Jutek : Journal of Usability and Technology Engineering Knowledge

journal homepage: https://ejournal.undar.or.id/index.php/Jutek

Artikel Tinjauan

Analisis Komparasi Validasi Magic Bytes dan MIME Type dalam Mitigasi Serangan Polyglot pada REST API

Rizky Parlika [a, *], Ayala Septama Rahanda [b]

[a, b] *Fakultas Ilmu Komputer, Universitas Pembangunan Nasional "Veteran", Surabaya, Indonesia*
email: [a] rizkyparlika.if@upnjatim.ac.id, [b] 23081010071@student.upnjatim.ac.id

INFO ARTICLE

Kata Kunci:
REST API,
file upload,
MIME Type,
Magic Bytes,
file polyglot

ABSTRAK

Sebagian besar waktu, validasi unggahan file pada layanan REST API masih menggunakan pemeriksaan MIME Type di header HTTP. Metode ini mudah digunakan dan tidak memakan banyak ruang, tetapi rentan terhadap modifikasi header dan tidak cukup efektif untuk melindungi dari serangan file baru, terutama file polyglot yang secara teknis merupakan file gambar yang sah tetapi juga mengandung payload berbahaya. Studi ini menetapkan lingkungan pengujian terkontrol yang terdiri dari backend REST API berbasis Node.js, pembangkitan dataset otomatis, dan bot serangan berbasis Python untuk mengevaluasi efektivitas validasi MIME Type dan Magic Bytes dalam mengurangi serangan unggahan file. Dataset uji coba terdiri dari 200 sampel, termasuk file gambar biasa, malware umum, malware dengan ekstensi yang diubah, dan file polyglot. Hasil eksperimen menunjukkan bahwa MIME Type masih dapat membiarkan file normal lolos dan memblokir malware umum dengan header non-gambar. Namun, ia tidak dapat menghentikan malware yang diubah namanya atau file polyglot yang tampak seperti gambar. Magic Bytes selalu memblokir malware umum dan malware yang diubah namanya tanpa memengaruhi file sah. Namun, ia masih tidak dapat mendeteksi file polyglot karena hanya memeriksa header biner di awal file. Uji latensi menunjukkan bahwa penggunaan Magic Bytes hanya menambah waktu pemrosesan yang sangat kecil, dalam rentang mikrodetik. Ketika memeriksa salah satu sampel polyglot dalam format heksadesimal, terlihat bahwa file tersebut mengandung header JPEG asli dan payload skrip PHP di tengah file. Hasil ini menunjukkan bahwa Magic Bytes dapat membantu melindungi dari kelemahan MIME, tetapi harus digunakan bersama alat lain untuk melindungi dari serangan file polyglot.

* Corresponding author.
E-mail address: rizkyparlika.if@upnjatim.ac.id (Rizky Parlika).

https://doi.org/xx.xxxx/Jutek.2026.xx.xxx
Received 26 January 2026; Received in revised form 30 January 2026; Accepted 10 February 2026

1 PENDAHULUAN

Peningkatan arsitektur layanan berbasis REST API telah menghasilkan aplikasi modern yang memungkinkan pengguna mengunggah file. Fitur ini umumnya ditemukan pada sistem manajemen dokumen, layanan berbagi media, dan aplikasi bisnis yang memerlukan pertukaran data antar platform (Al Hilmi and Yunan 2022; Oz et al. 2025). Sistem unggah file ini terlihat sederhana, namun merupakan salah satu titik rentan bagi file berbahaya untuk masuk. Sulit untuk memeriksa file yang dikirim pengguna tanpa proses inspeksi khusus (Hasibuan and Gultom 2018). Oleh karena itu, keamanan proses unggah file sangat penting dalam pengembangan layanan digital, terutama dalam arsitektur microservice terdistribusi (de Aguiar Monteiro et al. 2017).

Pada kenyataannya, sebagian besar pengembang masih memeriksa tipe konten (Content-Type) yang disediakan oleh header HTTP untuk menentukan apakah file tersebut dapat diterima atau tidak. Metode ini dipilih karena mudah diimplementasikan dan tidak membebani proses server-side (Riadi and Aristianto 2016). Namun, mengandalkan informasi dari klien merupakan masalah besar karena header ini dapat dengan mudah diubah menggunakan berbagai alat. Hal ini berarti server dapat menerima file berbahaya yang tampak seperti format tertentu tetapi sebenarnya mengandung malware dalam bentuk instruksi eksekusi atau payload (Pooj and Patil 2016). Penelitian sebelumnya telah menunjukkan metodologi ini; namun, penelitian terbaru menunjukkan bahwa teknik spoofing header telah cukup canggih untuk melewati firewall modern (Akhavani et al. 2025; Ginting, Sihombing, and Harahap 2012). Selain itu, jika pengaturan keamanan tidak tepat, browser dapat melakukan MIME sniffing, yang menjalankan file gambar sebagai skrip berbahaya (Kishnani and Das 2025).

Metode validasi Magic Bytes semakin banyak digunakan sebagai cara untuk mengevaluasi keaslian format file daripada tes berbasis header. Teknik ini memeriksa beberapa byte pertama file dan membandingkannya dengan pola yang menunjukkan format (Rizal, Ruuhwan, and Chandra 2020). Pendekatan ini dianggap lebih andal daripada tes MIME Type karena pemeriksaan didasarkan pada nilai biner. Metode ini digunakan oleh banyak sistem keamanan modern, seperti antivirus dan modul inspeksi konten, untuk mengidentifikasi jenis file dengan benar tanpa menggunakan ekstensi atau header (Ardiansyah, Hardi, and Gata 2020; Boiko, Moskalenko, and Shovkoplias 2023).

Metode berbasis Magic Bytes tidak sepenuhnya aman, meskipun lebih baik daripada validasi MIME. Metode serangan polyglot baru telah terungkap yang memengaruhi cara penyerang menyisipkan kode berbahaya ke dalam sistem (Koch et al. 2022). File polyglot dirancang agar terlihat seperti foto atau file media yang valid, tetapi sebenarnya mengandung skrip berbahaya atau kode instruksi di beberapa bagian (Cohen, Nissim, and Elovici 2020). Penyerang dapat membuat file yang lolos uji Magic Bytes dengan menempatkan byte penanda format di awal file dan menambahkan payload setelahnya. Filefile ini masih dapat dijalankan oleh beberapa server yang tidak memiliki langkah keamanan tambahan. Ini disebut stegomalware (Chaganti et al. 2021).

Penelitian sebelumnya telah mengkaji kelemahan MIME Type, efektivitas Magic Bytes, dan beberapa strategi penghindaran dalam prosedur unggah file (Koch et al. 2025; Pooj and Patil 2016). Namun, sebagian besar penelitian ini hanya fokus pada komponen deteksi tanpa mengaitkannya dengan arsitektur REST API yang umum digunakan dalam aplikasi modern (Deng et al. 2023; Du et al. 2024). Selain itu, belum ada pengujian sistematis terhadap serangan polyglot, terutama yang menggunakan simulasi serangan otomatis dan lingkungan eksperimental terisolasi yang dapat meniru kondisi nyata di backend (Koch et al. 2022; Oz et al. 2024).

