLM untuk menganalisis karakteristik atau konteks kode. Penelitian yang lebih baru juga mulai menggunakan sistem *multi-agent* untuk melakukan pemeriksaan bertingkat terhadap kandidat *secret*. Perkembangan tersebut menunjukkan bahwa penelitian *secret detection* terus diarahkan pada peningkatan kemampuan deteksi dan pengurangan kesalahan hasil deteksi[10], [11], [15].

Meskipun berbagai metode telah dikembangkan, penelitian terdahulu lebih banyak berfokus pada peningkatan metode deteksi dan analisis terhadap kode sumber, repositori, atau sistem kontrol versi. Sementara itu, penerapan deteksi hardcoded credential secara langsung pada lingkungan pengembangan masih terbatas, khususnya dalam bentuk ekstensi IDE yang mengombinasikan Pattern

Matching dan Shannon Entropy untuk memberikan peringatan selama proses penulisan kode. Kondisi tersebut menunjukkan adanya peluang untuk menerapkan mekanisme deteksi pada tahap yang lebih dekat dengan aktivitas pengembang[7], [8], [9].

Penelitian ini tidak bertujuan menggantikan alat pemindai repositori maupun pendekatan berbasis model yang telah dikembangkan sebelumnya. Penelitian ini mengambil posisi pada tahap yang lebih awal, yaitu ketika pengembang masih menulis dan mengubah kode program. Dengan menempatkan proses deteksi pada lingkungan pengembangan, *hardcoded credential* dapat diketahui sebelum kode dikirim ke repositori[7].

Berdasarkan permasalahan dan celah penelitian tersebut, penelitian ini mengusulkan **CredGuard**, yaitu ekstensi Visual Studio Code untuk mendeteksi *hardcoded credential* selama proses penulisan kode. CredGuard menggunakan *Pattern Matching* berbasis Regex untuk mendeteksi kredensial yang memiliki pola tertentu dan *Shannon Entropy* untuk menganalisis string dengan tingkat keacakan tinggi. Hasil deteksi ditampilkan sebagai peringatan pada editor agar pengembang dapat memeriksa bagian kode yang terindikasi mengandung kredensial.

Penerapan kedua metode tersebut memungkinkan proses deteksi mempertimbangkan dua karakteristik kredensial, yaitu pola yang dapat dikenali dan tingkat keacakan string. Mekanisme deteksi diterapkan secara lokal pada lingkungan pengembangan dan diarahkan untuk memberikan peringatan selama proses penulisan kode tanpa menggunakan analisis berbasis *machine learning* maupun LLM.

Untuk mengetahui kemampuan metode yang diterapkan, hasil deteksi CredGuard dievaluasi berdasarkan *confusion matrix* menggunakan metrik *Precision*, *Recall*, dan *F1-Score*. *Precision* digunakan untuk mengukur proporsi hasil deteksi positif yang benar, *Recall* digunakan untuk mengukur kemampuan sistem dalam menemukan kredensial yang terdapat pada data pengujian, sedangkan *F1-Score* digunakan untuk melihat keseimbangan antara *Precision* dan *Recall*. Evaluasi tersebut digunakan untuk mengetahui kemampuan deteksi yang dihasilkan oleh penerapan *Pattern Matching* dan *Shannon Entropy*[8].

Melalui penelitian ini, CredGuard diharapkan dapat membantu pengembang mengenali *hardcoded credential* lebih awal selama proses penulisan kode. Penerapan deteksi pada lingkungan pengembang diharapkan dapat melengkapi mekanisme keamanan yang dilakukan pada tahap repositori atau tahap berikutnya. Dengan demikian, penelitian ini berfokus pada pengembangan dan evaluasi ekstensi Visual Studio Code yang menerapkan *Pattern Matching* dan *Shannon Entropy* untuk mendeteksi *hardcoded credential* pada sisi pengembang

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana merancang dan membangun ekstensi Visual Studio Code berbasis *Shift-left Security* untuk mendeteksi *hardcoded credential* pada file berekstensi .js, .ts, .py, .php, dan .env secara otomatis dan *real-time*?
2. Bagaimana mengimplementasikan kombinasi *Pattern Matching* dan *Shannon Entropy* untuk mendeteksi *hardcoded credential* yang memiliki pola tertentu maupun karakteristik string acak?
3. Bagaimana mengevaluasi tingkat akurasi deteksi menggunakan metrik *Precision*, *Recall*, dan *F1-Score* guna menekan angka false positive?

