BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1978-1987
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1290

Otomatisasi Deteksi Dan Mitigasi Webshell PHP Menggunakan Algoritma Random Forest

Najib Khoirul Rizal, Wahyu Widodo*

Informatika, Sekolah Tinggi Manajemen Informatika dan Ilmu Komputer El Rahma Yogyakarta, Yogyakarta, Indonesia
Email: xrizal29@gmail.com, 2,*wahyu@stmikelrahma.ac.id
Email Penulis Korespondensi: wahyu@stmikelrahma.ac.id

Abstrak−Webshell PHP merupakan ancaman serius pada aplikasi berbasis web yang memungkinkan penyerang memperoleh akses tidak sah dan mengendalikan server dari jarak jauh. Variasi teknik obfuskasi menyebabkan webshell sulit dideteksi menggunakan pemindaian berbasis kata kunci konvensional. Penelitian ini bertujuan membangun sistem deteksi dan mitigasi webshell PHP secara real-time berbasis File Integrity Monitoring (FIM) menggunakan algoritma Random Forest. Fitur ekstraksi dibentuk dari pemanggilan fungsi opcode VLD (dengan fallback regex), frekuensi variabel PHP, signature webshell, serta fitur statistik (entropy dan compression ratio). Efisiensi pemrosesan dijaga melalui metode streaming pada berkas berukuran besar (≥ 5 MB), serta pemantauan FIM berbasis Watchdog yang dioptimasi dengan path filtering, event debouncing, dan asynchronous queue. Penelitian ini memberikan kontribusi berupa perancangan mekanisme ekstraksi fitur hibrid yang menggabungkan indikator instruksi internal (*opcode*) dan metrik keacakan statistik (*entropy*) untuk mengenali pola obfuskasi kompleks, serta pengintegrasian model klasifikasi *Random Forest* dengan arsitektur mitigasi aktif berbasis FIM yang mampu melakukan tindakan penanganan otomatis secara mandiri. Pada pengujian menggunakan 1.207 data uji dari dataset yang seimbang (20% test set), model Random Forest mencapai akurasi keseluruhan sebesar 97%. Secara spesifik, kelas webshell memperoleh precision 96%, recall 99%, dan F1-score 97% (555 True Positive dan 29 False Negative), sedangkan kelas normal memperoleh precision 99%, recall 95%, dan F1-score 97% (616 True Negative dan 7 False Positive). Batasan utama dalam penelitian ini dibatasi pada pemantauan berkas bersumber kode PHP, ketergantungan analisis pada environment server yang terinstal ekstensi VLD, serta skenario tindakan mitigasi yang berfokus pada pengarantinaan berkas secara real-time. Pengujian fungsional mengonfirmasi bahwa sistem mampu melakukan pemantauan, klasifikasi, karantina di luar web root, dan pengiriman notifikasi Telegram Bot secara otomatis dan real-time.

Kata Kunci: Webshell; Random Forest; File Monitoring; PHP; Keamanan Siber

Abstract− PHP webshells pose a serious threat to web-based applications, enabling attackers to gain unauthorized access and remotely control servers. Varied obfuscation techniques make these webshells difficult to detect using conventional keyword-based scanning. This research aims to develop a real-time PHP webshell detection and mitigation system based on File Integrity Monitoring (FIM) using the Random Forest algorithm. Feature extraction is derived from VLD opcode function calls (with a regex fallback), PHP variable frequency, webshell signatures, and statistical features (entropy and compression ratio). Processing efficiency is maintained through a streaming method for large files (≥ 5 MB) and a Watchdog-based FIM mechanism optimized with path filtering, event debouncing, and an asynchronous queue. The research contributes a hybrid feature extraction mechanism combining internal instruction indicators (opcodes) and statistical randomness metrics (entropy) to recognize complex obfuscation patterns and integrates a Random Forest classification model with an active FIM-based mitigation architecture capable of autonomous, automated response actions. Testing on 1,207 samples from a balanced dataset (20% test set) showed the Random Forest model achieving an overall accuracy of 97%. Specifically, the webshell class achieved 96% precision, 99% recall, and a 97% F1-score (555 True Positives and 29 False Negatives), while the normal class achieved 99% precision, 95% recall, and a 97% F1-score (616 True Negatives and 7 False Positives). The study is limited to monitoring PHP source code files, relies on a server environment with the VLD extension installed, and focuses mitigation actions on real-time file quarantine. Functional testing confirmed the system's ability to perform monitoring, classification, quarantine outside the web root, and Telegram Bot notifications automatically and in real-time.

Keywords: Webshell; Random Forest; File Monitoring; PHP; Cyber Security

1. PENDAHULUAN

Perkembangan teknologi informasi yang semakin pesat telah meningkatkan pemanfaatan aplikasi berbasis web di berbagai sektor, kondisi tersebut juga diikuti dengan meningkatnya ancaman keamanan siber yang dapat mengganggu kerahasiaan, integritas, dan ketersediaan informasi [1]. Berdasarkan Laporan Lanskap Keamanan Siber Indonesia 2024 dari BSSN, tercatat 330.527.636 trafik anomali, termasuk aktivitas *Advanced Persistent Threat* (APT), *ransomware*, *phishing*, serta 5.780 kasus *defacement*, yang sebagian besar menyasar situs pemerintah [2] Salah satu ancaman yang sering dimanfaatkan penyerang adalah *webshell*, yaitu skrip berbahaya yang memungkinkan akses tidak sah untuk mengendalikan server dari jarak jauh[3]. *Webshell* umumnya digunakan sebagai pintu masuk berbagai serangan lanjutan, seperti pencurian data, modifikasi berkas, hingga penyisipan konten berbahaya. Variasi teknik *obfuscation* dan pemanggilan fungsi secara dinamis menyebabkan webshell sulit dideteksi menggunakan pendekatan konvensional, hal ini dikarenakan pendekatan konvensional hanya mengandalkan pencarian kata kunci seperti *eval(), system()* dan fungsi berbahaya lainnya langsung dari file tersebut. Namun saat kode diubah menjadi teks acak seperti obfuscation kata kunci tersebut hilang dan dianggap lolos dari pemindaian. sehingga diperlukan sistem yang mampu melakukan deteksi melalui perilaku, statistik, dan dapat memitigasi secara *real-time*[4].

