

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

1st Yuaraina Dirgantariyki Ryandi

Fakultas Rekayasa Industri

Universitas Telkom

Bandung, Indonesia

yuarainaryandi@student.telkomuniversi-
ty.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/informasi 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/informasi 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 kalangan *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

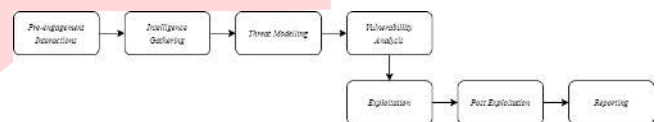
SQLSus adalah alat gratis MySQL Injection yang digunakan untuk melakukan eksploitasi *database* [7].

9. Havij

Havij adalah alat otomatis dikembangkan 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].



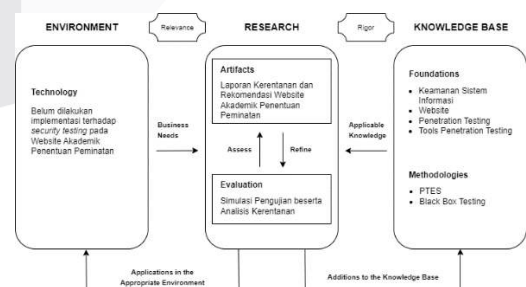
GAMBAR 1
METODE PTES

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].



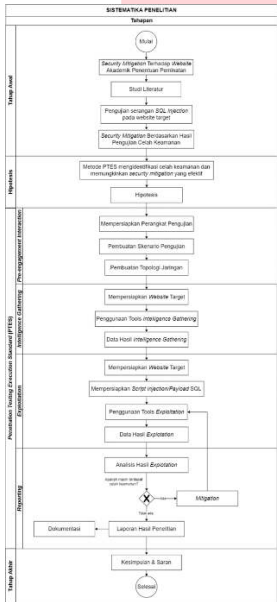
GAMBAR 1
MODEL KONSEPTUAL

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

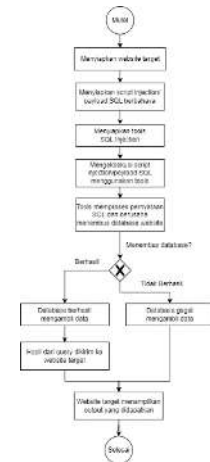
A. Pre-engagement Interactions

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

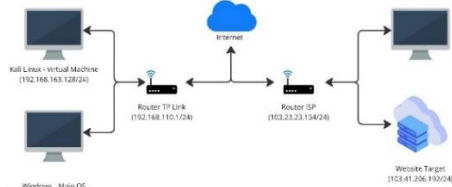
Dalam pengujian penetrasi, terdapat skenario pengujian yang akan dilakukan dengan tujuan menguji sistem dari *website* target dengan serangan *SQL Injection*.



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

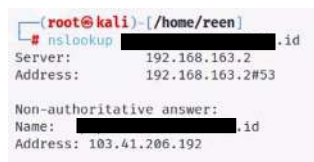
Pengumpulan informasi dilakukan menggunakan Nmap didapatkan informasi *port* dan *service* pada *website*.

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	teradataorhms?	
8080/tcp	open	http-proxy	
8084/tcp	open	websnp?	

GAMBAR 6
PENGUMPULAN INFORMASI DENGAN NMAP

3. Pengumpulan Informasi dengan Nslookup

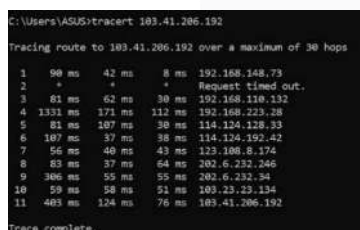
Pengumpulan informasi dilakukan menggunakan Nslookup didapatkan *IP Address* 103.41.206.192.



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*.