Masih ada perbedaan pendapat mengenai sejauh mana validasi Magic Bytes memengaruhi waktu pemrosesan dalam hal kinerja. Beberapa studi mengacu pada penundaan yang meningkat; namun, belum ada pengukuran standar yang dilakukan untuk memberikan analisis perbandingan antara metode berbasis MIME dan berbasis Magic Bytes (Atlidakis, Godefroid, and Polishchuk 2019).

Karena sedikit penelitian yang meneliti baik keamanan maupun kinerja, pengembang biasanya membuat keputusan berdasarkan apa yang mereka pikirkan daripada apa yang mereka ketahui.

Situasi ini menunjukkan adanya kesenjangan dalam penelitian yang perlu diisi. Saat ini, tidak ada analisis mendalam yang mengevaluasi efektivitas kedua teknik validasi dalam menangani serangan polyglot pada REST API, sambil menilai implikasi kinerja masing-masing metode. Selain itu, sedikit penelitian yang menggabungkan uji keamanan dengan pendekatan forensik untuk menganalisis struktur internal file polyglot yang berhasil melewati mekanisme pertahanan.

Penelitian ini mengajukan beberapa pertanyaan fundamental yang berasal dari masalah-masalah tersebut: Seberapa baik MIME Type dapat menangani berbagai jenis serangan berbasis file, seperti file yang diubah namanya dan file polyglot? Seberapa besar Magic Bytes dapat memperbaiki masalah-masalah ini? Dan seberapa besar Magic Bytes memperlambat kinerja REST API? Pertanyaan-pertanyaan ini menjadi dasar untuk metodologi eksperimental dan analisis dalam studi ini.

Tujuan studi ini adalah untuk mengevaluasi secara objektif efektivitas MIME Type dan Magic Bytes dalam mengatasi serangan polyglot pada REST API, serta mengevaluasi dampak kinerjanya melalui pencatatan latensi dengan resolusi milidetik. Penelitian ini juga menawarkan analisis forensik terhadap arsitektur internal file polyglot untuk menunjukkan bagaimana serangan-serangan ini dapat mengelabui mekanisme validasi saat ini. Temuan studi ini diharapkan memberikan pemahaman yang lebih komprehensif tentang kelebihan dan kekurangan dua prosedur validasi yang digunakan hingga saat ini.

Penelitian ini secara umum memberikan kontribusi yang beragam. Artikel ini memperkenalkan desain testbed yang sederhana untuk mengevaluasi metode validasi unggahan berkas pada REST API, yang mencakup backend Node.js, generator dataset uji, dan skrip serangan otomatis. Kedua, penelitian ini melakukan analisis perbandingan antara validasi berbasis MIME Type dan Magic Bytes dengan menganalisis pola deteksi (berkas yang diblokir dan yang lolos) pada berbagai jenis berkas, seperti berkas standar, malware yang diganti namanya, dan berkas polyglot. Ketiga, penelitian ini menilai dampak penggunaan Magic Bytes terhadap kinerja layanan dengan mendokumentasikan latensi pemrosesan sisi server. Keempat, makalah ini menyajikan studi kasus forensik sederhana menggunakan potongan heksadesimal dan representasi ASCII dari sampel file polyglot yang berhasil melewati validasi, untuk menunjukkan batasan metodologi berbasis Magic Bytes secara konkret.

2 METODOLOGI PENILITIAN

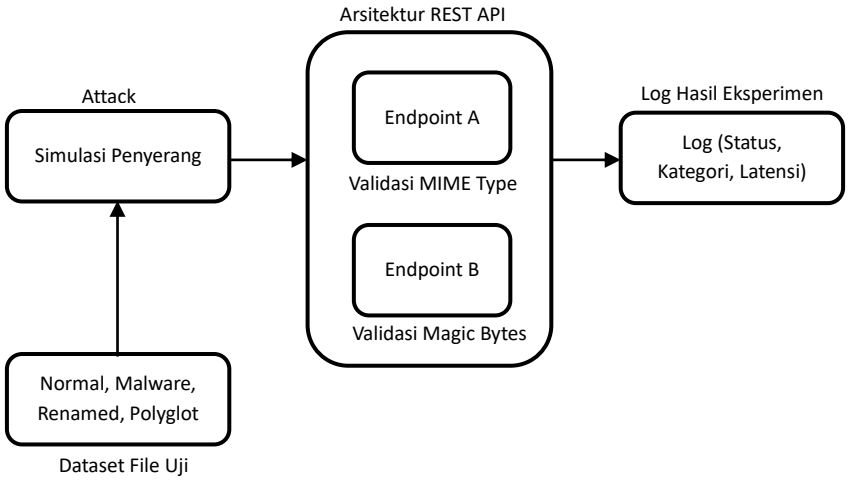
2.1 Jenis dan Pendekatan Penelitian

Penelitian ini diklasifikasikan sebagai penelitian kuantitatif yang menggunakan metodologi eksperimental. Tujuan utama penelitian ini adalah menganalisis mekanisme validasi unggahan file pada layanan REST API dengan membandingkan dua metodologi yang umum digunakan: validasi berbasis MIME Type dan validasi berbasis Magic Bytes. Kedua pendekatan tersebut diuji dalam lingkungan terkontrol dengan skenario serangan terorganisir untuk mengevaluasi perbedaan dalam efektivitas deteksi dan pengaruhnya terhadap kinerja sistem (Alazmi and De Leon 2022; Huang et al. 2019).

Tujuan studi ini adalah untuk menciptakan lingkungan uji yang secara akurat mensimulasikan proses pengunggahan file ke server REST API. Variabel independen dalam penelitian ini adalah proses validasi yang digunakan di ujung server (MIME Type dan Magic Bytes). Variabel dependen terdiri dari pengukuran keamanan, termasuk keandalan deteksi, dan metrik kinerja yang diwakili oleh latensi pemrosesan sisi server. Secara umum, eksperimen dimulai dengan menyiapkan lingkungan uji, mengumpulkan dataset file uji, menjalankan serangan otomatis dengan bot serangan, dan kemudian mengumpulkan serta menganalisis data log yang dihasilkan dari serangan tersebut.

2.2 Rancangan Lingkungan Uji

Lingkungan pengujian untuk penyelidikan ini adalah laboratorium tertutup yang terdiri dari berbagai komponen yang bekerja sama. Komponen utama meliputi server REST API berbasis Node.js, skrip bot serangan berbasis Python yang berfungsi sebagai klien penyerang, kumpulan berkas uji yang disimpan dalam direktori lokal, dan modul perekam log yang mencatat hasil setiap permintaan unggahan berkas. Bot serangan mengambil berkas dari dataset, mengirimkannya ke endpoint REST API dengan skenario header tertentu, dan server merespons dengan memeriksa berkas-berkas tersebut dan mencatat temuan. Penggunaan bot otomatis seperti ini merupakan metode uji keamanan yang efektif untuk menemukan kerentanan (Oz et al. 2024; Viglianisi et al. 2020).