Beberapa penelitian sebelumnya telah mengembangkan berbagai pendekatan untuk meningkatkan keamanan aplikasi web. Penelitian pertama mengimplementasikan Wazuh dengan memanfaatkan *File Integrity Monitoring*, *Active Response*, dan integrasi VirusTotal untuk mendeteksi serangan *defacement*, *webshell*, serta *reverse shell* melalui mekanisme pemantauan dan respons otomatis, namun bergantung pada *rate-limit* API eksternal [2]. Penelitian kedua

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1978-1987
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1290

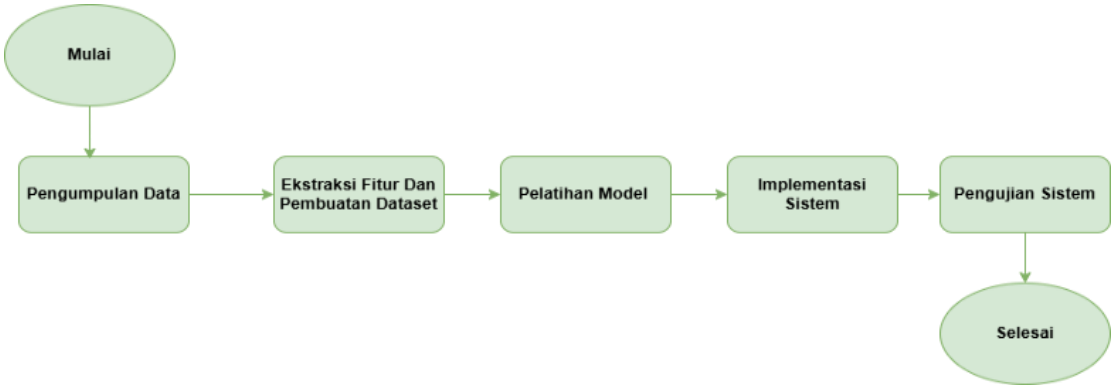
mengembangkan sistem *Security Information and Event Management* (SIEM) berbasis Wazuh yang terintegrasi dengan *Telegram Bot* untuk mendeteksi serangan *deface* dan memberikan notifikasi secara *real-time* kepada administrator, tetapi tidak memiliki mekanisme isolasi/mitigasi berkas secara otomatis[5]. Penelitian ketiga mengusulkan model *Security Operations* yang mengintegrasikan *firewall*, *File Integrity Monitoring*, dan *threat intelligence* sehingga mampu meningkatkan efektivitas deteksi dan respons terhadap berbagai serangan siber, namun belum dilengkapi *engine* pemutus ancaman berbasis *Machine Learning,* [6]. Penelitian keempat menerapkan pendekatan *deep learning* menggunakan CNN dan RNN-LSTM untuk mendeteksi webshell berdasarkan karakteristik kode, namun arsitektur *deep learning* membutuhkan konsumsi daya komputasi (*resource overhead*) yang tinggi serta latensi inferensi yang kurang ideal untuk pemantauan *real-time inline* [7]. sedangkan penelitian kelima berhasil membuktikan bahwa *Random Forest* akurat dalam membedakan kode webshell dan normal. Namun, tidak ada penangkap perubahan berkas otomatis dan tidak ada tindakan pengisolasian berkas secara langsung saat ancaman ditemukan [8].

Berdasarkan kajian tersebut terdapat kesenjangan mendasar yaitu belum adanya sistem terintegrasi yang menggabungkan *File Integrity Monitoring*, ekstraksi fitur struktur, entropi berkas, model *Machine Learning* lokal, serta aksi mitigasi aktif (*quarantine*) secara *Real-Time*. *Random Forest* dipilih dalam penelitian ini dikarenakan algoritma ini menggunakan banyak pohon keputusan dan menentukan hasil melalui voting terbanyak, sehingga dapat meminimalkan *overfitting*, sedangkan metode *RandomizedSearchCV* dipilih untuk mempercepat komputasi karena dapat mengoptimalkan parameter yang digunakan. Proses klasifikasi dilakukan menggunakan *Random Forest* berdasarkan fitur hasil ekstraksi[9]. Sistem juga menerapkan mekanisme pemantauan perubahan berkas (*File Integrity Monitoring*), sehingga setiap penambahan, perubahan, maupun pemindahan berkas PHP dapat dideteksi secara *real-time*. Pendekatan ini mengadopsi konsep *File Integrity Monitoring* yang berfungsi memantau perubahan berkas sebagai indikator awal adanya aktivitas mencurigakan pada sistem[6]. Berkas yang teridentifikasi sebagai webshell akan dipindahkan secara otomatis ke direktori *quarantine* sebagai bentuk mitigasi awal, kemudian sistem mengirimkan notifikasi kepada administrator melalui Telegram. Dengan pendekatan tersebut, diharapkan sistem mampu memberikan kontribusi praktis dalam meningkatkan efektivitas deteksi *webshell*, mempercepat proses respons terhadap ancaman, serta membantu menjaga integritas dan keamanan aplikasi web secara berkelanjutan.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Penelitian ini dilakukan melalui beberapa tahapan yang saling berkaitan, dimulai dari pengumpulan dataset hingga pengujian sistem deteksi webshell secara *real-time*. Setiap tahapan dirancang untuk menghasilkan sistem yang mampu mendeteksi webshell PHP menggunakan pendekatan algoritma *Random Forest*. Alur penelitian yang dilakukan ditunjukkan pada Gambar 1.



Gambar 1. Alur Tahapan Penelitian

Dari Gambar 1, dapat dijelaskan sebagai berikut:

a. Pengumpulan Data
   Tahap pertama dilakukan dengan mengumpulkan data berupa berkas PHP normal dan webshell dari beberapa repositori publik. Seluruh berkas kemudian divalidasi dan dilakukan penghapusan data duplikat untuk memastikan dataset yang digunakan memiliki kualitas yang baik sebelum memasuki tahap berikutnya.
b. Ekstraksi Fitur Pembuatan dataset
   Pada tahap ini setiap berkas PHP diproses untuk menghasilkan fitur-fitur yang digunakan pada proses klasifikasi. Hasil ekstraksi kemudian disusun menjadi dataset yang berisi atribut numerik beserta label kelas sehingga dapat digunakan sebagai data pelatihan dan pengujian model.
c. Pelatihan Model Random Forest
   Dataset yang telah terbentuk dibagi menjadi data pelatihan dan data pengujian. Selanjutnya dilakukan proses pelatihan menggunakan algoritma *Random Forest* dengan optimasi parameter menggunakan *Random Search Cross Validation* sehingga diperoleh model klasifikasi dengan konfigurasi terbaik.

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1978-1987
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1290

d. Implementasi Sistem
Model Random Forest yang telah dilatih diintegrasikan dengan mekanisme dalam menentukan keputusan deteksi terhadap setiap berkas PHP yang dianalisis.
e. Pengujian sistem
Tahap terakhir dilakukan untuk mengevaluasi kinerja model dan implementasi sistem. Pengujian model dilakukan menggunakan metrik *accuracy*, *precision*, *recall*, *F1-score*, dan *confusion matrix*. Selanjutnya dilakukan pengujian sistem secara *real-time* untuk memastikan proses deteksi, mitigasi melalui *quarantine*, serta pengiriman notifikasi menggunakan *Telegram Bot* dapat berjalan sesuai dengan rancangan.

2.2 Pengumpulan dan Pembentukan Dataset

Tahap ini bertujuan menyusun dataset yang representatif untuk membedakan karakteristik berkas sah (*normal*) dan kode berbahaya (webshell). Dataset yang berhasil dikumpulkan berjumlah 6.032 berkas PHP dari beberapa repository open source , yang ditunjukan pada tabel 1.

