Jurnal Pendidikan dan Teknologi Indonesia (JPTI)
Vol. 5, No. 8, Agustus 2025, Hal. 2228-2237
DOI: https://doi.org/10.52436/1.jpti.957
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Pengujian Keamanan Aplikasi *Mobile Learning Management System* berbasis *Deep Reinforcement Learning* dengan Model *Fuzzing* Adaptif

Rama Amindra Buana[*1], Yusuf Kurniawan[2]

[1,2]Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Indonesia
Email: [1]23223061@std.stei.itb.ac.id, [2]yusufk@itb.ac.id

Abstrak

Keamanan aplikasi *Learning Management System* (LMS) berbasis *mobile* menjadi perhatian utama seiring dengan meningkatnya penggunaan *platform* digital dalam kegiatan pembelajaran. Namun, pengujian keamanan secara manual dan metode *fuzzing* tradisional sering kali tidak efektif dalam mendeteksi kerentanan tersembunyi. Oleh karena itu, penelitian ini bertujuan untuk merancang dan mengimplementasikan model *fuzzing* berbasis *Deep Reinforcement Learning* (DRL) guna mengoptimalkan proses pengujian keamanan pada aplikasi LMS berbasis *mobile* dalam format APK. Model yang dikembangkan menggunakan algoritma *Deep Q-Network* (DQN) untuk mengeksplorasi komponen UI, *intent*, dan input dengan mengandalkan hasil analisis statis serta dataset payload dari OWASP dan FuzzDB. Sistem dikendalikan oleh agen DRL yang dilatih melalui interaksi bertahap dengan *environment Appium* dan *ADB*, dengan *reward function* yang mempertimbangkan pemicu API, deteksi *crash*, dan variasi aksi. Evaluasi dilakukan berdasarkan jumlah respon API yang dipicu, skenario *crash* yang dihasilkan, serta stabilitas dan konsistensi *reward* selama pelatihan. Hasil menunjukkan bahwa agen DRL mampu mempertahankan *reward* stabil di atas 500, memicu 11 *crash* unik, dan menjelajahi 95 aksi eksplorasi berbeda dengan jumlah aksi berulang yang minim. Penelitian ini menunjukkan bahwa pendekatan DRL dapat meningkatkan cakupan pengujian dan efektivitas deteksi kerentanan pada aplikasi LMS *mobile*. Temuan ini penting bagi pengembang dan institusi pendidikan dalam memperkuat keamanan aplikasi sebelum implementasi luas, serta berkontribusi pada pengembangan metode *fuzzing* otomatis berbasis kecerdasan buatan.

Kata kunci: *APK, Deep Reinforcement Learning, DQN, Eksplorasi UI, Fuzzing, Keamanan Aplikasi Mobile.*

Security Testing of Mobile Learning Management System Applications Based on Deep Reinforcement Learning with Adaptive Fuzzing Model

Abstract

The security of mobile-based Learning Management System (LMS) applications has become a major concern in line with the increasing use of digital platforms for educational activities. However, manual security testing and traditional fuzzing methods are often ineffective in detecting hidden vulnerabilities. Therefore, this study aims to design and implement a fuzzing model based on Deep Reinforcement Learning (DRL) to optimize the security testing process for APK-based LMS mobile applications. The proposed model adopts the Deep Q-Network (DQN) algorithm to explore UI components, intents, and input fields by leveraging static analysis results and fuzzing payload datasets from OWASP and FuzzDB. The system is controlled by a DRL agent trained through staged interactions within an Appium and ADB-based environment, using a reward function that accounts for triggered API responses, crash detection, and action diversity. Evaluation is conducted based on the number of triggered API endpoints, generated crash scenarios, and reward consistency throughout the training process. The results show that the DRL agent is able to maintain a stable reward above 500, trigger 11 unique crashes, and explore 95 different exploration actions with minimal number of repeated actions. This study shows that the DRL approach can improve the test coverage and effectiveness of vulnerability detection in mobile LMS applications. These findings are important for developers and educational institutions in strengthening application security before widespread implementation, and contribute to the development of AI-based automated fuzzing methods.

Keywords: *APK, Deep Reinforcement Learning, DQN, Fuzzing, Mobile Application Security, UI Exploration.*

p-ISSN: 2775-4227
e-ISSN: 2775-4219

1. PENDAHULUAN

Aplikasi *mobile Learning Management System* (LMS) telah menjadi salah satu sarana utama dalam mendukung pembelajaran daring, baik di lingkungan pendidikan formal maupun organisasi profesional. LMS *mobile* memberikan fleksibilitas bagi pengguna untuk mengakses materi pembelajaran, forum diskusi, maupun evaluasi secara *real time* melalui perangkat Android [1]. Namun, pertumbuhan ini juga disertai dengan peningkatan potensi ancaman keamanan. Laporan DBIR 2024 mencatat bahwa sektor pendidikan mengalami lebih dari 1.500 *data breach*, menjadikannya salah satu sektor paling rentan terhadap serangan siber [2]. Studi Greco dan Chianese (2024) menyoroti bahwa LMS menjadi target strategis karena menyimpan data akademik dan personal yang bernilai tinggi [3].

Arsitektur aplikasi LMS *mobile* mencakup berbagai komponen seperti aktivitas, *intent*, komunikasi antar-komponen, *permission*, dan *Application Programming Interface* (API) eksternal. Kompleksitas ini memperluas *attack surface* dan membuka peluang eksploitasi jika tidak diuji secara menyeluruh [4], [5], [6]. Berbagai pendekatan telah digunakan untuk menguji keamanan aplikasi *Android*, salah satunya adalah teknik *fuzzing*, yaitu metode otomatis yang menyuntikkan *input* acak atau termutasi untuk mengidentifikasi kerentanan [7]. Meskipun efektif, pendekatan *fuzzing* tradisional seperti *Monkey Testing* maupun *random event injection* sering kali tidak efektif dalam menjangkau seluruh jalur eksekusi atau komponen tersembunyi dalam aplikasi [8], [9].