Gambar 1. *Arsitektur testbed REST API pada penelitian*

Gambar 1 menunjukkan tata letak umum arsitektur lingkungan pengujian. Arsitektur *Testbed* REST API. Bot serangan dalam gambar ditampilkan sebagai klien yang terhubung ke dua *endpoint* berbeda pada *server* REST API: satu untuk memeriksa *MIME Types* dan satu untuk memeriksa *Magic Bytes*. Modul pencatatan menghubungkan kedua *endpoint* dan mencatat informasi kategori file, hasil validasi (diizinkan atau diblokir), serta latensi pemrosesan. Bot serangan menggunakan dataset file uji sebagai sumber data sebelum mengirimkannya ke *server*.

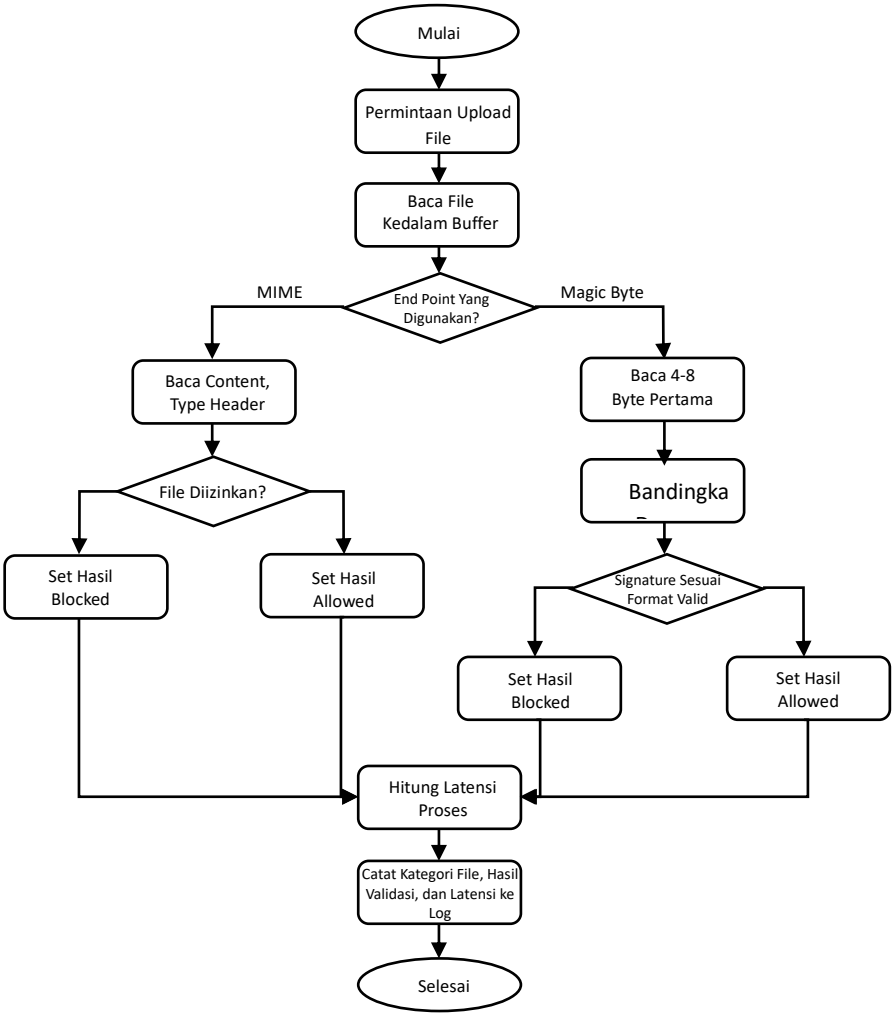
Uji coba dilakukan pada laptop dengan sistem operasi Windows 64-bit, 16 GB RAM DDR5, dan prosesor dengan 6 core dan 12 thread. Server REST API menggunakan Node.js versi 22.20.0 dengan framework Express dan modul manajemen unggahan file yang menyimpan file di memori daripada di disk. Hal ini berarti file hanya diproses di RAM. Skrip bot serangan dijalankan di sisi klien dengan Python versi 3.13.9, dan menggunakan perpustakaan HTTP untuk melakukan permintaan ke server. Semua bagian eksperimen dilakukan di workstation yang sama sehingga dapat diulang dengan mudah. Nilai latensi yang diukur sebagian besar mencerminkan biaya pemrosesan validasi, bukan penundaan dalam mengakses jaringan atau disk.

2.3 Desain REST API dan Mekanisme Validasi

Server REST API yang digunakan dalam studi ini memiliki dua endpoint utama yang digunakan untuk mengunggah file. Endpoint pertama menggunakan metode validasi berdasarkan MIME Type, sedangkan endpoint kedua menggunakan metode validasi berdasarkan Magic Bytes. Kedua endpoint tersebut menggunakan sistem unggah file yang menyimpan file secara sementara di memori. Dipilih metode ini untuk mengurangi dampak operasi input/output disk terhadap waktu pemrosesan. Dengan cara ini, perbedaan latensi yang tercatat lebih baik menunjukkan perbedaan kompleksitas validasi.

Ketika permintaan unggahan masuk, server hanya memverifikasi nilai Content-Type dalam header HTTP di endpoint dengan validasi MIME Type. File dianggap valid dan lolos tahap validasi jika nilai Content-Type sesuai dengan salah satu MIME Type gambar yang diizinkan, seperti image/jpeg, image/png, atau image/gif. File tidak diterima jika nilai Content-Type tidak terdaftar dalam daftar nilai yang diizinkan. Pendekatan ini mirip dengan cara kebanyakan pengembang menggunakan informasi header untuk menentukan jenis file.

Endpoint yang menggunakan validasi Magic Bytes, di sisi lain, memverifikasi isi file secara langsung. Server memeriksa beberapa byte pertama dari buffer file saat file diunggah dan membandingkannya dengan pola penanda format standar, seperti untuk file JPEG, PNG, atau GIF. Jika urutan byte pertama sesuai dengan salah satu pola yang diizinkan, file tersebut valid. Jika tidak, file tersebut tidak valid. Dengan cara ini, server tidak hanya memeriksa header; ia juga memeriksa struktur biner aktual file (Anwar, Fadlil, and Riadi 2020).



Gambar 2. *Flowchart mekanisme validasi file upload pada endpoint MIME Type dan Magic Bytes*

Kedua endpoint mencatat log informasi saat memproses setiap permintaan. Server mencatat identitas file atau nama sampel, kategori file berdasarkan label kebenaran (normal, malware, malware dengan ekstensi yang dimodifikasi, atau polyglot), hasil validasi (diizinkan atau diblokir), waktu mulai pemrosesan, dan waktu selesai pemrosesan untuk setiap file yang diterimanya. Perbedaan antara waktu mulai dan waktu selesai, diukur dalam milidetik, digunakan untuk menghitung latensi pemrosesan. Gambar 2 menunjukkan bagaimana proses validasi bekerja di kedua ujung secara konseptual. Diagram alur ini menggambarkan langkah-langkah dalam Mekanisme

Validasi MIME Type dan Magic Bytes, termasuk menerima permintaan, memeriksa header atau isi file, membuat keputusan, dan mencatat hasilnya.