Tabel 1. Sumber Dataset

No	Tipe	Link	Jumlah
1.	Webshell	https://github.com/Cyc1e183/PHP-Webshell-Dataset	2917
2.	Normal	https://github.com/wordpress/WordPress	1902
3.	Normal	https://github.com/laravel/laravel	27
4.	Normal	https://github.com/codeigniter4/CodeIgniter4	1186
Total			6032

Penggabungan data *normal* dan *webshell* dari berbagai sumber ini meningkatkan keberagaman dan representasi dataset, sehingga dapat mengoptimalkan kemampuan generalisasi model *Random Forest* dalam membedakan berkas normal dan *webshell* pada berbagai skenario aplikasi web. Tahap ekstraksi fitur bertujuan menghasilkan vektor numerik (V = [f_1, f_2, f_3, …, f_n) melalui analisis struktural dan statistik. Karakteristik fungsi serta variabel sensitif (seperti $_GET dan $_POST) diekstrak dari instruksi tingkat rendah. Analisis fungsi PHP ini diawali dengan mentransformasi kode sumber (*source code*) menjadi kode operasi (*opcode*) Zend Engine menggunakan bantuan *Vulcan Logic Disassembler* (VLD) [10]. Representasi visual dari hasil pembangkitan instruksi tingkat rendah tersebut ditunjukkan pada Gambar 2.

```
[redacted]:~$ php -d vld.active=1 -d vld.execute=0 shell.php
Finding entry points
Branch analysis from position: 0
1 jumps found. (Code = 62) Position 1 = -2
filename:       [redacted]/shell.php
function name:  (null)
number of ops:  7
compiled vars:  none
line     #* E I O op                                   fetch          ext  return  operands
-------------------------------------------------------------------------------------------
   1     0  E >   INIT_FCALL                                                    'system'
         1        FETCH_R                              global                   '_GET'
         2        FETCH_DIM_R                                    ~1        ~0, 'cc'
         3        SEND_VAL                                                  ~1
         4        DO_ICALL                                       $2
         5        ECHO                                                      $2
   4     6      > RETURN                                                    1

branch: #  0; line:     1-    4; sop:     0; eop:     6; out0:  -2
path #1: 0,
```

Gambar 2. Output vld

Hasil *opcode* tersebut kemudian diproses menggunakan Regular Expression (Regex) pada pustaka Python untuk mengekstraksi nama nama fungsi yang ditentukan seperti system(), eval(), base64_decode() melalui pola berikut: (?:INIT_FCALL|DO_FCALL|INIT_DYNAMIC_FCALL|INCLUDE_OR_EVAL).*?[\'"]([a-zA-Z0-9_\\]+)[\'"] [11]. Jika VLD gagal akibat kesalahan sintaks, mekanisme *fallback* memindai *source code* secara langsung, didukung sistem pemrosesan adaptif berbasis ambang batas 5 MB (*In-Memory Buffer* atau *Streaming*) untuk efisiensi RAM. Guna mendeteksi *webshell* terobfuscasi, tingkat keacakan distribusi karakter dihitung menggunakan metode *Shannon Entropy* melalui persamaan berikut:

$$H(X) = \sum_{i=1}^{n} p(r_i)\ log_2\ p(r_i) \quad (1)$$

Nilai *H(X)* menunjukkan derajat keacakan karakter di dalam berkas PHP, di mana *p(r_i)* merupakan probabilitas kemunculan karakter ke-*i* dan *n* mewakili jumlah total karakter yang diamati. Nilai entropi yang tinggi mengindikasikan struktur kode yang acak, yang menjadi indikator kuat bahwa berkas *webshell* tersebut telah mengalami proses penyamaran (*obfuscation*). Selain tingkat keacakan, tingkat keteraturan pola isi berkas juga diukur berdasarkan nilai rasio kompresibilitas berkas menggunakan persamaan berikut:

$$Compression\ Ratio\ = \frac{Ukuran\ Setelah\ Kompresi}{Ukuran\ Sebelum\ Kompresi} \quad (2)$$

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1978-1987
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1290

Nilai *compression ratio* dihitung dengan membandingkan ukuran berkas setelah dikompresi menggunakan algoritma zlib terhadap ukuran aslinya. Berkas PHP normal umumnya memiliki pola kode yang berulang sehingga menghasilkan rasio yang kecil (mudah dikompresi), sedangkan berkas *webshell* yang ter-obfuscasi cenderung menghasilkan rasio mendekati angka 1 karena sifat teksnya yang acak dan sulit dikompresi. Pendekatan ini memadukan ekstraksi *opcode* berbasis instruksi dengan metrik statistik teks (entropi dan rasio kompresi) untuk menghasilkan representasi fitur kaya yang efektif mengenali *webshell* konvensional maupun terobfuskasi[10][12].

2.3 Random Forest

Random Forest merupakan algoritma *ensemble learning* yang membangun sejumlah pohon keputusan (*decision tree*) melalui metode *bootstrap sampling* dan *random feature selection* untuk meningkatkan generalisasi serta mencegah *overfitting*[13][14]. Pada tahap klasifikasi, keputusan akhir ditentukan berdasarkan mekanisme *majority voting* atau nilai probabilitas dari seluruh pohon keputusan secara independen. Untuk memperoleh performa model yang optimal, proses optimasi hiperparameter dilakukan secara efisien menggunakan *RandomizedSearchCV*. Metode ini menguji kombinasi parameter taktis seperti *n_estimators, max_depth, min_samples_split, min_samples_leaf*, dan *max_features* melalui *k-fold cross validation* dengan metrik F1-score untuk menjaga keseimbangan *precision* dan *recall*[15]. Nilai rata-rata performa dari setiap lipatan dihitung menggunakan persamaan berikut:

$$CV_{score} = \frac{1}{k}\sum_{i=1}^{k} Score_i \quad (3)$$

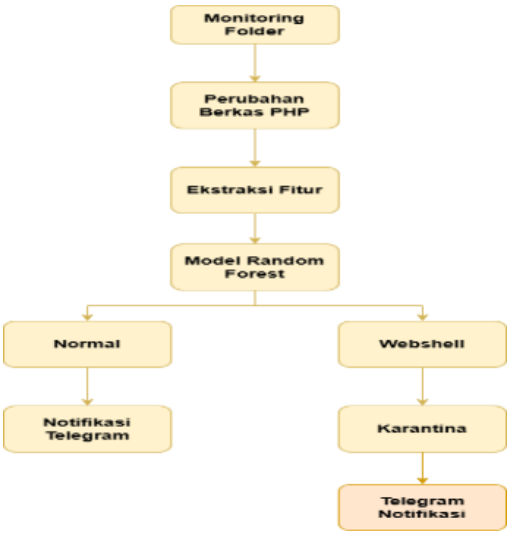
Persamaan di atas digunakan untuk mengevaluasi skor validasi silang (*cross-validation score*), di mana *Score* mewakili nilai F1-score pada lipatan *fold* ke-i dan k menyatakan jumlah total lipatan yang diuji. Kombinasi parameter dengan nilai *CVscore* tertinggi selanjutnya dipilih untuk membangun model final *Random Forest* yang tangguh dalam meminimalkan kesalahan klasifikasi berkas *webshell*.