Hop	IP / Host Name	TSP	Network	Country	Loss	Response
1	192.168.148.79			ID	0.00	Success
2	192.168.138.132			ID	0.00	Success
3	192.168.223.28			ID	0.00	Success
4	114.124.128.33			ID	0.00	Success
5	114.124.192.42			ID	0.00	Success
6	232.100.4.174			ID	0.00	Success
7	202.6.232.246			ID	0.00	Success
8	202.6.232.34			ID	0.00	Success
9	103.23.23.134			ID	0.00	Success
10	103.41.206.192			ID	0.00	Success
11	103.41.206.192			ID	0.00	Success

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 “/”

```

[10:14:02] [CRITICAL] all tested parameters do not appear to be injectable
[*] ending @ 10:14:02 /2023-06-07/
  
```

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/”

```

[01:39:28] [CRITICAL] all tested parameters do not appear to be injectable
[01:39:28] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 7466 times, 400 (Bad Request) - 97 times
[*] ending @ 01:39:28 /2023-06-08/
  
```

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/”

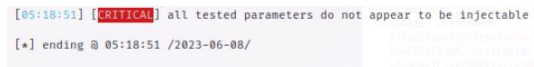
```

[04:06:10] [CRITICAL] all tested parameters do not appear to be injectable
[04:06:10] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 7460 times, 400 (Bad Request) - 97 times
[*] ending @ 04:06:10 /2023-06-08/
  
```

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.

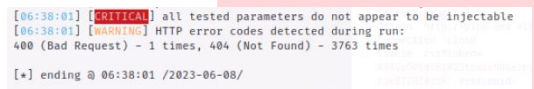
4) Menggunakan parameter “numberid=1202190036”



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.

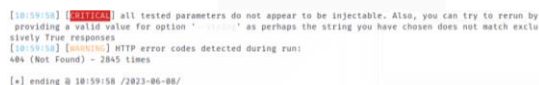
5) Menggunakan parameter “id=321” pada menu Seleksi Peminatan



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

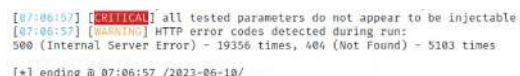


GAMBAR 16
HASIL SQLMAP DENGAN “id=1”

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*.

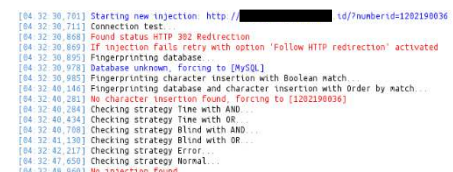
- c. Menggunakan parameter “mahasiswa/tukar_peminatan/”



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 Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

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

[illegible]

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 Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

