Analisis *Security Mitigation* Terhadap *Website* Akademik Penentuan Peminatan di Institusi XYZ Menggunakan Metode *Penetration Testing Execution Standard* (PTES)

1st Yuaraina Dirgantarizki Ryandi
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
yuarainaryandi@student.telkomuniversity.ac.id

2nd Umar Yunan Kurnia Septo Hediyanto
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
umaryunan@telkomuniversity.ac.id

3rd Adityas Widjajarto
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
adtwjrt@telkomuniversity.co.id

***Abstrak* — Dalam perkembangan teknologi pendidikan, keamanan informasi menjadi yang terpenting dalam melindungi data dan informasi dari ancaman atau serangan. Salah satunya serangan *SQL Injection* yang mengarah ke server database dengan menyisipkan pernyataan SQL berbahaya kemudian *database* mengeksekusi *query* tersebut. Penelitian dilakukan untuk mengetahui kondisi celah keamanan terhadap serangan *SQL Injection* pada *website* akademik penentuan peminatan di suatu institusi dan mengetahui mitigasi pada *website* tersebut. Pengujian ini perlu dilakukan karena *website* belum dilakukan uji keamanan karena saat proses pembuatannya tidak dilakukan proses *security testing* dan langsung memasuki tahapan *Go-Live*, sehingga kemungkinan terdapat celah keamanan, seperti kebocoran data pada *website* ataupun adanya perubahan data yang diperlukan, sehingga berdampak pada penyalahgunaan data/infomasi pengguna, berkurangnya kepercayaan pengguna, hingga merusak nama baik Institusi XYZ. Penelitian ini menerapkan kerangka kerja keamanan *Penetration Testing Execution Standard* (PTES) dengan bantuan *tools* disetiap tahapannya. Pengujian yang dilakukan berupa eksploitasi pada setiap tautan dengan parameter yang ditentukan. Hasil dari pengujian serangan *SQL Injection* terhadap *website* dengan lima *tools* yang dilakukan menunjukkan bahwa tidak terdapat celah keamanan yang ada, sehingga implementasi *security mitigation* tidak perlu dilakukan.**

***Kata kunci*— *Website*, PTES, *SQL Injection*, mitigasi keamanan**

I. PENDAHULUAN

Pada era saat ini perkembangan teknologi semakin pesat, terutama dalam sektor pendidikan. Tentunya hal tersebut perlu memperhatikan sebuah keamanan informasi yang dapat meminimalisir resiko, melindungi data, dan informasi dari ancaman atau serangan yang ada. Salah satu contohnya, serangan *SQL Injection* yang mengarah ke *server database* dengan menyisipkan pernyataan SQL berbahaya kemudian *database* mengeksekusi *query* tersebut. Serangan ini masih sering dilakukan dikarenakan kurangnya kesadaran keamanan para developer, kode yang rentan, hingga sistem yang tidak diperbarui. Karena hal tersebut, *SQL Injection* termasuk kategori *Top 3* (A03:2021 – *Injection*) pada OWASP *Top 10 Web Application Security Risk* [1]. Dengan adanya serangan tersebut penyerang dapat mencuri data, memanipulasi atau memodifikasi data guna menipu pengguna atau melakukan kejahatan lainnya.

Kondisi ini perlu menjadi perhatian bagi institusi khususnya sektor akademik atau pendidikan pengguna *web application*. Salah satu institusi pendidikan memiliki aplikasi penentuan peminatan berbasis *web application*. Hingga saat ini, website tersebut belum dilakukan uji keamanan karena saat proses pembuatan tidak dilakukan proses *security testing* dan langsung memasuki tahapan *Go-Live*, sehingga kemungkinan masih terdapat celah keamanan, seperti kebocoran data pada *website* ataupun adanya perubahan data yang diperlukan, sehingga berdampak pada penyalahgunaan data/infomasi pengguna, berkurangnya kepercayaan pengguna terhadap Institusi, hingga merusak nama baik Institusi XYZ. Maka dari hal tersebut, diperlukan adanya pengujian terhadap *website* akademik penentuan peminatan. Penelitian ini menerapkan metode atau kerangka kerja keamanan *Penetration Testing Execution Standard* (PTES) dengan tujuan mengetahui kemungkinan eksploitasi yang terjadi dalam sebuah sistem. Dalam simulasi penyerangannya akan disesuaikan juga dengan kebutuhan penelitian ini.

II. DASAR TEORI

A. Keamanan Sistem Informasi

Menurut G. J. Simons (1995), keamanan informasi adalah bagaimana usaha untuk dapat mencegah penipuan (*cheating*) atau mendeteksi adanya penipuan pada sistem yang berbasis informasi, di mana informasinya sendiri tidak memiliki arti fisik. Keamanan informasi juga harus memuat sejumlah aspek, yaitu *Confidentiality*, *Integrity*, dan *Availability* [2].

B. Security Mitigation

Security mitigation berkaitan dengan pengembangan sistem informasi yang memuat persyaratan keamanan *Confidentiality*, *Integrity*, dan *Availability*. *Security mitigation* berfokus pada strategi untuk membatasi dampak

ancaman terhadap sebuah sistem ataupun data yang disimpan yang mempengaruhi persyaratan keamanan CIA [3].

C. *Website* Akademik Penentuan Peminatan

Website akademik penentuan peminatan salah satu layanan akademik berupa *website* yang dikelola di Institusi XYZ. Website ini digunakan mahasiswa agar memudahkan mereka untuk memilih peminatan. Layanan yang ada, seperti pendaftaran peminatan, status pengajuan peminatan, hingga pengajuan untuk penukaran peminatan dengan mahasiswa lain.

D. *SQL Injection*

SQL Injection adalah jenis serangan yang diarahkan ke *server database* dengan mengirimkan pernyataan SQL berbahaya dikirim sebagai bagian dari *query* SQL dan *database* mengeksekusi *query* tersebut. Dengan adanya serangan tersebut penyerang dapat mengambil, menampilkan, menghapus, atau mengubah data yang diambil [4].

E. *Tools Penetration Testing*

1. Whois

Whois adalah alat yang digunakan untuk mengumpulkan informasi terkait *website* target sebelum melakukan penyerangan [5].

2. Nslookup

Nslookup (*name server lookup*) adalah alat *penetration testing* berupa *command line* sederhana yang digunakan untuk mendapatkan informasi terkait *IP Address* yang dimiliki suatu *website* maupun perangkat tertentu.

3. Nmap

Nmap (*Network Mapper*) adalah alat gratis digunakan untuk audit keamanan jaringan bertujuan menemukan *host* dan *service* suatu jaringan [5].