2.4 File Integrity Monitoring

File Integrity Monitoring (FIM) adalah mekanisme pemantauan sistem untuk mendeteksi perubahan tidak sah pada berkas atau direktori seperti aktivitas *create, modify, move,* dan *delete* yang mengindikasikan ancaman keamanan[16][17]. Pada penelitian ini, FIM diimplementasikan secara *real-time* menggunakan pustaka *Watchdog* pada Python yang bekerja melalui mekanisme *observer* dan *event handler*. Untuk mencegah lonjakan penggunaan CPU, sistem dioptimalkan melalui tiga strategi: pemfilteran jalur (*path filtering*) guna mengabaikan direktori dinamis (misalnya storage/ dan vendor), teknik penundaan peristiwa (*debouncing*) saat terjadi perubahan berturut-turut, serta pemrosesan analisis *opcode* secara asinkron (*asynchronous queue*) di latar belakang. Optimasi ini memungkinkan sistem melakukan pemantauan, deteksi, dan mitigasi otomatis secara cepat tanpa membebani daya komputasi utama server.

2.5 Implementasi Sistem

Implementasi sistem merupakan tahapan penerapan metode yang telah dirancang ke dalam suatu sistem deteksi webshell PHP secara real-time. Sistem dikembangkan menggunakan bahasa pemrograman Python dengan mengintegrasikan algoritma Random Forest sebagai metode klasifikasi, mekanisme File Integrity Monitoring menggunakan pustaka Watchdog, serta layanan Telegram Bot sebagai media notifikasi kepada administrator. Sistem bekerja dengan memantau direktori yang berisi berkas PHP secara terus-menerus. Setiap penambahan, perubahan, atau pemindahan berkas akan memicu proses ekstraksi fitur, yang meliputi ekstraksi opcode, perhitungan entropy, analisis variabel PHP, serta pencarian signature webshell. Selanjutnya, hasil ekstraksi fitur menjadi masukan bagi model Random Forest untuk menentukan apakah berkas tersebut termasuk kategori normal atau webshell[18]. Untuk alur lengkapnya terdapat pada **Gambar 4**.



Gambar 4. Alur Sistem

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1978-1987
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1290

Apabila hasil klasifikasi menunjukkan bahwa berkas merupakan webshell, sistem akan melakukan tindakan mitigasi secara otomatis dengan memindahkan berkas ke direktori *quarantine* untuk mencegah eksekusi lebih lanjut. Selain itu, sistem mengirimkan notifikasi kepada administrator melalui *Telegram Bot* yang berisi informasi mengenai nama berkas, lokasi penyimpanan, serta hasil klasifikasi. Sebaliknya, apabila berkas diklasifikasikan sebagai normal, sistem hanya mencatat hasil pemindaian dan mengirimkan notifikasi tanpa melakukan tindakan karantina. Implementasi tersebut memungkinkan proses deteksi dan respons terhadap ancaman webshell dilakukan secara otomatis segera setelah terjadi perubahan pada berkas PHP, sehingga dapat mempercepat proses mitigasi serta membantu menjaga integritas aplikasi web.

2.6 Pengujian

Pengujian sistem dilakukan untuk mengevaluasi kemampuan sistem dalam mendeteksi webshell PHP serta memastikan setiap komponen berjalan sesuai dengan rancangan. Pengujian meliputi pengujian performa model Random Forest dan pengujian fungsional terhadap mekanisme deteksi serta mitigasi secara real-time[19].

Pengujian performa model dilakukan menggunakan data uji yang tidak digunakan pada proses pelatihan. Hasil klasifikasi dievaluasi menggunakan *confusion matrix* dan metrik evaluasi yang terdiri dari *accuracy*, *precision*, *recall*, dan *F1-score*. Metrik tersebut digunakan untuk mengukur kemampuan model dalam membedakan berkas PHP normal dan webshell. Selanjutnya dilakukan pengujian fungsional untuk memastikan seluruh mekanisme sistem berjalan dengan baik. Pengujian meliputi kemampuan *Watchdog* dalam mendeteksi perubahan berkas secara *real-time*, proses ekstraksi fitur, klasifikasi menggunakan *Random Forest*, pemindahan berkas ke direktori *quarantine* apabila terdeteksi sebagai webshell, serta pengiriman notifikasi kepada administrator melalui Telegram Bot[20].

3. HASIL DAN PEMBAHASAN

3.1 Hasil Pembuatan Dataset

Setelah dilakukan proses validasi dan penghapusan berkas duplikat, diperoleh dataset akhir sebanyak 6.015 berkas yang terdiri atas 3.088 berkas PHP normal dan 2917 berkas webshell. Proses tersebut bertujuan untuk memastikan setiap data yang digunakan bersifat unik sehingga dapat mengurangi potensi bias pada proses pelatihan model. Komposisi dataset yang seimbang antara kedua kelas diharapkan dapat membantu model dalam mempelajari karakteristik masing-masing kelas secara lebih optimal.

```
  GNU nano 8.4                                        all_dataset.csv
filename,Entropy,Compression_ratio,eval,assert,system,exec,shell_exec,passthru,proc_open,popen,pcntl_exec,pcntl_fork,pcntl_s
9a1ba8d66603893ea59987c278cce305.php,4.981213391336292,0.2486957214353038,0,0,1,1,1,1,0,1,0,0,0,0,0,5,0,7,6,4,0,2,2,5,1,0,2,3,
22f6b3710499104 0a3b8ab1e32e2ca08.php,4.541467880199449,1.14,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
795e3b9b3aafa0feb562a3acd8882817.php,5.169203478478008,0.6435439560439561,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0ab432a5e2d8f5b4efffc1d570240bdd.php,5.761114290321789,0.2235362149384238,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
bbaba04910d91fc14a53f01115cb652b.php,5.823368417554647,0.3005222253608859,3,0,1,0,0,0,0,0,0,0,0,0,12,0,11,8,2,1,3,1,7,2,0,2,
38dc8383da0859dca82cf0c943dbf16d.php,6.028558478513042,0.7552332087006359,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
26fda294b2c808784b8f0ee4767288e4.php,4.021345141061048,0.1915402706573954,0,0,0,0,0,0,2,0,0,0,0,3,1,0,1,0,1,0,0,4,0,0,0,0,
105a8066a644a9410d3ebc16f06a2884.php,6.021978943834432,0.7596620079863092,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
2916dbfd9888417b9445ce42599934c6.php,4.796128496866428,0.8138297872340425,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
f2f210d772d18530f9609fb44e7b340e.php,5.43263425514405,0.283076923076923,0,0,0,0,0,0,0,0,0,0,0,2,0,1,1,0,0,1,1,1,1,0,0,0,1,
00dcb456f3a935f350b133db7dcaf830.php,5.225339181488643,0.6426140231449966,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
89cedd38d2b1d001a9f3aef16bd5ae23.php,5.419001906715257,0.2949790794979079,0,0,0,0,0,0,0,0,0,0,0,0,6,6,0,0,0,0,0,0,0,0,0,
e6a1992bd9e222991afb0b1d21c78566.php,5.238129413399892,0.6433378196500673,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
db77cad200c0c342abd56a62d699c598.php,4.974746028606628,0.7807017543859649,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,
07c417d64f115b8f8a6911ba006874d2.php,5.198639581314107,0.6453566621803499,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
fe7dfe14e6318a2e2a6e748560e6a144.php,5.511385297747667,0.2795754359363154,0,0,1,1,1,1,0,1,0,0,0,0,5,0,7,6,4,0,2,2,5,1,0,2,3,
```

