Available online at https://ejournal.almaata.ac.id/index.php/IJUBI
Indonesian Journal of Business Intelligence
Volume 6 | Issue 1 | June (2023)
ISSN 2621-3915 (PRINT), ISSN 2621-3923 (ONLINE), Published by **Alma Ata University Press**

IJUBI
Indonesian Journal
---- of ----
Business Intelligence

MENINGKATKAN KEAMANAN TERHADAP *SQL INJECTION* STUDI KASUS SISTEM KEPEGAWAIAN BNN

Rachmawati Yulia Andarini[1], Purwono Hendradi[2*], Setiya Nugroho[3]
[1,2,3]Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muhammadiyah Magelang.
*p_hendra@ummgl.ac.id
Jl. Mayjend. Bambang Soegeng, Mertoyudan, Magelang, Jawa Tengah, Indonesia

Keywords:

SQL Injection; captcha code;SiMPaN

Abstract

In the era of rapid digitalization, SQL Injection attacks are a serious threat to the security of data and sensitive information on websites. The vulnerability of the SiMPaN BNNK Magelang website to SQL Injection attacks can pose a risk of leakage of personal data and employee monthly reporting files. This research aims to improve website security at BNNK Magelang by using the features of a web-based application. This study uses a qualitative research method involving the stages of initiation, investigation, testing, verification, and recommendations to analyze the security of the SiMPaN BNNK Magelang web application, as well as provide recommendations for improvements based on the results of analysis and testing of SQL Injection attacks. The results show that by implementing security measures such as SSL and captcha codes, hacking attacks such as SQL Injection can be prevented. The use of SSL increases the security of data exchange, while the captcha code ensures that the user who enters is a human, not a computer or a bot so that login security on the website can be increased and the risk of hacking attacks can be minimized. Researchers provide a deeper understanding of SQL Injection attacks and provide practical security solutions to avoid these attacks. The conclusion that can be drawn is that the combination of adding SSL and captcha code can provide a more optimal level of security on a website's login system.

Kata Kunci:

SQL Injection; kode captcha;SiMPaN

Abstrak

Dalam era *digitalisasi* yang pesat, serangan *SQL Injection* menjadi ancaman yang serius bagi keamanan data dan informasi sensitif yang terdapat dalam *website. Rentannya website* SiMPaN BNNK Magelang terhadap serangan *SQL Injection* dapat menimbulkan risiko kebocoran data pribadi dan berkas pelaporan bulanan pegawai. Penelitian ini bertujuan untuk meningkatkan keamanan *website* di BNNK Magelang dengan menggunakan fitur dari aplikasi berbasis *web*. Penelitian ini menggunakan metode penelitian kualitatif yang melibatkan tahap *inisiasi, investigasi,* pengujian, *verifikasi*, dan rekomendasi untuk menganalisis keamanan aplikasi web SiMPaN BNNK Magelang, serta memberikan rekomendasi perbaikan berdasarkan hasil analisis dan pengujian terhadap serangan *SQL Injection*. Hasil penelitian menunjukkan bahwa dengan menerapkan langkah-langkah keamanan seperti *SSL* dan lode

Article history:

Received : 10 April 2023
Revised : 25 June 2023
Accepted : 30 June 2023

captcha, serangan *peretasan* seperti *SQL Injection* dapat dicegah. Penggunaan *SSL* meningkatkan keamanan pertukaran data, sementara *kode captcha* memastikan bahwa pengguna yang masuk adalah manusia, bukan komputer atau *bot* sehingga pengamanan *login* pada *website* dapat ditingkatkan dan risiko serangan *peretasan* dapat diminimalkan. Peneliti memberikan pemahaman lebih dalam tentang serangan *SQL Injection* dan menyediakan solusi keamanan yang praktis untuk menghindari serangan tersebut. Kesimpulan yang dapat diambil adalah kombinasi penambahan *SSL* dan *kode captcha* dapat memberikan tingkat keamanan yang lebih maksimal pada sistem *login* sebuah *website*.