4. Traceroute atau Tracert

Traceroute atau Tracert adalah *command line* sederhana bertujuan melacak lintasan atau rute yang digunakan paket dari sumber ke tujuannya yang berada di satu atau sejumlah jaringan.

5. SQLmap

SQLmap adalah alat gratis digunakan untuk *penetration testing* dengan mendeteksi secara otomatis dan mengeksploitasi kelemahan injeksi SQL dan dapat mengambil alih *database* dari *web server* [6].

6. Burpsuite

Burpsuite adalah alat *penetration testing* populer dikalangan *hacker* dan dikembangkan oleh Portswigger. Fitur umum yang tersedia dalam Burpsuite, yaitu *Automation scanner vulnerability, Proxy, Intruder, Repeater*, dan *Decoder* [5].

7. jSQL Injection

jSQL Injection adalah alat gratis dan *cross-platform* (Windows, Linux, Mac OS X, Solaris) yang digunakan mencari dan eksploitasi kerentanan *SQL Injection* untuk mendapatkan informasi *database* yang ada pada *website* target [5].

8. SQLSus

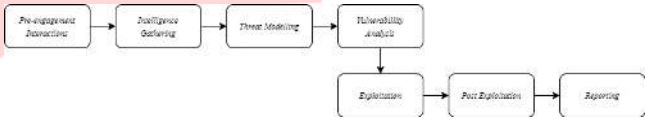
SQLSus adalah alat gratis MySQL Injection yang digunakan untuk melakukan eksploitasi *database* [7].

9. Havij

Havij adalah alat otomatis dikenbangkan oleh Perusahaan ITSecTeam yang digunakan untuk mencari dan melakukan eksploitasi berupa *SQL Injection* yang terdapat pada *website* target.

F. Metode *Penetration Testing Execution Standard* (PTES)

Metode PTES adalah Metode *Penetration Testing* yang dikembangkan oleh tim praktisi keamanan informasi dengan tujuan menjawab kebutuhan akan standar yang lengkap dan terkini dalam pengujian penetrasi (GeeksforGeeks, 2019) [8].



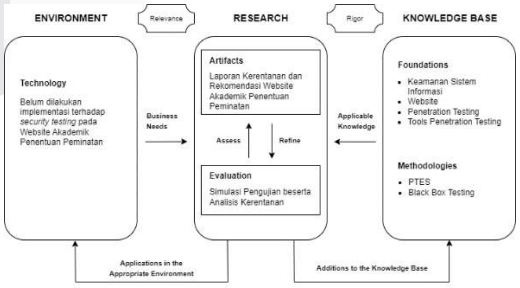
GAMBAR 1
METODE PTES

Pada gambar diatas terdapat tujuh tahapan PTES, yaitu *Pre-engagement Interaction, Intelligence Gathering, Threat Modelling, Vulnerability Analysis, Exploitation, Post Exploitation,* dan *Reporting.*

III. METODOLOGI PENELITIAN

A. Model Konseptual

Dalam penelitian Hevner, Model Konseptual digunakan untuk dapat memahami, menerapkan, dan mengevaluasi penelitian sistem informasi dengan menggunakan dua paradigma, yaitu *Design Science* dan *Behaviour Science*. Pada disiplin sistem informasi, kedua paradigma tersebut mempertemukan aktor, organisasi, dan teknologi (Hevner et al., 2010) [9].



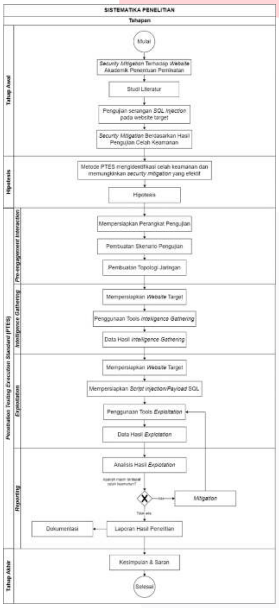
GAMBAR 1
MODEL KONSEPTUAL

Gambar diatas merupakan model konseptual dalam penelitian terbagi tiga aspek, yaitu *Environment, Research,* dan *Knowledge Base*. Bagian *Environment* terdapat teknologi yang dimana belum dilakukannya implementasi terhadap *security testing* pada *Website* Akademik Penentuan

Peminatan. Pada bagian *Research* memiliki fungsi untuk melakukan *Refine* (penyaringan) dan *Assess* (penilaian) yang merupakan faktor dalam penelitian ini. Faktor penelitian memiliki dua bagian, yaitu *Artifacts* berupa laporan kerentanan dan rekomendasi yang akan diberikan pada *website* akademik penentuan peminatan. Lalu, terdapat *Evaluation* yang berfokus pada simulasi pengujian yang akan dilakukan beserta analisis kerentanan yang didapatkan. Bagian *Knowledge Base* merupakan teori yang dijadikan sebagai acuan dalam melakukan penelitian. Pada *Foundation* atau dasar ilmu terdapat teori keamanan sistem informasi, *website, penetration testing*, serta dilengkapi dengan *beberapa tools penetration testing* yang mendukung selama penelitian. Terdapat *Methodologies* yang digunakan, yaitu PTES dan *Black Box Testing*.

B. Sistematika Penyelesaian Masalah

Sistematika penelitian digunakan dalam menjelaskan tahapan metode penelitian yang dilakukan dalam menyelesaikan masalah dalam penelitian.



GAMBAR 2
SISTEMATIKA PENELITIAN

IV. HASIL DAN PEMBAHASAN

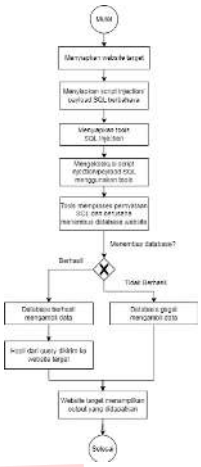
A. *Pre-engagement Interactions*

1. Hardware dan Software

Dalam pengujian dibutuhkan persiapan, seperti perangkat keras berupa *Main OS* dan *Virtual Machine.* Sedangkan, perangkat lunak berupa *Software Virtual Machine, Intelligence Gathering Tools* terdiri dari Whois, nslookup, Nmap, Traceroute, dan *Exploitation Tools* terdiri dari SQLmap, Burpsuite, jSQL Injection, SQLSus, dan Havij.