Gambar 5. Sample dataset

Berdasarkan Gambar 5 Hasil pembentukan dataset menunjukkan bahwa seluruh berkas berhasil direpresentasikan ke dalam sejumlah fitur yang digunakan sebagai masukan pada model klasifikasi. Setiap data terdiri atas atribut hasil ekstraksi beserta label kelas yang merepresentasikan kategori normal atau webshell.

3.2 Hasil Pelatihan Model

Setelah dataset berhasil dibentuk, dilakukan proses pelatihan menggunakan algoritma *Random Forest*. Optimasi parameter dilakukan menggunakan *Random Search Cross Validation* untuk memperoleh kombinasi parameter yang memberikan performa terbaik terhadap data pelatihan. Rentang parameter yang digunakan pada proses optimasi ditunjukkan pada Tabel 2.

Tabel 2. Tabel Rentang Parameter

No	Nama Parameter	Nilai
1.	n_estimators	100-300
2.	max_depth	15-25
3.	min_samples_split	2-20
4.	min_samples_leaf	1-10
5.	max_features	Sqrt2 dan log2

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1978-1987
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1290

Setelah seluruh proses optimasi selesai, diperoleh kombinasi parameter terbaik sebagaimana ditunjukkan pada **Tabel 3.**

Tabel 3. Tabel Hasil Kombinasi Parameter Terbaik

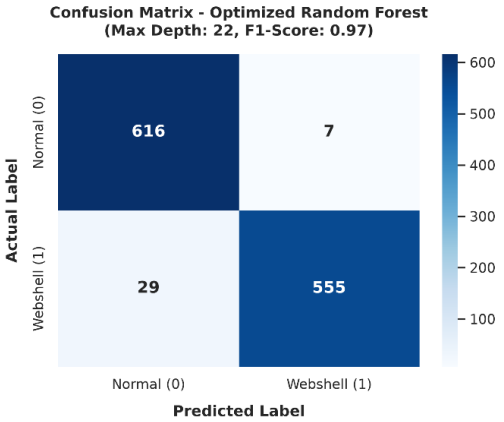
No	Nama Parameter	Nilai
1.	n_estimators	229
2.	max_depth	22
3.	min_samples_split	3
4.	min_samples_leaf	1
5.	max_features	sqrt

Model dengan parameter terbaik kemudian disimpan dalam format *Pickle (.pkl)* dan digunakan pada tahap implementasi sistem sehingga proses deteksi dapat dilakukan tanpa melakukan pelatihan ulang.. Hasil evaluasi klasifikasi report ditunjukkan pada Tabel 4.

Tabel 4. Tabel Hasil Train Split

No	Kelas1	Precision	recall	F1-score	support
1.	Webshell	96	99	97	623
2.	Normal	99	95	97	584
Accuracy				97	1207

Hasil evaluasi menunjukkan bahwa model mampu mengklasifikasikan berkas PHP normal dan webshell dengan tingkat akurasi yang tinggi. Nilai precision dan recall yang diperoleh juga menunjukkan bahwa model memiliki kemampuan yang baik dalam mengenali webshell tanpa menghasilkan terlalu banyak kesalahan klasifikasi.



Gambar 6. Confusion Matrix

Berdasarkan Gambar 6, model *Random Forest* berhasil mengklasifikasikan 616 berkas normal (*True Negative*) dan 555 berkas *webshell* (*True Positive*) secara tepat dengan akurasi 97%. Meski demikian, terdapat 7 berkas normal yang salah terdeteksi sebagai *webshell* (*False Positive*/FP) dan 29 berkas *webshell* yang terlewat (*False Negative*/FN). Evaluasi terhadap 7 kasus *False Positive* ini sangat krusial mengingat aksi otomatis sistem adalah melakukan karantina terhadap berkas yang terdeteksi.

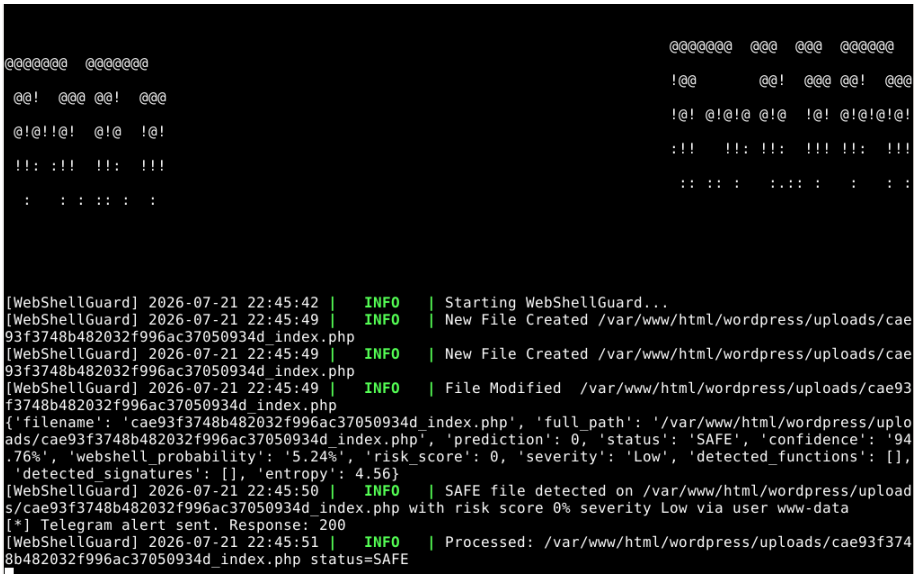
Dalam lingkungan produksi, jika berkas normal yang dikarantina merupakan komponen penting aplikasi web (seperti skrip *controller* atau modul fungsi), situs web dapat mengalami gangguan fungsi (*downtime*) atau menampilkan pesan kesalahan *500 Internal Server Error*. Risiko ini merupakan implikasi (*trade-off*) yang tidak terhindarkan dari penggunaan sistem keamanan berbasis pencegahan otomatis (*preventive/active response*).

