Analisis Manfaat Rooting Serta Dampak Penerapan Custom ROM Terhadap Kinerja Perangkat Android

Fatahillah Aswam Qowa'id[1], Agus prihanto[2]

[1,2]Teknik Informatika/Teknik Informatika, Universitas Negeri Surabaya
[1]fatahillah.19068@mhs.unesa.ac.id
[2]agusprihanto@unesa.ac.id

***Abstrak*—Sistem operasi Android bawaan pabrik (Stock ROM) umumnya membatasi akses sistemik dari penggunanya dan sering menyertakan berbagai aplikasi pra-instal (bloatware) yang secara tidak langsung membebani kinerja perangkat keras. Makalah ini menyajikan studi eksperimental komprehensif mengenai efektivitas proses rooting dan implementasi Custom ROM sebagai upaya optimalisasi kinerja, sembari menganalisis risiko keamanan yang menyertainya. Menggunakan metode eksperimen kuantitatif dengan pendekatan *within-subjects design* pada ponsel Xiaomi Redmi Note 9, pengujian membandingkan dua kondisi utama: kelompok kontrol dengan Stock ROM dan kelompok eksperimen yang menjalankan Custom ROM dengan hak akses root aktif. Hasil pengujian menunjukkan peningkatan performa yang sangat signifikan, dibuktikan dengan lonjakan skor total AnTuTu Benchmark sebesar 28,75% dan skor 3DMark sebesar 12,86%. Peningkatan ini diakibatkan oleh efisiensi manajemen memori dan penerapan mode performa pada prosesor. Namun, pemberian hak akses root tersebut terbukti mematikan mekanisme isolasi (sandboxing) bawaan sistem Android. Hal ini ditunjukkan secara nyata melalui kemudahan injeksi memori oleh aplikasi pihak ketiga, yang menegaskan tingginya risiko kerentanan eksploitasi data.**

***Kata Kunci*— Android, Rooting, Custom ROM, Overclocking, Benchmark, Integritas Memori.**

I. PENDAHULUAN

Sistem operasi Android saat ini mendominasi lanskap pasar global berkat arsitekturnya yang berbasis *open-source* dan tingkat fleksibilitas yang tinggi bagi para pengembang maupun pengguna. Namun, dalam praktiknya, mayoritas produsen perangkat keras seluler (OEM) menerapkan batasan sistemik yang sangat ketat sebelum perangkat didistribusikan kepada konsumen akhir. Batasan ini, yang sering kali diimplementasikan melalui penguncian *bootloader* dan pembatasan hak akses administratif lain, pada dasarnya dirancang dengan tujuan mitigasi risiko keamanan dan menjaga stabilitas sistem operasi bawaan pabrik (Stock ROM). Sayangnya, kebijakan keamanan ini sering kali disertai dengan penanaman berbagai aplikasi pra-instal (*bloatware*) serta kustomisasi antarmuka pabrikan yang berjalan terus-menerus di latar belakang sistem. Kondisi tersebut secara langsung membebani alokasi *Random Access Memory* (RAM) dan memonopoli siklus kerja prosesor, yang berujung pada tidak optimalnya potensi perangkat keras, terutama pada perangkat ponsel pintar kelas menengah (*mid-range*).

Seiring dengan perkembangan teknologi perangkat lunak, kebutuhan komputasi pengguna modern terus meningkat secara eksponensial. Hal ini sangat terasa pada sektor hiburan digital, di mana aplikasi kompetitif masa kini seperti *Mobile Legends* menuntut alokasi pemrosesan grafis (GPU) dan prosesor (CPU) yang konsisten dan intensif. Untuk mencapai performa *rendering* yang mulus dengan *frame rate per second* (FPS) yang presisi dan stabil, perangkat membutuhkan ketersediaan sumber daya perangkat keras yang tidak terhalang oleh beban sistem latar belakang maupun pembatasan suhu (*thermal throttling*) yang dikonfigurasi secara ketat oleh pabrikan. Kesenjangan antara kebutuhan komputasi tinggi dan keterbatasan Stock ROM inilah yang mendorong banyak pengguna tingkat lanjut untuk mencari metode optimalisasi alternatif di luar batas pedoman standar pabrik.

Salah satu metode optimalisasi komputasi yang paling radikal dan terbukti efektif adalah melalui proses *rooting* dan instalasi Custom ROM. *Rooting* merupakan proses modifikasi eskalasi hak akses untuk mendapatkan kendali penuh tingkat tertinggi (*superuser*) atas akar subsistem operasi Android. Dengan hak akses administratif yang absolut ini, pengguna dapat memodifikasi partisi sistem, menghapus *bloatware* hingga ke akarnya, serta memasang *kernel* kustom yang memungkinkan terjadinya praktik *overclocking*. Penerapan *overclocking* secara paksa menginstruksikan prosesor untuk bekerja mencapai, atau bahkan melebihi, batas frekuensi maksimal yang ditetapkan pabrik secara konstan (melalui penerapan *performance governor*), sehingga gawai mampu menangani beban komputasi berat dengan latensi yang jauh lebih rendah. Penggantian Stock ROM dengan Custom ROM yang mengusung antarmuka minimalis lebih lanjut menyediakan ruang pita lebar memori yang signifikan bagi aplikasi pengguna.