2.4 Dataset dan Skenario Serangan

Ada 200 file dalam dataset uji untuk studi ini. File-file tersebut dibagi menjadi beberapa kelompok untuk menunjukkan kombinasi berbeda antara situasi normal dan serangan. Kelompok pertama adalah file biasa, yaitu gambar dalam format JPEG, PNG, atau GIF yang diambil dari sumber publik. Jenis kedua adalah file malware biasa, yaitu file yang mengandung skrip berbahaya, termasuk webshell atau instruksi untuk menjalankan perintah (Pan et al. 2021). Jenis ketiga adalah malware yang diubah namanya, yaitu file berbahaya yang ekstensi aslinya telah diubah menjadi ekstensi gambar. Jenis keempat adalah file polyglot, yaitu file hibrida yang memiliki struktur header gambar yang sah tetapi juga mengandung payload berbahaya di bagian lain file (Chandra and Aldiansyah 2025). Tabel 1 menunjukkan ringkasan komposisi dataset.

Tabel 1. *Ringkasan Dataset Uji*

Kategori File	Jumlah	Deskripsi
Normal	100	File gambar valid (JPG, PNG, GIF) yang tidak mengandung payload berbahaya
Malware	50	File berisi skrip berbahaya (seperti webshell atau script eksekusi perintah)
Renamed Malware	30	File malware yang ekstensi aslinya diubah menjadi ekstensi gambar (.jpg)
Polyglot	20	File hibrida dengan header gambar valid dan payload berbahaya yang disisipkan di

Data set tersebut disusun sebagian secara manual dan sebagian menggunakan skrip Python. Hal ini dilakukan untuk mengurangi bias manusia dan memudahkan pengulangan proses. Penelitian ini mendapatkan file normal dari kumpulan foto yang tersedia secara gratis dan memilihnya untuk memastikan formatnya kompatibel dengan jenis file yang diizinkan oleh server. Skrip singkat yang menunjukkan cara umum untuk mengeksploitasi sistem, termasuk skrip yang menjalankan perintah yang disediakan melalui parameter, digunakan untuk membuat file malware. Isi berkas malware yang diganti namanya tidak diubah, tetapi ekstensi berkas diubah menjadi ekstensi gambar untuk membuatnya terlihat seperti upaya kamuflase dasar. Pada saat yang sama, berkas polyglot dibuat dengan menjaga bagian awal berkas sebagai header gambar yang sah, lalu menambahkan payload berbahaya pada offset yang telah ditentukan. Dengan cara ini, berkas masih dapat dikenali sebagai gambar, tetapi memiliki bagian yang mungkin digunakan dalam situasi tertentu (Birnbaum, Kraetzer, and Dittmann 2023).

Sebuah perangkat lunak Python bernama attack bot melakukan serangan terhadap layanan REST API. Skrip ini memeriksa semua file dalam dataset satu per satu dan mengirimkan masing masing file ke kedua tujuan secara otomatis. Bot serangan mengirim permintaan unggah ke kedua endpoint validasi MIME dan endpoint validasi Magic Bytes untuk setiap sampel. Pola pengiriman dapat diatur secara berurutan atau acak, namun untuk penyelidikan ini, pola berurutan digunakan agar lebih mudah untuk memberi label dan menganalisis data.

Perubahan header dilakukan pada kategori file terinfeksi agar eksperimen dapat menunjukkan situasi serangan yang sebenarnya. Bot serangan mengatur nilai header Content-Type menjadi tipe gambar yang diizinkan, seperti image/jpeg, untuk file malware yang diganti namanya dan file polyglot, meskipun konten file sebenarnya berbeda. Dengan cara ini, endpoint yang hanya memeriksa MIME Types akan mendapatkan informasi yang salah, tetapi endpoint Magic Bytes akan terus memeriksa isi file. Metode spoofing header ini berguna untuk memeriksa seberapa baik setiap

metode validasi dapat menangani serangan yang memanfaatkan manipulasi protokol tingkat aplikasi (Deng et al. 2023; Sarpong et al. 2021).

2.5 Pengolahan dan Analisis Data

Setelah semua skenario serangan dijalankan, log server REST API digunakan untuk mengumpulkan data. Setiap baris dalam log merupakan permintaan untuk mengunggah file dan berisi informasi tentang label kebenaran file (normal, malware, malware yang diubah namanya, atau polyglot), hasil keputusan validasi di endpoint yang diuji (diizinkan atau diblokir), dan nilai penundaan pemrosesan. Kemudian, data dari kedua endpoint digabungkan menjadi satu dataset dan diproses dengan skrip analisis untuk membuat ringkasan jumlah file yang diizinkan dan dilarang dalam setiap kategori dan oleh setiap metode validasi.

Untuk melakukan analisis keamanan, hitunglah berapa banyak file yang diizinkan dan dilarang dalam setiap kategori file untuk setiap endpoint. Penelitian ini menunjukkan bagaimana sistem berperilaku seiring waktu, termasuk apakah file normal selalu diterima, apakah malware konvensional atau malware yang diganti namanya berhasil dicegah, dan berapa banyak file polyglot yang lolos mekanisme validasi. Tujuan utama analisis ini adalah untuk menentukan kapan sistem berfungsi sebagaimana mestinya (misalnya, ketika memblokir file berbahaya) dan kapan tidak berfungsi sebagaimana mestinya (misalnya, ketika membiarkan file berbahaya masuk) (OWASP Foundation 2023a, 2023b).

Untuk melakukan analisis kinerja, perlu diketahui berapa lama waktu yang dibutuhkan oleh setiap mekanisme validasi untuk memproses. Dilakukan pengambil nilai latensi untuk setiap permintaan dan menggabungkannya untuk mendapatkan nilai ringkasan, seperti latensi rata-rata di endpoint berbasis Magic Bytes dan endpoint berbasis MIME Type. Penelitian ini menggunakan statistik ini untuk melihat seberapa banyak waktu pemrosesan tambahan yang ditambahkan oleh inspeksi biner Magic Bytes dibandingkan dengan validasi berbasis MIME Type, yang hanya menggunakan header. Juga untuk melihat apakah perbedaan tersebut signifikan dalam praktik nyata.

Pekerjaan ini mencakup analisis kuantitatif dan pemeriksaan forensik dasar terhadap sampel file polyglot yang berhasil melewati validasi Magic Bytes (Belkind, Dubin, and Dvir 2023; Birnbaum et al. 2023). Pada tahap ini, isi file ditampilkan dalam format heksadesimal dan ASCII. Hal ini memudahkan untuk melihat bagian header yang valid, penempatan payload berbahaya, dan struktur keseluruhan file. Analisis ini tidak dimaksudkan sebagai cara baru untuk menemukan hal-hal, melainkan contoh kualitatif yang menunjukkan bagaimana file polyglot dapat menipu sistem validasi berbasis Magic Bytes dan menunjukkan bahwa metode ini tidak bekerja dengan baik terhadap serangan file hibrida.