Untuk memitigasi dampak dari kesalahan deteksi tersebut, desain mekanisme karantina pada penelitian ini diterapkan secara aman. Berkas yang terdeteksi *webshell* tidak langsung dihapus permanen dari penyimpanan, melainkan dipindahkan ke direktori terisolasi di luar ***web root***. Dengan pendekatan ini, ancaman *webshell* langsung dinetralisir secara *real-time*, sementara berkas normal yang terdampak kesalahan deteksi (FP) tetap tersimpan utuh dan dapat dipulihkan (*restore*) dengan mudah oleh administrator tanpa kehilangan data.

3.3 Pengujian Deteksi Dan Mitigasi

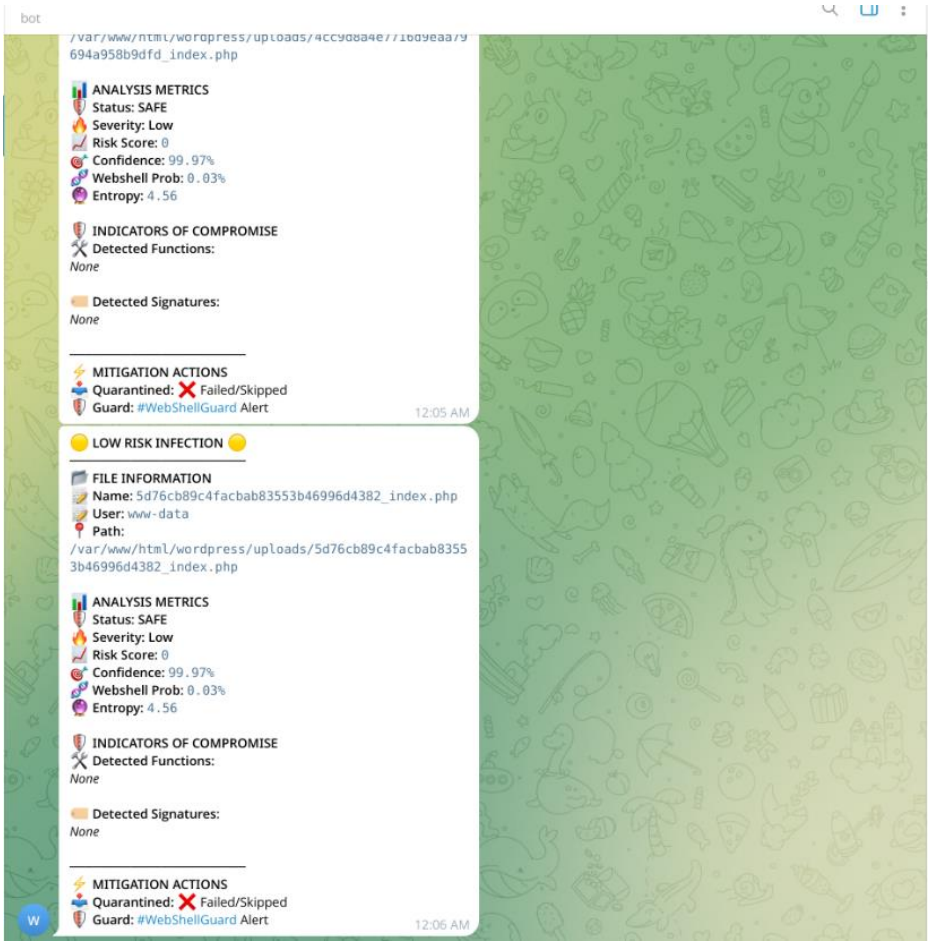
Pengujian pertama dilakukan menggunakan berkas PHP normal dengan menambahkan atau memodifikasi berkas pada direktori yang dipantau oleh sistem. Hasil pengujian menunjukkan bahwa sistem berhasil mendeteksi perubahan berkas dan secara otomatis menjalankan proses ekstraksi fitur serta klasifikasi menggunakan model Random Forest. Hasil pengujian ditunjukkan pada Gambar 7 dan Gambar 8.

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1978-1987
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1290



Gambar 7. Pegujian File Normal

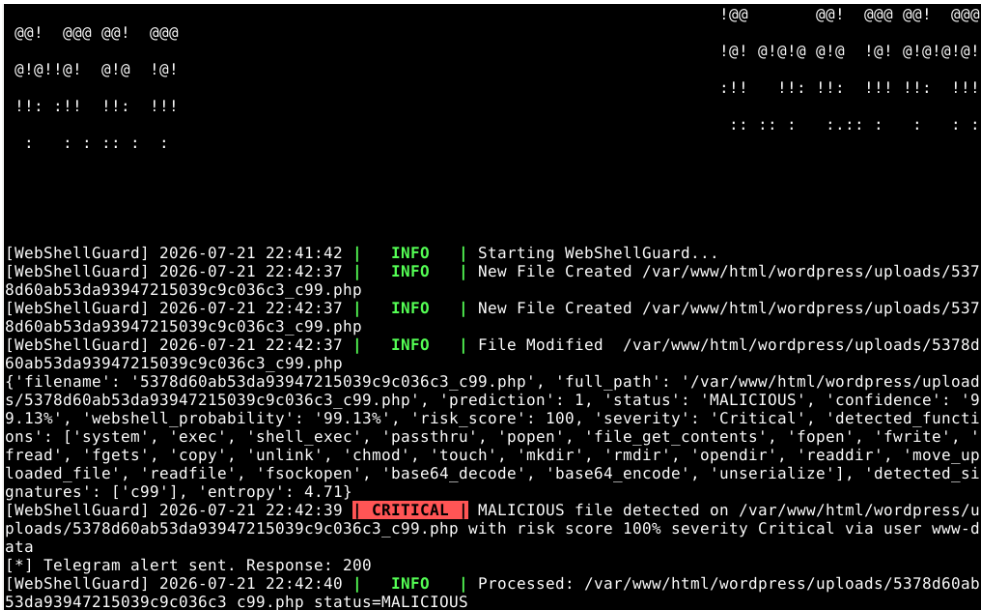
Gambar 7 menunjukkan log aktivitas sistem saat mendeteksi penambahan atau perubahan berkas PHP normal pada direktori pemantauan. Terlihat keluaran hasil klasifikasi oleh model Random Forest.



Gambar 8. Notifikasi Telegram File Normal

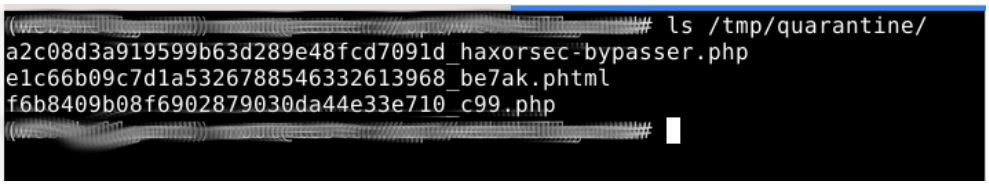
Berdasarkan hasil klasifikasi, berkas dikenali sebagai PHP normal sehingga sistem tidak melakukan tindakan karantina. Meskipun demikian, sistem tetap mengirimkan notifikasi kepada administrator melalui Telegram sebagai informasi bahwa proses pemindaian telah berhasil dilakukan. Pengujian berikutnya dilakukan menggunakan beberapa sampel webshell PHP yang ditempatkan pada direktori pemantauan. Setelah berkas terdeteksi, sistem secara otomatis melakukan ekstraksi fitur dan proses klasifikasi.

