Uji Penetrasi Keamanan *Website* Dinas Komunikasi dan Informatika

Website Security Penetration Test for The Communications and Informatics Office

Rimba Dirgantara[1], Rezki Kurniati[2], Nurmi Hidayasari[3*]
[1,2,3]Jurusan Teknik Informatika, Politeknik Negeri Bengkalis
E-mail: [1]mcsegypt17@gmail.com, [2]rezki@polbeng.ac.id, [3*]nurmihidayasari@polbeng.ac.id
Corresponding author

Abstrak

Website resmi Dinas Komunikasi dan Informatika Kabupaten XYZ memiliki peran penting dalam menyebarkan informasi pemerintah kepada masyarakat, tetapi kerentanannya terhadap serangan seperti *SQL Injection*, *Brute Force*, XSS, CSRF dan DDoS mengancam integritas *data* dan ketersediaan layanan. Penelitian ini bertujuan untuk mengidentifikasi celah keamanan, menerapkan mitigasi dan mengukur tingkat keamanan menggunakan sistem penilaian CVSS. Kebaruan penelitian ini terletak pada integrasi metode mitigasi manual dan otomatis serta penerapan notifikasi *real-time* berbasis *Telegram* menggunakan *Snort* sebagai *Intrusion Detection System* (IDS). Hasil penelitian menunjukkan bahwa mitigasi yang diterapkan, seperti validasi *input*, sanitasi, penggunaan *prepared statements* dan CAPTCHA, mampu menghilangkan keberhasilan serangan *SQL Injection* dan memperkuat perlindungan terhadap potensi serangan *Brute Force*. Serangan XSS, CSRF, dan DDoS tetap gagal berkat proteksi bawaan. Dengan kombinasi mitigasi dan sistem notifikasi *real-time*, penelitian ini memberikan kontribusi nyata dalam meningkatkan keamanan website pemerintah, menyediakan langkah konkret untuk mengurangi risiko serangan, dan mempercepat respons terhadap ancaman. Temuan ini relevan untuk penelitian masa depan dalam pengamanan *website* sektor publik.

Kata kunci: *Penetrasi Testing, SQL Injection, Bruteforce*, XSS, CSRF

Abstract

The official website of the XYZ Regency Communication and Informatics Office plays an important role in disseminating government information to the public, but its vulnerability to attacks such as SQL Injection, Brute Force, XSS, CSRF and DDoS threatens data integrity and service availability. This study aims to identify security gaps, implement mitigations and measure the level of security using the CVSS scoring system. The novelty of this study lies in the integration of manual and automatic mitigation methods and the implementation of real-time notifications based on Telegram using Snort as an Intrusion Detection System (IDS). The results of the study show that the mitigations implemented, such as input validation, sanitization, use of prepared statements and CAPTCHA, are able to eliminate the success of SQL Injection attacks and strengthen protection against potential Brute Force attacks. XSS, CSRF, and DDoS attacks still fail thanks to built-in protection. With the combination of mitigation and a real-time notification system, this study makes a real contribution to improving the security of government websites, providing concrete steps to reduce the risk of attacks, and accelerating the response to threats. These findings are relevant for future research in securing public sector websites

Keywords: Penetrasi Testing, SQL Injection, Bruteforce, XSS, CSRF

1. PENDAHULUAN

Aplikasi berbasis *website* adalah salah satu inovasi teknologi yang berkembang pesat, dibangun dengan menggunakan HTML, PHP, JavaScript, CSS dan basis data untuk memenuhi berbagai kebutuhan informasi dan layanan [1]. Seiring perkembangan teknologi,

ancaman keamanan terhadap aplikasi berbasis website juga semakin kompleks. Serangan seperti *SQL Injection*, *Brute Force*, XSS, CSRF dan DDoS telah menjadi ancaman utama yang dapat membahayakan kerahasiaan, integritas, dan ketersediaan *data*.

SQL Injection adalah teknik yang digunakan penyerang untuk memodifikasi pernyataan *SQL* guna mengeksploitasi komunikasi antara halaman *web* dan basis data [2]. *Brute Force* melibatkan upaya sistematis untuk menebak kredensial *login* menggunakan alat otomatis [3]. XSS memungkinkan penyerang menyisipkan skrip berbahaya yang dapat memanipulasi konten atau mencuri data pengguna [4], sedangkan CSRF memungkinkan penyerang mengirimkan permintaan tidak sah melalui pengguna yang sah tanpa disadari [5]. DDoS di sisi lain, menggunakan sejumlah besar perangkat yang dikendalikan untuk melumpuhkan layanan dengan membanjiri *server target* [6]. Untuk mendeteksi dan mencegah serangan ini, perangkat lunak seperti *Snort*, sebuah sistem deteksi intrusi *open-source* yang dapat digunakan untuk mengenali pola serangan berdasarkan aturan yang telah ditentukan [7].