Pendahuluan

Di era perkembangan digitalisasi yang sangat pesat ini,*SQL Injection* merupakan sebuah bahasa pemrograman seperti *PHP* atau *Perl mengakses database* melalui *SQL query* atau *xampp. SQL Injection* merupakan suatu teknik peretasan yang memungkinkan penyerang mendapatkan akses yang tidak sah ke dalam *database* kemudian menyerang atau mengubah data-data yang berada di dalam *database*. Ada beberapa jenis *SQL Injection* diantaranya *Tautologies, Logically Incorrect Query, Union Query, Piggy Backed Query, Stored Procedures. Tautologi* adalah serangan untuk menghasilkan kondisi *TRUE* dengan menggunakan '=' (sama) dengan *query*. Jenis *SQL Injection* ini dapat melewati halaman *otentikasi* dan mendapatkan data yang *diekstraksi*. Beberapa pekerjaan telah dilakukan untuk mencegah serangan pada *SQL Injection*. Pada tahun 2020, Hlaing dan Khaing menghadirkan pendekatan yang *mendeteksi token kueri* dengan *leksikon berbasis* kata yang dicadangkan untuk *mendeteksi Serangan Injeksi Bahasa Kueri Terstruktur (SQLI)*. Sistem yang diusulkan dilakukan dengan menggunakan dua pendekatan, yaitu: membuat *leksikon* dan *tokenize* pernyataan *query input* dan setiap *token string dideteksi* ke *leksikon* kata yang telah ditentukan untuk mencegah *SQLI*. Berdasarkan percobaan yang dilakukan, sistem yang diusulkan mampu memberikan pencegahan yang berhasil dari berbagai permintaan jahat untuk *injeksi* [1].

Badan Narkotika Nasional Kabupaten Magelang telah mengembangkan *website* untuk absensi non pegawai negeri/polri secara *lokal*. Manfaat *website* bagi pegawai non pns/polri adalah dapat mengurangi *human error* dan meningkatkan *efektivitas* kerja. Aplikasi Sistem Informasi Manajemen Kepegawaian ASN(SiMPaN) dapat *diakses* oleh semua pegawai sehingga tidak menutup kemungkinan terjadi berbagai serangan yang dilakukan oleh *cracker,* sedangkan data dan informasi di dalamnya bersifat sensitif karena berhubungan dengan data pribadi dan berkas pelaporan bulanan pegawai. *Website* Sistem Informasi Manajemen Kepegawaian ASN (SiMPaN) menggunakan Bahasa Pemrograman *PHP*, dimana *Engine PHP* menerjemahkan *PHP* ke *website* yang ada, apa yang sebenarnya dilihat oleh pengguna di *browser* adalah bukan *file PHP* itu sendiri melainkan *output* dari semua perintah *PHP* yang dikirimkan kembali oleh *website.* Metode yang digunakan dalam pengembangan sistem web SiMPaN hampir sama dengan Perancangan Web E-Shop Pada Toko Sandy yaitu menggunakan metode SDLC (System Development Life Cycle) dan software yang digunakan ialah Php dan Mysql yang berfungsi untuk merancang dan mendesain yaitu bahasa pemrograman PHP, database MySQL, text editor Notepad++, dan XAMPP [2].

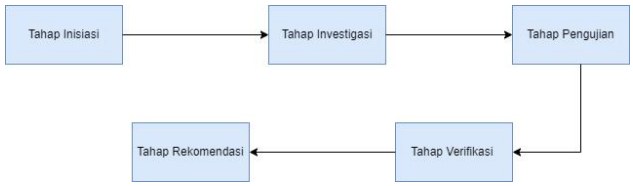
Beberapa Penelitian sebelumnya yang membahas tentang Meningkatkan Keamanan Terhadap *SQL Injection* dengan berbagai metode seperti Peningkatkan Keamanan Webserver Aplikasi Pelaporan Pajak Daerah Menggunakan Metode *Penetration Testing Execution* Standar disajikan pengujian 7 jenis

kerentanan yang dibandingkan antara hasil *exploitasi* dan hasil sesudah perbaikan [3]. Kemudian Penelitian ini menggunakan model *Webqual 4.0*. Metode *WebQual 4.0* [4]. Analisa keamanan menggunakan metode Open Web Application Security Project (OWASP) dan Tool Red Hawk dan OWASP Zap [5]. Metode penelitian yang digunakan yaitu OWASP Top-10 2017 yang menghasilakan jumlah 5 subdomain yang teridentifikasi dengan melakukan scanning menggunakan tool TheHarvester [6]. Penelitian lainnya yang membahas analisa keamanan aplikasi data pokok pendidikan (dapodik) metode yang digunakan metode kualitatif dengan menggunakan beberapa *tools* berupa perangkat lunak dan cara-cara tertentu yang lazim digunakan untuk menguji keamanan aplikasi dengan tahapan :Tahap *Inisiasi*, Tahap *Investigasi*, Tahap Pengujian, dan Tahap *Verifikasi* [7].