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

```

10:20:15.811 Starting injection http://[redacted]@phaloxen.com/
10:20:15.853 Connection lost.
10:20:15.884 Found status 302 redirector
10:20:15.884 IP injection fails retry with option "Follow HTTP redirector" activated
10:20:15.872 PostgreSQL database
10:20:15.916 database unknown. According to [redacted]
10:20:15.945 PostgreSQL character injection with Boolean match
10:20:15.986 PostgreSQL database and character injection with Order by match...
10:20:16.016 No character injection. According to [redacted]
10:20:16.164 Checking strategy 1st with AND
10:20:16.254 Checking strategy 2nd with OR
10:20:16.336 Checking strategy 3rd with AND
10:20:16.423 Checking strategy 4th with OR
10:20:16.510 Checking strategy 5th with AND
10:20:16.597 Checking strategy 6th with OR
10:20:16.684 Checking strategy 7th with AND
10:20:16.771 Checking strategy 8th with OR
10:20:16.858 Checking strategy 9th with AND
10:20:16.945 Checking strategy 10th with OR
10:20:17.032 Checking strategy 11th with AND
10:20:17.119 Checking strategy 12th with OR
10:20:17.206 Checking strategy 13th with AND
10:20:17.293 Checking strategy 14th with OR
10:20:17.380 Checking strategy 15th with AND
10:20:17.467 Checking strategy 16th with OR
10:20:17.554 Checking strategy 17th with AND
10:20:17.641 Checking strategy 18th with OR
10:20:17.728 Checking strategy 19th with AND
10:20:17.815 Checking strategy 20th with OR
10:20:17.902 Checking strategy 21st with AND
10:20:17.989 Checking strategy 22nd with OR
10:20:18.076 Checking strategy 23rd with AND
10:20:18.163 Checking strategy 24th with OR
10:20:18.250 Checking strategy 25th with AND
10:20:18.337 Checking strategy 26th with OR
10:20:18.424 Checking strategy 27th with AND
10:20:18.511 Checking strategy 28th with OR
10:20:18.598 Checking strategy 29th with AND
10:20:18.685 Checking strategy 30th with OR
10:20:18.772 Checking strategy 31st with AND
10:20:18.859 Checking strategy 32nd with OR
10:20:18.946 Checking strategy 33rd with AND
10:20:19.033 Checking strategy 34th with OR
10:20:19.120 Checking strategy 35th with AND
10:20:19.207 Checking strategy 36th with OR
10:20:19.294 Checking strategy 37th with AND
10:20:19.381 Checking strategy 38th with OR
10:20:19.468 Checking strategy 39th with AND
10:20:19.555 Checking strategy 40th with OR
10:20:19.642 Checking strategy 41st with AND
10:20:19.729 Checking strategy 42nd with OR
10:20:19.816 Checking strategy 43rd with AND
10:20:19.903 Checking strategy 44th with OR
10:20:20.000 Checking strategy 45th with AND
10:20:20.087 Checking strategy 46th with OR
10:20:20.174 Checking strategy 47th with AND
10:20:20.261 Checking strategy 48th with OR
10:20:20.348 Checking strategy 49th with AND
10:20:20.435 Checking strategy 50th with OR
10:20:20.522 Checking strategy 51st with AND
10:20:20.609 Checking strategy 52nd with OR
10:20:20.696 Checking strategy 53rd with AND
10:20:20.783 Checking strategy 54th with OR
10:20:20.870 Checking strategy 55th with AND
10:20:20.957 Checking strategy 56th with OR
10:20:21.044 Checking strategy 57th with AND
10:20:21.131 Checking strategy 58th with OR
10:20:21.218 Checking strategy 59th with AND
10:20:21.305 Checking strategy 60th with OR
10:20:21.392 Checking strategy 61st with AND
10:20:21.479 Checking strategy 62nd with OR
10:20:21.566 Checking strategy 63rd with AND
10:20:21.653 Checking strategy 64th with OR
10:20:21.740 Checking strategy 65th with AND
10:20:21.827 Checking strategy 66th with OR
10:20:21.914 Checking strategy 67th with AND
10:20:22.001 Checking strategy 68th with OR
10:20:22.088 Checking strategy 69th with AND
10:20:22.175 Checking strategy 70th with OR
10:20:22.262 Checking strategy 71st with AND
10:20:22.349 Checking strategy 72nd with OR
10:20:22.436 Checking strategy 73rd with AND
10:20:22.523 Checking strategy 74th with OR
10:20:22.610 Checking strategy 75th with AND
10:20:22.697 Checking strategy 76th with OR
10:20:22.784 Checking strategy 77th with AND
10:20:22.871 Checking strategy 78th with OR
10:20:22.958 Checking strategy 79th with AND
10:20:23.045 Checking strategy 80th with OR
10:20:23.132 Checking strategy 81st with AND
10:20:23.219 Checking strategy 82nd with OR
10:20:23.306 Checking strategy 83rd with AND
10:20:23.393 Checking strategy 84th with OR
10:20:23.480 Checking strategy 85th with AND
10:20:23.567 Checking strategy 86th with OR