Website resmi Dinas Komunikasi dan Informatika Kabupaten XYZ memainkan peran penting dalam menyebarkan informasi pemerintah, memfasilitasi interaksi dengan masyarakat dan mempromosikan potensi daerah. Sebagai media strategis, *website* ini mendukung transparansi dan partisipasi publik. Namun, fungsi penting ini juga membuatnya menjadi target serangan siber. *Website* pemerintah menyimpan *data* sensitif sehingga ancaman seperti *SQL Injection*, *Brute Force* dan DDoS dapat mengancam integritas *data* dan ketersediaan layanan. Selain itu, keamanan *website* ini tidak hanya berfungsi untuk melindungi *data*, tetapi juga untuk menjaga kepercayaan publik terhadap pemerintah.

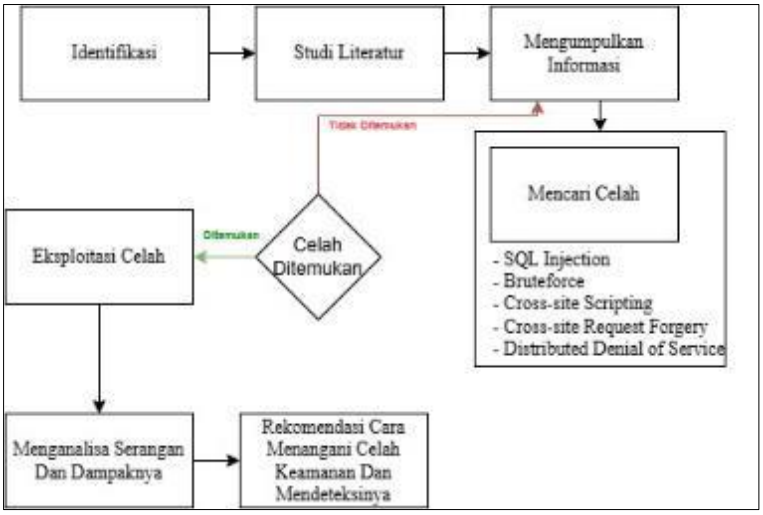
Sebagai tanggapan, diperlukan penerapan praktik keamanan yang komprehensif, seperti validasi *input*, sanitasi *input*, penggunaan *prepared statements* dan pembaruan *framework* untuk mencegah *SQL Injection*. Penggunaan CAPTCHA dan pembatasan upaya *login* menjadi solusi penting untuk mencegah serangan *Brute Force*, sementara mekanisme *escaping* dan *token* CSRF efektif melindungi dari XSS dan CSRF. Untuk deteksi dini serangan, *Snort* dapat digunakan sebagai solusi deteksi intrusi yang memberikan peringatan secara *real-time*, mendukung respons cepat terhadap ancaman.

Kebaruan penelitian ini terletak pada integrasi metode mitigasi manual, serta penerapan sistem notifikasi *real-time* berbasis *Telegram* yang jarang ditemukan dalam penelitian serupa. Berdasarkan latar belakang ini, penelitian bertujuan untuk menguji keamanan *website* Dinas Komunikasi dan Informatika Kabupaten XYZ terhadap lima jenis serangan, menganalisis dampaknya dan mengusulkan langkah mitigasi yang komprehensif untuk meningkatkan keamanan secara signifikan.

2. METODE PENELITIAN

2.1 Metode Penelitian

Metode penelitian yang digunakan dalam penelitian ini adalah *Penetrasi Testing*. *Penetration Testing* merupakan metode untuk menguji sistem keamanan komputer dan jaringan dengan cara mensimulasikan serangan [8]. Metode untuk mencapai tujuan penelitian dijabarkan pada Diagram blok berikut:



Gambar 1 Tahapan Penelitian

Gambar 1 merupakan prosedur penelitian untuk melakukan uji keamanan *web* aplikasi Dinas Komunitas dan Informatikan Kabupaten XYZ untuk mencapai tujuan. Berikut ini adalah prosedur penelitian yang meliputi:

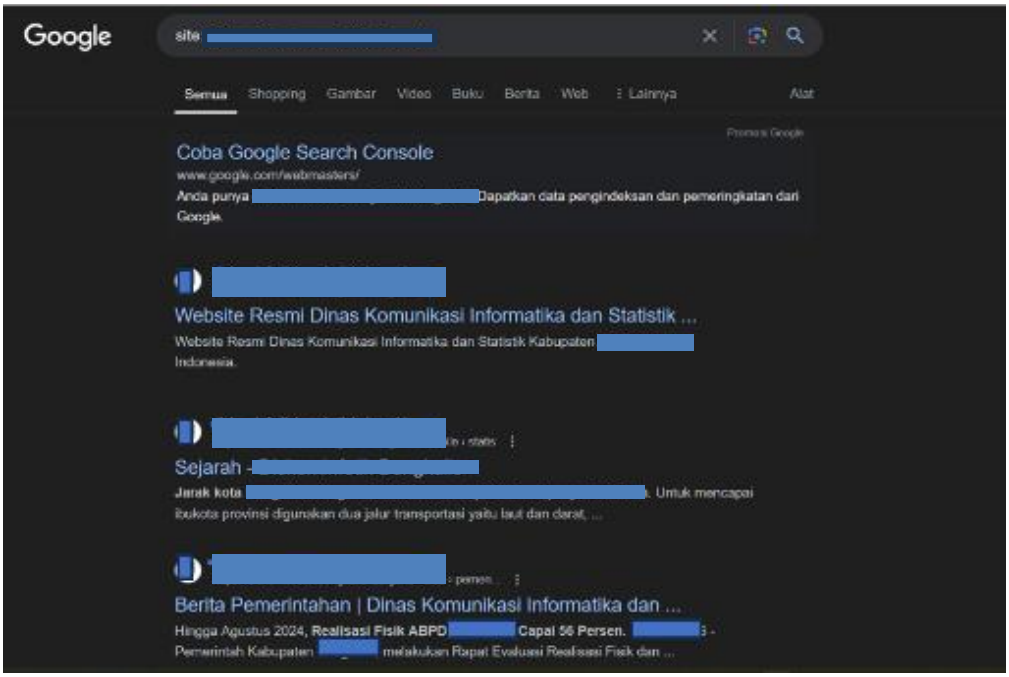
1. Identifikasi Masalah:
   Merumuskan masalah yang diidentifikasi dari objek penelitian serta menentukan batasan-batasannya, sehingga dapat memberikan arah yang lebih jelas.
2. Studi Literatur
   Mengumpulkan sumber-sumber yang berkaitan dengan penetrasi testing, serangan *SQL Injection*, serangan *Bruteforce*, penggunaan alat *SQLMap* dan *Hydra*. serangan XSS, serangan CSRF, serangan *DDoS* dan penggunaan *Snort*.
3. Mengumpulkan Informasi
   Mengumpulkan informasi terkait objek yang sedang diuji, informasi dikumpulkan tanpa berinteraksi langsung dengan objek tersebut. Cara mengumpulkan informasi yang digunakan antara lain yaitu pencarian di internet (*google dork*), WHOIS.
4. Mencari Celah
   Eksplorasi terhadap objek yang diuji dengan membuka setiap halaman yang ada dan mencoba setiap parameter *GET* dan *POST, form input* untuk memicu indikasi kerentanan.
5. Eksploitasi Celah
   Pengujian dilakukan secara manual dan automasi menggunakan dua alat yaitu *SQLMap* dan *Hydra* untuk serangan *SQL Injection* dan serangan *Bruteforce* dan untuk serangan *Cross-Site Scripting* dan *Cross-Site Requests Forgery* dilakukan secara manual dan serangan *DDoS* [2, 3, 9, 10].
6. Menganalisis Serangan Dan Dampaknya
   Memahami sepenuhnya jenis serangan, mekanisme dan konsekuensi dari serangan yang telah dilakukan.
7. Rekomendasi Cara Menangani Celah Keamanan Dan Mendeteksinya
   Memberi rekomendasi cara menangani celah keamanan yang dieksploitasi pada tahap eksploitasi celah dan kemudian mendeteksi serangan siber menggunakan IDS.

3. HASIL DAN PEMBAHASAN

3.1 Mengumpulkan Informasi

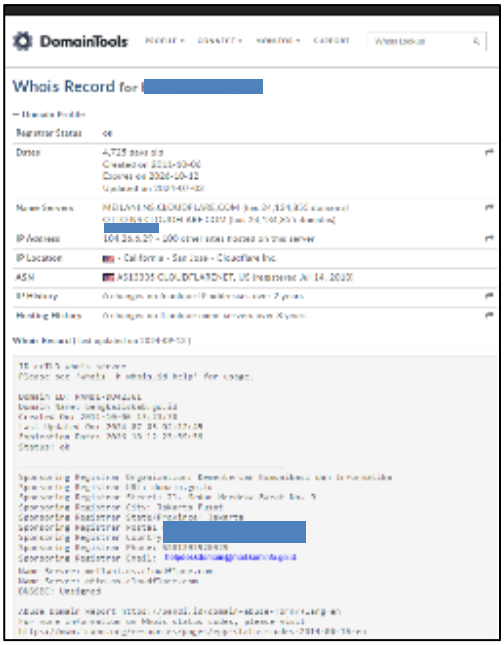
Pada tahap ini dilakukan pencarian informasi menggunakan *Google Dork* dan *whois* untuk mengetahui apakah *domain* yang diuji terdeteksi oleh *google* dan informasi terkait informasi kapan *domain* dibuat, kapan di *update*, kapan masa habis *domain*.