2. Skenario Pengujian SQL Injection

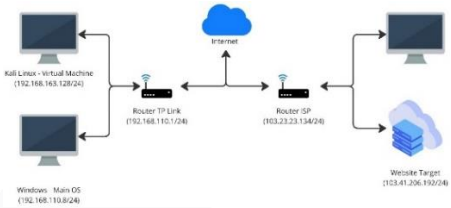
Dalam pengujian penetrasi, terdapat skenario pengujian yang akan dilakukan dengan tujuan menguji sistem dari *website* target dengan serangan *SQL Injection*.



GAMBAR 3
SKENARIO PENGUJIAN *SQL INJECTION*

3. Topologi Eksperimen

Topologi jaringan yang digunakan sebagai gambaran eksperimen terkait perangkat dan jaringan saling terhubung atau berkomunikasi satu sama lain.



GAMBAR 4
TOPOLOGI JARINGAN

B. *Intelligence Gathering*

1. Pengumpulan Informasi dengan Whois

Pengumpulan informasi dilakukan menggunakan Whois dengan didapatkan informasi *website*, sebagai berikut:

Information Gathering Result

Domain ID	PANDI-DO3307516
Domain Name	xxxxx.id
Created On	2020-09-22 11:09:09
Last Updated On	2022-09-18 10:09:09
Expiration Date	2023-09-22 00:09:09
Status	serverTransferProhibited
Status	clientTransferProhibited
Registrar Organization	Kilat Domain
Registrar URL	www.kilatdomain.id
Registrar Street	Pakuwon Tower, Lantai 9 Unit F dan G Jalan Casablanca Raya, Kav. 88
Registrar City	Jakarta Selatan
Registrar State/Province	Jakarta
Registrar Postal Code	12870
Registrar Country	ID
Registrar Phone	02129682828
Registrar Email	registrar@kilatdomain.id
Name Server	ns1.kilatdomain.id
Name Server	ns2.kilatdomain.id
DNSSEC	*Unsigned*

GAMBAR 5
PENGUMPULAN INFORMASI DENGAN WHOIS

Untuk hasil yang didapatkan Whosis berupa informasi informasi domain yang ada pada *website.*

2. Pengumpulan Informasi dengan Nmap

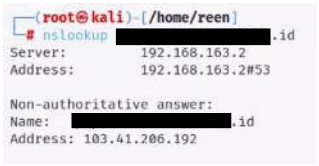
Pengumpulan informasi dilakukan menggunakan Nmap didapatkan informasi *port* dan *service* pada *website.*

PORT	STATE	SERVICE	VERSION
22/tcp	open	ssh	OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux protocol 2.0)
53/tcp	open	domain	(generic dns response: NOTIMP)
80/tcp	open	http	nginx 1.18.0 (Ubuntu)
83/tcp	open	http	Apache httpd 2.4.10 ((Debian))
88/tcp	open	http	Apache httpd 2.4.10 ((Debian))
443/tcp	open	ssl/http	nginx 1.18.0 (Ubuntu)
8001/tcp	open	vcom-tunnel?	
8002/tcp	open	teradataordbms?	
8080/tcp	open	http-proxy	
8084/tcp	open	websnp?	

GAMBAR 6
PENGUMPULAN INFORMASI DENGAN NMAP

3. Pengumpulan Informasi dengan Nslookup

Pengumpulan informasi dilakukan menggunakan Nslookup didapatkan *IP Address* 103.41.206.192.



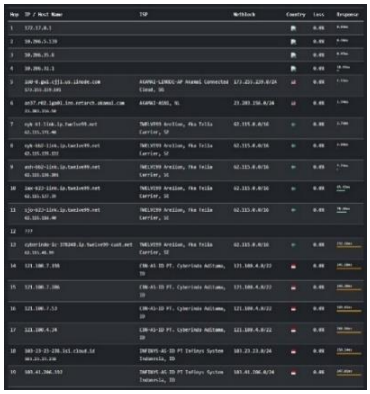
GAMBAR 7
PENGUMPULAN INFORMASI DENGAN NSLOOKUP

4. Pengumpulan Informasi dengan Traceroute

Pengumpulan informasi dilakukan menggunakan Traceroute untuk melacak rute pengiriman paket *IP Address* dari *website* target.

GAMBAR 8
PENGUMPULAN INFORMASI DENGAN TRACEROUTE

Didapatkan sebelas rute yang dilewati. Untuk informasi lebih detail pengecekan Traceroute melalui *online website*.



GAMBAR 9
HASIL TRACEROUTE *ONLINE*

GAMBAR 10
LOKASI IP ADDRESS HASIL TRACEROUTE

Hasil didapatkan sejumlah informasi detail terkait rute yang dilewati termasuk *ISP, Netblock, Country, Loss,* dan *Response.*

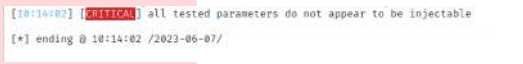
C. *Exploitation*

1. Pengujian SQL Injection Menggunakan SQLmap

Pengujian digunakan dua metode, yaitu GET dan POST.

IV.3.1.1 Metode GET

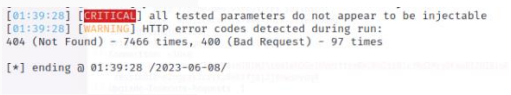
1) Tidak menggunakan parameter atau kosong "/"



GAMBAR 11
HASIL SQLMAP DENGAN PARAMETER KOSONG

Hasil pengujian "***all tested parameters do not appear to be injectable***" artinya alat tidak menemukan adanya parameter yang rentan untuk dapat diinjeksi, maka tidak ada informasi atau data sensitif yang dapat diperoleh dari *database website* target.

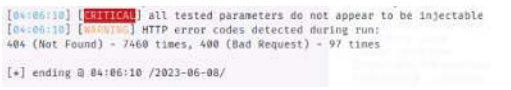
2) Menggunakan parameter "mahasiswa/seleksi_peminatan/"



GAMBAR 12
HASIL SQLMAP DENGAN "mahasiswa/seleksi_peminatan/"

Didapatkannya status kode 404 (*Not Found*) artinya sumber daya yang diminta tidak ditemukan selama injeksi dan status kode 400 (*Bad Request*) artinya permintaan yang dikirim klien tidak *valid* atau kesalahan sintaksis pada *server*. Hasil pengujian "***all tested parameters do not appear to be injectable***" yang artinya alat tidak menemukan adanya parameter rentan untuk diinjeksi, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

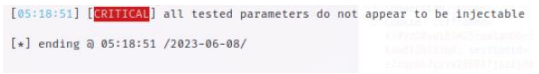
3) Menggunakan parameter "mahasiswa/tukar_peminatan/"



