BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan sebagai berikut:

1. RASP hanya dapat mendeteksi dan mencegah 2 dari 6 serangan yang disimulasikan dari kategori *client-side* dan *server-side*. Celah keamanan yang mengakibatkan 2 serangan lolos ini dikarenakan adanya *human error*, karena tidak menulis kode sesuai dengan standar keamanan yang sesui *guideline* OWASP MASVS dan dokumentasi dari android. RASP juga tidak melakukan validasi input pada level aplikasi, sehingga semua *payload* akan ditransfer langsung ke API atau *backend server*. Sedangkan WAF hanya dapat melakukan pencegahan jika serangan tersebut sudah dikirimkan ke API atau sudah masuk ke *network layer*.
2. RASP hanya dapat mengamankan pada level aplikasi saja, jika *request* sudah dikirimkan maka WAF akan bekerja untuk memvalidasi *request* tersebut, jadi WAF hanya bekerja pada *backend* atau *network layer pada aplikasi*.
3. Penggunaan WAF untuk mengamankan API masih sangat disarankan, tetapi para sysadmin ataupun DevOps perlu melalukan pembaharuan terhadap rules yang akan digunakan, karena beberapa WAF sudah dapat di *bypass* oleh attacker atau pihak yang tidak bertanggung jawab. Untuk RASP sendiri dapat meminimalisir serangan pada aplikasi, sifatnya yang selalu memperbarui parameter deteksinya akan sangat membantu untuk serangan-serangan yang akan terjadi kedepannya, namun perlu diingat menggunakan RASP bukan berarti aplikasi akan langsung aman dari serangan. Karena pada simulasi serangan, ternyata banyak metode serangan yang tidak dapat dicegah RASP dan

semuanya berasal dari kelalaian *developer* dalam membangun aplikasi, sehingga tetap perlu mengikuti *base security guideline* yang sudah diterapkan oleh OWASP MASVS maupun dokumentasi dari bahasa pemrograman yang digunakan.

5.2 Saran

Untuk mengembangkan penelitian ini agar menjadi lebih baik pada penelitian selanjutnya, penulis memberikan saran sebagai berikut:

1. Menggunakan target APK dengan fitur yang lebih banyak, sehingga dapat menjalankan jenis serangan yang lebih beragam lagi untuk melihat performa dari RASP.
2. Menggunakan versi RASP yang terbaru.
3. Melakukan *reverse engineering* terhadap *core* RASP.
4. Tidak hanya berpatok pada satu product RASP saja