Di BNNK Magelang sendiri tidak ada pegawai yang paham akan *cyber security*,dan di BNNK Magelang sendiri juga pernah terjadi *peretasan*. *Peretasan* itu terjadi sebanyak 3 kali,yaitu dengan cara *attacker* mencoba berbagai kombinasi untuk mencoba *akses* masuk dalam *login website* Sistem Informasi Manajemen Kepegawaian ASN (SiMPaN). Maka penelitian ini mengajukan peningkatan keamanan *website* di BNNK Magelang yang berfungsi untuk membandingkan manakah yang lebih baik untuk mengamankan *login* pada sebuah *website* dan cara untuk menghindari terjadinya *Human Error*, disini peneliti mencoba menguji dengan cara menambahkan *SSL* dan menambahkan *kode Captcha* pada bagian *login website* yang didahului dengan menguji keamanan menggunakan tahapan :Tahap *Inisiasi,* Tahap *Investigasi*, Tahap Pengujian, Tahap *Verifikasi* dan Tahap Rekomendasi.

Metode

Penelitian ini menggunakan metode penelitian kualitatif yang berdasarkan pada teori-teori. Penelitian ini menggunakan pendekatan deskriptif analisis yang sebelumnya dibaca terlebih dahulu. Sumber data dikumpulkan melalui teknik studi literatur berbagai sumber, diantaranya buku, jurnal, artikel, dan skripsi yang sejalan dan berkaitan dengan judul yang dipilih oleh peneliti. Dalam tahapannya penelitian ini menggunakan teknis analisis dengan membaca data yang kemudian dibahas untuk kemudian disimpulkan. Berdasarkan studi yang telah peneliti baca peneliti memilih menerapkan metode dari hasil penelitian bastian et al., 2020 [7], dalam analisa keamanan aplikasi data pokok pendidikan (dapodik) metode yang digunakan metode kualitatif dengan menggunakan beberapa *tools* berupa perangkat lunak dan cara-cara tertentu yang lazim digunakan untuk menguji keamanan aplikasi dengan Tahap-tahap yang dilakukan adalah sebagai berikut :



Gambar 1. Tahap Penelitian

Tahap *inisiasi* (analisis kebutuhan dan pengumpulan data), pada tahap ini dilakukan penelusuran dan pengkajian *literatur-literatur* yang berhubungan dengan keamanan aplikasi.

Tahap *investigasi (identifikasi* proses sistem), pada tahap ini dilakukan penyelidikan terhadap *web server program website* yang digunakan.

Tahap pengujian (*identifikasi* bukti sesuai *klasifikasi* sistem), pada tahap ini dilakukan pengujian terhadap keamanan aplikasi dengan menggunakan *tools*, yaitu peneliti akan melakukan tes serangan dengan *SQL Injection* (*Username*(acak) +' OR 1=1 --).

Tahap *verifikasi* (pengukuran tingkat keamanan dengan *SQLI* dan memberikan rekomendasi perbaikan), pada tahap ini dilakukan *verifikasi* terhadap keamanan aplikasi untuk pemberitahuan kepada *admin* untuk dilakukan perbaikan-perbaikan atas dasar hasil *investigasi* dan pengujian pada aspek pemrograman [7]. Setelah dilakukan serangan *SQL Injetion* (*Username*(acak) +' OR 1=1 --), ditemukan celah pada bagian *login*, kemudian peneliti menambahkan *SSL* pada bagian *login* karena celah *security* yang terdapat di bagian *login* karena tidak ada *filtering,* atau bisa juga menambahkan *kode captcha* pada bagian bawah *login* karena *website* SiMPaN dibuat *localhost* maka tidak menutup kemungkinan akan terjadinya *human error* maka ditambahkannya *kode captcha* untuk menghindari *human error*. Kemudian kita tes lagi apabila tes gagal maka

penambahan *2 faktor autentichationnya* berhasil.

Pembahasan

Tahap *inisiasi* (analisis kebutuhan dan pengumpulan data) pada tahap ini penelitian yang dilakukan oleh [9], melakukan *penetrasi* secara paksa dalam serangan *SQL Injection* pada *website*. Dengan cara mengumpulkan data berupa dua rancangan *website* yang rentan akan *serangan SQLI* tanpa menggunakan teknik *maxlength* dan *input type number*. Kemudian rancangan *website* yang kedua merupakan pengembangan dengan menggunakan teknik *maxlength* dan *input type number* untuk mencegah *penetrasi* paksa dalam *SQL Injection* dengan pengujian dari *server* penyedia layanan *hosting* serta pengumpulan data *sintax SQL Injection* yang biasa digunakan *hacker* untuk *website*.