GAMBAR 13
HASIL SQLMAP DENGAN "mahasiswa/seleksi_peminatan/"

Didapatkannya status kode 404 (*Not Found*) artinya sumber daya yang diminta tidak ditemukan selama injeksi dan status kode 400 (*Bad Request*) artinya permintaan yang dikirim klien tidak valid atau kesalahan sintaksis pada *server*. Hasil pengujian "***all tested parameters do not appear to be injectable***" artinya alat tidak menemukan adanya parameter yang rentan untuk diinjeksi, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

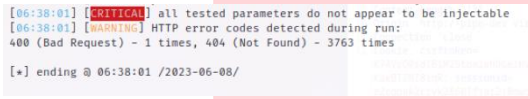
4) Menggunakan parameter “numberid=1202190036”



GAMBAR 14
HASIL SQLMAP DENGAN “numberid=1202190036”

Hasil pengujian “***all tested parameters do not appear to be injectable***” artinya alat tidak menemukan adanya parameter yang rentan untuk dapat diinjeksi, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

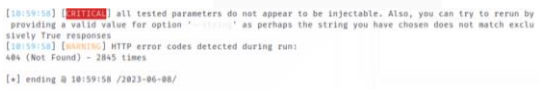
5) Menggunakan parameter “id=321” pada menu Seleksi Peminatan



GAMBAR 15
HASIL SQLMAP DENGAN “id=321”

Didapatkannya status kode 404 (*Not Found*) artinya sumber daya yang diminta tidak ditemukan selama injeksi dan status kode 400 (*Bad Request*) artinya permintaan yang dikirim klien tidak valid atau kesalahan sintaksis pada *server*. Hasil pengujian “***all tested parameters do not appear to be injectable***” artinya alat tidak menemukan adanya parameter yang rentan untuk diinjeksi, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

6) Menggunakan parameter “id=1” pada menu Tukar Peminatan

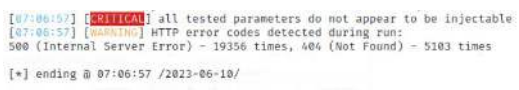


GAMBAR 16
HASIL SQLMAP DENGAN “id=1”

Didapatkannya status kode 404 (*Not Found*) artinya sumber daya yang diminta tidak ditemukan selama proses injeksi. Hasil pengujian “***all tested parameters do not appear to be injectable***” artinya alat tidak menemukan adanya parameter yang rentan untuk diinjeksi, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

IV.3.1.2 Metode POST

a. Menggunakan parameter “*username*” dan “*password*” pada *Login Page*



GAMBAR 17
HASIL SQLMAP DENGAN “*username*” dan “*password*”

Didapatkannya status kode 500 (*Internal Server Error*) artinya terjadinya kesalahan pada internal *server* yang menghalangi proses permintaan dan status kode 404 (*Not Found*) artinya sumber daya yang diminta tidak ditemukan selama injeksi. Hasil pengujian “***all tested parameters do not appear to be injectable***” artinya alat tidak menemukan adanya parameter yang rentan untuk diinjeksi, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

2. Pengujian SQL Injection Menggunakan Burpsuite

Pengujian digunakan dua metode, yaitu GET dan POST.

IV.3.2.1 Metode GET

a. Tidak menggunakan parameter atau kosong “/”

GAMBAR 18
HASIL BURPSUITE DENGAN PARAMETER KOSONG

Didapatkan status kode 400 (*Bad Request*) artinya permintaan yang dikirim klien tidak valid atau kesalahan sintaksis pada *server* dan status kode 404 (*Not Found*) artinya sumber daya yang diminta tidak ditemukan selama injeksi. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

b. Menggunakan parameter “mahasiswa/seleksi_peminatan/”

GAMBAR 19
HASIL BURPSUITE “mahasiswa/seleksi_peminatan/”

Didapatkannya status status kode 200 (OK) artinya respon permintaan dan server dianggap *valid*, namun *website* tidak menampilkan apapun. Didapatkan status kode 400 (*Bad Request*) artinya permintaan yang dikirim klien tidak *valid* atau kesalahan sintaksis pada *server* dan status kode 404 (*Not Found*) artinya sumber daya yang diminta tidak ditemukan selama injeksi. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

c. Menggunakan parameter “mahasiswa/tukar_peminatan/”

GAMBAR 20
HASIL BURPSUITE “mahasiswa/tukar_peminatan/”

Didapatkannya status status kode 200 (OK) artinya respon permintaan dan server dianggap valid, namun *website* tidak menampilkan apapun. Didapatkan status kode 400 (*Bad Request*) artinya permintaan yang dikirim klien tidak *valid* atau

kesalahan sintaksis pada *server* dan status kode 404 (*Not Found*) artinya sumber daya yang diminta tidak ditemukan selama injeksi. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

d. Menggunakan parameter "numberid=1202190036"

GAMBAR 21
HASIL BURPSUITE "numberid=1202190036"

Didapatkannya status status kode 200 (OK) artinya respon permintaan dan server dianggap *valid*, namun *website* tidak menampilkan apapun. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang dapat diperoleh dari *database website* target.

e. Menggunakan parameter "id=321" pada menu Seleksi Peminatan

GAMBAR 22
HASIL BURPSUITE "id=321"

Didapatkannya status status kode 200 (OK) artinya respon permintaan dan server dianggap *valid*, namun *website* tidak menampilkan apapun. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

f. Menggunakan parameter "id=1" pada menu Tukar Peminatan
Didapatkannya status kode 200 (OK) sebanyak 125 dari 125 daftar payload. Ini menunjukkan permintaan *server* dan *respons* dianggap *valid*. Hal ini adanya kemungkinan injeksi dilakukan berhasil tanpa menyebabkan kesalahan pada *server*. Namun, tidak terdapat perbedaan dari masing *payload* mengenai ukuran data. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

IV.3.2.2 Metode POST

a. Menggunakan parameter "*username*" dan "*password*" pada *Login Page*
Didapatkannya status kode 200 (OK) sebanyak 125 dari 125 daftar *payload*. Ini menunjukkan proses permintaan *server* dan *respons* dianggap valid. Hal ini adanya kemungkinan injeksi dilakukan berhasil tanpa menyebabkan kesalahan pada server. Namun, tidak terdapat perbedaan dari masing *payload* mengenai ukuran data. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

3. Pengujian SQL Injection Menggunakan jSQL Injection
Pengujian digunakan dua metode, yaitu GET dan POST.

IV.3.3.1 Metode GET

a. Tidak menggunakan parameter atau kosong "/"

GAMBAR 23
HASIL JSQL INJECTION PARAMETER KOSONG

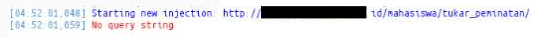
Hasil pengujian "*No query string*", sehingga tidak dapat melakukan *SQL Injection* karena parameter tidak mengandung *query* tertentu. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

b. Menggunakan parameter "mahasiswa/seleksi_peminatan/"

GAMBAR 25
HASIL JSQL INJECTION "mahasiswa/seleksi_peminatan/"

Hasil pengujian "*No query string*", sehingga tidak dapat melakukan *SQL Injection* karena parameter tidak mengandung *query* tertentu. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi atau data sensitif yang diperoleh dari *database website* target.