1. *Google Dork*
   Menggunakan mesin pencari terkenal yaitu *google* dan memanfaatkan operator seperti *site* untuk mengetahui apakah domain terdeteksi oleh *google*.



Gambar 2 Hasil *Google Dork*

2. *Whois*
   Menggunakan alat *online* untuk mengetahui informasi terkait informasi kapan *domain* dibuat, kapan di*update*, kapan masa habis *domain*.



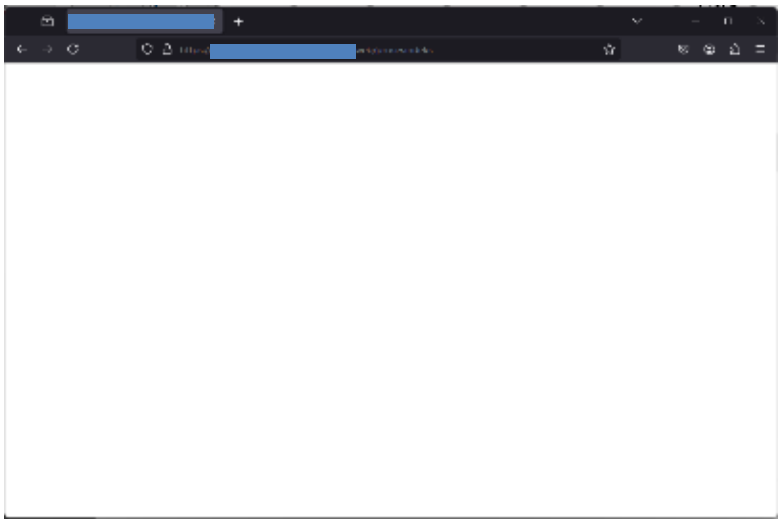
Gambar 3 Hasil *Whois*

3. *Subdomain Scanner*
   Subdomain Scanner merupakan alat yang digunakan untuk menemukan *subdomain* yang terkait dengan domain yang sedang diuji. Ini dilakukan untuk mengetahui ruang lingkup *website* yang dihosting pada *server* yang sama.

Gambar 4 Hasil *Subdomain Scanner*

3.2 Mencari Celah

1. Serangan *SQL Injection* Manual
   Dengan memberikan karakter khusus (*single quote*) ke parameter thn_mulai maka respon dari *website* adalah *blank* atau putih seperti gambar 5, ini merupakan salah satu indikasi celah keamanan *SQL Injection*.