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1978-1987
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1290



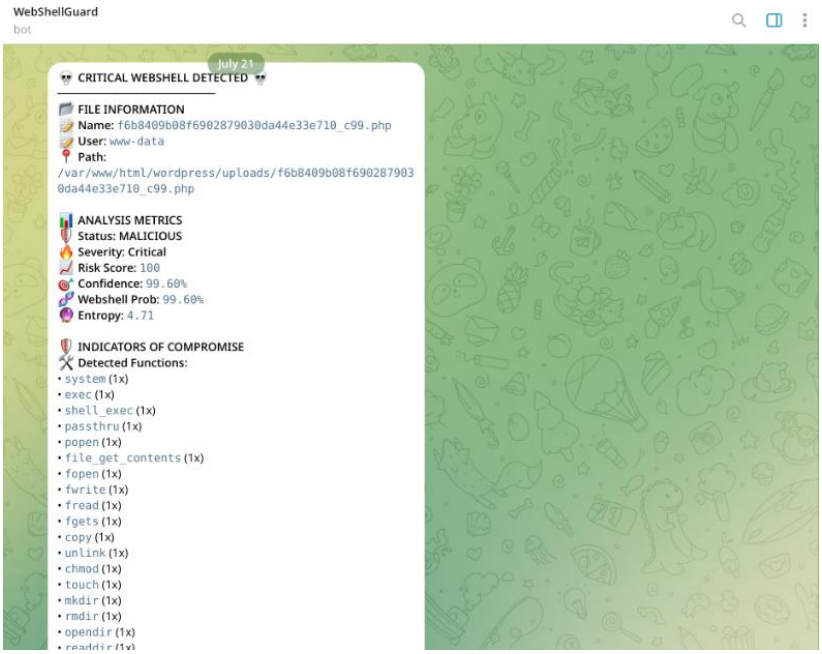
Gambar 9. Pengujian File Webshell

Berdasarkan Gambar 9 menunjukkan bahwa webshell berhasil diidentifikasi sebagai berkas berbahaya. Setelah proses identifikasi selesai, sistem secara otomatis memindahkan berkas ke direktori *quarantine* sebagai tindakan mitigasi yang dapat ditunjukan pada Gambar 10.



Gambar 10. Hasil Karantina atau Mitigasi

Setelah proses identifikasi selesai, sistem secara otomatis memindahkan berkas ke direktori *quarantine* sebagai tindakan mitigasi. Setelah itu sistem mengirimkan notifikasi ke telegram Telegram yang berisi informasi nama berkas, serta status tindakan yang dilakukan. Yang dapat ditunjukan pada Gambar 11.



Gambar 11. Notifikasi Webshell Telegram

Berdasarkan Gambar 11 Terdapat beberapa informasi yang dikirimkan yaitu berupa nama berkas yang terdeteksi, fungsi fungsi terdeteksi dan notifikasi bahwa berkas tersebut telah dikarantina.

3.4 Pembahasan

Berdasarkan hasil pengujian dan evaluasi performa, penelitian ini mencatatkan efektivitas deteksi *webshell* berbasis *opcode* VLD dan *Random Forest* yang terintegrasi langsung dengan aksi karantina *real-time*. Jika disandingkan dengan temuan M. Djamalyanto et al.[5] dan F. Arman et al. [2] yang mengandalkan pemantauan berbasis SIEM dan *File Integrity Monitoring* (FIM), pemantauan tingkat sistem terbukti sangat handal dalam menjaga integritas berkas, namun F. Arman et al. [2] mencatatkan adanya kendala latensi pemrosesan serta visualisasi *alert* saat menangani pemrosesan data berukuran besar. Temuan dalam penelitian ini berhasil mengatasi kendala pemrosesan tersebut melalui penerapan *path filtering*, *event debouncing*, *asynchronous queue*, dan metode *streaming* memori sehingga pemrosesan berkas tetap stabil pada lingkungan bertrafik tinggi.

Dari sudut pandang pemantauan lalu lintas jaringan, temuan T. Arya Saputra et al. [6] (yang menggabungkan FIM dengan WAF) dan M. Firman Prayogi et al. [8] membuktikan keunggulan penyaringan serangan di lapisan HTTP (WAF) yang mampu merespons ancaman secara cepat dengan rata-rata waktu respons masing-masing 4 detik dan 1,3 detik (serta akurasi >99% pada temuan M Firman Prayogi et al.[8] yang juga mengandalkan *Random Forest*). Meski demikian, pendekatan berbasis trafik memiliki fokus area yang berbeda dengan penelitian ini yang melakukan inspeksi mendalam pada logika instruksi internal berkas PHP, sehingga analisis tetap presisi meskipun *webshell* terselubung (*obfuscated*) atau diunggah melalui jalur yang tidak terdeteksi oleh WAF. Sementara itu, temuan R. Yuranda and E. S. Negara**.**[7] yang berfokus pada analisis *code-level* mengonfirmasi bahwa ekstraksi *opcode* PHP menggunakan model *Deep Learning* seperti *Convolutional Neural Networks* (CNN) menghasilkan akurasi klasifikasi yang tinggi. Penelitian ini memperkuat validitas penggunaan *opcode* sebagai fitur deteksi utama, sekaligus membuktikan bahwa ekstraksi *opcode* yang dikombinasikan dengan algoritma *Random Forest* dan metrik statistik mampu memberikan performa klasifikasi yang kompetitif dengan beban komputasi lebih ringan, serta dapat diintegrasikan langsung ke dalam arsitektur mitigasi aktif secara mandiri.

4. KESIMPULAN

Penelitian ini berhasil mengimplementasikan sistem deteksi dan mitigasi *webshell* PHP secara otomatis dengan mengintegrasikan pemantauan *real-time* berbasis Watchdog, analisis *opcode* VLD, dan model *Random Forest*. Penelitian ini memberikan kontribusi berupa perancangan mekanisme ekstraksi fitur hibrid yang menggabungkan indikator instruksi internal (*opcode*) dan metrik keacakan statistik (*entropy*) untuk mengenali pola obfuskasi kompleks, serta pengintegrasian model klasifikasi *Random Forest* dengan arsitektur mitigasi aktif berbasis FIM yang mampu melakukan tindakan penanganan otomatis secara mandiri. Kendala performa pada server bertrafik tinggi ditangani melalui optimasi *path filtering*, *event debouncing*, dan pemrosesan *asynchronous queue*, sementara efisiensi memori pada berkas besar dijaga melalui metode *streaming*. Meski demikian, sistem ini memiliki batasan operasional; mekanisme karantina otomatis terhadap *False Positive* tetap berisiko menimbulkan *downtime* atau hilangnya dependensi fungsi apabila berkas aplikasi yang diubah secara langsung (*live update/editing*) keliru terdeteksi sebagai *webshell*. Selain itu, cakupan dataset masih terbatas pada beberapa *framework* umum. Untuk penelitian selanjutnya, disarankan memperluas variasi dataset, serta menerapkan ekstraksi fitur *hybrid* menggabungkan TF-IDF kode sumber dan *opcode* VLD dengan algoritma seperti XGBoost atau LightGBM guna meminimalkan *False Positive* dan memperkuat deteksi *webshell* terobfuskasi.