1.3 Tujuan

Adapun tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Merancang dan membangun ekstensi Visual Studio Code berbasis *Shift-Left Security* yang dapat mendeteksi *hardcoded credential* pada file berekstensi .js, .ts, .py, .php, dan .env secara otomatis dan *real-time* selama proses penulisan kode.
2. Mengimplementasikan kombinasi *Pattern Matching* dan *Shannon Entropy* untuk mendeteksi *hardcoded credential* yang memiliki pola tertentu maupun karakteristik string acak.
3. Mengevaluasi tingkat akurasi deteksi menggunakan metrik *Precision*, *Recall*, dan *F1-Score*.

1.4 Manfaat

Penelitian dan pengembangan ekstensi Visual Studio Code ini diharapkan dapat memberikan manfaat yang signifikan, baik secara teoritis maupun praktis, antara lain:

1.4.1 Manfaat Akademis

1. Memberikan kontribusi pada kajian keamanan perangkat lunak, khususnya mengenai penerapan konsep *Shift-Left Security* melalui deteksi *hardcoded credential* pada lingkungan pengembangan (*Integrated Development Environment*).
2. Memberikan hasil evaluasi empiris terhadap penerapan kombinasi *Pattern Matching* (Regex) dan *Shannon Entropy* dalam mendeteksi *hardcoded credential* berdasarkan metrik *Precision*, *Recall*, dan *F1-Score*.

1.4.2 Manfaat Praktis

1. Bagi Pengembang Perangkat Lunak: Memberikan alat bantu berupa ekstensi Visual Studio Code yang dapat mendeteksi dan memberikan peringatan terhadap *hardcoded credential* selama proses penulisan kode sehingga pengembang dapat melakukan pemeriksaan sebelum kode dikirim ke repositori.
2. Bagi Pengembang Keamanan Perangkat Lunak: Memberikan contoh penerapan deteksi *hardcoded credential* pada sisi pengembang sebagai bagian dari pendekatan *Shift-Left Security* yang menempatkan proses keamanan lebih awal dalam siklus pengembangan perangkat lunak.
3. Bagi Peneliti Selanjutnya: Menjadi referensi untuk penelitian selanjutnya dalam pengembangan metode deteksi *hardcoded credential*, baik melalui pengembangan metode berbasis aturan maupun integrasi dengan pendekatan lain seperti analisis konteks, *machine learning*, atau *Large Language Model*.

1.5 Batasan Masalah

Agar penelitian ini lebih terarah dan tidak menyimpang dari tujuan yang telah ditetapkan, maka diberikan batasan masalah sebagai berikut:

1. Lingkungan Pengembangan

Sistem yang dibangun berupa ekstensi yang hanya beroperasi pada *Visual Studio Code* sebagai lingkungan pengembangan.

2. Cakupan Ekstensi File

Proses pemindaian dibatasi pada file berekstensi .js (JavaScript), .ts (TypeScript), .py (Python), .php (PHP), dan .env (*environment variables*).

3. Ruang Lingkup Hardcoded Credential

Objek deteksi dibatasi menjadi dua kategori sebagai berikut:

- a. Kredensial berpola, yaitu kredensial yang memiliki karakteristik atau format tertentu dan dideteksi menggunakan *Pattern Matching* (Regex). Deteksi mencakup AWS Access Key, Google Cloud API Key, Stripe Secret Key, GitHub Personal Access Token, JSON Web Token (JWT), Private Key (PEM), Password Hardcoded, dan Slack Bot Token.
- b. Kredensial berupa string acak, yaitu string tanpa pola khusus yang dianalisis menggunakan *Shannon Entropy*. Analisis dibatasi berdasarkan panjang minimum string dan nilai ambang batas (*threshold*) entropi yang ditentukan dalam penelitian.

4. Batasan Metode Deteksi

Pattern Matching (Regex) digunakan untuk mendeteksi kredensial berdasarkan pola yang telah ditentukan, sedangkan *Shannon Entropy* digunakan untuk menganalisis tingkat keacakan string. Penelitian ini tidak menggunakan *machine learning* maupun *Large Language Model* dalam proses deteksi.