Akan tetapi, perolehan performa komputasi yang maksimal dan tanpa batas ini membawa konsekuensi sekunder yang amat fatal terhadap integritas keamanan dari sistem operasi itu sendiri. Arsitektur keamanan Android pada dasarnya bertumpu pada mekanisme isolasi aplikasi (*Application Sandboxing*) dan verifikasi tanda tangan digital untuk mencegah suatu program mengakses atau mengintervensi direktori program lain. Ketika hak akses root dibuka bebas, benteng pertahanan fundamental ini seketika runtuh. Modifikasi tingkat lanjut menggunakan perangkat lunak manipulasi pihak ketiga, seperti alat injeksi memori atau pemodifikasi *manifest*, dapat dengan mudah menyusup ke dalam memori virtual yang sedang berjalan untuk mengubah instruksi logika dan nilai autentikasi secara *real-time*. Kondisi kritis ini menjadikan sistem berada pada titik kerentanan tertingginya, di mana tidak ada lagi tembok

penghalang antara aplikasi yang aman dan skrip berbahaya yang berpotensi mengeksploitasi data lokal pengguna.

Berdasarkan latar belakang konseptual tersebut, penelitian ini dilaksanakan untuk menganalisis secara empiris fenomena *trade-off* (kompromi) dua sisi antara peningkatan eksponensial dari kinerja dan degradasi fatal pada keamanan perangkat bersistem operasi Android. Melalui metode pendekatan eksperimental kuantitatif, makalah ini bertujuan untuk mengukur secara presisi tingkat lonjakan performa yang diperoleh dari implementasi Custom ROM, injeksi *root*, dan *overclocking* melalui serangkaian uji *benchmark* sintetis maupun waktu nyata. Di saat yang bersamaan, analisis teknis juga akan dititikberatkan pada bukti nyata tentang bagaimana mekanisme keamanan bawaan sistem dikompromikan tanpa sisa akibat eskalasi hak akses tersebut, dengan harapan dapat menyajikan pandangan akademis yang utuh, berimbang, dan faktual terkait modifikasi sistem operasi Android.

II. TINJAUAN PUSTAKA

A. *Arsitektur Dasar dan Konsep Hak Akses (Rooting)*

Sistem operasi Android memiliki sejarah panjang yang bermula pada tahun 2003, awalnya dirancang oleh perusahaan Android.inc sebagai sistem operasi untuk kamera digital, sebelum akhirnya beralih fokus ke perangkat *smartphone* pada tahun 2004. Setelah diakuisisi oleh Google pada tahun 2005, sistem operasi ini dibangun dengan mendasarkan proyeknya pada inti (kernel) Linux. Android kemudian dipromosikan sebagai sistem operasi *open-source* gratis di bawah naungan Open Handset Alliance pada tahun 2007. Sebagai turunan modifikasi dari sistem operasi Linux, Android memiliki struktur perizinan hirarkis.

Dalam ekosistem ini, istilah "root" mengacu pada profil administrator atau tingkat akses tertinggi yang diwarisi dari sistem operasi Unix/Linux. Pada kondisi bawaan pabrik, produsen mengunci hak akses root ini demi alasan keamanan pengguna. Melakukan eskalasi hak akses (*rooting*) memungkinkan pengguna bertransformasi menjadi administrator super yang dapat mengubah konfigurasi sistem, memasang perangkat lunak tanpa batasan, mengelola perizinan pengguna, mengubah berkas apapun, hingga memodifikasi pengaturan keamanan fundamental. Akses root ini memberikan keleluasaan untuk menghapus *bloatware* yang mengganggu, memasang *firmware* kustom, serta mendongkrak performa prosesor (*overclocking*). Namun, hak akses absolut ini layaknya pedang bermata dua; ia dapat dimanfaatkan oleh individu tidak bertanggung jawab untuk melakukan manipulasi data (*cheating*) maupun peretasan dengan mencuri *cookies* dalam jaringan nirkabel yang sama.

B. *Custom Recovery, Magisk, dan Custom ROM*

Untuk menerapkan modifikasi pada tingkat sistem partisi Android, pengguna membutuhkan jembatan berupa *Custom Recovery*. *Custom Recovery* adalah sebuah sistem operasi mini yang menggantikan sistem *recovery* bawaan pabrik (Stock Recovery), yang diciptakan melalui kolaborasi komunitas pengembang Android seperti Koushik Dutta (pencipta ClockworkMod/CWM) dan tim pengembangan TWRP (Team Win Recovery Project). Alat ini jauh lebih fleksibel, mengizinkan instalasi perangkat lunak (termasuk Custom ROM), proses *rooting*, serta *backup* dan *restore* seluruh memori sistem perangkat.