10:20:23.654 Checking strategy 87th with AND
10:20:23.741 Checking strategy 88th with OR
10:20:23.828 Checking strategy 89th with AND
10:20:23.915 Checking strategy 90th with OR
10:20:24.002 Checking strategy 91st with AND
10:20:24.089 Checking strategy 92nd with OR
10:20:24.176 Checking strategy 93rd with AND
10:20:24.263 Checking strategy 94th with OR
10:20:24.350 Checking strategy 95th with AND
10:20:24.437 Checking strategy 96th with OR
10:20:24.524 Checking strategy 97th with AND
10:20:24.611 Checking strategy 98th with OR
10:20:24.698 Checking strategy 99th with AND
10:20:24.785 Checking strategy 100th with OR
10:20:24.872 Checking strategy 101st with AND
10:20:24.959 Checking strategy 102nd with OR
10:20:25.046 Checking strategy 103rd with AND
10:20:25.133 Checking strategy 104th with OR
10:20:25.220 Checking strategy 105th with AND
10:20:25.307 Checking strategy 106th with OR
10:20:25.394 Checking strategy 107th with AND
10:20:25.481 Checking strategy 108th with OR
10:20:25.568 Checking strategy 109th with AND
10:20:25.655 Checking strategy 110th with OR
10:20:25.742 Checking strategy 111th with AND
10:20:25.829 Checking strategy 112th with OR
10:20:25.916 Checking strategy 113th with AND
10:20:26.003 Checking strategy 114th with OR
10:20:26.090 Checking strategy 115th with AND
10:20:26.177 Checking strategy 116th with OR
10:20:26.264 Checking strategy 117th with AND
10:20:26.351 Checking strategy 118th with OR
10:20:26.438 Checking strategy 119th with AND
10:20:26.525 Checking strategy 120th with OR
10:20:26.612 Checking strategy 121st with AND
10:20:26.699 Checking strategy 122nd with OR
10:20:26.786 Checking strategy 123rd with AND
10:20:26.873 Checking strategy 124th with OR
10:20:26.960 Checking strategy 125th with AND
10:20:27.047 Checking strategy 126th with OR
10:20:27.134 Checking strategy 127th with AND
10:20:27.221 Checking strategy 128th with OR
10:20:27.308 Checking strategy 129th with AND
10:20:27.395 Checking strategy 130th with OR
10:20:27.482 Checking strategy 131st with AND
10:20:27.569 Checking strategy 132nd with OR
10:20:27.656 Checking strategy 133rd with AND
10:20:27.743 Checking strategy 134th with OR
10:20:27.830 Checking strategy 135th with AND
10:20:27.917 Checking strategy 136th with OR
10:20:28.004 Checking strategy 137th with AND
10:20:28.091 Checking strategy 138th with OR
10:20:28.178 Checking strategy 139th with AND
10:20:28.265 Checking strategy 140th with OR
10:20:28.352 Checking strategy 141st with AND
10:20:28.439 Checking strategy 142nd with OR
10:20:28.526 Checking strategy 143rd with AND
10:20:28.613 Checking strategy 144th with OR
10:20:28.700 Checking strategy 145th with AND
10:20:28.787 Checking strategy 146th with OR
10:20:28.874 Checking strategy 147th with AND
10:20:28.961 Checking strategy 148th with OR
10:20:29.048 Checking strategy 149th with AND
10:20:29.135 Checking strategy 150th with OR
10:20:29.222 Checking strategy 151st with AND
10:20:29.309 Checking strategy 152nd with OR
10:20:29.396 Checking strategy 153rd with AND
10:20:29.483 Checking strategy 154th with OR
10:20:29.570 Checking strategy 155th with AND
10:20:29.657 Checking strategy 156th with OR
10:20:29.744 Checking strategy 157th with AND
10:20:29.831 Checking strategy 158th with OR
10:20:29.918 Checking strategy 159th with AND
10:20:30.005 Checking strategy 160th with OR
10:20:30.092 Checking strategy 161st with AND
10:20:30.179 Checking strategy 162nd with OR
10:20:30.266 Checking strategy 163rd with AND
10:20:30.353 Checking strategy 164th with OR
10:20:30.440 Checking strategy 165th with AND
10:20:30.527 Checking strategy 166th with OR
10:20:30.614 Checking strategy 167th with AND
10:20:30.701 Checking strategy 168th with OR
10:20:30.788 Checking strategy 169th with AND
10:20:30.875 Checking strategy 170th with OR
10:20:30.962 Checking strategy 171st with AND
10:20:31.049 Checking strategy 172nd with OR
10:20:31.136 Checking strategy 173rd with AND
10:20:31.223 Checking strategy 174th with OR
10:20:31.310 Checking strategy 175th with AND
10:20:31.397 Checking strategy 176th with OR
10:20:31.484 Checking strategy 177th with AND
10:20:31.571 Checking strategy 178th with OR
10:20:31.658 Checking strategy 179th with AND
10:20:31.745 Checking strategy 180th with OR
10:20:31.832 Checking strategy 181st with AND
10:20:31.919 Checking strategy 182nd with OR
10:2
```