Untuk meningkatkan efisiensi eksplorasi input, sejumlah penelitian mulai mengeksplorasi integrasi *machine learning* dalam proses *fuzzing*. *NeuFuzz* memanfaatkan *Deep Neural Network* untuk memprediksi jalur rentan dalam aplikasi Android [8], sementara *APIRL* menggabungkan algoritma *Deep Q-Netwrok* (DQN) dengan transformer untuk pengujian REST API secara adaptif [10]. Di sisi lain, *DRLFCfuzzer* mengintegrasikan format *constraints* dalam proses *reward shaping*, dan *RLF* memodifikasi AFL dengan strategi eksplorasi berbasis *Q-learning* [11], [12]. Penelitian [13] menunjukkan bahwa pendekatan DRL dapat diarahkan untuk menemukan urutan eksploitasi yang rentan dalam sistem kompleks seperti *smart contracts*. Pendekatan serupa dapat diadopsi untuk menyusun strategi eksplorasi adaptif dalam konteks *fuzzing* aplikasi *mobile*. Namun, pendekatan tersebut umumnya terbatas pada pengujian *file* atau API, tanpa menyentuh eksplorasi berbasis antarmuka pengguna (*UI exploration*), yang justru menjadi elemen penting dalam interaksi aplikasi *mobile* [14].

Sementara itu, beberapa model seperti *ARES* dan *FATE* memang telah dirancang untuk mengeksplorasi aplikasi Android secara otomatis [15], dan strategi DRL adaptif terbukti mampu meningkatkan *coverage* jalur eksekusi aplikasi [11]. Akan tetapi, pendekatan tersebut masih jarang mengintegrasikan hasil analisis statis seperti struktur UI, *intent*, dan *endpoint* API sebagai komponen pembentuk *state*. Padahal, informasi tersebut sangat krusial dalam meningkatkan relevansi aksi agen terhadap arsitektur internal aplikasi, khususnya dalam konteks pengujian keamanan aplikasi LMS *mobile* yang kompleks dan kaya fitur.

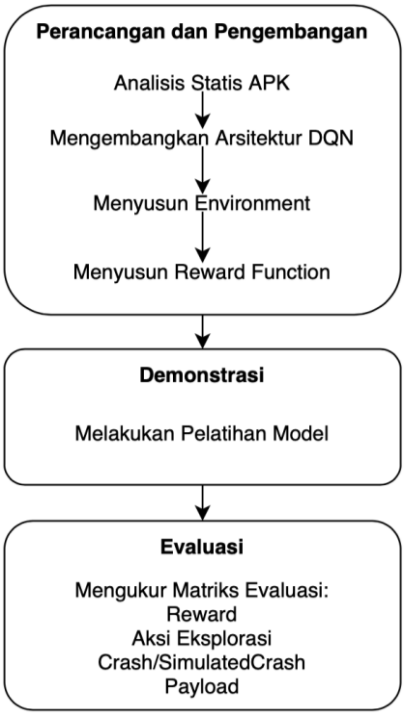
Berdasarkan hal tersebut, Terdapat *research gap* dalam integrasi antara hasil analisis statis — seperti *intent*, elemen UI, dan API — dengan strategi eksplorasi adaptif berbasis DRL. Sebagian besar pendekatan sebelumnya belum menggabungkan informasi struktural ini secara sistematis ke dalam pembentukan *state* dan *action space* agen DRL. Penelitian ini berupaya mengisi celah tersebut dengan merancang sistem *fuzzing* yang secara eksplisit mengeksplorasi antarmuka pengguna, *intent*, dan *input* aplikasi berdasarkan hasil *reverse engineering* terhadap APK LMS *mobile*.

Penelitian ini bertujuan untuk merancang model *fuzzing* otomatis berbasis DRL. Sistem ini difokuskan untuk meningkatkan efektivitas eksplorasi *input* dan deteksi kerentanan dalam aplikasi LMS *mobile* berbasis APK. Pendekatan yang digunakan memanfaatkan algoritma DQN dan menggabungkan data analisis statis serta respons dinamis aplikasi untuk membentuk *environment* dan *reward function*yang adaptif.

2. METODE PENELITIAN

Penelitian ini menggunakan pendekatan *Design Science Research Methodology* (DSRM)[16], yang cocok untuk riset yang berorientasi pada pembangunan artefak teknologi. DSRM mencakup enam tahap: identifikasi masalah, perumusan tujuan, desain dan pengembangan, demonstrasi, evaluasi, serta komunikasi hasil penelitian. Dalam konteks ini, DSRM digunakan untuk merancang, mengimplementasikan, dan mengevaluasi sistem *fuzzing* otomatis berbasis DRL untuk pengujian keamanan aplikasi LMS berbasis APK. Alur penyelarasan DSRM dalam penelitian ini dapat terlihat pada Gambar 1. Fokus utama penelitian ini adalah pengembangan agen DRL yang mampu mengeksplorasi UI dan *intent* aplikasi secara adaptif untuk mendeteksi kerentanan.

p-ISSN: 2775-4227
e-ISSN: 2775-4219



Gambar 1. Metode Penelitian *Fuzzing* Berbasis DRL

2.1. Pengumpulan Data dan Analisis Statis APK

Langkah pertama dalam proses desain adalah melakukan analisis statis terhadap APK LMS *mobile* untuk memperoleh struktur internal alikasi. *Tools* yang digunakan meliputi:

- JADX dan APKTool: untuk mendekompilasi APK dan mengekstrak *AndroidManifest.xml*, daftar komponen, dan struktur *intent*;
- ADB Logcat: untuk menangkap sinyal API dan *exception;* serta
- MobSF: untuk identifikasi permission dan API *calls*.