Setelah Custom Recovery terpasang, pengguna dapat mengganti sistem operasinya dengan Custom ROM. Custom ROM merupakan sistem operasi turunan Android yang telah dimodifikasi oleh pengembang pihak ketiga. Keunggulan Custom ROM meliputi performa dan konsumsi daya baterai yang lebih dioptimalkan, penambahan fitur kustomisasi tema, dan kebebasan mutlak dari perangkat lunak bawaan (*bloatware*), sehingga perangkat menjadi jauh lebih ringan. Lebih dari itu, Custom ROM memungkinkan perangkat yang telah dihentikan pembaruannya oleh pabrikan untuk tetap mencicipi versi Android terkini.

Pada sistem yang telah dimodifikasi ini, akses *superuser* wajib dikendalikan menggunakan pengelola perizinan, seperti aplikasi SuperSU atau Magisk. Perangkat lunak ini bertindak layaknya penjaga pintu sistem, memantau secara ketat dan memberikan notifikasi otorisasi kepada pengguna ketika ada sebuah aplikasi yang mencoba meminta izin akses *root*. Tanpa adanya aplikasi pengelola ini, sistem operasi yang telah di-*root* secara otomatis akan menolak secara mutlak semua instruksi izin ke *kernel*.

C. *Kerentanan Integritas Memori dan Sandboxing*

Dalam mengevaluasi dampak *rooting* terhadap keamanan, penting untuk memahami mekanisme *Application Sandboxing* bawaan Android 10, di mana setiap aplikasi dikunci dengan *User ID* (UID) unik yang mencegah aplikasi tersebut membaca alokasi RAM milik aplikasi lainnya. Akan tetapi, batasan keamanan isolasi ini otomatis runtuh apabila sebuah aplikasi diberikan izin *superuser* melalui Magisk.

Aplikasi seperti Game Guardian memanifestasikan potensi peretasan akibat akses *root*. Alat ini mengakses direktori sistem inti Linux (seperti /proc/[PID]/mem) untuk memindai ruang memori virtual secara *real-time*, menyaring pencarian bilangan bulat (integer) atau desimal (float), dan melakukan isolasi alamat heksadesimal. Menggunakan *system calls* kernel Linux seperti *ptrace*, alat ini dapat menyuntikkan (menimpa) data baru secara paksa. Lebih jauh, fitur *Speedhack* bekerja dengan mencegat fungsi *system calls* penunjuk waktu (*clock_gettime*), memanipulasi rentang waktu (*delta time*) agar aplikasi target melipatgandakan kecepatan siklus logikanya. Fitur *Unrandomizer* pada alat tersebut bahkan mampu mencari dan menimpa algoritma generator angka acak (*Pseudo-Random Number Generator*) menjadi nilai statis yang konstan.

Sebagai tambahan eksploitasi statis, perangkat lunak seperti Lucky Patcher membedah kerentanan lapisan aplikasi dengan memanfaatkan akses *root* untuk meretas hak baca-tulis (R/W) di direktori partisi /data/app/. Ia memodifikasi *bytecode* dalam fail *classes.dex* yang berisi logika operasional, menghancurkan protokol keamanan *Signature Verification* sistem dengan

melakukan *patching* pada fungsi *framework.jar*, dan mengelabui otentikasi lisensi API *Google Play* dengan metode *API Hooking*. Alat ini juga mampu mengedit isi kontrak *AndroidManifest.xml* untuk memanipulasi izin spesifik dari sebuah aplikasi tanpa disadari oleh penggunanya.

D. *Kajian Penelitian Terdahulu*

Tinjauan literatur didukung oleh tiga penelitian akademik terdahulu yang relevan. Penelitian pertama oleh Fajar Shodiq (2021) mengkaji integritas data *forensic* pasca-*rooting*, menggunakan metodologi perbandingan nilai *hash* (algoritma MD5 dan SHA1) partisi pengguna. Shodiq membuktikan bahwa akses *rooting* melalui perantara Custom Recovery (TWRP) tetap mampu menjaga orisinalitas bukti digital pada partisi data.

Penelitian kedua oleh Basthanda dkk. (2024) memvalidasi kinerja pengoperasian Custom ROM (Evolution X 7.3) sebagai pengganti MIUI (Stock ROM) pada gawai POCO F3. Melalui penambahan modul pembaru *driver* GPU dan penyesuaian termal *Magisk*, penelitian ini secara eksperimental mengkonfirmasi bahwa pengaplikasian modifikasi sistem Custom ROM sangat jitu dalam mengekstrak potensi sirkuit terpadu (*chipset*) dan berhasil mereduksi ralat (*bug*) dari OS pabrikan.

Penelitian ketiga dari Asrori dkk. (2020) meneliti bahaya infeksi *malware* berbasis hak akses *root* menggunakan teknik perpaduan analisis statis dan dinamis. Secara statis, peneliti membongkar *file* APK dan menemukan bahwa *malware* meminta izin tidak wajar untuk menginstal paket atau mengubah pengaturan di *AndroidManifest.xml*. Dari segi pemantauan lingkungan isolasi (*sandbox*) dinamis, *malware* tersebut terbukti menjalankan *shell script* eksploitasi untuk membobol celah pertahanan pada kernel Linux, memuluskan eskalasi paksa ke tingkat *superuser* agar dapat mengirimkan data curian korban langsung ke peladen *Command and Control* (C&C).