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 Injection*, maka tidak ada informasi/data sensitif yang diperoleh dari *database website* target.

IV.3.3.2 Methode POST

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

[illegible]

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 Injection*, 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 “/”

```
[+] Session "[REDACTED].id" loaded
sqlsus> start
[!] - FATAL - find_select_columns() FAILED... exiting
```

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/”

```
[+] Session "[REDACTED]id" loaded
sqlsus> start
[!] - FATAL - find_select_columns() FAILED... exiting
```

GAMBAR 32
HASIL SOLSUS 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/”

```
[+] Session "[REDACTED].id" loaded
sqlsus> start
[!] - FATAL - find_select_columns() FAILED... exiting
```

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”

```
[+] Session "██████████.id" loaded
sqlsus> start
[!] - FATAL - find_select_columns() FAILED... exiting
```

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

```
[+] Session "██████████.id" loaded
sqlsus> start
[!] - FATAL - find_select_columns() FAILED... exiting
```

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

```
[+] Session "██████████.id" loaded
sqlsus> start
[!] - FATAL - find_select_columns() FAILED... exiting
```

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

```
[+] Session "██████████.id" loaded
sqlsus> start
[!] - FATAL - find_select_columns() FAILED... exiting
```

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”


```

Finding columns count(MySQL,MySQL 2006): 31
Error: 302 Found
Finding columns count(MySQL,MySQL 2006): 32
Error: 302 Found
Cannot find column count!
Target Not Vulnerable :

```

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

```

Finding columns count(MySQL,MySQL 2006): 31
Error: 302 Found
Finding columns count(MySQL,MySQL 2006): 32
Error: 302 Found
Cannot find column count!
Target Not Vulnerable :

```

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

```

Finding columns count(MySQL,MySQL 2006): 31
Error: 302 Found
Finding columns count(MySQL,MySQL 2006): 32
Error: 302 Found
Cannot find column count!
Target Not Vulnerable :

```

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*

```

Finding columns count(MySQL,MySQL 2006): 31
Error: 403 Forbidden
Finding columns count(MySQL,MySQL 2006): 32
Error: 403 Forbidden
Cannot find column count!
Target Not Vulnerable :

```

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	JSQ Injection	SQLMap	Havij
GET	Terdapat parameter status 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 tool harus memasukkan query string atau spesifik parameter	Tidak ditemukan celah SQL Injection pada parameter kosong “?”	Tidak dapat melakukan SQL Injection pada parameter kosong “?” karena tool harus memasukkan 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 tool harus memasukkan 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 tool harus memasukkan 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 tool harus memasukkan 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 tool harus memasukkan 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 dapat melakukan 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 dapat melakukan SQL Injection pada parameter “id=321” pada menu Seleksi Peminatan	Tidak ditemukan celah SQL Injection pada parameter “id=321” pada menu Seleksi Peminatan	Tidak dapat melakukan 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 dapat melakukan SQL Injection pada parameter “id=1” pada menu Tukar Peminatan	Tidak ditemukan celah SQL Injection pada parameter “id=1” pada menu Tukar Peminatan	Tidak dapat melakukan 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 dapat melakukan 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 Dikategorikan 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

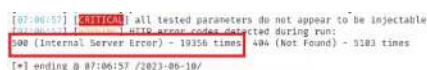
Django secara *default* terlindungi dari serangan *SQL Injection* karena menggunakan *Object Relational Mapping* (ORM) yang berarti developer tidak perlu menuliskan *query* SQL secara langsung, melainkan menggunakan API QuerySet bawaan. Django kemudian mengonversi dari *query* Python menjadi *query* 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 Firewall* (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 disimpulkan:

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 *level* 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.

- [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).

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).