Data hasil analisis ini diformat menjadi file CSV dan JSON:

- *intents_filtered.csv* untuk daftar intent eksplisit;
- *clickable_ui_elements.csv* untuk elemen UI dan koordinat;
- *api_url_keywords.json* dan *api_logcat.csv* untuk aktivitas API yang dipicu saat eksplorasi

Selain *dataset* hasil analisis statis, digunakan pula *dataset payload fuzzing*. *Dataset payload fuzzing* yang digunakan dalam pemicuan *intent* dan *input* aplikasi dikonstruksi dari repositori publik *OWASP SecLists* [17] dan *FuzzDB* [18]. Kedua dataset ini dipilih karena menyediakan koleksi *payload* untuk berbagai skenario serangan, seperti *file upload bypass*, API *fuzzing*, SQL *injection*, dan *login attack*. *Payload* dari dataset ini kemudian difilter dan disesuaikan untuk skenario *input* teks maupun parameter *intent* dalam *environment* simulatif. Ringkasan distribusi *payload* ditampilkan pada Tabel 1.

Tabel 1. Distribusi *Payload* per Kategori

Kategori	*SecLists*	*FuzzDB*	Total
Command Injection	1.948.918	390.230	2.339.148
File Upload Fuzzing	2.348.781	806.497	3.155.278
API_Fuzzing	911.197	1.864	913.061
Login Attack	241.368	–	241.368
File Upload Bypass	62.205	–	62.205
SQL_Injection	8.794	3.309	12.103
Path_Traversal	12.228	–	12.228
Total *Payload*	5.533.491	1.201.900	6.735.391

p-ISSN: 2775-4227
e-ISSN: 2775-4219

2.2. Perancangan Model DRL

Model *fuzzing* otomatis dalam penelitian ini dirancang untuk mensimulasikan interaksi pengguna terhadap aplikasi Android melalui representasi *environment* berbasis RL. *Environment* ini merepresentasikan kondisi aplikasi sebagai *state* dan memungkinkan pengambilan *action* eksploratif yang dapat memicu perubahan perilaku aplikasi. Secara umum, *environment* dikembangkan untuk merepresentasikan kerentanan potensial yang bisa dipicu melalui input eksploratif. Sementara itu, agen dilatih untuk menemukan strategi interaksi yang mampu mengarahkan sistem ke dalam kondisi rawan (misalnya *crash*, *exception*, atau *error* API).

Agen DRL yang digunakan dalam penelitian ini mengadopsi algoritma DQN dengan tiga lapis jaringan saraf: *input*-128-*hidden*-25-*output*. DQN dipilih karena keunggulannya dalam lingkungan diskret dengan jumlah aksi terbatas dan *reward* jarang. Strategi pembelajaran yang digunakan mencakup

- *Epsilon-greedy exploration*;
- *Replay Buffer*; serta
- *Target network* untuk stabilitas [5][6].

Model dibangun menggunakan *PyTorch* dan *Stable-Baselines3*.

2.3. Pembentukan *Environment* Simulatif

Environment dibentuk dalam format *OpenAI Gym*, dengan nama *FuzzingEnvironment*. *State* direpresentasikan sebagai vektor berdimensi 10 yang mencerminkan:

- Jumlah *intent*, elemen UI aktif, *login* status;
- Mode layar (*fullscreen*/*split*), API *triggered* dari *logcat*; serta
- *Noise* acak untuk mencegah *overfitting* terhadap struktur tertentu.

Sementara itu, *action space* mencakup:

- Tap elemen UI;
- Pemicu intent dengan *payload*;
- Pengisian *input* teks; dan
- Simulasi *file upload*.

2.4. Strategi Pembelajaran dan *Reward Function*

Dalam proses pembelajaran, nilai Q dari setiap aksi diperbarui berdasarkan formulasi DQN sebagai berikut:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma a' maxQ(s',a') - Q(s,a)] \qquad (1)$$

Dengan:

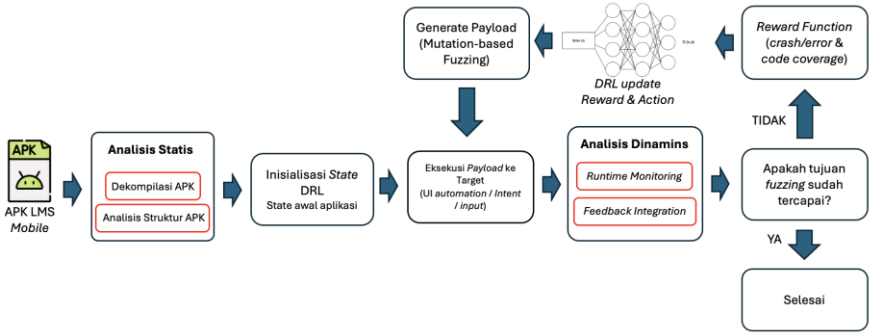
- $Q(s,a)$: nilai Q dari aksi a pada *state* s;
- α: *learning rate*;
- r: *reward* yang diterima setelah melakukan aksi;
- γ: *discount factor*.

Nilai Q digunakan untuk mengarahkan pemilihan aksi pada iterasi selanjutnya, baik dalam mode eksplorasi maupun eksploitasi.