5. Batasan Pengujian Akurasi

Evaluasi akurasi deteksi dilakukan menggunakan confusion matrix berdasarkan nilai True Positive (TP), False Positive (FP), True Negative (TN), dan False Negative (FN). Hasil tersebut digunakan untuk menghitung metrik Precision, Recall, dan F1-Score.

6. Data Pengujian

Pengujian dilakukan secara lokal menggunakan dataset pengujian berupa potongan kode sumber yang mengandung dummy credential dan data bukan kredensial. Dataset digunakan untuk menguji kemampuan sistem dalam membedakan data yang termasuk *hardcoded credential* dan data yang tidak termasuk *hardcoded credential*.

7. Pengujian Performa

Pengujian performa dilakukan sebagai pengujian tambahan untuk mengamati kinerja ekstensi selama proses deteksi. Pengujian ini tidak menjadi bagian dari evaluasi akurasi utama yang diukur menggunakan Precision, Recall, dan F1-Score.

8. Keterbatasan Sistem

Ekstensi hanya menganalisis string yang terdapat secara literal di dalam file yang dipindai. Sistem tidak melakukan data flow analysis dan tidak mendeteksi kredensial yang telah dienkripsi, di-encode, atau di-obfuscate.



BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian ekstensi CredGuard, dapat diperoleh beberapa kesimpulan sebagai berikut.

1. CredGuard berhasil dirancang dan dibangun sebagai ekstensi Visual Studio Code berbasis konsep Shift-Left Security untuk mendeteksi hardcoded credential pada file JavaScript (.js), TypeScript (.ts), Python (.py), PHP (.php), dan Environment Variables (.env). Penerapan Shift-Left Security ditunjukkan dengan penempatan proses deteksi langsung pada lingkungan pengembangan ketika developer masih menulis atau mengubah kode, sehingga hasil deteksi dapat diketahui sebelum source code dikirim ke repository. Kemampuan pemindaian otomatis dan real-time dibuktikan melalui implementasi event perubahan dokumen dengan mekanisme debounce sebesar 300 milidetik serta hasil Black-box Testing yang menunjukkan bahwa perubahan dokumen dapat memicu proses pemindaian dan menghasilkan Diagnostic Warning sesuai skenario pengujian. Pengembangan dilakukan menggunakan Extreme Programming melalui tiga iterasi, di mana hasil pengujian pada setiap iterasi menjadi dasar penyempurnaan pada iterasi berikutnya.
2. Kombinasi Pattern Matching (Regex) dan Shannon Entropy berhasil diimplementasikan sebagai mekanisme deteksi hardcoded credential dengan peran yang berbeda dan saling melengkapi. Pattern Matching digunakan untuk mengenali credential yang memiliki format atau pola tertentu melalui delapan kategori deteksi, yaitu AWS Access Key, Google Cloud API Key, Stripe Secret Key, GitHub Personal Access Token, JSON Web Token (JWT), Private Key (PEM), Password Hardcoded, dan Slack Bot Token. Credential yang memenuhi pola yang telah ditentukan dapat dideteksi menggunakan Pattern Matching tanpa bergantung pada hasil

perhitungan Shannon Entropy. Sebaliknya, Shannon Entropy digunakan untuk menganalisis kandidat string yang tidak memiliki pola baku yang dapat dikenali oleh Regex tetapi memiliki karakteristik keacakan tinggi, sehingga kandidat tersebut tetap dapat dideteksi melalui mekanisme entropy tanpa bergantung pada hasil deteksi Regex. Kedua metode dijalankan secara terpisah pada dokumen yang sama, kemudian hasil deteksi digabungkan dan dilakukan deduplikasi sebelum ditampilkan sebagai Diagnostic Warning. Hasil Black-box Testing menunjukkan bahwa fungsi deteksi credential berpola, deteksi credential pada deklarasi TypeScript, penyaringan kandidat high-entropy, dan deteksi CG009 telah menghasilkan keluaran sesuai dengan hasil yang diharapkan.