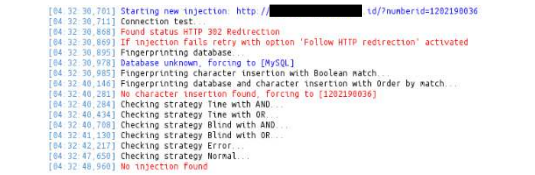
c. Menggunakan parameter "mahasiswa/tukar_peminatan/"



GAMBAR 26
HASIL JSQL INJECTION "mahasiswa/tukar_peminatan/"

Hasil pengujian "*No query string*", sehingga tidak dapat melakukan *SQL Injection* karena parameter tidak mengandung *query* tertentu. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

d. Menggunakan parameter "numberid=1202190036"

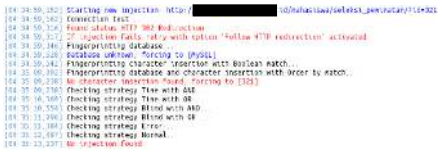


GAMBAR 27
HASIL JSQL INJECTION "numberid=1202190036"

Didapatkannya status kode 302 (*Redirection*) artinya sumber daya yang diminta ditemukan, namun server memberikan *respons* untuk mengalihkan klien ke lokasi sumber daya yang

berbeda. Hasil pengujian "***No injection found***" artinya alat tidak menemukan adanya titik injeksi yang rentan terhadap *SQL Injectio*n, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

e. Menggunakan parameter "id=321" pada menu Seleksi Peminatan



GAMBAR 28
HASIL JSQL INJECTION "id=321"

Didapatkannya status kode 302 (*Redirection*) artinya sumber daya yang diminta ditemukan, namun server memberikan *respons* untuk mengalihkan klien ke lokasi sumber daya yang berbeda. Hasil pengujian "***No injection found***" artinya alat tidak menemukan adanya titik injeksi yang rentan terhadap *SQL Injectio*n, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

f. Menggunakan parameter "id=1" pada menu Tukar Peminatan

GAMBAR 29
HASIL JSQL INJECTION "id=1"

Didapatkannya status kode 302 (*Redirection*) artinya sumber daya yang diminta ditemukan, namun server memberikan *respons* untuk mengalihkan klien ke lokasi sumber daya yang berbeda. Hasil pengujian "***No injection found***" artinya alat tidak menemukan adanya titik injeksi yang rentan terhadap *SQL Injectio*n, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

IV.3.3.2 Metode POST

a. Menggunakan parameter "*username*" dan "*password*" pada *Login Page*

GAMBAR 30
HASIL JSQL INJECTION "*username*" dan "*password*"

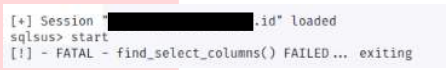
Didapatkannya status kode 200 (OK) artinya permintaan server dan respons kepada server dianggap valid. Hal ini menunjukkan adanya kemungkinan injeksi dilakukan berhasil tanpa menyebabkan kesalahan pada server. "***Found 1 ignored <form> in HTML body***" artinya *tools* mengabaikan elemen <form> yang digunakan untuk *input* pengguna karena tidak relevan dimana *tools* hanya fokus terhadap kerentanan parameter pada tautan/URL. Hasil pengujian "***No injection found***" artinya alat tidak menemukan adanya titik injeksi yang rentan terhadap *SQL Injectio*n, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

4. Pengujian SQL Injection Menggunakan SQLSus

Pengujian digunakan dua metode, yaitu GET dan POST.

IV.3.4.1 Metode GET

a. Tidak menggunakan parameter atau kosong "/"



GAMBAR 31
HASIL SQLSUS PARAMETER KOSONG

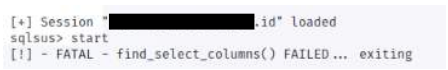
Didapatkannya pesan "***FATAL – find_select_columns( ) FAILED … exiting***" artinya alat tidak menemukan adanya kolom *database* pada *website* target yang diidentifikasi untuk dilakukan *SQL Injection*, sehingga tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

b. Menggunakan parameter "mahasiswa/seleksi_peminatan/"

GAMBAR 32
HASIL SQLSUS PARAMETER "mahasiswa/seleksi_peminatan/"

Didapatkannya pesan "***FATAL – find_select_columns( ) FAILED … exiting***" artinya alat tidak menemukan adanya kolom *database* pada *website* target yang diidentifikasi untuk dilakukan *SQL Injection*, sehingga tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

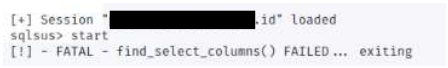
c. Menggunakan parameter "mahasiswa/tukar_peminatan/"



GAMBAR 33
HASIL SQLSUS PARAMETER "mahasiswa/tukar_peminatan/"

Didapatkannya pesan "***FATAL – find_select_columns( ) FAILED … exiting***" artinya alat tidak menemukan adanya kolom *database* pada *website* target yang diidentifikasi untuk dilakukan *SQL Injection*, sehingga tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

d. Menggunakan parameter “numberid=1202190036”



GAMBAR 34
HASIL SQLSUS PARAMETER “numberid=1202190036”

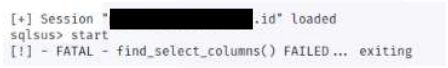
Didapatkannya pesan “***FATAL – find_select_columns( ) FAILED … exiting***” artinya alat tidak menemukan adanya kolom *database* pada *website* target yang diidentifikasi untuk dilakukan *SQL Injection*, sehingga tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

e. Menggunakan parameter “id=321” pada menu Seleksi Peminatan

GAMBAR 35
HASIL SQLSUS PARAMETER “id=321”