Fungsi *reward* dirancang secara bertingkat untuk merefleksikan nilai strategis dari setiap aksi yang dilakukan agen. Sebagai contoh, aksi yang menyebabkan *crash* aplikasi diberikan *reward* tinggi, sedangkan aksi yang hanya mengeksplorasi aktivitas baru atau memicu API baru diberikan *reward* sedang. Selain itu, *payload* dari kategori prioritas tinggi seperti *SQL Injection* diberikan *reward* tambahan. Sebaliknya, aksi yang berulang seperti klik di tempat sama atau penggunaan *payload* yang sama akan dikenakan penalti. Pendekatan ini mengikuti prinsip *reward shaping* dalam DRL untuk memastikan pembelajaran yang lebih efisien dan bermakna yang telah diterapkan secara efektif dalam domain keamanan aplikasi *mobile* maupun *smart contract* [13], [14].

Agen DRL dilatih menggunakan algoritma DQN dengan arsitektur jaringan tiga lapis. Proses pelatihan menggunakan pendekatan ε-*greedy* untuk eksplorasi, dan target *network* untuk stabilitas pembelajaran. Setiap episode terdiri dari maksimum 250 langkah, dengan mekanisme *early stopping* jika *reward* mencapai ambang tertentu. *Log reward* dan aksi dicatat setiap episode untuk mengevaluasi konvergensi dan stabilitas agen. Gambar 2 menunjukkan keseluruhan arsitektur sistem yang akan direalisasikan.

p-ISSN: 2775-4227
e-ISSN: 2775-4219



Gambar 2. Alur Kerja *Fuzzing* Berbasis DRL

2.5. Prosedur Evaluasi

Evaluasi dilakukan pada *environment* simulatif berbasis APK LMS yang telah dianalisis. Agen DRL dilatih selama 100 episode dan diuji pada 30 episode tambahan dengan $\varepsilon = 0$. Kriteria evaluasi meliputi:

- *Reward* total per episode dan *moving average reward*;
- Jumlah aksi eksplorasi unik (*intent*/API/UI);
- Jumlah *crash*/*error* yang terpicu; dan
- Konsistensi strategi agen (evaluasi konvergensi dan diversifikasi aksi).

Hasil *reward* dan log aksi dikumpulkan dan divisualisasikan untuk menunjukkan kestabilan strategi agen. Tujuan evaluasi bukan membandingkan dengan pendekatan tradisional (itu dilakukan di bagian hasil), tetapi menilai apakah agen telah belajar strategi eksplorasi yang optimal dalam lingkup *environment* yang tersedia. Strategi ini juga diperkuat dengan penggunaan replay buffer dan target *network* yang telah terbukti meningkatkan stabilitas pembelajaran dalam pengujian keamanan berbasis DRL [19].

3. HASIL DAN PEMBAHASAN

Bagian ini menyajikan hasil dari proses perancangan dan implementasi model *fuzzing* berbasis DRL yang telah dijelaskan pada bagian metode sebelumnya. Fokus pembahasan mencakup struktur arsitektur sistem, mekanisme lingkungan simulatif dan agen DRL, serta hasil pelatihan awal yang dilakukan terhadap *environment* berbasis APK LMS *mobile*. Pembahasan dilakukan secara bertahap mulai dari gambaran arsitektur, perincian aksi dan *state* dalam *environment*, penggunaan *dataset payload* untuk eksplorasi, hingga evaluasi awal terhadap kinerja agen DRL berdasarkan *reward* dan *log* pelatihan. Tujuan dari bagian ini adalah untuk menunjukkan bahwa sistem yang dirancang telah berjalan secara fungsional, adaptif, dan sesuai dengan tujuan desain. Seluruh pengujian dilakukan dalam lingkungan simulatif dan berbasis data hasil analisis manual terhadap satu APK LMS *mobile*.

3.1. Hasil

Sebagai hasil dari tahapan perancangan dan pengembangan, sistem *fuzzing* yang dirancang terdiri dari dua komponen utama, yaitu: (1) lingkungan simulatif berbasis *Android* sebagai *environment* DRL, dan (2) agen DQN yang bertindak sebagai aktor eksplorasi. Seluruh struktur arsitektur ini dibangun berdasarkan metode yang telah dijelaskan sebelumnya, dengan penyesuaian terhadap karakteristik aplikasi target dan kompleksitas data *input* yang tersedia.

Secara khusus, file *intents_filtered.csv* digunakan untuk menyusun daftar *intent* eksplisit yang menjadi bagian dari ruang aksi agen, sedangkan *clickable_ui_elements.csv* memuat koordinat elemen UI yang dapat diklik serta komponen input teks yang relevan. File *api_url_keywords_fixed.json* berisi *keyword* API yang digunakan sebagai indikator untuk mendeteksi pemicuan API dalam simulasi, dan file *api_logcat.csv* merepresentasikan *log* respons sistem terhadap aksi, termasuk pemetaan koordinat UI terhadap jalur API. Gabungan file-file ini menjadi dasar pembentukan *state vector* berdimensi 10 yang mencerminkan status aplikasi saat ini, termasuk jumlah *intent* dan UI aktif, status *login*, mode layar, serta indikator aktivitas API. Dengan pendekatan ini, *environment* DRL yang dibentuk tidak hanya bersifat generik, tetapi kontekstual terhadap struktur internal APK target.