3 HASIL DAN PEMBAHASAN

3.1 Gambaran Umum Hasil Eksperimen

Data set uji coba terdiri dari 200 file: 100 file normal, 50 file malware, 30 file malware yang diubah namanya, dan 20 file polyglot. Semua file ini digunakan dalam eksperimen. Bot serangan mengirimkan setiap berkas ke dua endpoint secara berurutan: yang memeriksa MIME Type dan yang memeriksa Magic Bytes. Jadi, untuk setiap sampel, dua respons diperoleh, masing-masing dari dua mekanisme validasi. Server menangani semua permintaan unggahan berkas tanpa kesalahan HTTP atau kegagalan eksekusi yang dapat mengganggu eksperimen.

Server mencatat label kategori file (normal, malware, malware yang diubah namanya, atau polyglot), hasil keputusan validasi (diizinkan atau diblokir), dan angka latensi pemrosesan untuk setiap permintaan. Kemudian, data dari kedua endpoint digabungkan dan digunakan untuk membuat ringkasan jumlah file yang diizinkan dan dilarang untuk setiap kategori file dan teknik validasi, serta ringkasan statistik latensi untuk setiap endpoint.

3.2 Hasil Validasi pada Endpoint MIME Type

Endpoint pertama menggunakan sistem validasi berbasis MIME Type yang hanya memeriksa nilai Content-Type dalam header HTTP. Dalam pengujian ini, file biasa dikirim dengan header Content-Type yang sesuai dengan format aslinya (seperti image/jpeg atau image/png), sementara file malware dikirim dengan header yang sesuai dengan jenis aslinya, seperti script atau text. Bot serangan mengubah nilai Content-Type menjadi tipe gambar yang diizinkan, seperti image/jpeg, untuk file malware yang diubah namanya dan file polyglot, terlepas dari isi file sebenarnya.

Hasil pengujian menunjukkan bahwa endpoint MIME Type dapat menangani semua file normal, sesuai dengan yang dimaksudkan. File yang dikirim dengan header non-gambar yang umumnya digunakan oleh malware dilarang karena tidak sesuai dengan daftar MIME Type yang diizinkan. Namun, sistem ini tidak berfungsi ketika penyerang menyembunyikan file berbahaya dengan mengubah nama file dan mengubah header Content-Type menjadi tipe gambar. Sistem (Ginting et al. 2012; Pooj and Patil 2016) menyatakan bahwa semua sampel malware yang diubah namanya dan file polyglot dianggap aman. Kegagalan ini menunjukkan bahwa validasi berbasis atribut eksternal sangat mudah diretas, yang menciptakan celah keamanan serius pada server (Akhavani et al. 2025).

3.3 Hasil Validasi pada Endpoint Magic Bytes

Endpoint kedua menggunakan sistem validasi berbasis Magic Bytes dengan membaca beberapa byte pertama file dan membandingkannya dengan signature format gambar yang didukung. Keputusan endpoint ini didasarkan pada struktur biner file (Ardiansyah et al. 2020; Sethi 2024), bukan nilai Content-Type di header seperti endpoint MIME Type.

Endpoint Magic Bytes menerima dan mengizinkan semua file normal yang sebenarnya merupakan gambar selama pengujian. File malware umum yang tidak memiliki struktur header gambar yang sesuai selalu diblokir. File malware yang diubah namanya melakukan hal yang sama. Meskipun ekstensi dan header HTTPnya terlihat seperti gambar, signature internal file tidak cocok dengan format gambar yang diizinkan, sehingga semua sampel dalam kelompok ini berhasil diblokir. Di sisi lain, endpoint Magic Bytes membiarkan file polyglot dengan header gambar yang valid lolos sebagai file normal.

3.4 Perbandingan Hasil Validasi MIME Type dan Magic Bytes

Tabel 2 menampilkan ringkasan kuantitatif hasil validasi untuk kedua endpoint pada setiap jenis file. Tabel ini menunjukkan berapa banyak file yang dapat dilewatkan dan diblokir oleh masing-masing teknik validasi.

Tabel 2. *Hasil validasi per kategori file dan mekanisme validasi*

File	*Endpoint*	*Jumlah*	*Allowed*	*Blocked*
Normal	MIME Type	100	100	0
Normal	Magic Bytes	100	100	0
Malware	MIME Type	50	0	50
Malware	Magic Bytes	50	0	50
Malware Renamed	MIME Type	30	30	0
Malware Renamed	Magic Bytes	30	0	30
Polyglot	MIME Type	20	20	0
Polyglot	Magic Bytes	20	20	0

Tabel 2 menunjukkan bahwa kedua sistem validasi berperilaku sama terhadap file normal dan file malware biasa: mereka membiarkan file normal lolos dan mencegah malware biasa. Ada perbedaan besar antara kelompok file malware yang diganti namanya dan kelompok file polyglot.

Dalam kategori malware yang diganti namanya, endpoint MIME Type membiarkan semua sampel lolos, tetapi endpoint Magic Bytes menghentikan semuanya. Baik MIME Type maupun Magic Bytes membiarkan semua sampel polyglot lolos karena mereka tidak menemukan risiko apa pun dalam kategori polyglot.

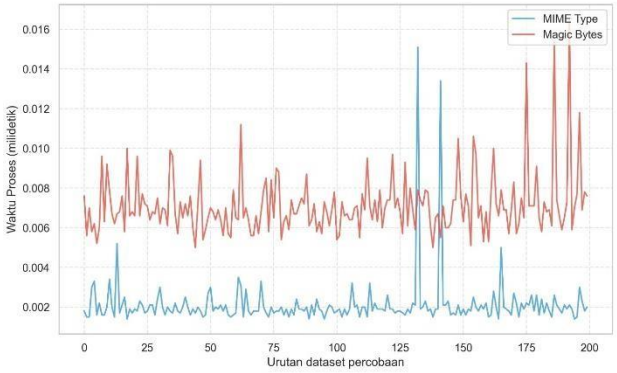
3.5 Hasil Pengukuran Latensi

Studi ini menganalisis waktu pemrosesan di kedua ujung proses serta hasil validasi. Perbedaan antara waktu mulai dan selesai pemrosesan permintaan unggahan file di sisi server digunakan untuk menghitung latensi. Kemudian, nilai latensi untuk setiap permintaan di rata-rata untuk mendapatkan nilai rata-rata untuk setiap teknik validasi. Tabel 3 menampilkan ringkasan hasil pengukuran.

Tabel 3. *Rata-rata latensi pemrosesan masing-masing endpoint*

Endpoint	***Rata-rata Latensi (ms)***
MIME Type	0.0021 ms
Magic Bytes	0.0071 ms

Secara umum, endpoint Magic Bytes memiliki latensi yang sedikit lebih tinggi dibandingkan dengan endpoint MIME Type. Hal ini konsisten dengan penerapan prosedur pembacaan dan pencocokan signature biner dalam Magic Bytes (Birnbaum et al. 2023; Cohen et al. 2020). Gambar 3 menampilkan ilustrasi perbandingan ini. Gambar tersebut menunjukkan bahwa perbedaan latensi antara kedua metode tersebut jauh di bawah 1 ms dan relatif kecil dibandingkan dengan latensi bagian lain dari sistem.