Tahap *investigasi (identifikasi* proses sistem), pada tahap ini, mencari tau bahasa pemrograman apa yang digunakan pada pembuatan *website*, menggunakan sistem *database* apa pada sistem, dan mencari tahu *website* tersebut menerapkan *server lokal* atau bukan. *Website* SiMPaN menggunakan Bahasa Pemrograman *PHP*, dimana *Engine PHP* menerjemahkan *PHP* ke *website* yang ada, apa yang sebenarnya dilihat oleh pengguna di *browser* adalah bukan *file PHP* itu sendiri melainkan *output* dari semua perintah *PHP* yang dikirimkan kembali oleh *website*. Sedangkan untuk pengolahan *database website* SiMPaN menggunakan *XAMPP. Website* SiMPaN menerapkan *server lokal.*

Tahap pengujian (*identifikasi* bukti sesuai *klasifikasi* sistem), pada tahap ini dilakukan pengujian terhadap keamanan aplikasi dengan menggunakan *tools*, yaitu peneliti akan melakukan tes serangan dengan *SQL Injection* (*Username*(acak) +' OR 1=1 --). Kemudian akan menganalisis hasilnya apakah serangan berhasil atau tidak. Setelah dilakukan pengujian ke 10 kali percobaan untuk membuat keamanan pada *login website* agar tidak mudah *diretas* oleh *peretas* [10]. Dengan melalui beberapa tahap yaitu tahap analisis,tahap ini dilakukan dengan melakukan wawancara tidak terstruktur kepada salah satu pegawai BNNK yang bertanggung jawab terhadap sistem pengelolaan data keluar/masuk, hasil dari wawancara ialah adanya jalur komunikasi yang belum aman antara *client* dengan *server* sehingga meskipun *server* SiMPaN berada pada jaringan *lokal*, dimungkinkan adanya serangan yang dilakukan oleh pihak internal ataupun orang lain berkaitan dengan penyadapan sehingga orang tersebut dapat merubah data yang ada di dalam *server*. Tahap selanjutnya ialah tahap *design*, tahap ini dilakukan untuk menggambarkan *topology* dari rancangan jaringan yang akan dibangun. Selanjutnya tahap *simulation prototyping*, pada tahap ini dilakukan simulasi komunikasi data menggunakan SSL menggunakan *virtual* sistem operasi dengan menggunakan *virtual box*. Selanjutnya tahap *implementation*,tahap *implementasi* yaitu tahapan penerapan SSL yang digunakan oleh server SiMPaN BNNK Magelang. Selanjutnya adalah tahap *monitoring*,tahap ini dilakukan *capture* menggunakan *wireshark* dan dilakukan perbandingan antara data yang lewat sebelum menggunakan *SSL* dan data yang lewat setelah menggunakan *SSL*. Selanjutnya tahap Management, tahap ini dilakukan terkait dengan kebijakan yang diterapkan pihak BNNK Magelang dalam mengakses sistem SiMPaN ini. Dimana setelah dilakukan analisis mengenai paket data yang lewat sebelum dengan menggunakan wireshark, apakah *SSL* ini akan diterapkan pada saat melakukan *akses* ke *server* SiMPaN ataupun tidak [11]. Penelitian ini berfungsi untuk membandingkan manakah yang lebih baik untuk mengamankan *login* pada sebuah *website* dan cara untuk menghindari terjadinya *Human Error*, disini peneliti mencoba menguji dengan cara menambahkan *SSL* dan menambahkan *kode Captcha* pada bagian *login website*.

A. Menguji dengan SSL

Pertama peneliti akan melakukan pengujian terhadap *website* SiMPaN yang telah dibuat. Untuk mengetahui apakah hasil yang dikeluarkan oleh *website* SiMPaN telah sesuai dengan yang diharapkan atau belum. Untuk proses menguji celah keamanan *website* SiMPaN peneliti menggunakan metode *SQL Injection* pada bagian *login*.

```
if (isset($_POST['login'])){
    // berfungsi menangkap data yang dikirim
    $user = $_POST['username'];
    $pass = md5($_POST['password']);

    // berfungsi menyeleksi data user dengan username dan password yang sesuai
    $sql = mysqli_query($koneksi,"SELECT * FROM users INNER JOIN data_p ON users.no_PPNPN=data_p.no_PPNP WHERE username='$user' AND passwor
    //berfungsi menghitung jumlah data yang ditemukan
    $cek = mysqli_num_rows($sql);
```

Gambar 2. Skrip Awal SiMPaN

Pada gambar 2 Menampilkan skrip awal dari *website* SiMPaN,sebelum ditambahkan keamanan dan *filtering* pada *login*. Yang akan diuji kerentaannya menggunakan *SQL Injection* dan hasilnya berada di bawah ini:

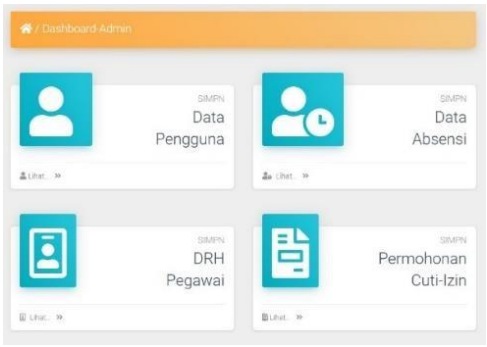
Gambar 3. Tampilan *Website Login*

Gambar 4. Menguji *Login* Simpan dengan *SQL Injection*

Hasil dari pengujian kerentaan pada gambar 4 ialah,tanpa adanya sebuah *filter* di *codingan* awal sangat mudah dibobol tanpa mengetahui *password* dari seorang *user* atau pun *admin*. Dengan menggunakan bahasa *SQL* yang ada di *phpMyAdmin*. Caranya yaitu dengan *(Username* +'-- *)* kemudian masukan *password* acak. Peneliti juga menguji kerentaan *login* menggunakan cara lain yaitu jika tanpa mengetahui *Username* dan juga *Password* bahasa yang digunakan berbeda dengan yang sebelumnya karena penggunaan logikanya juga berbeda. Dengan cara(*Username*(acak) +' OR 1=1 --) [12]. OR 1=1 dalam bahasa *SQL* artinya ketika tidak ada *filter* salah satu dari *password* maupun *username* benar walaupun dengan *username* asal tetap akan bisa masuk. Hasil jika *bypass* berhasil akan apabila ditampilkan di database seperti gambar dibawah ini :

Gambar 5. Hasil Tes *SQL Injection* di *Database*

Gambar 5 menampilkan hasil tes *SQL Injection* di database. Kemudian akan langsung masuk ke halaman *dashboard* pada *website*, Kejadian seperti ini akan sangat berbahaya yang kita tau dimana di dalam *website* ini banyak terdapat data-data penting para pegawai BNNK MAGELANG. Pengecekan *SQL Injection* ini dilakukan agar kita mengetahui kekurangan apa yang ada pada *querynya*. Setelah ini penulis akan melakukan *injection* tanpa mengetahui baik dari *username* dan juga *password.*

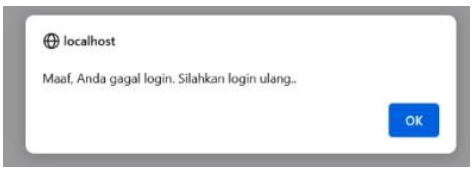


Gambar 6. *SQL Injection* Berhasil Masuk Ke Halaman *Dashboard*

Pada gambar 6 menampilkan hasil dari pengujian kerentaan dengan *SQL Injection* berhasil masuk ke Halaman *Dashboard*. Dengan sangat mudah untuk dimasuki padahal tanpa mengetahui *password* dan *username* hal seperti ini perlu diperbaiki untuk pengamanan secara lebih baik lagi. Karena *authentication* adalah keamanan paling awal yang seharusnya tercipta. Setelah ditambahkan keamanan, untuk mengujinya gunakan *username* dan *password* yang kuat agar *website* tidak mudah terkena *brute force attack*.

Gambar 7. Skrip Untuk Menambahkan *SSL* Pada *Login Website* Simpan

Pada gambar 7 menampilkan skrip untuk menambahkan *SSL* pada *login Website* SiMPaN. Yang berguna untuk keamanan pada *website* yang memungkinkan dalam pembuatan *website* dibuat pembatasan percobaan *login* agar tidak mudah terkena *brute force attack* untuk mencoba kombinasi *username* dan *password* ribuan kali.

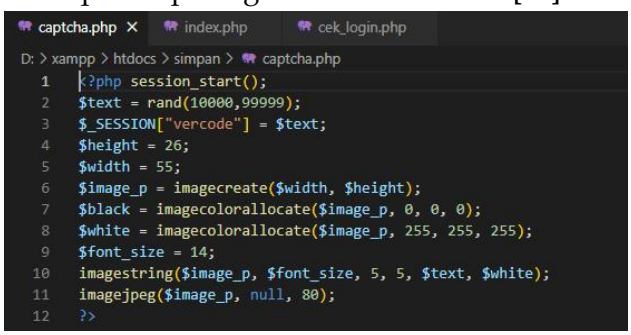


Gambar 8. Hasil Pengujian Kerentaan *Login* dengan *SQL Injection* Setelah Ditambahkan *SSL*