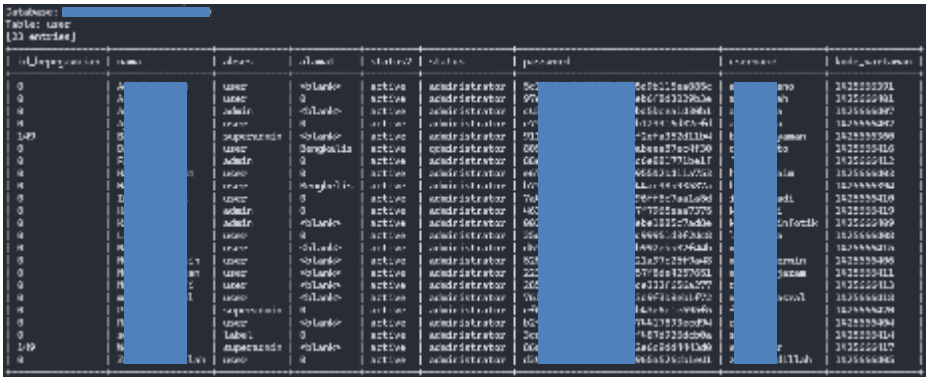


Gambar 5 Indikasi Celah *SQL Injection*

2. Serangan *Bruteforce* manual
   Telah dilakukan sepuluh kali percobaan login untuk melihat respon dari halaman *login*, hasilnya tidak ada CAPTCHA dan pembatasan upaya login ini mengindikasikan bahwa halaman *login* dapat dilakukan serangan *bruteforce*.
3. Serangan *Cross-site Scripting* (XSS) Manual
   Telah dilakukan upaya sepuluh *payload* atau *syntax* yang dapat memicu *alert* atau *popup* pada *browser*, *payload* di dapatkan dari *https://github.com/payloadbox/*XSS-*payload-list*. Hasil pengujian menunjukkan bahwa untuk skenario yang diuji, *website* tidak rentan terhadap serangan XSS *reflected*. Tidak ada *payload* yang berhasil memicu *alert* atau mengubah perilaku *website* secara signifikan.
4. Serangan *Cross-site Request Forgery* (CSRF) Manual
   Telah dilakukan pembuatan *form input* yang berisikan *username* dan *password* yang dikirim kan ke halaman */dashboard/profile*, setelah *form* tersebut dibuka dan mengirimkan data *username* dan *password* ke halaman tujuan, respon dari website yaitu menolak *request*, ini mengindikasikan bahwa *website* sudah menggunakan proteksi untuk serangan ini.

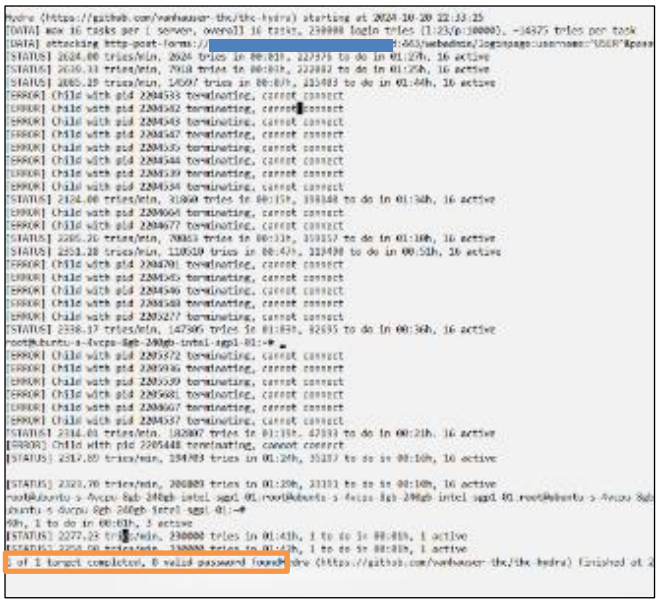
3.3 Eksploitasi Celah

1. Serangan *SQL Injection* menggunakan alat automasi *SQLMap*
   Dari penggunakan alat automasi *SQLMap*, hasil dari eksploitasi dapat mengekstraksi *data* pada tabel *users*, tabel ini berikan kredensial pengguna *website*.



Gambar 6 Tabel *Users*

2. Serangan *Bruteforce* menggunakan alat automasi *Hydra*
   Dari penggunakan alat automasi *Hydra*, hasil dari penggunakan alat yaitu tidak ditemuakan kombinasi kredensial yang benar, ini mengartikan bahwa setiap *user* telah menggunakan kombinasi *password* yang kuat.



Gambar 7 Hasil *Bruteforce*

3. Serangan *Distributed Denial of Service*
   Serangan dilakukan pada *port* 80 yaitu layanan *webserver*, serangan dilakukan dengan dua alat yaitu *Hping3* dan *Xerxes* dieksekusi secara bersamaan. Pada saat kedua alat ini dijalankan, keduanya akan mengirimkan paket dalam jumlah besar secara terus menerus. Namun, karena *DNS website* telah diarahkan ke *Cloudflare*, maka beban trafik jaringan akan di distribusikan oleh *Cloudflare*. Dengan teknologi yang dimilikinya, *Cloudflare* dapat menyaring dan mengelola lalu lintas tersebut sebelum mencapai *server* asli, sehingga mengurangi dampak serangan *DDoS*. Hasilnya *website* masih tetap di akses tanpa ada gangguan layanan.

3.4 Menganalisa Serangan Dan Dampaknya

1. Kerentanan dari serangan *SQL Injection*: Validasi *input* tidak memadai, tidak menggunakan *query* dinamis tanpa parameterisasi, kurangnya sanitasi *input*, *framework* tidak diperbarui. Dari lima analisis kerentanan hal ini dapat menyebabkan dampak penyerang dapat mencuri data sensitive dari basis *data* seperti informasi pengguna.
2. Kerentanan dari serangan *Bruteforce*: Tidak adanya fitur CAPTCHA pada *form login*, tidak ada pembatasan upaya *login*. Dari dua analisis kerentanan hal ini dapat menyebabkan Meningkatkan risiko penyerang berhasil memperoleh akses tidak sah ke akun pengguna, terutama jika kata sandi yang digunakan lemah.

