BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan dari pengujian yang sudah dilakukan peneliti dengan judul "*Penetration Testing* Aplikasi *Web* dengan *Open Web Application Security Project* (*OWASP*) pada *Website* Dinas Kominfo Banyumas Melalui *Kali Linux*" adalah sebagai berikut:

1. Berdasarkan pemindaian *vulnerability scanning* menggunakan *nuclei cent templates* dan *zaproxy*, ditemukan dua kerentanan yaitu *Cloud Metadata Potentially Exposed* yang dikategorikan kedalam *OWASP Top 10 A05:2021 Security Misconfiguration* dengan *severity high* dan *SQL Injection Time Based* pada *database PostgreSQL* yang dikategorikan kedalam *OWASP Top 10 – A01:2021 Injection* dengan *severity high.*
2. Hasil pengujian eksploitasi terhadap dua kerentanan yang ditemukan berhasil mengidentifikasi bahwa kerentanan *Cloud Metadata Potentially Exposed* menjadi kerentanan dengan *severity informational* atau *none* berdasarkan *report CVSS* dengan nilai 0.0, sedangkan kerentanan *SQL Injection* teridentifikasi menjadi kerentanan dengan *severity medium* berdasarkan report *CVSS dengan* nilai 4.8. pengujian aplikasi terhadap kerentanan *XSS* juga dilakukan peneliti sehingga menghasilkan kerentanan *XSS* dengan *severity informational* atau *none.*
3. Mitigasi yang direkomendasikan penguji terhadap kerentanan yang ditemukan yaitu kerentanan *SQL Injection Time-Based* berupa penggunaan *prepared statements* atau *parameterized queries*. Ini membuat *query SQL* disiapkan dengan placeholders, dan input pengguna disisipkan setelah *query* dikompilasi. Ini memastikan bahwa input pengguna diperlakukan sebagai data, bukan sebagai bagian dari *query SQL*. Rekomendasi ini dapat mencegah penyerang menyisipkan kode *SQL* karena *input* pengguna tidak akan pernah digabungkan secara langsung ke dalam *query SQL*.

5.2 Saran

Berdasarkan pengujian yang sudah berhasil dilakukan. Ada beberapa saran yang dapat meningkatkan *severity* kerentanan, diantaranya sebagai berikut:

1. Mengidentifikasi konfigurasi server *nginx* yang dilindungi *WAF* dengan melakukan *bypass* terhadap *firewall* agar dapat mengetahui kenapa setiap permintaan yang tidak memenuhi syarat akan dialihkan ke halaman utama *website.*
2. *Bypassing WAF* untuk menghindari mekanisme proteksi *WAF* sehingga *SQLMap* dapat menampilkan *database* yang lebih akurat dan server dapat memberikan respon sesuai dengan *request* dari penguji.
3. Menggunakan *payload XSS* yang lebih kompleks dan lebih banyak, sehingga memungkinkan input dapat melewati proteksi dari *WAF*.