III. *METODE PENELITIAN*

A. *Desain dan Subjek Penelitian*

Penelitian ini menggunakan metode eksperimen kuantitatif dengan mengadopsi pendekatan *Within-Subjects Design* atau *Repeated Measures*. Pemilihan desain ini didasarkan pada prinsip di mana subjek eksperimen yang sama akan mengalami semua kondisi atau perlakuan pengujian secara berulang. Penggunaan satu perangkat yang sama sangat krusial untuk mengeliminasi bias yang mungkin timbul akibat variasi silikon atau degradasi perangkat keras, sehingga seluruh perbedaan hasil yang tercatat murni berasal dari intervensi perangkat lunak. Subjek perangkat keras yang digunakan secara konsisten dalam penelitian ini adalah ponsel pintar Xiaomi Redmi Note 9 (dengan nama kode: Merlin). Sementara itu, subjek perangkat lunak yang diuji merupakan perbandingan langsung antara *firmware* bawaan pabrik (Stock ROM) dengan sistem yang dimodifikasi (Custom ROM).

B. *Variabel Penelitian*

Untuk mengukur dampak modifikasi sistem secara presisi, arsitektur pengujian dikategorikan ke dalam tiga variabel utama:

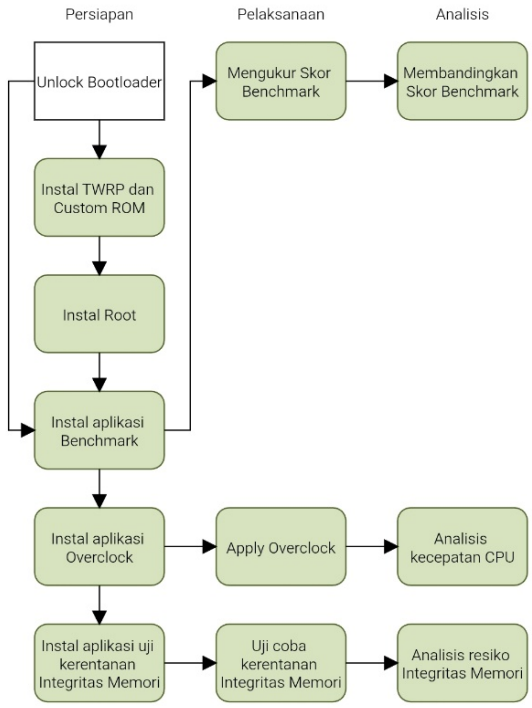
- *Variabel Independen (Status Root dan Sistem)*: Merupakan kondisi sistem yang dimanipulasi oleh peneliti. Terdapat dua kondisi utama, yaitu kondisi standar di mana perangkat Android berjalan pada pengaturan pabrikan, dan kondisi modifikasi di mana perangkat telah diinstal Custom ROM serta berhasil mendapatkan hak akses *root* secara penuh.
- *Variabel Dependen (Kinerja dan Keamanan)*: Merupakan hasil atau efek yang diukur akibat perubahan variabel independen. Variabel ini mencakup tingkat kinerja komputasi murni dari perangkat keras, serta kemampuan fungsional perangkat saat menjalankan aplikasi khusus yang menuntut otorisasi *superuser* (eksploitasi keamanan).
- *Variabel Kontrol*: Variabel yang dipertahankan konstan untuk memastikan validitas komparasi. Variabel kontrol meliputi penggunaan model perangkat keras yang identik (Xiaomi Redmi Note 9) dan versi basis sistem operasi yang setara, yakni Android versi 10.

C. *Instrumen Pengujian*

Penelitian ini mengandalkan serangkaian perangkat lunak analitik untuk merekam dan memodifikasi data. Pengukuran performa mentah komputasi didokumentasikan menggunakan instrumen aplikasi *benchmark* sintetis seperti AnTuTu Benchmark dan 3DMark. Pemantauan aktivitas dan modifikasi parameter *kernel* (termasuk praktik *overclocking*) diimplementasikan melalui aplikasi Kernel Adiutor dan CPU Monitor. Sedangkan untuk pengujian kerentanan arsitektur isolasi sistem dan integritas memori, peneliti mengeksekusi skenario injeksi menggunakan aplikasi Game Guardian dan Lucky Patcher, di bawah pengawasan manajer *systemless root*, yakni Magisk.

D. *Prosedur Pelaksanaan Eksperimen*

Alur eksperimen dalam penelitian ini dirancang secara sistematis melalui tiga tahapan terstruktur yang berjalan sekuensial:



Gbr. 1 Alur penelitian

- *Tahap Persiapan Sistem:* Fase awal ini difokuskan pada penyiapan lingkungan perangkat keras. Proses diawali dengan membuka kunci keamanan pabrikan (*Unlock Bootloader*). Setelah terbuka, alur eksperimen bercabang: sebagian langsung diinstal aplikasi *benchmark* guna merekam data acuan performa dasar (*baseline*) pada Stock ROM. Alur utama berlanjut pada modifikasi mendalam, yakni dengan melakukan instalasi (flashing) lingkungan Custom Recovery (TWRP), instalasi Custom ROM berbasis *Android Open Source Project* (AOSP), dan menanamkan akses *root* via Magisk. Tahap ini diakhiri dengan memasang seluruh aplikasi uji *benchmark*, instrumen *overclocking*, dan alat uji kerentanan memori.