Didapatkannya pesan “***FATAL – find_select_columns( ) FAILED … exiting***” artinya alat tidak menemukan adanya kolom *database* pada *website* target yang diidentifikasi untuk dilakukan *SQL Injection*, sehingga tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

f. Menggunakan parameter “id=1” pada menu Tukar Peminatan

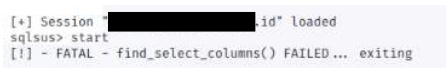


GAMBAR 36
HASIL SQLSUS PARAMETER “id=1”

Didapatkannya pesan “***FATAL – find_select_columns( ) FAILED … exiting***” artinya alat tidak menemukan adanya kolom *database* pada *website* target yang diidentifikasi untuk dilakukan *SQL Injection*, sehingga tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

IV.3.4.2 Metode POST

a. Menggunakan parameter “*username*” dan “*password*” pada *Login Page*



GAMBAR 37
HASIL SQLSUS PARAMETER “*username*” dan *“password”*

Didapatkannya pesan “***FATAL – find_select_columns( ) FAILED … exiting***” artinya alat tidak menemukan adanya kolom *database* pada *website* target yang diidentifikasi untuk dilakukan *SQL Injection*, sehingga tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

Pengujian *SQL Injection* Menggunakan Havij

Pengujian digunakan dua metode, yaitu GET dan POST.

IV.3.5.1 Metode GET

a. Tidak menggunakan parameter atau kosong “/”

GAMBAR 38
HASIL HAVIJ PARAMETER KOSONG

Didapatkan hasil “***Target url must have an input parameter!***” yang artinya alat tidak dapat mengidentifikasi menggunakan URL/tautan ini dikarenakan harus mencantumkan spesifik parameter. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

b. Menggunakan parameter “mahasiswa/seleksi_peminatan/”

GAMBAR 39
HASIL HAVIJ PARAMETER “mahasiswa/seleksi_peminatan/”

Didapatkan hasil “***Target url must have an input parameter!***” yang artinya alat tidak dapat mengidentifikasi menggunakan URL/tautan ini dikarenakan harus mencantumkan spesifik parameter. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

c. Menggunakan parameter “mahasiswa/tukar_peminatan/”

GAMBAR 40
HASIL HAVIJ PARAMETER “mahasiswa/tukar_peminatan/”

Didapatkan hasil “***Target url must have an input parameter!***” yang artinya alat tidak dapat mengidentifikasi menggunakan URL/tautan ini dikarenakan harus mencantumkan spesifik parameter. Karena tidak dapat melakukan *SQL Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

d. Menggunakan parameter “numberid=1202190036”

GAMBAR 41
HASIL HAVIJ PARAMETER "numberid=1202190036"

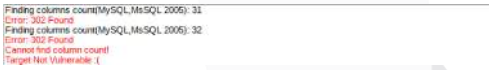
Didapatkannya status kode 302 (*Redirection*) artinya sumber daya yang diminta ditemukan, namun *server* memberikan *respons* untuk mengalihkan klien ke lokasi sumber daya yang berbeda. Hasil pengujian menggunakan Havij menunjukkan "***Target Not Vulnerable:(***" artinya alat tidak menemukan adanya kerentanan *SQL Injection* selama pengujian, sehingga tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

e. Menggunakan parameter "id=321" pada menu Seleksi Peminatan

GAMBAR 42
HASIL HAVIJ PARAMETER "id=321"

Didapatkannya status kode 302 (*Redirection*) artinya sumber daya yang diminta ditemukan, namun *server* memberikan *respons* untuk mengalihkan klien ke lokasi sumber daya yang berbeda. Hasil pengujian menggunakan Havij menunjukkan "***Target Not Vulnerable:(***" artinya alat tidak menemukan adanya kerentanan *SQL Injection* selama pengujian, sehingga tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

f. Menggunakan parameter "id=1" pada menu Tukar Peminatan



GAMBAR 43
HASIL HAVIJ PARAMETER "id=1"

Didapatkannya status kode 302 (*Redirection*) artinya sumber daya yang diminta ditemukan, namun *server* memberikan *respons* untuk mengalihkan klien ke lokasi sumber daya yang berbeda. Hasil pengujian menggunakan Havij menunjukkan "***Target Not Vulnerable:(***" artinya alat tidak menemukan adanya kerentanan *SQL Injection* selama pengujian, sehingga tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

IV.3.5.2 Metode POST

a. Menggunakan parameter "*username*" dan "*password*" pada *Login Page*

GAMBAR 44
HASIL HAVIJ PARAMETER "*username*" dan *"password"*

Didapatkannya status kode 403 (*Forbidden*) artinya klien tidak memiliki hak akses ke konten secara tidak sah, sehingga *server* menolak memberikan sumber daya yang diminta. Hal ini diindikasikan terdapat pembatasan keamanan pada sumber daya pada *server web*, WAF yang secara otomatis memblokir permintaan, ataupun kesalahan konfigurasi pada *server* itu sendiri. Hasil pengujian menggunakan Havij menunjukkan "***Target Not Vulnerable:(***" artinya alat tidak menemukan adanya kerentanan *SQL Injection* selama pengujian, sehingga tidak ada informasi atau data sensitif yang diperoleh dari *database website* target.