Sebagai pelengkap terhadap data struktur aplikasi, sistem ini juga memperkaya *attack surface* dengan menggunakan *dataset payload* dari repositori publik OWASP *SecLists* dan *FuzzDB*. *Dataset payload* diklasifikasikan ke dalam beberapa kategori berdasarkan jenis eksploitasi, seperti SQL *Injection*, *Path Traversal*, dan *File Upload Bypass*. *Dataset* ini membantu memperkaya variasi input yang dieksplorasi agen, dan digunakan dalam *reward function* untuk memberikan nilai tambah bagi *payload* dari kategori prioritas tinggi.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Lingkungan *fuzzing* dikembangkan dalam bentuk *environment OpenAI Gym*, yang diberi nama *FuzzingEnvironment*. Lingkungan ini dibentuk dari *parsing* hasil analisis statis serta pemetaan *payload* ke dalam JSON terstruktur. *Environment* menerima aksi eksplorasi berupa tap pada UI (dengan koordinat spesifik) atau pemicuan *intent* dengan parameter berisi *payload*. Setiap aksi yang dilakukan agen mengubah kondisi lingkungan (*state*) dan menghasilkan *reward* sebagai sinyal pembelajaran.

Representasi *state* terdiri dari vector dengan 10 dimensi yang menggambarkan situasi aplikasi secara ringkas namun informatif. *State* ini diubah menjadi tensor menggunakan PyTorch, dan digunakan sebagai *input* ke dalam model DQN untuk prediksi nilai aksi. Selain *environment* dan *state*, didefinisikan juga daftar aksi yang dapat dipilih oleh agen (*action space*). Daftar lengkap *state* dan *action* yang digunakan dalam penelitian ini dapat dilihat pada Tabel 2 dan Tabel 3.

Agen DRL dilatih menggunakan algoritma DQN dengan arsitektur jaringan saraf (*input*, 128, 25, *output*). *Reward function* dirancang secara bertingkat:

- +10 untuk *crash* (*SimulatedCrash*),
- +5 untuk *payload* dari kategori prioritas tinggi (misal SQLi),
- +3 untuk *payload* dari kategori baru,
- +2 untuk pemicu API *intent*/*logcat*,
- +1 untuk eksplorasi aktivitas atau komponen baru,
- −0.5 hingga −2 untuk aksi berulang atau *payload* yang sama.

Untuk memperoleh konfigurasi terbaik, dilakukan eksplorasi terhadap delapan kombinasi nilai *hyperparameter*, dengan memvariasikan tiga parameter utama: ukuran *replay buffer*, ukuran *batch*, dan jumlah unit pada *hidden layer* pertama dalam arsitektur DQN. Nilai lainnya seperti *learning rate* dan *discount factor* dipertahankan konstan berdasarkan nilai umum yang direkomendasikan dalam literatur. Konfigurasi akhir yang digunakan dalam penelirian ini dapat terlihat pada Tabel 4.

Pelatihan dilakukan selama 100 episode menggunakan konfigurasi hyperparameter seperti pada Tabel 4. Pada fase awal pelatihan (episode 1–30), *reward* per episode masih menunjukkan fluktuasi tinggi di kisaran 350–400, mencerminkan ketidakstabilan agen dalam memilih aksi eksploratif. Hal ini merupakan konsekuensi dari fase eksplorasi awal (*exploration phase*) dalam strategi *epsilon-greedy*, yaitu karena agen masih mencoba berbagai kemungkinan aksi tanpa preferensi terhadap efektivitasnya. Memasuki episode ke-40 hingga ke-100, total *reward* mulai meningkat secara signifikan dan cenderung stabil di atas angka 500.

Tabel 2. *State Space* Agen DRL

Fitur *State*	Tipe Data
Hash dari nama aktivitas saat ini	*Float* [0,1]
Jumlah elemen UI yang bisa diklik	*Integer* ≥ 0
Jumlah *intent* yang tersedia	*Integer* ≥ 0
Status *Login*	*Biner* (0 atau 1)
Jumlah *task* terbaru	*Integer* ≥ 0
Mode Window	*Biner* (0 atau 1*)*
Jumlah *keyword* API yang ditemukan dari *logcat*	*Integer* ≥ 0
Noise acak 1 (*random float* [0,1])	*Float* [0,1]
Noise acak 2 (*random float* [0,1])	*Float* [0,1]
Noise acak 3 (*random float* [0,1])	*Float* [0,1]

Tabel 3. *Action Space* Agen DRL

Jenis Aksi	Deskripsi	Catatan
Intent	Mengeksekusi *intent* eksplisit berdasarkan hasil *parsing AndroidManifest.xml*	Jumlah *intent* tergantung APK target
UI *Tap*	Melakukan klik pada elemen UI menggunakan koordinat X dan Y	Jumlah aksi tergantung jumlah elemen UI
Input Text	Mengisi *field input* dengan *payload* seperti SQLi atau XSS	Tergantung jumlah *field input* dalam UI
File Upload	Menyimulasikan pengunggahan *file* berbahaya (misalnya *.exe, .php*)	Aktif jika *field upload* tersedia di UI

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Tabel 4. *Hyperparameter* Agen DRL

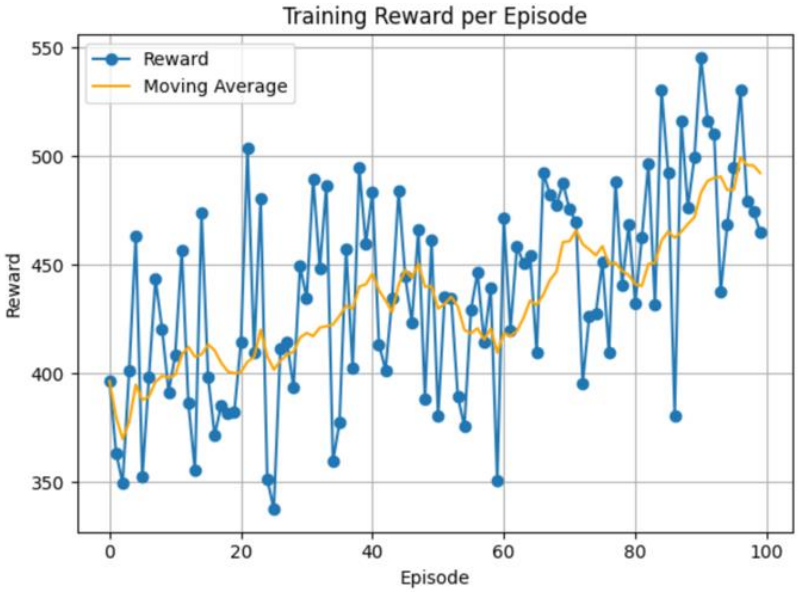
Parameter	Nilai
Replay buffer size	20.000
Batch size	32
Hidden input (neurons)	128
Learning rate	0.0005
Discount factor	0.99
Epsilon decay	0.995