- *Tahap Pelaksanaan Eksperimen*: Pengujian inti mulai dioperasikan pada tahap ini. Langkah pertama adalah perekaman skor kuantitatif *benchmark* dari kinerja aktual CPU, GPU, dan alokasi memori RAM pasca-modifikasi. Peneliti kemudian menerapkan pengaturan spesifik pada aplikasi kontrol *kernel*, secara spesifik mengubah *CPU Governor* ke mode *Performance* untuk memaksa sirkuit perangkat keras beroperasi pada ambang batas frekuensi maksimalnya (*Apply Overclock*). Setelah itu, dilaksanakan skenario *penetration testing* menggunakan izin akses *root* untuk memindai ruang operasi serta menyuntikkan (injeksi) modifikasi variabel fiktif langsung ke dalam alamat memori (RAM) aktif secara *real-time*.

E. *Teknik Analisis Data*

Tahap pengujian diakhiri dengan evaluasi komprehensif terhadap himpunan data primer yang berhasil direkam pada fase pelaksanaan. Analisis dilakukan dengan metode komparatif pada skor *benchmark*, membandingkan angka performa Stock ROM dan Custom ROM untuk menghitung persentase peningkatannya secara matematis. Peneliti juga melakukan observasi perilaku kecepatan *clock CPU* dan penanganan stabilitas termalnya (*thermal throttling*). Di ranah keamanan, data hasil uji coba penetrasi dibedah untuk mengevaluasi dampak langsung dari hak eskalasi *superuser* terhadap runtuhnya sistem perlindungan keamanan isolasi (*sandbox*) Android, serta seberapa dalam tingkat kerentanan gawai terhadap injeksi instruksi data berbahaya.

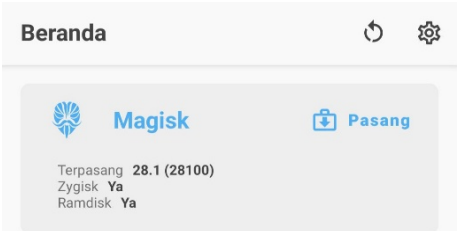
IV. *HASIL DAN PEMBAHASAN*

A. *Hasil Implementasi Sistem Tingkat Lanjut*

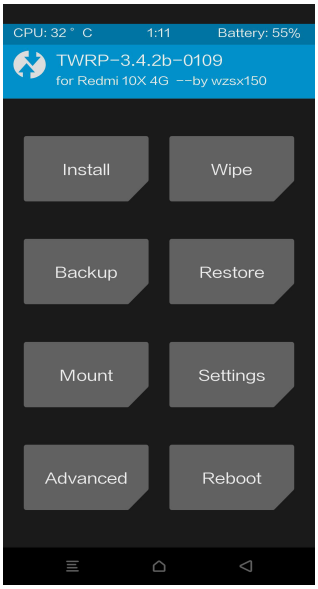
Proses implementasi diawali dengan membuka kunci *bootloader* perangkat keras Xiaomi Redmi Note 9 (Merlin) menggunakan perangkat lunak Mi Flash Unlock. Langkah ini mensyaratkan aktivasi opsi "OEM Unlocking" dan "USB Debugging" pada menu Opsi Pengembang (Developer Options). Status *bootloader* yang terbuka (*unlocked*) merupakan prasyarat mutlak yang mengizinkan penulisan ulang (flashing) pada partisi sistem pabrikan.

Setelah partisi terbuka, sistem pemulihan bawaan diganti dengan Custom Recovery. Pemasangan dilakukan melalui antarmuka baris perintah (command line) menggunakan Minimal ADB Fastboot dengan mengeksekusi perintah fastboot flash recovery twrp.img untuk menanamkan TWRP versi 3.4.2b. Melalui lingkungan TWRP inilah Stock ROM bawaan Xiaomi (MIUI) dihapus total (wipe) dan digantikan oleh Custom ROM Nusantara Project V2.8 varian Gapps. Transisi ke sistem operasi berbasis *Android Open Source Project* (AOSP) ini menghasilkan antarmuka yang sangat bersih dan menghilangkan puluhan aplikasi latar belakang (*bloatware*) bawaan pabrik. Hak akses administrator kemudian dibuka menggunakan skrip *systemless root* (Magisk). Pemilihan Magisk didasarkan pada kemampuannya memodifikasi sistem tanpa menyentuh partisi sistem secara langsung, serta hadirnya fitur *Magisk Hide* untuk menyembunyikan status *root* dari deteksi aplikasi perbankan.