D. *Reporting*

1. Perbandingan Hipotesis dengan Hasil Alat Pengujian

Metode	Parameter	Hipotesis Awal	Hasil Pengujian dengan Alat				
			SQLmap	Burpsuite	jSQL Injection	SQLSus	Havij
GET	Tanpa parameter atau kosong "/"	Terdapat celah keamanan *SQL Injection* pada parameter kosong "/"	Tidak ditemukan celah *SQL Injection* pada parameter kosong "/"	Tidak ditemukan celah *SQL Injection* pada parameter kosong "/"	Tidak dapat melakukan *SQL Injection* pada parameter kosong "/" karena *tools* harus mencantumkan *query string* atau spesifik parameter	Tidak ditemukan celah *SQL Injection* pada parameter kosong "/"	Tidak dapat melakukan *SQL Injection* pada parameter kosong "/" karena *tools* harus mencantumkan spesifik parameter
	mahasiswa/ seleksi_peminatan	Terdapat celah keamanan *SQL Injection* pada parameter "mahasiswa/ seleksi_peminatan "	Tidak ditemukan celah *SQL Injection* pada parameter "mahasiswa/ seleksi_peminatan"	Tidak ditemukan celah *SQL Injection* pada parameter "mahasiswa/ seleksi_peminatan"	Tidak dapat melakukan *SQL Injection* pada parameter "mahasiswa/ seleksi_peminatan" karena *tools* harus mencantumkan *query string* atau spesifik parameter	Tidak ditemukan celah *SQL Injection* pada parameter "mahasiswa/ seleksi_peminatan"	Tidak dapat melakukan *SQL Injection* pada parameter "mahasiswa/ seleksi_peminatan" karena *tools* harus mencantumkan spesifik parameter
	mahasiswa/ tukar_peminatan"	Terdapat celah keamanan *SQL Injection* pada parameter "mahasiswa/ tukar_peminatan"	Tidak ditemukan celah *SQL Injection* pada parameter "mahasiswa/ tukar_ peminatan"	Tidak ditemukan celah *SQL Injection* pada parameter "mahasiswa/ tukar_ peminatan"	Tidak dapat melakukan *SQL Injection* pada parameter "mahasiswa/ tukar_ peminatan" karena *tools* harus mencantumkan *query string* atau spesifik parameter	Tidak ditemukan celah *SQL Injection* pada parameter "mahasiswa/ tukar_ peminatan"	Tidak dapat melakukan *SQL Injection* pada parameter "mahasiswa/ tukar_ peminatan" karena *tools* harus mencantumkan *query string* atau spesifik parameter
	numberid= 1202190036	Terdapat celah keamanan *SQL Injection* pada parameter "numberid= 1202190036"	Tidak ditemukan celah *SQL Injection* pada parameter "numberid= 1202190036"	Tidak ditemukan celah *SQL Injection* pada parameter "numberid= 1202190036"	Tidak ditemukan celah *SQL Injection* pada parameter "numberid= 1202190036"	Tidak ditemukan celah *SQL Injection* pada parameter "numberid= 1202190036"	Tidak ditemukan celah *SQL Injection* pada parameter "numberid= 1202190036"
	id=321 pada menu Seleksi Peminatan	Terdapat celah keamanan *SQL Injection* pada parameter "id=321" pada menu Seleksi Peminatan	Tidak ditemukan celah *SQL Injection* pada parameter "id=321" pada menu Seleksi Peminatan	Tidak ditemukan celah *SQL Injection* pada parameter "id=321" pada menu Seleksi Peminatan	Tidak ditemukan celah *SQL Injection* pada parameter "id=321" pada menu Seleksi Peminatan	Tidak ditemukan celah *SQL Injection* pada parameter "id=321" pada menu Seleksi Peminatan	Tidak ditemukan celah *SQL Injection* pada parameter "id=321" pada menu Seleksi Peminatan
	id=1 pada menu Tukar Peminatan	Terdapat celah keamanan *SQL Injection* pada parameter "id=1" pada menu Tukar Peminatan	Tidak ditemukan celah *SQL Injection* pada parameter "id=1" pada menu Tukar Peminatan	Tidak ditemukan celah *SQL Injection* pada parameter "id=1" pada menu Tukar Peminatan	Tidak ditemukan celah *SQL Injection* pada parameter "id=1" pada menu Tukar Peminatan	Tidak ditemukan celah *SQL Injection* pada parameter "id=1" pada menu Tukar Peminatan	Tidak ditemukan celah *SQL Injection* pada parameter "id=1" pada menu Tukar Peminatan
POST	*username* dan *password* pada *Login Page*	Terdapat celah keamanan *SQL Injection* pada parameter "*username*" dan "*password*" pada *Login Page*	Tidak ditemukan celah *SQL Injection* pada parameter "*username*" dan "*password*" pada *Login Page*	Tidak ditemukan celah *SQL Injection* pada parameter "*username*" dan "*password*" pada *Login Page*	Tidak ditemukan celah *SQL Injection* pada parameter "*username*" dan "*password*" pada *Login Page*	Tidak ditemukan celah *SQL Injection* pada parameter "*username*" dan "*password*" pada *Login Page*	Tidak ditemukan celah *SQL Injection* pada parameter "*username*" dan "*password*" pada *Login Page*

GAMBAR 45
HIPOTESIS DAN HASIL PENGUJIAN

Hasil dari pengujian menggunakan lima alat yang digunakan menunjukkan bahwa tidak ditemukannya celah keamanan atau kerentanan yang dapat ditembus pada *Website* Akademik Penentuan Peminatan.

2. Faktor Penyebab Website Dikaregorikan Aman dan Sulit Dieksploitasi

Dari pengujian *website* dilakukan memiliki hasil yang gagal atau tidak berhasil ditemukannya celah untuk melakukan injeksi. Ini dapat dikatakan *website* sudah terbukti aman dari serangan *SQL Injection* dengan memiliki hasil nilai

keamanan yang sangat baik. Terdapat beberapa faktor kemungkinan yang menyebabkan *website* tidak dapat dieksploitasi, seperti:

1. *Website* target dibangun menggunakan *framework* Django. Ini diketahui dari hasil pengujian Burpsuite saat mengeluarkan *respons* dengan status kode 404 (*Not Found*).

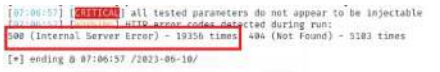
GAMBAR 46
KODE 404 (*NOT FOUND)* BURPSUITE

Django secara *default* terlindungi dari serangan *SQL Injection* karena menggunakan *Object Relational Mapping* (ORM) yang berarti developer tidak perlu menuliskan *query* SQL secara langsung, melainkan menggunakan API QuerySet bawaan. Django kemudian mengonversi dari *query* Python menjadi *query* SQL yang membuatnya berkomunikasi dengan *database*. Selain itu, karena *query* Django dibangun menggunakan *query parametrization* yang dimana kode *query* SQL ditentukan secara terpisah. Jika parameter dibuat oleh pengguna yang membuat tidak terlalu aman, sehingga parameter tersebut di-*escape* oleh *driver database* yang mendasarinya (docs.djangoproject.com, 2023) [10].

2. Kemungkinan besar *Web Application Firewal* (WAF) sudah diimplementasikan dan aktif saat proses *hosting website* menjadikan serangan *SQL Injection* sulit dilakukan atau menembus *database* dari *website* target. Ini diketahui dari hasil pengujian *tools* Havij dan SQLmap. Salah satu *output* Havij menghasilkan status kode 403 (*Forbidden*) artinya klien tidak memiliki hak akses ke konten secara tidak sah, sehingga server menolak memberikan sumber daya yang diminta. Hal ini diindikasikan terdapat pembatasan keamanan pada sumber daya pada *server web*, WAF secara otomatis memblokir permintaan, ataupun kesalahan konfigurasi pada *server* itu sendiri.