3.5 Rekomendasi Penanganan Celah Keamanan dan Solusi Deteksi Serangan Siber

1. Lakukan validasi *input*.
   Terapkan validasi *input* yang ketat untuk memastikan bahwa semua *input* pengguna sesuai dengan format yang diharapkan.
2. Sanitasi *input*.
   Terapkan sanitasi *input* untuk membersihkan semua *input* pengguna dari karakter khusus yang dapat digunakan untuk menyusun serangan *SQL Injection*.
3. Menggunakan *Query Builder*.

Penggunaan *library* atau *framework* yang menyediakan fitur *query builder* untuk membangun *query SQL* secara dinamis, sehingga menyertakan mekanisme parameterisasi dan validasi *input* secara otomatis.

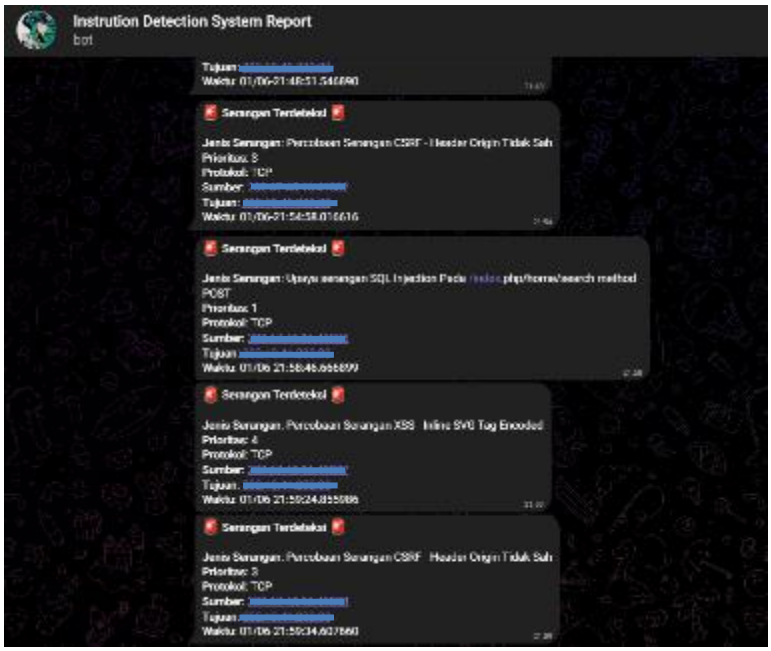
4. Menggunakan *Prepare Stetament*
   Dengan menggunakan *prepared stetament*, *query* dan nilai *input* pengguna dapat dipisahkan sehingga mengurangi risiko penyerangan menyuntikkan kode berbahaya.
5. Perbarui versi *framework*
   Dengan memperbarui *framekwork* ke versi terakhir, memungkinkan *framework* terbaru telah menambal *CVE-2022-40835*. Proses perbarui akan sedikit sulit dikarenakan diharuskan menulis ulang semua kode.
6. Menerapkan fitur CAPTCHA
   Penerapan *captcha* '*Iam not a robot*' pada sistem ini bertujuan untuk membedakan antara pengguna manusia dan *bot*. Dengan menyajikan visual pada setiap sesi *login*, sistem dapat mencegah upaya otomatis seperti serangan *bruteforce* yang sering dilakukan oleh peretas yang menggunakan *bot* untuk menebak kata sandi secara acak.
7. Menerapkan pembatasan upaya *login*
   Dengan menerapkan pembatasan percobaan, maka akan meningkatkan keamanan sistem *login* dari serangan *bruteforce*
8. Mengimplementasikan *snort* sebagai alat deteksi serangan siber.
   Mengimplementasikan *Snort* sebagai alat deteksi serangan siber merupakan langkah strategis dalam meningkatkan keamanan sebuah sistem. *Snort* mampu mendeteksi berbagai jenis serangan siber, seperti *SQL Injection*, *Brute Force*, XSS, CSRF dan DDoS. Deteksi ini dilakukan dengan menganalisis pola serangan yang terjadi dan mencocokkannya dengan aturan yang telah ditetapkan dalam sistem *Snort*. Melalui pemahaman mendalam terhadap pola-pola serangan, *Snort* dapat diatur untuk mengenali aktivitas mencurigakan secara efektif dengan menyesuaikan aturan yang dapat memberikan peringatan dini terhadap potensi ancaman keamanan.