Gbr. 2 Bootloader Terbuka



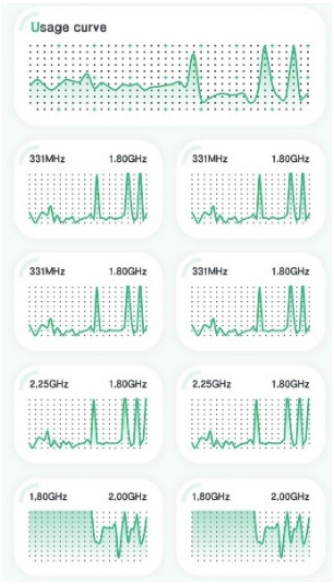
Gbr. 3 Status magisk (rooted)



Gbr. 4 Antarmuka custom recovery TWRP

B. *Konfigurasi Overclocking dan Pengendalian Beban Kerja*

Tahap krusial dalam mendongkrak performa adalah penerapan *overclocking* yang diatur melalui aplikasi Kernel Adiutor dengan akses *root*. Pada kondisi Stock ROM standar, *CPU Governor* beroperasi pada mode efisiensi (*interactive/schedutil*), di mana frekuensi prosesor berfluktuasi sangat rendah dari 224 MHz hingga batas 1.8 GHz, tergantung pada beban aplikasi untuk menghemat daya. Melalui modifikasi *kernel*, *CPU Governor* dipaksa beralih ke mode *Performance*. Konfigurasi ini mengunci frekuensi aktif prosesor agar terus bekerja pada ambang batas puncaknya secara konstan, yakni 1.8 GHz hingga 2.0 GHz pada *big cluster* CPU

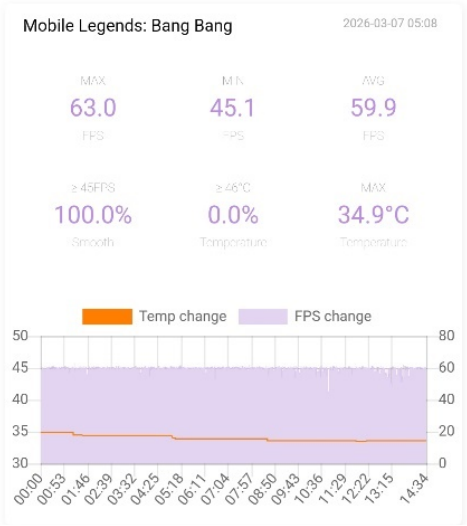


Gbr. 5 Grafik CPU sebelum overclock

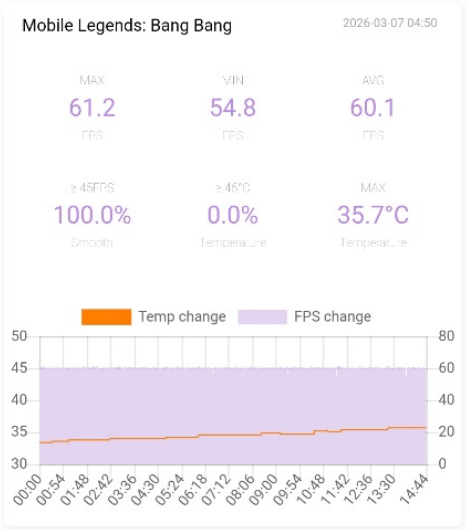
Gbr. 6 Grafik CPU sesudah overclock

Untuk membuktikan dampak stabilitas konfigurasi ini di skenario dunia nyata, peneliti menginisiasi pengujian pengendalian beban kerja menggunakan fitur pemutaran ulang (*replay*) pada aplikasi permainan *Mobile Legends: Bang Bang*. Pemilihan metode *replay* memastikan bahwa sirkuit CPU dan GPU memproses beban kerja matematis dan *rendering* instruksi visual yang seratus persen identik antara kondisi standar dan kondisi *overclock*. Hasil rekaman metrik menunjukkan peningkatan stabilitas *Frame Rate Per Second*

(FPS) dari rata-rata 59.9 menjadi 60.1 FPS. Kendati peningkatannya tampak minor secara angka absolut, fluktuasi *frame drop* (penurunan drastis FPS) berhasil dieliminasi. Namun, modifikasi ini menghasilkan kompromi termal, di mana pemrosesan maksimal yang konstan memicu peningkatan suhu operasional perangkat secara linear.