Pencatatan data *reward*, langkah, dan rata-rata *reward* berjalan (*moving average*) dilakukan secara otomatis oleh sistem, dan hasilnya disimpan ke dalam file *training_log.csv*. Data tersebut digunakan untuk memvisualisasikan kinerja agen selama pelatihan, sebagaimana ditunjukkan pada Gambar 3. Log ini menjadi acuan untuk mengevaluasi kestabilan proses pembelajaran dan mendeteksi perubahan strategi eksplorasi dari waktu ke waktu. Selain pencatatan *reward*, sistem juga menyimulasikan deteksi *crash* atau *error* melalui respons aksi yang memuat pola-pola *payload* berbahaya seperti *shutdown*, *panic*, *.php*, atau *.exe*, yang secara eksplisit ditandai dengan sinyal "*SimulatedCrash*". Deteksi ini digunakan dalam fungsi *reward* sebagai indikator keberhasilan eksplorasi ke kondisi rawan, dengan *reward* tinggi (+10) jika *crash* berhasil dipicu.

Moving average reward (per 10 episode) menunjukkan tren konvergensi hingga mendekati nilai 530, sebagaimana ditunjukkan pada Gambar 3. Grafik ini menggambarkan kemampuan agen dalam membentuk strategi eksplorasi yang lebih efektif dari waktu ke waktu. Strategi yang dimaksud yaitu dalam menghindari aksi repetitif dan memaksimalkan *reward* berbasis pemicuan *intent*, eksplorasi aktivitas baru, serta penggunaan *payload* kategori prioritas tinggi.

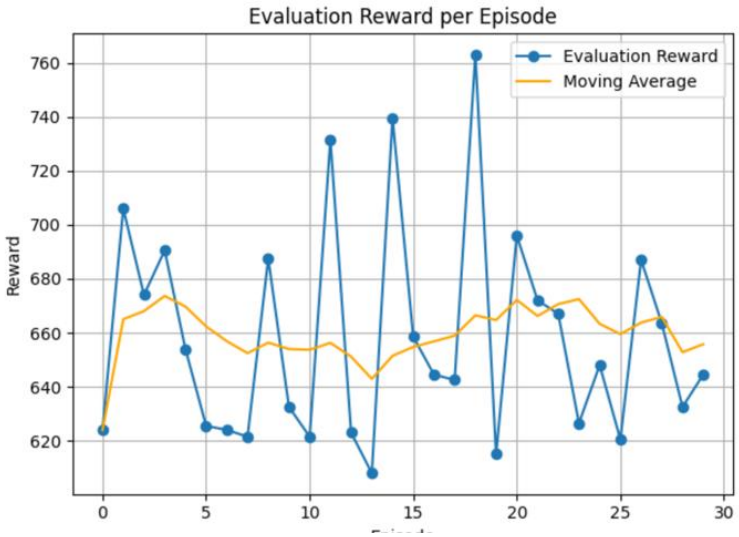
Selama pelatihan, *log* aksi agen menunjukkan kecenderungan untuk menggabungkan aksi klik UI dengan pemicuan *intent* yang bervariasi. Hal ini menunjukkan bahwa agen mampu membentuk urutan aksi yang tidak hanya acak, tetapi juga kontekstual terhadap kondisi *state environment*. Agen juga cenderung menghindari aksi yang sebelumnya telah dikenai penalti seperti klik di posisi yang sama atau penggunaan *payload* yang berulang. Strategi ini sejalan dengan bentuk fungsi *reward* bertingkat yang digunakan, yang mendorong eksplorasi elemen baru dan pemberian nilai lebih pada aksi strategis.

Evaluasi dilakukan pada 30 episode tambahan tanpa eksplorasi ($\varepsilon = 0$) untuk menguji kemampuan agen menjalankan strategi optimal. Hasil evaluasi menunjukkan bahwa agen mampu mempertahankan *reward* tinggi secara konsisten, dengan *reward* maksimum sebesar 763 dan minimum sebesar 608. Nilai rata-rata *reward* evaluasi mendekati 660. Ini menandakan bahwa kebijakan (*policy*) yang dipelajari agen memiliki generalisasi yang cukup baik terhadap *state* yang ditemui. Gambar 4 menggambarkan grafik lengkap hasil evaluasi dari Agen DRL.



Gambar 3. Grafik *Training Reward* DRL-*based Fuzzing*

p-ISSN: 2775-4227
e-ISSN: 2775-4219



Gambar 4. Grafik *Evaluation Reward* DRL-*based Fuzzing*

Selain *reward*, kinerja eksplorasi dievaluasi dari tiga aspek:

- Jumlah aksi eksplorasi unik (*intent*/API/UI);
- Jumlah *crash* atau *error* yang berhasil dipicu;
- Jumlah payload prioritas tinggi yang digunakan secara efektif.

Hasil evaluasi agen DRL ini dirangkum dalam Tabel 5.

