BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil implementasi, pengujian, dan analisis yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut:

1. Sebuah *library* keamanan *android* yang canggih dan efektif telah berhasil dirancang dan diimplementasikan. Dengan menggunakan pendekatan berlapis (*multi-layered*) yang menggabungkan analisis statis, *heuristik*, dan pemanfaatan kode *native* (C++), *library* ini tidak hanya fungsional tetapi juga selaras dengan standar keamanan global OWASP MASVS. Keberhasilannya divalidasi melalui pengujian *white box*, pengujian *black box* dengan tingkat keberhasilan 100% pada skenario yang dirancang, serta penilaian ahli dengan koefisien *Aiken's V* sebesar 0.75 yang menunjukkan validitas isi yang tinggi.
2. Penelitian ini membuktikan bahwa pendekatan yang digunakan sangat efektif dalam menembus mekanisme *root hiding* modern. Pengujian secara spesifik menunjukkan bahwa *library* mampu mendeteksi status *root* pada perangkat yang menggunakan *Magisk Hide, Zygisk*, dan bahkan *Shamiko*. Temuan ini menjawab rumusan masalah mengenai kemampuan *library* dalam menghadapi tantangan keamanan yang relevan saat ini.
3. Penelitian ini juga berhasil memetakan batasan dan tantangan keamanan generasi berikutnya. Kelemahan teridentifikasi bukan lagi pada metode *root hiding* umum, melainkan pada *framework hooking* canggih seperti *Lsposed* beserta modul-modulnya (contoh: *Hide My Applist*). Modul-modul ini bekerja

dengan memanipulasi respons *API* pada *level* yang lebih dalam, yang menjadi tantangan utama selanjutnya.

Dengan demikian, rumusan masalah telah terjawab: *library* keamanan telah berhasil diimplementasikan sebagai solusi yang kuat dan tervalidasi, yang bahkan mampu mengatasi teknik penghindaran deteksi modern. Sekaligus, penelitian ini berhasil mendefinisikan peta jalan yang sangat jelas untuk evolusi selanjutnya guna menghadapi ancaman yang lebih canggih.

5.2 Saran

Saran yang diajukan terbagi menjadi dua kategori, saran untuk pengembangan *library* secara spesifik dan saran untuk penelitian lebih lanjut di bidang ilmu terkait.

5.2.1 Saran untuk Pengembangan Proyek

1. Mengatasi Lsposed dengan Deteksi Anomali *Runtime*

Mengingat tantangan utama adalah *hooking framework* Lsposed, disarankan untuk mengembangkan modul baru yang berfokus pada analisis perilaku (*behavioral analysis*). Ini dapat mencakup pendeteksian anomali pada pemanggilan fungsi sistem kritis atau memeriksa *memory map* proses untuk mengidentifikasi jejak dari *framework hooking* seperti Frida atau Xposed/Lsposed.

2. Meningkatkan Resiliensi Internal (*Self-Protection*)

Untuk meningkatkan ketahanan *library* terhadap rekayasa balik, disarankan untuk menerapkan teknik *obfuscation* yang lebih agresif menggunakan ProGuard/R8. Selain itu, dapat dikembangkan mekanisme *anti-tampering* yang

memeriksa integritas kode *library* itu sendiri saat *runtime* untuk memastikan tidak ada modifikasi oleh penyerang.

3. Meningkatkan Adaptabilitas Terhadap Ancaman Baru

Untuk mengatasi kelemahan daftar ancaman yang statis, sangat disarankan untuk membangun sistem konfigurasi jarak jauh (*remote config*). Hal ini memungkinkan untuk mengunduh daftar jejak ancaman terbaru (misalnya, nama paket atau *signature* file baru) dari server secara berkala, sehingga *library* dapat beradaptasi tanpa perlu merilis ulang aplikasi utama.

5.2.2 Saran untuk Penelitian Selanjutnya

1. Pemanfaatan *Machine Learning* untuk Deteksi Anomali

Penelitian selanjutnya dapat mengeksplorasi penggunaan model *machine learning* yang ditanamkan di dalam aplikasi. Model ini dapat dilatih untuk mengenali pola normal dari penggunaan CPU, memori, dan panggilan sistem, sehingga dapat mendeteksi anomali yang disebabkan oleh *hooking* atau *malware* (termasuk *zero-day*) tanpa bergantung pada *signature* yang telah diketahui..

2. Eksplorasi Deteksi Berbasis Fitur Perangkat Keras (*Hardware-Assisted Detection*)

Untuk tingkat keamanan tertinggi, penelitian masa depan dapat berfokus pada pemanfaatan fitur keamanan yang tertanam di perangkat keras, seperti ARM TrustZone atau *Trusted Execution Environment* (TEE). Pemeriksaan keamanan

yang dijalankan di dalam lingkungan terisolasi ini secara teoritis jauh lebih sulit untuk dimanipulasi oleh ancaman yang berjalan di sistem operasi utama..

3. Studi Komparatif Kinerja dan Dampak *User Experience* (UX)

Disarankan untuk melakukan studi penelitian yang mengukur secara kuantitatif dampak dari penerapan berbagai lapisan keamanan (seperti) terhadap kinerja aplikasi (konsumsi baterai, latensi) dan pengalaman pengguna. Hasil studi ini dapat memberikan panduan bagi para pengembang dalam menyeimbangkan antara tingkat keamanan yang diinginkan dengan performa dan kenyamanan pengguna.