Gbr. 7 Grafik FPS sebelum overclock



Gbr. 8 Grafik FPS sesudah overclock

C. *Haasil Pengukuran Komputasi dan Benchmark Sintetis*

Pengukuran komparatif kemampuan komputasi perangkat lunak dianalisis menggunakan aplikasi *benchmark* sintetis terkemuka. Hasil pengukuran menunjukkan lonjakan performa yang amat signifikan dan konsisten di seluruh lini pengujian keras:

- *Lonjakan Skor Keseluruhan AnTuTu:* Kalkulasi akhir dari pengujian AnTuTu Benchmark menunjukkan peningkatan eksponensial sebesar 28,75%, melonjak dari skor awal 254.943 pada kondisi pabrikan menjadi 328.248 pasca-modifikasi.
- *Kinerja CPU dan GPU:* Peningkatan arsitektur pemrosesan menyumbang kenaikan skor CPU sebesar 32,16% (109.745 menjadi 145.043). Lonjakan ini ditopang oleh kapabilitas *Multi-Core* dan Operasi Matematika CPU yang kini dieksekusi tanpa interupsi *thermal throttling* bawaan. Di sektor grafis, skor GPU naik sebesar 25,38% (dari 22.513 ke 28.227), selaras dengan hasil uji aplikasi 3DMark yang mencatat apresiasi skor keseluruhan dari 1.150 menjadi 1.298 (naik 12,86%).
- *Efisiensi Manajemen Memori dan UX*: Menghilangnya *bloatware* bawaan MIUI secara dramatis membebaskan alokasi pita lebar (bandwidth) memori. Skor komponen Memori (MEM) melesat 26,64% menjadi 74.834, yang mengindikasikan tingkat latensi RAM yang jauh lebih responsif. Kondisi ini secara langsung mendongkrak pengalaman pengguna atau *User Experience* (UX) sebesar 26,02%, membuktikan bahwa perangkat menengah mampu melampaui kelasnya jika dibebaskan dari restriksi pabrik

Tabel I
Skor partial AnTuTu

Part	Process	Score	
		Sebelum	Sesudah
Cpu	Operasi Matematika	27733	36978
	Algoritma Umum	22156	29941
	Multi-Core	56312	73132
	Ai	3544	4992
Gpu	Marooned OpenGL ES 3.0	13363	16497
	Coastline OpenGL ES 3.0	9150	11730
Memory	Bandwidth RAM	13498	16872
	Latensi RAM	23140	28220
	IO Aplikasi ROM	10271	13695
	Pembacaan Sekuensial ROM	1636	2241
	Penulisan Sekuensial ROM	1101	1448
	Akses Acak ROM	1620	2050
	Campuran Akses Multi-Acak ROM	192	246
	Campuran Akses Acak ROM	3349	4651
	ROM AI	766	957
	Kinerja Multi-utas ROM AI	3519	4454
Ux	Pengalaman Pengguna	18790	24419
	Keamanan Data	14565	17981
	Pemrosesan Data	3874	5235
	Pemrosesan Dokumen	18149	21867
	Penguraian Gambar	3354	4533
	Pemrosesan Gambar	578	792
	UX CTS	2100	2500
	Dekode UX	1118	1452
	Edisi UX	1065	1365

Tabel II
Skor total AnTuTu

	SEBELUM	SESUDAH
CPU	109745	145043
GPU	22513	28227
MEMORY	59092	74834
UX	63593	80144
TOTAL SCORE	254943	328248

Tabel III
Skor 3dMark

Score		**Sebelum**	**Sesudah**
Graphic	Test 1	8,21	9,22
	Test 2	3,99	4,55
Graphic Score		1242	1402
Physics	Test 1	17,41	19,61
	Test 2	9,96	11,29
	Test 3	4,88	5,49
Physics score		914	1031
Overall score		1150	1298

D. *Analisis Risiko Keamanan dan Integritas Memori*

Pencapaian performa yang menembus batas fabrikasi ini harus dibayar mahal dengan runtuhnya tembok pertahanan sistem isolasi atau *Application Sandboxing* bawaan Android 10. Pengujian *penetration testing* pada lingkungan yang telah di-*root* memvalidasi hipotesis kerentanan integritas memori tingkat tinggi.

Menggunakan alat eksekutor memori Game Guardian, aplikasi ini terbukti mampu menembus sistem keamanan inti dengan mengakses direktori partisi /proc/[PID]/mem. Dalam kondisi standar, mekanisme perlindungan *sandbox* akan secara otomatis memblokir aplikasi "A" untuk membaca ruang memori RAM yang dialokasikan bagi aplikasi "B". Namun, dengan izin *superuser*, alat ini sukses memindai dan menyuntikkan (injeksi) rentetan nilai *integer* baru langsung ke dalam alamat memori (RAM) pada aplikasi simulasi *Minecraft* secara *real-time*. Keberhasilan mengubah variabel inventori dan logika waktu ini menunjukkan bahwa *malware* apapun yang berhasil meretas akses *root* dapat dengan mudah mencuri atau memanipulasi parameter token autentikasi di dalam RAM tanpa terdeteksi.

Eksploitasi keamanan ini diperparah oleh pembuktian modifikasi statis menggunakan utilitas Lucky Patcher. Melalui eksploitasi hak baca/tulis (R/W), utilitas ini terbukti sanggup membongkar direktori /data/app/, mendekompilasi struktur paket (APK), dan menimpa *bytecode* dalam fail *classes.dex*. Lebih fatal lagi, perangkat ini menghancurkan perlindungan *Signature Verification* dengan melakukan penambalan (*patching*) pada *framework.jar* Android, serta mengubah struktur perizinan krusial di *AndroidManifest.xml*. Fakta lapangan ini secara absolut mendemonstrasikan bahwa gawai yang beroperasi dalam mode *root* kehilangan seluruh sertifikasi digital aslinya, menjadikannya lahan yang sangat rentan terhadap penyisipan instruksi *backdoor* tak kasat mata.