Tabel 1 Aturan *Snort* Untuk Deteksi Serangan

Jenis Serangan	**Aturan**
DDoS	alert tcp any any -> $HOME_NET 80 (flags: S; msg:"Serangan DDOS"; detection_filter: track by_dst, count 10000, seconds 60; sid:1000002;)
SQL Injection	alert tcp any any -> any 80 (msg:"Percobaan SQL Injection Universal"; pcre:"/(\%27|\'|\-\-|\%23)/"; sid:1000004;)
XSS	alert tcp any any -> any 80 (msg:"Percobaan Serangan XSS - SCRIPT ALERT ENCODED"; pcre:"/%3Cscript%3E.*?alert%2e8.*?%29.*?%3C%2Fscript%3E/"; sid:1000005;)
Bruteforce	alert tcp any any -> any 80 (msg:"Percobaan Serangan Bruteforce"; content:"POST"; http_method; content:"/pentest/index.php/home"; http_uri; threshold:type both, track by_src, count 10, seconds 60; sid:1000011;)
CSRF	alert tcp any any -> any 80 (msg:"Percobaan Serangan CSRF - Header Origin Tidak Sah"; content:"POST"; http_method; content:"/pentest/index.php/admin/change_password"; http_uri; content:"Origin: null"; http_header; nocase; offset:0; sid:1000013;)

Setelah membuat aturan yang sesuai untuk mendeteksi serangan *SQL Injeciton*, *Bruteforce*, XSS, CSRF, DDoS gunakan bahasa *scripting bash* untuk membuat *bot telegram* sebagai notifikasi pemberitahuan secara *real-time*.

Gambar 8 *Bash Scripting* Untuk Mengirim Notifikasi *Real-Time*



Gambar 9 Contoh Notifikasi Serangan di *Telegram*

Tabel 2 Perbandingan Tingkat Keamanan Sebelum dan Setelah Mitigasi Keamanan

Jenis Serangan	Sebelum Mitigasi	Setelah Mitigasi	Peningkatan Keamanan
SQL Injection	Serangan berhasil melakukan *injection* dan mengekstrak data dari basis data.	Serangan *injection* tidak ada yang berhasil.	Validasi *input*, sanitasi *input*, penggunaan *prepared statement*, pembaruan *framework*.
Bruteforce	Serangan tidak berhasil menemukan kombinasi kredensial yang tepat.	Serangan tetap tidak berhasil, namun penerapan mitigasi diperlukan untuk mencegah potensi serangan.	CAPTCHA, pembatasan upaya *login*

XSS	Serangan tidak berhasil karena karakter spesial sudah di-*escaping* oleh fitur *website*.	Tidak diperlukan mitigasi tambahan karena proteksi sudah ada.	Tidak diperlukan peningkatan keamanan lebih lanjut karena perlindungan terhadap XSS sudah diterapkan.
CSRF	Serangan tidak berhasil karena website sudah menerapkan *token* CSRF di setiap *input*.	Tidak diperlukan mitigasi tambahan karena proteksi sudah ada.	Tidak diperlukan peningkatan keamanan lebih lanjut karena proteksi terhadap CSRF sudah diterapkan.
DDoS	Serangan tidak berhasil karena *website* sudah dilindungi oleh *Cloudflare*.	Tidak diperlukan mitigasi tambahan karena proteksi DDoS sudah ada.	Tidak diperlukan peningkatan keamanan lebih lanjut karena proteksi DDoS sudah diterapkan oleh *Cloudflare*.

Setelah melakukan perbandingan antara serangan sebelum dan sesudah mitigasi pada beberapa jenis kerentanannya, yaitu *SQL Injection*, *Brute Force*, XSS, CSRF dan DDoS, dapat disimpulkan bahwa mitigasi yang diterapkan berhasil meningkatkan keamanan *website*. Langkah mitigasi yang diterapkan meliputi validasi *input*, sanitasi *input*, penggunaan *prepared statements*, penerapan CAPTCHA, pembatasan upaya *login* dan proteksi tambahan lainnya. Setelah mitigasi diterapkan, serangan yang sebelumnya berhasil, seperti *SQL Injection* tidak lagi dapat mengekstrak *data* dari basis data. Sementara itu, serangan lainnya seperti *Brute Force*, XSS dan CSRF tetap tidak berhasil karena sistem sudah memiliki perlindungan bawaan yang memadai.

Untuk mengukur efektivitas mitigasi yang diterapkan, digunakan sistem penilaian CVSS (*Common Vulnerability Scoring System*). CVSS menghitung skor berdasarkan dampak terhadap kerahasiaan, integritas, dan ketersediaan sistem. Hasil pengukuran menunjukkan adanya peningkatan skor keamanan secara signifikan setelah mitigasi diterapkan yang mengindikasikan pengurangan risiko signifikan terhadap kerentanan yang ditemukan sebelum mitigasi.