3. Hasil pengujian akhir menunjukkan bahwa CredGuard memperoleh Precision sebesar 80,12%, Recall sebesar 86,00%, dan F1-Score sebesar 82,96% pada dataset pengujian yang terdiri atas 300 sampel. Dataset tersebut terdiri atas 150 sampel Secret dan 150 sampel Non-Secret. Berdasarkan hasil confusion matrix, diperoleh 129 True Positive, 21 False Negative, 118 True Negative, dan 32 False Positive. Hasil tersebut menunjukkan bahwa CredGuard memiliki kemampuan yang cukup baik dalam menemukan hardcoded credential pada dataset pengujian, meskipun masih terdapat kesalahan deteksi berupa False Positive dan False Negative. Hasil pengujian juga menunjukkan peningkatan setelah penyempurnaan pada iterasi ketiga, dengan Precision meningkat dari 74,48% menjadi 80,12%, Recall dari 72,00% menjadi 86,00%, dan F1-Score dari 73,20% menjadi 82,96%. Berdasarkan kriteria keberhasilan yang telah ditetapkan, hasil pengujian akhir CredGuard memperoleh Precision sebesar 80,12%, Recall sebesar 86,00%, dan F1-Score sebesar 82,96%. Ketiga nilai tersebut telah melampaui batas minimum 75%, sehingga hasil pengujian akhir memenuhi kriteria keberhasilan yang ditetapkan dalam penelitian.

5.2 Saran

Berdasarkan hasil penelitian dan keterbatasan sistem yang telah ditemukan, beberapa pengembangan yang dapat dilakukan pada penelitian selanjutnya adalah sebagai berikut.

1. Pengembangan selanjutnya dapat memperluas cakupan jenis *hardcoded credential* dan bahasa pemrograman yang didukung sehingga mekanisme deteksi dapat diterapkan pada lingkungan pengembangan yang lebih beragam.
2. Mekanisme Shannon Entropy dan Pattern Matching dapat disempurnakan lebih lanjut, khususnya pada proses *filtering* dan pengenalan konteks kode, untuk mengurangi kemungkinan terjadinya *False Positive* dan *False Negative*.
3. Penelitian selanjutnya dapat mengembangkan mekanisme analisis konteks, seperti *data flow analysis* atau pendekatan berbasis *machine learning* dan Large Language Model, untuk melengkapi keterbatasan pendekatan berbasis pola dan karakteristik string.
4. Pengujian dapat diperluas menggunakan dataset yang lebih besar dan lebih beragam, termasuk data dari repositori perangkat lunak yang telah melalui proses pelabelan dan validasi, sehingga kemampuan generalisasi sistem dapat dievaluasi dengan lebih baik.
5. Pengujian performa dan penggunaan sumber daya dapat dikembangkan secara lebih mendalam untuk mengetahui pengaruh proses pemindaian terhadap penggunaan CPU, memori, serta respons Visual Studio Code selama proses pengembangan.

DAFTAR PUSTAKA

- [1] V. Chornii, Y. Martseniuk, A. Partyka, and O. Harasymchuk, "Information security risks associated with the uncontrolled storage of secrets in source code ★," 2025.
- [2] S. K. Basak, L. Neil, B. Reaves, and L. Williams, "SecretBench: A Dataset of Software Secrets," Mar. 2023, [Online]. Available: <http://arxiv.org/abs/2303.06729>
- [3] A. Krause, J. H. Klemmer, N. Huaman, D. Wermke, Y. Acar, and S. Fahl, "Pushed by Accident: A Mixed-Methods Study on Strategies of Handling Secret Information in Source Code Repositories," Jun. 2026, [Online]. Available: <http://arxiv.org/abs/2211.06213>
- [4] M. R. Rahman, N. Imtiaz, M. A. Storey, and L. Williams, "Why secret detection tools are not enough: It's not just about false positives - An industrial case study," *Empir. Softw. Eng.*, vol. 27, no. 3, May 2022, doi: 10.1007/s10664-021-10109-y.
- [5] "Human error triggers Mercedes-Benz sensitive data leak - Polymer." Accessed: May 13, 2026. [Online]. Available: <https://www.polymerhq.io/blog/human-error-triggers-mercedes-benz-sensitive-data-leak/>
- [6] "How to Avoid Source Code Breaches: Lessons From the Mercedes-Benz GitHub Token Leak - VicOne." Accessed: Jun. 16, 2026. [Online]. Available: <https://vicone.com/blog/how-to-avoid-source-code-breaches-lessons-from-the-mercedes-benz-github-token-leak/>
- [7] A. Dawoud, S. Finster, N. Coppik, and V. Ashiwal, "Better Left Shift Security! Framework for Secure Software Development," in *Proceedings - 9th IEEE European Symposium on Security and Privacy Workshops, Euro S and PW 2024*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 642–649. doi: 10.1109/EuroSPW61312.2024.00078.
- [8] S. K. Basak, J. Cox, B. Reaves, and L. Williams, "A Comparative Study of Software Secrets Reporting by Secret Detection Tools," Jul. 2023, [Online]. Available: <http://arxiv.org/abs/2307.00714>