REFERENCES

[1] A. I. A. Azkiya and B. Santoso, “Indonesian Journal of Computer Science,” *Indones. J. Comput. Sci.*, vol. 13, no. 1, pp. 1227–1240, 2023, [Online]. Available: http://ijcs.stmikindonesia.ac.id/ijcs/index.php/ijcs/article/view/3135

[2] F. Arman, A. Wahyu Azinar, A. Senja Fitrani, and A. Eviyanti, “Implementasi Monitoring Dan Pencegahan Serangan Defacement Judi Online Menggunakan Wazuh,” *JATI (Jurnal Mhs. Tek. Inform.*, vol. 10, no. 1, pp. 1309–1317, 2026, doi: 10.36040/jati.v10i1.16988.

[3] A. Ikhwanudin, T. Toifur, A. Syamhalim, M. Yusuf, F. A. Yusri, and M. J. Ardiansyah, “Evaluasi Keamanan Fitur Unggah Berkas Pada Situs Web,” vol. 10, no. 2, pp. 2066–2072, 2026, doi: 10.36040/jati.v10i2.17273.

[4] H. J. Lee, S. J. Hwang, M. Pratiwi, and Y. H. Choi, *Obfuscated PHP Webshell Detection Using the Webshell Tailored TextRank Algorithm*, vol. 1, no. 1. Association for Computing Machinery, 2024. doi: 10.1145/3605098.3635940.

[5] M. Djamalyanto, L. Widyawati, Husain, and I. P. Hariyadi, “Implementasi Security Information And Event Management Untuk Untuk Mencegah Serangan Deface Pada Server Terintegrasi Telegram,” vol. 11, no. 1, pp. 31–42, 2025, doi: 10.30742/melekitjournal.v11i1.398.

[6] T. Arya Saputra, S. Wahyu, M. Hadi Arfian, and N. Budi Santoso, “Implementation of IT Security Operations Management Application For Cyber Security Threat Monitoring,” JEPIN (Jurnal Edukasi dan Penelitian Informatika), vol. 12, no. 1, pp. 63–74, 2026, doi: 10.26418/jp.v12i1.105673.

[7] R. Yuranda and E. S. Negara, “Application of Deep Learning Algorithm for Web Shell Detection in Web Application Security System,” *J. Sisfokom (Sistem Inf. dan Komputer)*, vol. 13, no. 3, pp. 330–336, 2024, doi: 10.32736/sisfokom.v13i3.2234.

[8] M. Firman Prayogi, A. Fahrudi Setiawan, and F. Xaverius Ariwibisono, “Perancangan Sistem Keamanan Web Menggunakan Metode Random Forest,” *JATI (Jurnal Mhs. Tek. Inform.*, vol. 9, no. 6, pp. 10650–10657, 2025, doi: 10.36040/jati.v9i6.15457.

[9] Z. Wang, H. Wang, S. Yuan, and Z. Tian, “Incremental learning research for webshell detection,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 37, no. 8, pp. 1–27, 2025, doi: 10.1007/s44443-025-00235-8.

[10] Y. Zhao *et al.*, “Malicious webshell family dataset for webshell multi-classification research,” *Vis. Informatics*, vol. 8, no. 1, pp. 47–55, 2024, doi: 10.1016/j.visinf.2023.06.008.

[11] Y. , I. R. , Syamsul Bahri, "Analisa Log Web Server Untuk Mengetahui Pola Perilaku Pengunjung Website Menggunakan Teknik Regular Expressions," *Coding J. Komput. dan Apl.*, vol. 7, no. 01, pp. 120–130, 2019, doi: 10.26418/coding.v7i01.32692.
[12] A. Hannousse and S. Yahiouche, "Multi-language Webshell dataset," 2021, *Mendeley Data*: V2. doi: 10.17632/wt8m6bcwbr.2.
[13] J. Caesario, Nofiyati, and D. K. Wibowo, "Identification and Classification of Cyber Attacks on ELDIRU UNSOED using Random Forest Algorithm," *J. Tek. Inform.*, vol. 6, no. 4, pp. 2785–2794, 2025, doi: 10.52436/1.jutif.2025.6.4.5239.
[14] I. Siswanto *et al.*, "Implementasi Algoritma Random Forest untuk Deteksi Serangan Siber pada Jaringan Komputer," Siematic, vol. 1, no. 1, p. 2025. https://ejurnal.ibisa.ac.id/index.php/ji/article/view/439
[15] L. Oriana, A. Dwi, S. Wati, A. P. Ramahdani, N. N. Safira, and E. Ismanto, "Peningkatan Kinerja Model Random Forest untuk Deteksi Kecurangan Kartu Kredit Menggunakan RandomizedSearchCV," *J. Fasilkom*, vol. 15, no. 2, pp. 229–237, 2025, , doi: 10.37859/jf.v15i2.9832.
[16] A. Kamil, D. Rizaludin, and A. T. Ni'mah, "Implementasi Wazuh FIM (File Integrity Monitoring) untuk Perlindungan Keamanan Sistem Informasi pada Unit Kegiatan Mahasiswa di Universitas Trunojoyo Madura," *Sains Data J. Stud. Mat. dan Teknol.*, vol. 2, no. 2, pp. 80–92, 2024, doi: 10.52620/sainsdata.v2i2.127.
[17] G. A. Magapu, S. Sai, R. Kodandapuram, V. Ashok, and R. Tatiparthi, "Detect, React, Recover: A Hands-On Ransomware Defense System", ResearchGate, 2025, doi: 10.13140/RG.2.2.36642.34242.
[18] E. A. Winanto, Y. Novianto, S. Sharipuddin, I. S. Wijaya, and P. A. Jusia, "Peningkatan Performa Deteksi Serangan Menggunakan Metode Pca Dan Random Forest," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 11, no. 2, pp. 285–290, 2024, doi: 10.25126/jtiik.20241127678.
[19] Z. Alamin and Ritzkal, "Real-Time Phishing Detection Using Google Safe Browsing API and Machine Learning," *Journix J. Informatics Comput.*, vol. 1, no. 2, pp. 51–62, 2025, doi: 10.63866/journix.v1i2.8.
[20] T. Meliana, J. Jemakmun, and S. Suryayusra, "Pelatihan Sistem Deteksi Intruksi Menggunakan Wazuh Berbasis Notifikasi Real-Time Telegram di Universitas Bina Darma," *J. Pengabdi. Masy. Bangsa*, vol. 4, no. 4, pp. 1282–1289, 2026, doi: 10.59837/jpmba.v4i4.4413.