Gambar 8 Menampilkan pengujian kerentaan *Login* dengan *SQL Injection*. Setelah ditambahkan *SSL*. Hasilnya peretasan tidak dapat dilakukan,karena *SSL* membuat pertukaran data antara *website* anda dengan pengunjung menjadi aman. *SSL* akan *mengenkripsi* data tersebut sehingga *hacker* tak akan bisa membacanya. Karena *hacker* tersebut tak memiliki "kunci" untuk membuka gembok *enkripsi* data itu. fungsi SSL adalah mengamankan data pribadi, seperti nama, alamat dari para penjahat *cyber*. Cara kerja SSL yaitu dengan mengunci *cryptographic key* ke informasi perusahaan yang akan dikl. Data pun akan *terenkripsi* dengan baik selama proses transfer sehingga pihak ketiga tidak akan bisa masuk dan mencuri informasi sensitif [13].

B. Menguji dengan *Kode Captcha*

Peneliti mencoba menambahkan *kode captcha* sebelum *login*. Langkah pertama yaitu peneliti membuat Skrip *Captcha*, Tes ini melibatkan user untuk mengetikkan hasil tebakan huruf atau angka yang ada pada image. kemudian SiMPaN file dengan nama *captcha.php* yang berisi tentang kolom untuk *verfikasi* kode dalam bentuk teks ,dan akan memunculkan gambar dengan angka didalamnya dengan panjang 26 lebar 55 sebagai kode *verifikasi captcha*, yang ditampilkan pada gambar dibawah ini [14].

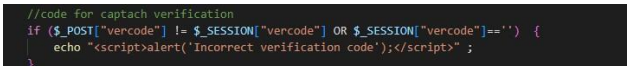


Gambar 9. Skrip Untuk Membuat *Captcha*

Pada gambar 9 peneliti menampilkan skrip untuk membuat *captcha*. Kemudian untuk memasang *kode captchanya*.Pada kode di bawah diantara Sampai pada index.php dibawah skrip peneliti memasukkan username dan password kemudian peneliti menambahkan skrip kode seperti gambar dibawah ini:

Gambar 10. Skrip Untuk Menampil *Kode Captcha* Di *Login*

Pada gambar 10 peneliti menampilkan skrip Sampai pada index.php yang berfungsi untuk memanggil *kode captcha*. Kemudian menambahkan skrip untuk memverifikasi *kode captcha* karena *kode captcha* tidak akan tampil ketika belum menambahkan *verifikasi kode captcha* seperti gambar dibawah ini:



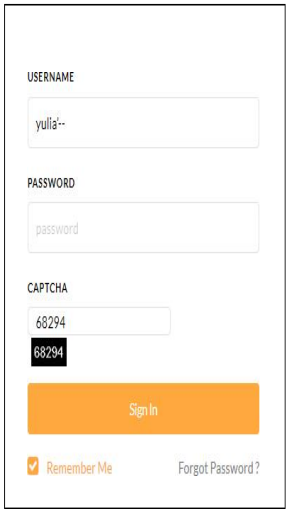
Gambar 11. Skrip Untuk *Verifikasi Kode Captcha*

Pada gambar 11 peneliti menampilkan skrip if ($_POST["vercode"] sampai </skript>" ;}.yang akan berfungsi untuk *verifikasi kode captcha*. Kemudian untuk hasilnya setelah dijalankan di laman *login* simpan,akan ditampilkan pada gambar di bawah ini.

Gambar 12. Hasil Setelah Ditambahkan *Captcha*

Pada gambar 12 peneliti menampilkan hasil setelah menambahkan *kode captcha* pada *login*. Peneliti mencoba menerapkan *2-factor authentication* [15], dengan menambahkan *kode captcha* sebelum *login*. *Captcha* adalah suatu bentuk uji tantangan- tantangan *(challange-response test)* yang digunakan dalam perkomputeran untuk memastikan bahwa jawaban tidak dihasilkan oleh suatu komputer. Proses ini biasanya melibatkan suatu komputer (*server*) yang meminta seorang pengguna untuk menyelesaikan suatu uji sederhana yang dapat dihasilkan dan dinilai oleh komputer tersebut. Karena komputer lain tidak dapat memecahkan *captcha*, pengguna manapun yang dapat memberikan jawaban yang benar akan dianggap sebagai manusia. Maka dari

pengujian tersebut, pengujian tersebut kadang disebut sebagai uji Turing balik, karena dikelola oleh mesin dan ditujukan untuk manusia, kebalikan dari uji Turing standar yang biasanya dikelola oleh manusia dan ditujukan untuk suatu mesin. *captcha* umumnya menggunakan huruf dan angka dari *citra terdistorsi* yang muncul di layar.