Gambar 3. *Grafik perbandingan rata-rata latensi pemrosesan pada endpoint MIME Type dan Magic Bytes.*

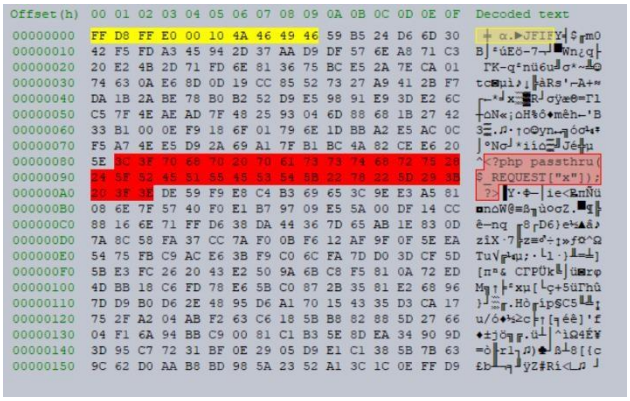
Gambar 3 menunjukkan bahwa latensi rata-rata di endpoint Magic Bytes hanya sedikit lebih tinggi daripada di endpoint MIME Type, sekitar 0,005 ms. Nilai ini berada dalam rentang mikrodetik, sehingga tidak menambah banyak pada waktu respons permintaan HTTP secara keseluruhan dibandingkan dengan komponen latensi lainnya seperti jaringan dan pemrosesan aplikasi. Dengan kata lain, pemeriksaan Magic Bytes tidak menambah banyak biaya. Penemuan ini menjadi dasar untuk penyelidikan selanjutnya mengenai trade-off antara keamanan dan kinerja.

3.6 Hasil Analisis Forensik File Polyglot

Untuk mendapatkan pemahaman lebih lanjut tentang faktorfaktor yang berkontribusi pada kegagalan identifikasi dalam berkas polyglot, dilakukan pemeriksaan forensik dasar pada salah satu sampel polyglot yang diizinkan oleh endpoint Magic Bytes. Isi berkas ditampilkan dalam format heksadesimal dan ASCII. Awal berkas mengandung serangkaian byte yang sesuai dengan signature gambar, seperti “FF D8 FF” untuk JPEG. Hal ini berarti berkas tersebut secara struktural sah sebagai berkas gambar.

Hexdump menunjukkan serangkaian karakter yang, ketika diperiksa dalam format ASCII, membentuk skrip eksekusi, seperti pola atau bentuk payload serupa. Hal ini terjadi pada titik tertentu

setelah bagian header. Masih terdapat byte penutup yang valid untuk format gambar yang digunakan pada akhir berkas. Gambar 4 menunjukkan struktur ini, dengan bagian header, lokasi payload berbahaya, dan bagian footer berkas yang ditandai.



Gambar 4. *Nilai heksadesimal dan representasi ASCII sampel file polyglot.*

Gambar 4 menunjukkan hexdump dari salah satu file sampel polyglot yang digunakan dalam eksperimen. Bagian yang diarsir kuning pada offset 00000000 menampilkan urutan byte “FF D8 FF E0 00 10 4A 46 49 46” dan teks JFIF di kolom teks terdekodasi. Ini adalah penanda header JPEG yang valid. Blok merah, di sisi lain, menunjukkan urutan byte yang membentuk skrip PHP “” di kolom teks terdekodasi. Rentang offset-nya berkisar dari sekitar 00000080 hingga 000000A0. Struktur ini menunjukkan bahwa metode Magic Bytes masih menganggap file tersebut sebagai gambar karena header-nya valid untuk file JPEG. Namun, bagian tengah file mengandung payload webshell yang dapat dieksekusi saat file diolah dengan cara yang tidak aman (Du et al. 2024; Wu et al. 2019).

Hasil penelitian menunjukkan bahwa validasi unggahan file berbasis MIME Type memiliki efektivitas yang sangat terbatas. Teknik ini hanya mampu memblokir file berbahaya yang secara “jujur” menyatakan jenis filenya melalui header HTTP, tetapi gagal mendeteksi malware ketika penyerang mengubah ekstensi file atau memalsukan header Content-Type. Seluruh sampel malware yang telah diubah namanya maupun file polyglot berhasil melewati validasi karena pemeriksaan hanya bergantung pada informasi dari sisi klien, tanpa analisis terhadap isi file. Temuan ini menegaskan bahwa MIME Type tidak layak digunakan sebagai satu-satunya mekanisme keamanan pada layanan unggah file REST API (Akhavani et al. 2025; Ginting et al. 2012; Kishnani and Das 2025; Pooj and Patil 2016).

Penerapan validasi Magic Bytes terbukti memberikan peningkatan keamanan yang signifikan dibandingkan MIME Type. Dengan memverifikasi signature biner file secara langsung, teknik ini berhasil mendeteksi dan memblokir malware tipikal serta malware yang telah dimodifikasi namanya, tanpa mengganggu file normal. Namun demikian, Magic Bytes masih memiliki keterbatasan dalam menghadapi serangan file polyglot. File jenis ini dirancang dengan signature awal yang valid sebagai file gambar, sementara payload berbahaya disisipkan pada bagian lain file, sehingga pemeriksaan yang hanya berfokus pada beberapa byte awal tidak mampu membedakan file berbahaya dari file normal. Akibatnya, seluruh sampel polyglot dalam pengujian tetap lolos dari proses (Belkind et al. 2023; Boiko et al. 2023; Cohen et al. 2020; Rizal et al. 2020).

Analisis lebih lanjut terhadap file polyglot menunjukkan bahwa kelemahan utama pendekatan berbasis signature terletak pada asumsi bahwa struktur file bersifat statis. Dalam praktiknya, penyerang dapat memanfaatkan fleksibilitas struktur file dengan menempatkan header format yang sah di awal file dan menyembunyikan kode eksekusi di bagian lain. Karena perbedaan antara file aman dan berbahaya tidak lagi berada pada header, metode Magic Bytes yang hanya memeriksa signature awal akan selalu melewatkan serangan jenis ini. Oleh karena itu, deteksi file polyglot memerlukan analisis konten yang lebih mendalam serta penerapan lapisan keamanan tambahan (Chaganti et al. 2021; Koch et al. 2022; Müller et al. 2021; Sethi 2024).

Dari sisi performa, penggunaan Magic Bytes memang menghasilkan latensi pemrosesan yang sedikit lebih tinggi dibandingkan validasi MIME Type. Namun, selisih waktu yang dihasilkan kurang dari 1 ms, sehingga tidak memberikan dampak yang signifikan terhadap kinerja aplikasi web secara keseluruhan. Temuan ini menunjukkan bahwa pemeriksaan konten file sederhana seperti Magic Bytes masih layak diterapkan pada lingkungan produksi, karena peningkatan keamanan yang diperoleh jauh lebih besar dibandingkan penurunan performa yang terjadi.

Secara praktis, hasil penelitian ini menegaskan bahwa validasi MIME Type tidak boleh digunakan secara mandiri dalam pengembangan REST API yang menyediakan fitur unggah file. Magic Bytes dapat dijadikan sebagai lapisan validasi minimum yang lebih andal, tetapi tetap tidak cukup untuk menghadapi seluruh bentuk serangan, khususnya file polyglot. Oleh karena itu, pengembang disarankan untuk mengombinasikan Magic Bytes dengan mekanisme keamanan lain, seperti pembatasan tipe file, penyimpanan file pada direktori non-eksekusi, serta penerapan prinsip least privilege dan multi-layer security, guna meminimalkan risiko eksploitasi unggahan file.