3.2. Diskusi

Hasil pelatihan menunjukkan bahwa fungsi *reward* bertingkat mampu mendorong pembentukan strategi eksploratif yang stabil dan adaptif. Agen tidak hanya menjalankan aksi secara acak, tetapi menunjukkan pola yang terarah dalam memilih kombinasi aksi *intent*, UI, dan *input*. *Reward* mulai stabil di atas 500 setelah sekitar 40 episode, menandakan bahwa model DQN berhasil belajar dari pengalaman. Hal ini sejalan dengan pendekatan *reward shaping* dalam DRL yang telah diterapkan dalam domain keamanan aplikasi *mobile*.

Tren konvergensi *reward* ini mendukung temuan [15] yang menunjukkan efektivitas DRL dalam eksplorasi UI untuk pengujian aplikasi *Android*. Namun, penelitian ini memberikan kontribusi tambahan dengan mengintegrasikan hasil analisis statis (jumlah *intent*, elemen UI, status *login*, API) ke dalam definisi *state*. Selain itu, sistem juga menggunakan *dataset* eksploitasi nyata dari *OWASP SecLists* dan *FuzzDB*, bukan hanya mutasi acak atau *input* generik. Hal ini menjadikan proses eksplorasi lebih relevan terhadap vektor serangan nyata dalam konteks APK LMS *mobile*.

Jika dibandingkan dengan metode *fuzzing* tradisional seperti *Monkey Testing*, pendekatan ini menunjukkan performa eksplorasi yang lebih unggul. Dalam *baseline* percobaan, *Monkey* menghasilkan banyak aksi klik berulang namun hanya menjangkau permukaan UI dan tidak menghasilkan *crash* signifikan. Sementara itu, sistem berbasis DRL mampu menemukan 11 *crash* unik dan menjaga diversitas aksi dengan penalti pada tindakan repetitif.

Sistem ini juga menunjukkan keunggulan dibanding pendekatan DRL yang hanya fokus pada API seperti pada penelitian [10]. Pendekatan mereka menggunakan DQN untuk REST API *fuzzing*, sedangkan penelitian ini memperluas cakupan eksplorasi ke UI, *intent*, dan *file input*. Hal ini memberikan ruang yang lebih luas untuk mendeteksi celah keamanan yang tidak dapat dijangkau oleh metode *single-vector fuzzing*. Integrasi umpan balik dari *error log* dan *crash report* ke dalam *reward* terbukti mampu mengarahkan eksplorasi ke kondisi rawan.

Tabel 5. Hasil Evaluasi Eksplorasi Agen DRL

Matrik Evaluasi	Nilai Rata-Rata
Total *reward* (30 eps)	660
Aksi eksplorasi unik	95
Crash / *Simulated Crash*	11
Payload prioritas tinggi	28
Payload berulang (dikenai penalti)	4

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Implikasi dari hasil ini sangat relevan dalam konteks peningkatan keamanan aplikasi *mobile*, khususnya pada sektor pendidikan. Sistem DRL-*based fuzzing* yang dirancang dapat digunakan sebagai alat bantu otomatis untuk menguji keamanan APK sebelum digunakan secara luas. Dengan karakter adaptifnya, sistem ini juga dapat digunakan ulang untuk aplikasi lain dengan struktur serupa, cukup dengan mengupdate hasil analisis statis. Dengan demikian, pendekatan ini berkontribusi pada efisiensi pengujian dan peningkatan ketahanan aplikasi terhadap serangan

4. KESIMPULAN

Penelitian ini berhasil merancang dan mengimplementasikan sistem *fuzzing* berbasis DRL untuk pengujian keamanan aplikasi LMS *Mobile*. Model ini menggunakan algoritma DQN untuk mengarahkan eksplorasi aksi secara adaptif terhadap elemen UI, *intent*, dan *input* aplikasi. Evaluasi menunjukkan bahwa agen mampu mempelajari strategi eksplorasi yang efektif, dengan *reward* yang stabil dan jumlah aksi unik yang tinggi, serta mampu memicu *crash* dalam skenario pengujian *grey-box*. Pendekatan ini terbukti lebih efisien dibandingkan metode *fuzzing* tradisional seperti *Monkey Testing*, khususnya dalam hal variasi aksi dan kemampuan menjangkau jalur eksekusi yang lebih dalam.

Temuan ini memberikan kontribusi pada pengembangan metode pengujian keamanan yang otomatis, adaptif, dan *reusable* untuk aplikasi *mobile*. Sistem dapat diterapkan pada berbagai aplikasi sejenis dengan hanya memperbarui hasil analisis statis dan *dataset payload*. Namun demikian, penelitian ini memiliki beberapa keterbatasan yang dapat dijadikan bahan pengembangan lanjutan. Pertama, pelatihan dan evaluasi dilakukan hanya terhadap satu APK dalam lingkungan simulatif, sehingga generalisasi terhadap berbagai jenis aplikasi masih perlu dibuktikan. Kedua, proses pemetaan *payload* ke dalam konteks *intent* atau *input* belum sepenuhnya kontekstual; dibutuhkan mekanisme *payload prioritization* atau *context-aware payload matching* untuk meningkatkan efektivitas *input*.

Penelitian selanjutnya dapat memperluas cakupan pengujian ke jenis aplikasi lain, serta mengeksplorasi teknik optimasi *payload* berbasis konteks agar lebih presisi dalam memicu kerentanan. Integrasi dengan metode evaluasi berbasis *coverage-guided fuzzing* atau algoritma DRL lanjutan seperti PPO dan A3C juga berpotensi meningkatkan efektivitas model secara signifikan.