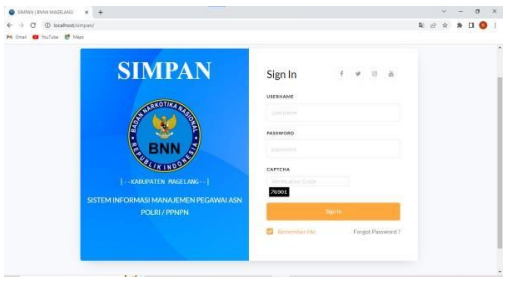
Gambar 13. Menguji *Login* Simpan dengan *SQL Injection* Setelah Menambahkan *Captcha*

Pada gambar 13 peneliti menampilkan pengujian *Login* SiMPaN dengan *SQL Injection* setelah menambahkan *captcha* pada *login website* SiMPaN sebelum menambahkan *SSL*. Apakah bisa *mengakses login* atau tidak.

Hasil

Gambar 14 menampilkan pengujian *Login* SiMPaN dengan *SQL Injection* setelah menambahkan *captcha* sebelum ditambahkan *SSL*. Hasilnya peretasan tidak dapat dilakukan, *SQL Injection* dapat juga dicegah dengan cara menerapkan keamanan *website* melalui penggunaan *captcha*, *OTP*, *anti injeksi* dibagian *sintax query* database pada *source code* dan menerapkan *validasi* tertentu seperti membatasi jumlah *penginputan* karakter serta mengubah jenis *penginputan* pada *form login*.

Gambar 14. Hasil Pengujian *Login* Simpan dengan *SQL Injection* Setelah Menambahkan *Captcha*



Gambar 15. Tampilan *Login* Setelah Ditambahkan *SSL* dan *Kode Captcha*

Tahap *verifikasi* (pengukuran tingkat keamanan dengan *SQLI*), pada tahap ini dilakukan *verifikasi* terhadap keamanan aplikasi untuk pemberitahuan kepada *admin* untuk dilakukan perbaikan-perbaikan atas dasar hasil *investigasi* dan pengujian pada aspek pemrograman [7].Setelah dilakukan serangan *SQL Injetion* (*Username*(acak) +' OR 1=1 --), ditemukan celah pada bagian *login*.

Tahap rekomendasi (memberikan rekomendasi perbaikan), pada tahap ini peneliti menambahkan *SSL* pada bagian *login* karena celah *security* yang terdapat di bagian *login* karena tidak ada *filtering,* atau bisa juga menambahkan *kode captcha* pada bagian bawah *login* karena *website* SiMPaN dibuat *localhost* maka tidak menutup kemungkinan akan terjadinya *human error* maka ditambahkannya *kode captcha* untuk menghindari *human error*. Kemudian kita tes lagi apabila tes gagal maka penambahan 2 *faktor autentichationnya* berhasil.

Kesimpulan dan Saran

Berdasarkan hasil penelitian diatas dapat disimpulnkan bahwa langkah-langkah pengujian keamanan aplikasi, termasuk pencegahan serangan *SQL Injection* dan serangan *bruce force attacker*, dapat meminimalisir kerentanan keamanan dalam aplikasi. Penambahan *kode captcha* dan implementasi *SSL* merupakan upaya yang efektif dalam meningkatkan keamanan aplikasi dan melindungi data serta pengguna dari serangan berbahaya. Implikasi dari informasi yang diberikan adalah perlunya pengujian keamanan aplikasi, pemahaman bahasa pemrograman dan sistem database, perlindungan terhadap serangan *SQL Injection*, serta tindakan perbaikan dan pemantauan yang berkelanjutan guna menjaga keamanan aplikasi. Saran untuk penelitian selanjutnya dalam bidang keamanan aplikasi adalah meliputi pengembangan teknik pencegahan yang lebih

efektif terhadap serangan *SQL Injection*, pengujian keamanan yang lebih komprehensif untuk mengatasi serangan lainnya seperti *XSS* atau *Injeksi* perintah, *integrasi* keamanan sejak awal pada tahap pengembangan aplikasi, penelitian tentang teknologi keamanan terkini seperti *AI* atau *Blockchain,* serta penelitian terhadap kerentanan dalam sistem manajemen basis data yang digunakan. Penelitian ini dapat memberikan kontribusi dalam meningkatkan keamanan aplikasi dan *mengidentifikasi* solusi yang lebih *efektif* dalam menghadapi serangan keamanan yang beragam.