4 KESIMPULAN

Studi ini menunjukkan bahwa sistem validasi unggahan file yang hanya mengandalkan MIME Type tidak memadai untuk menangani pola serangan modern. Dalam kasus uji, endpoint berbasis MIME Type masih dapat membiarkan file-file biasa lolos dan menghentikan malware umum yang tidak mencoba menyembunyikan dirinya. Namun, mereka sepenuhnya gagal saat harus menangani malware yang diubah namanya dan file polyglot. Kedua jenis file berbahaya ini dapat melewati validasi dengan mengubah header Content-Type menjadi tipe gambar yang diizinkan. Teknik Magic Bytes, di sisi lain, dapat membedakan antara file gambar asli dan malware biasa serta malware yang telah diganti namanya. Teknik ini memblokir semua sampel dalam kedua kategori tersebut. Namun, Magic Bytes masih tidak dapat mendeteksi file polyglot karena header file yang diperiksa masih valid sebagai gambar, sehingga semua sampel polyglot terlihat seperti file normal.

Dalam hal kinerja, penggunaan Magic Bytes memakan waktu sedikit lebih lama daripada validasi MIME Type karena server harus membaca dan mencocokkan beberapa byte pertama isi file. Uji latensi, di sisi lain, menunjukkan bahwa perbedaan rata-rata waktu pemrosesan antara kedua metode hanya beberapa mikrodetik, yang tidak terlalu besar dibandingkan dengan bagian latensi lain dalam layanan web. Dengan kata lain, Magic Bytes membuat hal-hal lebih aman tanpa memperlambatnya terlalu banyak, sehingga kekhawatiran tentang biaya kinerja tidak terlalu penting untuk skenario validasi dasar seperti ini.

Studi ini memberikan beberapa kontribusi penting. Pertama, menunjukkan lingkungan uji terintegrasi yang memiliki backend REST API berbasis Node.js, generator dataset uji otomatis, dan skrip serangan yang meniru perilaku penyerang dengan mengubah header dan mengirim banyak file sekaligus. Kedua, penelitian ini menawarkan penilaian empiris sistematis terhadap dua sistem validasi unggahan file yang umum digunakan, yaitu MIME Type dan Magic Bytes, di berbagai kategori file termasuk file normal, malware umum, malware yang diubah namanya, dan file polyglot. Ketiga, penelitian ini memperkaya analisis kuantitatif dengan pemeriksaan forensik sederhana terhadap sampel file polyglot, memudahkan pengamatan detail struktur internal file yang berhasil melewati validasi menggunakan potongan heksadesimal dan representasi ASCII.

Hasil ini memiliki dampak nyata pada cara layanan REST API dibangun. Validasi berdasarkan MIME Type telah terbukti tidak memadai dan tidak boleh menjadi satu-satunya lapisan perlindungan, terutama dalam layanan yang memungkinkan pengguna mengunggah file dari sumber yang tidak sepenuhnya tepercaya. Pengembang harus mengimplementasikan validasi berbasis Magic Bytes sebagai persyaratan dasar untuk mencegah infiltrasi malware yang umum dan berganti nama ke dalam sistem, mengingat dampaknya yang relatif kecil terhadap kinerja (Huang et al. 2019). Fakta bahwa Magic Bytes tidak dapat mendeteksi file polyglot menunjukkan bahwa lapisan keamanan

tambahan diperlukan (pertahanan berlapis). Penggunaan teknologi sanitasi konten seperti Content Disarm and Reconstruction (CDR), yang dapat mendeteksi dan menetralkan ancaman tersembunyi (Belkind et al. 2023; Koch et al. 2025), serta pengaturan keamanan ketat yang mengikuti standar industri seperti OWASP File Upload Cheat Sheet (OWASP Foundation 2023b), harus menjadi bagian dari strategi mitigasi modern. Penelitian selanjutnya dapat memperluas fokus untuk mencakup berbagai format file dan variasi payload yang lebih luas, serta menyelidiki integrasi Magic Bytes dengan sistem deteksi berbasis kecerdasan buatan (Machine Learning) untuk menghadapi perkembangan serangan stegomalware dan varian ancaman serupa di masa depan.

DAFTAR PUSTAKA

de Aguiar Monteiro, L., W. H. Carvalho Almeida, R. Rodrigues Hazin, A. Cavalcanti de Lima, S. K. Gomes Silva, and F. Silva Ferraz. 2017. “Survey on Microservice Architecture-Security, Privacy and Standardization on Cloud Computing Environment.” *International Journal on Advances in Security* 11:201 –213. https://www.academia.edu/download/118530699/download_full.pdf#page=211.

Akhavani, Seyed Ali, Bahruz Jabiyev, Ben Kallus, Cem Topcuoglu, Sergey Bratus, and Engin Kirda. 2025. “WAFFLED: Exploiting Parsing Discrepancies to Bypass Web Application Firewalls.” *ArXiv* 42(7). doi:10.48550/arXiv.2503.10846.

Alazmi, Suliman, and Daniel Conte De Leon. 2022. “A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners.” *IEEE Access* 10:33200–219. doi:10.1109/ACCESS.2022.3161522.

Anwar, F., A. Fadlil, and I. Riadi. 2020. “Analisis Validasi Image PNG File Upload Menggunakan Metadata Pada Aplikasi Berbasis Web.” *Edu Komputika Journal* 7(1):10–15. doi:10.15294/edukomputika.v7i1.38722.

Ardiansyah, A., N. Hardi, and W. Gata. 2020. “Identifikasi Dan Recovery File JPEG Dengan Metode Signature-Based Carving Dalam Model Automata.” *Komputika : Jurnal Sistem Komputer* 9(1):75–83. doi:10.34010/komputika.v9i1.2733.

Atlidakis, Vaggelis, Patrice Godefroid, and Marina Polishchuk. 2019. “Restler: Stateful Rest Api Fuzzing.” *International Conference on Software Engineering (ICSE)* 2019-May:748–58. doi:10.1109/ICSE.2019.00083.

Belkind, E., R. Dubin, and A. Dvir. 2023. “Open Image Content Disarm and Reconstruction.” *Arxiv*. doi:10.48550/ARXIV.2307.14057.

Birnbaum, B., C. Kraetzer, and J. Dittmann. 2023. “Stego-Malware Attribution: Simple Signature and Content-Based Features Derived and Validated from Classical Image Steganalysis on Five Exemplary Chosen.” *SECURWARE 2023 : The Seventeenth International Conference on Emerging Security Information, Systems and Technologies*. https://personales.upv.es/thinkmind/dl/conferences/securware/securware_2023/securware_2023_1_70_30047.pdf.

Boiko, Maksym, Viacheslav Moskalenko, and Oksana Shovkoplias. 2023. “Advanced File Carving: Ontology, Models and Methods.” *Radioelectronic and Computer Systems* (3(107)):204–16. doi:10.32620/reks.2023.3.16.

Chaganti, Raj, Vinayakumar R, Mamoun Alazab, and Tuan Pham. 2021. “Stegomalware: A Systematic Survey of Malware Hiding and Detection in Images, Machine Learning Models and Research Challenges.” *ArXiv*. doi:10.36227/techrxiv.16755457.v1.

