BAB V
PENUTUP

5.1 Kesimpulan

1. Berdasarkan hasil pengujian yang dilakukan menggunakan metode fuzzing dengan AFL++, ditemukan sebanyak 76 file crash, di antaranya 6 file crash bersifat unik dan menunjukkan potensi kerentanan pada aplikasi target.

2. Analisis lanjutan terhadap hasil crash menunjukkan bahwa jenis kerentanan adalah Stack Buffer Overflow, yang disebabkan oleh tidak adanya pembatasan ukuran input yang diterima oleh aplikasi. Kondisi ini memungkinkan input yang melebihi kapasitas buffer untuk mengganggu alur eksekusi program, bahkan berpotensi menyebabkan Arbitrary Code Execution, yang merupakan ancaman serius terhadap keamanan sistem. Namun, hasil crash ini masih belum bisa dikatakan sebuah kerentanan dan baru dikatakan sebuah bug, karena belum terdapat bukti bahwa kondisi tersebut dapat dilakukan eksploitasi.

3. Untuk memitigasi risiko tersebut, perlu diterapkan mekanisme validasi dan sanitasi input yang efektif, khususnya dalam membatasi panjang data yang diterima sesuai kapasitas buffer. Dengan menerapkan langkah-langkah ini, kemungkinan terjadinya buffer overflow dapat diminimalisasi, sehingga meningkatkan keamanan dan stabilitas perangkat lunak secara menyeluruh.

5.2 Saran

1. Penelitian selanjutnya sebaiknya dapat mengkombinasikan AFL++ dengan teknik fuzzing lain seperti grammar based fuzzing serta generation based fuzzing sehingga dapat membantu meningkatkan kualitas eksplorasi jalur

eksekusi, serta menemukan kerentanan yang lebih mendalam dan kompleks.

2. Disarankan untuk menggunakan seed corpus yang lebih beragam sehingga proses fuzzing dapat mengeksplorasi lebih banyak jalur eksekusi pada program target. Keanekaragaman input awal memungkinkan fuzzer untuk menemukan kerentanan yang tersembunyi dalam berbagai kondisi input, serta meningkatkan cakupan kode secara keseluruhan.

3. Pertimbangkan menggunakan perangkat lunak fuzzer lain seperti Radamsa, LibFuzzer, atau Honggfuzz sebagai pembanding atau pelengkap dari AFL++. Setiap fuzzer memiliki pendekatannya masing-masing, sehingga kombinasi atau perbandingan hasil dari beberapa fuzzer dapat memberikan wawasan yang lebih menyeluruh mengenai ketahanan aplikasi terhadap berbagai jenis input yang tidak valid.

4. Gunakan beragam opsi yang disediakan oleh perangkat lunak yang akan diuji agar jangkauan kode atau jalur eksekusi dapat dieksplorasi secara maksimal oleh *fuzzer* sehingga dapat benar-benar memastikan bahwa cakupan kode tercakup keseluruhan