- [9] S. Yun, Y. Zhang, Z. Shi, L. Wang, Y. Wang, and Y. Bai, "A Large-Scale Empirical Study of Secret Keys Leakage in Hugging Face Spaces," *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE '26)*, April 12–18, 2026, Rio de Janeiro, Brazil, vol. 1, doi: 10.1145/3744916.3773117.
- [10] M. N. Rahman, S. Ahmed, Z. Wahab, S. M. Sohan, and R. Shahriyar, "Secret Breach Detection in Source Code with Large Language Models," Jul. 2025, [Online]. Available: <http://arxiv.org/abs/2504.18784>
- [11] N. J. D. La Croix, T. Ahmad, and F. Han, "Enhancing Secret Data Detection Using Convolutional Neural Networks With Fuzzy Edge Detection," *IEEE Access*, vol. 11, pp. 131001–131016, 2023, doi: 10.1109/ACCESS.2023.3334650.
- [12] M. El, A. Tebib, M. Graa, O.-E.-K. Aktouf, and P. Andre, *IDE Plugins for Secure Android Applications Development: Analysis & Classification Study*.
- [13] "CUTE Extension for VS Code," 2022.
- [14] "Shannon Entropy in Artificial Intelligence and Its Applications Based on Information Theory".
- [15] B. Wang *et al.*, "Argus: A Multi-Agent Sensitive Information Leakage Detection Framework Based on Hierarchical Reference Relationships," Dec. 2025, doi: 10.1145/3744916.3773208.
- [16] Y. Huang, Y. Li, W. Wu, J. Zhang, and M. R. Lyu, "Your Code Secret Belongs to Me: Neural Code Completion Tools Can Memorize Hard-Coded Credentials," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 2515–2537, Jul. 2024, doi: 10.1145/3660818.
- [17] R. Feng, Z. Yan, S. Peng, and Y. Zhang, "Automated Detection of Password Leakage from Public GitHub Repositories," in *Proceedings - International Conference on Software Engineering*, IEEE Computer Society, Jul. 2022, pp. 175–186. doi: 10.1145/3510003.3510150.
- [18] A. Brasoveanu, M. Moodie, and R. Agrawal, "IssueGuard: Real-Time Secret Leak Prevention Tool for GitHub Issue Reports," in *CEUR Workshop Proceedings*, CEUR-WS, 2026, pp. 1–9. doi: 10.1145/nnnnnnn.nnnnnnn.

- [19] A. Akhtar, B. Bakhtawar, and S. Akhtar, "EXTREME PROGRAMMING VS SCRUM: A COMPARISON OF AGILE MODELS," *International Journal of Technology, Innovation and Management (IJTIM)*, vol. 2, p. 2022, doi: 10.54489/ijtim.v2i1.77.
- [20] A. C. Djamen, A. Mewengkang, J. P. A. Runtuwene, and P. T. D. Rompas, "Design of Employee Archive Application using Extreme Programming Model at PT. PLN Persero Wilayah Suluttenggo," *Scitepress*, Apr. 2023, pp. 321–325. doi: 10.5220/0009010603210325.
- [21] E. Susilawati and D. F. Fauzi, "http://infor.seaninstitute.org/index.php/infokum/index INFOKUM is licensed under a Creative Commons Attribution-Non Commercial 4.0 International License (CC BY-NC 4.0) MOBILE WEB-BASED VILLAGE SERVICE APPLICATION DESIGN WITH EXTREME PROGRAMMING METHODOLOGY," *JURNAL INFOKUM*, vol. 10, no. 1, 2021, [Online]. Available: <http://infor.seaninstitute.org/index.php/infokum/index>
- [22] I. Widiastuti and A. Cakranegara, "International Journal of Multidisciplinary Research and Literature DEVELOPMENT OF THE PUBLIC SERVICE MANAGEMENT INFORMATION SYSTEM THROUGH EXTREME PROGRAMMING IN SUKARAME DISTRICT, BANDAR LAMPUNG," *International Journal of Multidisciplinary Research and Literature IJOMRAL*, vol. 1, no. 3, pp. 241–360, 2022, doi: 10.53067/ijomral.v1i3.