Referensi

[1] J. Harefa, G. Prajena, A. Muhamad, E. Valin, S. Dewa, and S. Yuliandry, "SEA WAF : The Prevention of SQL Injection Attacks on Web Applications SEA WAF : The Prevention of SQL Injection Attacks on Web Applications," no. April, 2021, doi: 10.25046/aj060247.

[2] M. Metode, F. Multiple, C. Decision, M. Fmcdm, and D. Yogyakarta, "Indonesian Journal of Business Intelligence," vol. 3, no. 2, pp. 54–60, 2020.

[3] Y. A. Pohan, "Meningkatkan Keamanan Webserver Aplikasi Pelaporan Pajak Daerah Menggunakan Metode Penetration Testing Execution Standar," *J. Sistim Inf. dan Teknol.*, vol. 3, pp. 1–6, 2021, doi: 10.37034/jsisfotek.v3i1.36.

[4] N. Annggela, R. Andryani, U. Bina, and D. Palembang, "Analisis Kualitas Sistem Dapodik Untuk Pendataan Satuan Pendidikan di Kota Palembang Dengan Metode Webqual," 2019.

[5] P. Hendradi, "Analisis Keamanan E-learning Menggunakan Open Web Application Security Project (OWASP) studi kasus MOCA UNIMMA," *J. Inform.*, vol. 22, no. 02, pp. 132–138, 2022, [Online]. Available: https://jurnal.darmajaya.ac.id/index.php/JurnalInformatika/article/view/3327.

[6] S. Hidayatulloh and D. Saptadiaji, "Penetration Testing pada Website Universitas ARS Menggunakan Open Web Application Security Project (OWASP)," *J. Algoritm.*, vol. 18, no. 1, pp. 77–86, 2021, doi: 10.33364/algoritma/v.18-1.827.

[7] A. Bastian, H. Sujadi, and L. Abror, "Analisis Keamanan Aplikasi Data Pokok Pendidikan (DAPODIK) Menggunakan Penetration Testing Dan SQL Injection," *INFOTECH J.*, vol. 6, no. 2, pp. 65–70, 2020.

[8] R. Pangalila, "Penetration Testing Server Sistem Informasi Manajemen Dan Website Universitas Kristen Petra," *J. Teknol. Inf.*, vol. 3, no. 2, p. pp.271-p.276, 2015, [Online]. Available: http://publication.petra.ac.id/index.php/teknik-informatika/article/view/3145.

[9] S. P. Sitorus and R. A. Habibi, "Teknik Pencegahan *Penetrasi* SQL Injeksi Dengan Pengaturan Input Type Number dan Batasan Input Pada Form Login Website," *U-NET J. Tek. Inform.*, vol. 4, no. 2, pp. 26–33, 2020, doi: 10.52332/u-net.v4i2.303.

[10] A. D. Djayali, "Analisa Serangan SQL Injection pada Server pengisian Kartu Rencana Studi (KRS) Online," *J. Manaj. Inform. dan Komput.*, vol. 1, no. 1, pp. 16–24, 2020, [Online]. Available: https://jurnal.aikomternate.ac.id/index.php/jaminfokom.

[11] R. P. Adi, "Analisis Paket Data Pada Jalur Komunikasi SSL dengan Menggunakan Tools Wireshark untuk Keamanan Jalur Komunikasi (Studi Kasus Server Sistem Akademik Terpadu SMK Negeri 2 Salatiga)."

[12] M. Alenezi, M. Nadeem, and R. Asif, "SQL injection attacks countermeasures assessments," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 21, no. 2, pp. 1121–1131, 2020, doi: 10.11591/ijeecs.v21.i2.pp1121-1131.

[13] F. Rohimudin *et al.*, "Rancang Bangun Infrastruktur Cloud Full," vol. 28, no. 2, 2022, doi: 10.36309/goi.v28i2.171.

[14] D. Rusmana, "Rancang Bangun Pengaman Sistem Login Menggunakan Metode Captcha," *Incomtech*, vol. 10, no. 1, pp. 46–52, 2021, [Online]. Available: https://ejournal.istn.ac.id/index.php/incomtech/article/view/1061%0Ahttps://

ejournal.istn.ac.id/index.php/incomtech/article/download/1061/729.

[15] N. Nuryati *et al.*, "Two Factor Authentication Sistem Inventarisasi Barang dan Manajemen Dana Bantuan Operasional Sekolah Dinas Pendidikan Nasional," vol. 4, no. 2, pp. 1129–1136, 2022, doi: 10.47065/bits.v4i2.2297.