GAMBAR 47
KODE 403 (*FORBIDDEN)* HAVIJ

Kemudian, salah satu *output* SQLmap menghasilkan status kode 500 (*Internal Server Error*) artinya terjadinya kesalahan pada *internal server* yang menghalangi proses permintaan. Hal ini diindikasikan server menyembunyikan informasi sebagai bentuk perlindungan yang diterapkan oleh WAF.



GAMBAR 48
KODE 500 (*INTERNAL SERVER ERROR)* SQLMAP

Selain dari hasil pengujian eksploitasi, didapatkan juga saat proses *Intelligence Gathering* saat pencarian *traceroute* menggunakan *online website*.

GAMBAR 49
NAMA ISP TERLACAK TRACEROUTE

Didapatkan perusahaan ISP dan jika ditelusuri perusahaan sudah otomatis menyediakan WAF untuk perlindungan *website* dari berbagai kemungkinan serangan yang ada.

3. Developer kemungkinan menonaktifkan notifikasi atau tampilan *error* pada *website*, sehingga selama pengujian tidak terdapat celah website yang terlihat. Ini dapat diketahui dari hasil pengujian *tools* Burpsuite.
4. Developer sudah menerapkan validasi *input* karakter *whitelisting* dan *blacklisting* atau *filtering* terhadap metakarakter, seperti (&, ;, `, ', \, “, |, *, ?, ~, <, >, ^, (,), [,], {, }, $, \n, \r) [4].
5. Developer sudah mengatur pembatasan format pengisian, misalnya pada *form username* hanya dibatasi dengan penggunaan huruf maksimal 30 karakter.

3. Rekomendasi

Meskipun dari hasil pengujian gagal atau tidak berhasil, penelitian ini akan memberikan rekomendasi kedepannya agar developer tetap meningkatkan kualitas level keamanan website tersebut, sebagai berikut:

Rekomendasi secara umum:

a. Membuat protokol menjadi *HTTPS (Hypertext Transfer Protocol Secure)* dengan memberikan sertifikat SSL yang akan mengamankan koneksi pengguna dengan *server* protokol SSL (*Secure Sockets Layer*)/TLS (*Transport Layer Security*).
b. Mengganti *port* dengan tidak menggunakan *common port database* biasanya, seperti HTTP pada *port* 80 dan HTTPS pada *port* 443.
c. Menambahkan MFA (*Multifactor Authentication*) yang digunakan saat melakukan login.
d. Membuat pengguna selama mengakses *website* menggunakan VPN (*Virtual Private Network*).

Rekomendasi pencegahan serangan *SQL Injection:*

a. Menggunakan versi *framework* Django dan *database* (MySQL) terbaru dengan *update* berkala agar mendapatkan pembaruan keamanan dan peningkatan kinerja, serta kestabilan dari versi terbaru yang digunakan.
b. Menghindari penggunaan query raw(), extra(), RawSQL, dan *query* SQL secara langsung, serta sebaiknya tidak menggunakan tanda kutip di sekitar **%s** placeholder/variabel pengikat dalam *string* SQL dengan tujuan terhindar dari serangan *SQL Injection*.
c. Selalu menyimpan *password* dan data rahasia dengan format terenkripsi agar penyerang tidak mudah dalam melakukan peretasan data.
d. Perlunya pemisahan *database* antara *username* dengan *password* bertujuan mengantisipasi jika penyerang melakukan peretasan karena harus memahami struktur *database website* dan berguna ketika salah satu *database* terekspose yang lainnya masih terlindungi.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil penelitian analisis *security mitigation* terhadap *Website* Akademik Penentuan Peminatan menggunakan Metode *Penetration Testing Execution Standard* (PTES) dapat simpulkan:

1. Hasil dari pengujian serangan *SQL Injection* terhadap *website* akademik penentuan peminatan berdasarkan analisis dengan lima alat pengujian atau *tools* yang dilakukan menunjukkan bahwa tidak terdapat celah keamanan yang ada. Ini dapat dikatakan *website* tersebut sudah terbukti aman dari serangan *SQL Injection* dengan memiliki hasil nilai keamanan yang sangat baik.
2. Saat ini tidak diperlukan *security mitigation* karena berdasarkan hasil pengujian lima *exploitation tools* tidak ditemukan celah keamanan dari hasil *profiling SQL Injection*.

B. Saran

Terdapat saran untuk pengembangan *website* kedepannya, sebagai berikut:

1. Kepada developer berupa rekomendasi dan solusi perbaikan celah keamanan yang ada demi meningkatkan kualitas *leve*l keamanan *website* menggunakan acuan dari hasil penelitian yang dilakukan dengan harapan agar *website* tersebut terhindar dari kerentanan atau serangan lainnya yang akan datang.
2. Melakukan pengujian secara berkala dengan beragam jenis eksploitasi dan alat pengujian terkini untuk menemukan berbagai celah keamanan yang ada.

REFERENSI

[1] OWASP Top 10 Team, “A03:2021 – injection,” A03 Injection - OWASP Top 10:2021, https://owasp.org/Top10/A03_2021-Injection/ (accessed Jul. 10, 2023).

[2] Siregar, Keamanan Informasi, https://www.djkn.kemenkeu.go.id/kanwil-rsk/baca-artikel/13120/Keamanan-Informasi.html (accessed Jul. 6, 2023).

[3] V. Page, M. Dixon, and I. Choudhury, “Security risk mitigation for information systems,” BT Technology Journal, vol. 25, no. 1, pp. 118–127, 2007. doi:10.1007/s10550-007-0014-8

[4] R. Hermawan, “Teknik Uji Penetrasi web server Menggunakan SQL injection Dengan SQLmap DI kalilinux,” STRING (Satuan Tulisan Riset dan Inovasi Teknologi), vol. 6, no. 2, p. 210, 2021. doi:10.30998/string.v6i2.11477

[5] “Kali tools: Kali linux tools,” Kali Linux, https://www.kali.org/tools/ (accessed Jul. 3, 2023).

[6] “Sqlmap®,” sqlmap, https://sqlmap.org/ (accessed Jul.16, 2023).

[7] Sativouf, “News,” sqlsus, https://sqlsus.sourceforge.net/ (accessed Jul. 17, 2023).

[8] P. Philip, “Penetration Testing Execution Standard (PTES),”GeeksforGeeks,https://www.geeksforgeeks.org/penetration-testing-execution-standard-ptes/ (accessed Jul. 28, 2023).

[9] A. R. Hevner and S. Chatterjee, Design Research in Information Systems: Theory and Practice. Springer, 2010.

[10] Django Software Foundation, “Django,” Django Project,https://docs.djangoproject.com/en/4.2/ (accessed Jul. 5, 2023).