V. KESIMPULAN

Berdasarkan serangkaian pengujian eksperimental dan analisis komprehensif yang telah dilakukan terhadap implementasi Custom ROM Nusantara Project serta modifikasi hak akses *root* pada perangkat Xiaomi Redmi Note 9, penelitian ini menghasilkan simpulan yang tegas terkait fenomena pertukaran (*trade-off*) antara optimalisasi performa dan degradasi keamanan sistem.

Pertama, dari perspektif komputasi, modifikasi perangkat lunak tingkat sistem secara empiris terbukti mampu mengekstrak potensi maksimal dari perangkat keras keras secara signifikan. Hal ini divalidasi melalui lonjakan performa pada pengujian *benchmark* sintetis, di mana skor total AnTuTu meningkat pesat sebesar 28,75% (dari 254.943 menjadi 328.248), serta kenaikan skor kapabilitas pemrosesan grafis 3DMark sebesar 12,86%. Peningkatan masif ini dikontribusikan oleh dua faktor teknis utama: efisiensi manajemen pita lebar RAM yang drastis akibat penghapusan aplikasi latar belakang bawaan pabrik (*debloating*), serta modifikasi parameter *kernel* yang memaksa prosesor bekerja stabil pada frekuensi puncaknya melalui penerapan mode *Performance governor* (*overclocking*).

Kedua, dari perspektif keamanan informasi, perolehan performa ekstrem tersebut harus dibayar dengan hancurnya arsitektur keamanan fundamental Android. Pemberian hak akses *superuser* secara absolut melumpuhkan mekanisme perlindungan isolasi aplikasi (*Application Sandboxing*). Pengujian eksploitasi mengonfirmasi bahwa utilitas pihak ketiga seperti Game Guardian dapat dengan mudah melakukan intersepsi direktori dan injeksi nilai memori secara *real-time* pada aplikasi target yang sedang aktif beroperasi. Lebih lanjut, alat manipulasi statis seperti Lucky Patcher secara absolut membuktikan hilangnya integritas verifikasi tanda tangan digital (*signature verification*), yang memungkinkan perombakan struktur *bytecode* aplikasi dan manipulasi perizinan *AndroidManifest.xml* secara lokal. Kondisi tanpa pelindung ini menempatkan perangkat pada tingkat kerentanan paling tinggi terhadap infeksi *malware* eksploitatif dan potensi pencurian data.

Sebagai konklusi akhir, terdapat kompromi yang sangat jelas dan tidak dapat dihindari antara akselerasi kinerja dan kerentanan sistem. Praktik modifikasi *rooting* dan pemasangan Custom ROM sangat direkomendasikan secara eksklusif untuk tujuan merevitalisasi performa gawai usang atau difungsikan murni sebagai perangkat hobi operasional (*gaming* kompetitif). Pengguna sangat diimbau untuk tidak mengaplikasikan modifikasi sistem terbuka ini pada gawai komunikasi utama (*daily driver*) yang memuat kredensial krusial, aplikasi

perbankan digital, maupun aset dompet elektronik, demi mencegah eskalasi serangan fatal terhadap privasi data dan keamanan finansial.

REFERENSI

[1] Android Open Source Project. (n.d.). Android Source Code. Google. Diakses dari https://source.android.com.

[2] Android Open Source Project. (n.d.). System and Kernel Security. Google. Diakses dari https://source.android.com

[3] Asrori, N., Riadi, I., & Prayudi, Y. (2020). Analisis Perilaku Malware Android Untuk Memperoleh Hak Akses Root Menggunakan Metode Analisis Statis Dan Dinamis. Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), 4(4), 784–791. https://doi.org/10.29207/resti.v4i4.2255.

[4] Basthanda, F. D., Solehudin, A., & Purwantoro, P. (2024). Analisis Optimasi Performa Perangkat Android dengan Modifikasi Custom ROM. Jurnal Ilmiah Wahana Pendidikan, 10(13), 198-209. https://zenodo.org/records/12741652.

[5] Elgin, B. (2005, 17 Agustus). Google Buys Android for Its Mobile Arsenal. Bloomberg. Diakses dari https://www.bloomberg.com.

[6] Hoog, A. (2011). Android Forensics: Investigation, Analysis and Mobile Security for Google Android. Waltham, MA: Elsevier.

[7] LineageOS Project. (n.d.). About LineageOS. Diakses dari https://lineageos.org.

[8] Shodiq, F. (2021). Analisis Rooting pada Perangkat Android [Skripsi, Universitas Telkom]. Repository Universitas Telkom. https://repository.telkomuniversity.ac.id/pustaka/164264/analisis-rooting-pada-perangkat-android.html.

[9] XDA Developers. (n.d.). All About Custom ROMs. Diakses dari https://www.xda-developers.com.

[10] XDA Developers. (n.d.). Rooting Guide. Diakses dari https://www.xda-developers.com.