Tabel 3 Pengukuran Tingkat Keamanan Dengan CVSS v.3.0

Jenis Serangan	**Skor CVSS Sebelum Mitigasi**	***Vector CSVV* Sebelum mitigasi**	**Skor CVSS Setelah Mitigasi**
SQL Injection	7.5 (tinggi)	CVSS:3.0/AV:A/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H	0.0 (tidak ada)
Bruteforce	0.0 (tidak ada)	CVSS:3.0/AV:A/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N	0.0 (tidak ada)
XSS	0.0 (tidak ada)	CVSS:3.0/AV:A/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N	0.0 (tidak ada)
CSRF	0.0 (tidak ada)	CVSS:3.0/AV:A/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N	0.0 (tidak ada)
DDoS	0.0 (tidak ada)	CVSS:3.0/AV:A/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N	0.0 (tidak ada)

4. KESIMPULAN DAN SARAN

Dari hasil pengujian keamanan yang melibatkan lima teknik serangan, dapat disimpulkan bahwa website ini memiliki celah keamanan terhadap serangan *SQL Injection*. Untuk mengatasi celah ini, dapat diterapkan beberapa langkah mitigasi, antara lain: validasi *input*, sanitasi *input*, penggunaan *Query Builder*, penerapan *prepared statements*, serta memperbarui *framework* aplikasi ke versi terbaru. Agar dapat mendeteksi serangan secara lebih menyeluruh, disarankan untuk menggunakan *software* IDS (*Intrusion Detection System*) *Snort* pada *server* aplikasi. Penggunaan IDS bertujuan untuk mendeteksi upaya serangan siber yang dapat segera

diidentifikasi dan dicegah, sehingga mengurangi risiko gangguan terhadap data atau layanan yang sedang berjalan. Selain itu, untuk meningkatkan efektivitas deteksi serangan, sangat disarankan untuk melanjutkan penelitian ini dengan memanfaatkan teknologi berbasis *Artificial Intelligence* (AI). AI dapat membantu dalam mendeteksi dan menganalisis pola serangan atau anomaly baru secara *real-time*, memungkinkan respons yang lebih cepat dan akurat untuk mencegah serangan lebih lanjut.

DAFTAR PUSTAKA

[1] Y. Christian and D. Alfath, "Perancangan Sistem Manajemen Kerja Harian Berbasis Website Menggunakan Framework Codeigniter di Universitas Internasional Batam," 2021. [Online]. Available: https://journal.uib.ac.id/index.php/combines

[2] T. Anugrah, "PENETRATION TESTING KEAMANAN WEBSITE STIE SAMARINDA MENGGUNAKAN TEKNIK SQL INJECTION DAN XSS," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 1, Jan. 2024, doi: 10.23960/jitet.v12i1.3882.

[3] M. R. Sampurna, "NetPLG Journal of Network and Computer Applications Implementasi Hydra, FFUF, dan WFUZZ dalam Brute Force DVWA," vol. 1, no. 2, 2022, [Online]. Available: https://jurnal.netplg.com/

[4] S. Suroto and A. Asman, "ANCAMAN TERHADAP KEAMANAN INFORMASI OLEH SERANGAN CROSS-SITE SCRIPTING (XSS) DAN METODE PENCEGAHANNYA," 2021. [Online]. Available: http://www.hackers.com?yid=

[5] S. Fadilla Yulia Fauzan, "Analisis Metode Web Security PTES (Penetration Testing Execution And Standart) Pada Aplikasi E-Learning Universitas Negeri Padang", [Online]. Available: http://ejournal.unp.ac.id/index.php/voteknika/

[6] J. Wang and L. Wang, "SDN-Defend: A Lightweight Online Attack Detection and Mitigation System for DDoS Attacks in SDN," vol. 22, no. 21, Nov. 2022, doi: 10.3390/s22218287.

[7] Dhuha Sabri Ghazi, H. S. Hamid, M. J. Zaiter, and A. S. Ghazi Behadili, "Snort Versus Suricata in Intrusion Detection," *Iraqi Journal of Information and Communication Technology*, vol. 7, no. 2, pp. 73–88, Dec. 2024, doi: 10.31987/ijict.7.2.290.

[8] M. A. Z. Risky and Y. Yuhandri, "Optimalisasi dalam Penetrasi Testing Keamanan Website Menggunakan Teknik SQL Injection dan XSS," *Jurnal Sistim Informasi dan Teknologi*, pp. 215–220, Aug. 2021, doi: 10.37034/jsisfotek.v3i4.68.

[9] R. Hermawan, "STRING (Satuan Tulisan Riset dan Inovasi Teknologi) TEKNIK UJI PENETRASI WEB SERVER MENGGUNAKAN SQL INJECTION DENGAN SQLMAP DI KALILINUX."

[10] A. A. Wabi, I. Idris, O. M. Olaniyi, and J. A. Ojeniyi, "Impact Analysis and Features for DDOS Attacks Detection in SDN," Apr. 2023. [Online]. Available: www.matjournals.com