Chandra, JC, and M. Aldiansyah. 2025. “Rancang Bangun Sistem Deteksi Malware Dalam File Gambar Menggunakan Analisis Metadata Dan Virustotal.” *Bit (Fakultas Teknologi Informasi Universitas Budi Luhur)*. https://journal.budiluhur.ac.id/bit/article/view/4009.

Cohen, Aviad, Nir Nissim, and Yuval Elovici. 2020. “MalJPEG: Machine Learning Based Solution for the Detection of Malicious JPEG Images.” *IEEE Access* 8:19997–11. doi:10.1109/ACCESS.2020.2969022.

Deng, Gelei, Yuekang Li, Yi Liu, Tianwei Zhang, Yang Liu, Zhiyi Zhang, Guo Yu, and Dongjin Wang. 2023. “{NAUTILUS}: Automated {RESTful}{API} Vulnerability Detection.” *USENIX Association*.

https://www.usenix.org/conference/usenixsecurity23/presentation/deng-gelei.

Du, Wenlong, Jian Li, Ruijie Zhao, Junmin Zhu, Yijun Wang, Zhi Xue, Yanhao Wang, Libo Chen, and Zhengguang Han. 2024. "Vulnerability-Oriented Testing for {RESTful}{APIs}." *USENIX Association*. https://www.usenix.org/conference/usenixsecurity24/presentation/du.

Ginting, E. Santi Flora, P. Sihombing, and A. Salim Harahap. 2012. "Pengolahan File Dan Pendeteksian File Yang Dicurigai Adanya Virus Dengan Metode Mime Type." *Skripsi, Prog. Studi Ilmu Komputer, Univ. Sumatera Utara, Medan*. https://repositori.usu.ac.id/handle/123456789/74453.

Hasibuan, M. Siddik, and L. Mashur Gultom. 2018. "Analisis Serangan Deface Menggunakan Backdoor Shell Pada Website." *Techno.Com* 17(4):415–23. doi:10.33633/tc.v17i4.1887.

Al Hilmi, Muhammad Anis, and Rahul Ken Yunan. 2022. "Pengujian Keamanan Fitur Upload File Pada Sistem Aplikasi Web." *Jurnal Informatika Politeknik Harapan Bersama* 7(1):37–42. doi:10.30591/jpit.v7i1.3336.

Huang, Jin, Yu Li, Junjie Zhang, and Rui Dai. 2019. "Uchecker: Automatically Detecting Php-Based Unrestricted File Upload Vulnerabilities." *Institute of Electrical and Electronics Engineers Inc.* 581–92. doi:10.1109/DSN.2019.00064.

Kishnani, U., and S. Das. 2025. "Securing the Web: Analysis of HTTP Security Headers in Popular Global Websites." *ArXiv* 15416 LNCS:87–106. doi:10.1007/978-3-031-80020-7_5.

Koch, L., S. Oesch, A. Sadovnik, and B. Weber. 2025. "On the Abuse and Detection of Polyglot Files." *ArXiv*. doi:10.48550/ARXIV.2407.01529.

Koch, Luke, Sean Oesch, Amul Chaulagain, Mary Adkisson, Samantha Erwin, and Brian Weber. 2022. "Toward the Detection of Polyglot Files." *ACM International Conference Proceeding Series, Association for Computing Machinery* 120–28. doi:10.1145/3546096.3546106.

Müller, Jens, Dominik Noss, Christian Mainka, Vladislav Mladenov, and Jörg Schwenk. 2021. "Processing Dangerous Paths." *In 28th Annual Network and Distributed System Security Symposium, NDSS 2021, The Internet Society*. doi:10.14722/ndss.2021.23109.

OWASP Foundation. 2023a. "API Security Top 10 2023." https://owasp.org/APISecurity/editions/2023/en/0x11-t10/.

OWASP Foundation. 2023b. "File Upload Cheat Sheet." https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html.

Oz, H., G. S. Tuncay, A. Aris, A. Kharraz, and S. Uluagac. 2025. "Beyond the Upload Button: A 10-Year Retrospective on Security Issues within File Upload." *IEEE Communications Magazine* 63(2):104–10. doi:10.1109/MCOM.001.2400253.

Oz, Harun, Abbas Acar, Ahmet Aris, Güliz Seray Tuncay, Amin Kharraz, and Selcuk Uluagac. 2024. "(In) Security of File Uploads in Node. Js." *Proceedings of the ACM Web Conference, Association for Computing Machinery* 1573–84. doi:10.1145/3589334.3645342.

Pan, Zulie, Yuanchao Chen, Yu Chen, Yi Shen, and Xuanzhen Guo. 2021. "Webshell Detection Based on Executable Data Characteristics of Php Code." *Wireless Communications and Mobile Computing* 2021. doi:10.1155/2021/5533963.

Pooj, K., and S. Patil. 2016. "Understanding File Upload Security for Web Applications." *International Journal of Engineering Trends and Technology (IJETT)* 42(7):342–47. doi:10.14445/22315381/ijett-v42p261.

Riadi, I., and E. I. Aristianto. 2016. "An Analysis of Vulnerability Web Against Attack Unrestricted Image File Upload." *Computer Engineering and Applications* 5(1):19–28. doi:10.18495/comengapp.v5i1.161.

Rizal, R., R. Ruuhwan, and S. Chandra. 2020. "Signature File Analysis Using The National Institute Standard Technology Method Base on Digital Forensic Concepts." *Jurnal Informatika Universitas Pamulang* 5(3):364. doi:10.32493/informatika.v5i3.6073.

Sarpong, Pious Akwasi, Lawrence Sakyi Larbi, Daniel Paa Paa, Issah Bala Abdulai, Richard Amankwah, and Akwasi Amponsah. 2021. "Performance Evaluation of Open Source Web Application Vulnerability Scanners Based on OWASP Benchmark." *International Journal of Computer Applications* 174(18):15–22. doi:10.5120/ijca2021921070.

Sethi, P. C. 2024. "FILE CARVING: ANALYZING DATA RETRIEVAL IN DIGITAL FORENSICS." *International Journal of Creative Research Thoughts* 12(4). https://www.researchgate.net/profile/Purna-

Chandra-Sethi/publication/381203350_FILE_CARVING_ANALYZING_DATA_RETRIEVAL_IN_DIGITAL_FORENSICS/links/6661bce8de777205a3114162/File-Carving-Analyzing-Data-Retrieval-in-Digital-Forensics.pdf.

Viglianisi, Emanuele, Fondazione Bruno, Kessler Trento, Italy Michael Dallago, and Mariano Ceccato. 2020. "Resttestgen: Automated Black-Box Testing of Restful Apis." *IEEE 13th International Conference on Software Testing, Verification and Validation (ICST)* 142–52. doi:10.1109/ICST46399.2020.00024.

Wu, Yixin, Yuqiang Sun, Cheng Huang, Peng Jia, and Luping Liu. 2019. "Session-Based Webshell Detection Using Machine Learning in Web Logs." *Security and Communication Networks* 2019. doi:10.1155/2019/3093809.