UCAPAN TERIMA KASIH

Penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada Kementerian Komunikasi dan Digital (Komdigi) Republik Indonesia yang telah memberikan beasiswa sehingga memungkinkan berlangsungnya studi dan penelitian ini. Penulis juga sangat berterima kasih kepada Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, yang telah menyediakan fasilitas akademik yang sangat baik, dukungan yang berkelanjutan, dan lingkungan yang mendukung untuk penelitian. Kontribusi dari kedua lembaga tersebut sangat penting bagi keberhasilan penyelesaian karya ini.

DAFTAR PUSTAKA

[1] K. Thangavel, ‘Learning Management Systems (LMS) in Higher Education: Enhancing Teaching, Learning, And Administrative Processes’, Aug. 2024, doi: 10.34293/eduspectra.v6i2.09.

[2] C. D. Hylender, P. Langlois, A. Pinto, and S. Widup, ‘2024 Data Breach Investigations Report’, 2024. Accessed: Feb. 28, 2025. [Online]. Available: https://www.verizon.com/business/resources/Te3/reports/2024-dbir-data-breach-investigations-report.pdf

[3] D. Greco and L. Chianese, ‘Exploiting LLMs for E-Learning: A Cybersecurity Perspective on AI-Generated Tools in Education’, in *2024 IEEE International Workshop on Technologies for Defense and Security, TechDefense 2024 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 237–242. doi: 10.1109/TechDefense63521.2024.10863662.

[4] P. Hung, J. Lam, C. Wong, and T. Chan, ‘A Study on Using Learning Management System with Mobile App’, in *Proceedings - 2015 International Symposium on Educational Technology, ISET 2015*, Institute of Electrical and Electronics Engineers Inc., Mar. 2016, pp. 168–172. doi: 10.1109/ISET.2015.41.

[5] H. Abdullah and S. R. M. Zeebaree, ‘Android Mobile Applications Vulnerabilities and Prevention Methods: A Review’, in *Proceedings of 2021 2nd Information Technology to Enhance E-Learning and other Application Conference, IT-ELA 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 148–153. doi: 10.1109/IT-ELA52201.2021.9773615.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

[6] A. O. Lamina, M. F. Yussuf, T. Oyinloye, P. Oladokun, and V. K. Brown, 'Developing a framework for enhancing security testing of android applications', *World Journal of Advanced Research and Reviews*, vol. 23, no. 2, pp. 2585–2598, Aug. 2024, doi: 10.30574/wjarr.2024.23.2.2588.

[7] X. Zhang, W. Shen, Z. Liang, L. Cui, and Y. Wang, 'Research and Application of Automated Testing Technology for Data Security Vulnerabilities', in *2024 4th International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, IEEE, Dec. 2024, pp. 01–05. doi: 10.1109/ICMNWC63764.2024.10872029.

[8] Y. Wang, Z. Wu, Q. Wei, and Q. Wang, 'NeuFuzz: Efficient Fuzzing with Deep Neural Network', *IEEE Access*, vol. 7, pp. 36340–36352, 2019, doi: 10.1109/ACCESS.2019.2903291.

[9] M. A. Schneider, M. F. Wendland, A. Akin, and S. Senturk, 'Fuzzing of Mobile Application in the Banking Domain: A Case Study', in *Proceedings - Companion of the 2020 IEEE 20th International Conference on Software Quality, Reliability, and Security, QRS-C 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 485–491. doi: 10.1109/QRS-C51114.2020.00087.

[10] M. Foley and S. Maffeis, 'APIRL: Deep Reinforcement Learning for REST API Fuzzing', London, 2025. Accessed: Mar. 02, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2412.15991

[11] F. Gao, Y. Wang, L. Situ, and L. Wang, 'Deep Learning-Based Hybrid Fuzz Testing', *International Journal of Software and Informatics*, vol. 11, no. 3, pp. 335–355, 2021, doi: 10.21655/ijsi.1673-7288.00261.

[12] X. Liang and T. Xiao, 'RLF: Directed Fuzzing based on Deep Reinforcement Learning', in *Proceedings - 2022 International Conference on Machine Learning, Control, and Robotics, MLCR 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 127–133. doi: 10.1109/MLCR57210.2022.00032.

[13] J. Su, H. N. Dai, L. Zhao, Z. Zheng, and X. Luo, 'Effectively Generating Vulnerable Transaction Sequences in Smart Contracts with Reinforcement Learning-guided Fuzzing', in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Sep. 2022. doi: 10.1145/3551349.3560429.

[14] A. Helin, L. J. Gunn, and T. Toosi, 'Efficient fuzzing payload generation for mobile application security testing', Aalto University, 2024. Accessed: May 16, 2025. [Online]. Available: https://aaltodoc.aalto.fi/server/api/core/bitstreams/9791a045-6ee7-4311-9721-cf26facb267b/content

[15] A. Romdhana, A. Merlo, M. Ceccato, and P. Tonella, 'Deep Reinforcement Learning for Black-box Testing of Android Apps', *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 4, Jul. 2022, doi: 10.1145/3502868.

[16] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, 'A design science research methodology for information systems research', *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, Dec. 2007, doi: 10.2753/MIS0742-1222240302.

[17] D. Miessler, J. Haddix, and I. J. P. Portal, 'Seclists'. Accessed: Feb. 24, 2025. [Online]. Available: https://github.com/danielmiessler/SecLists

[18] A. Muntner, 'FuzzDB'. Accessed: Feb. 24, 2025. [Online]. Available: https://github.com/fuzzdb-project/fuzzdb

[19] X. Chen *et al.*, 'WebFuzzAuto: An Automated Fuzz Testing Tool Integrating Reinforcement Learning and Large Language Models for Web Security', in *2024 12th International Conference on Information Systems and Computing Technology, ISCTech 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ISCTech63666.2024.10845318.