Analisis *Security Mitigation* Terhadap *Website* Akademik Penunjan Administrasi di Institusi XYZ Mengunakan Metode *Penetration Testing Execution Standard* (PTES)

1st Akhmad Dzihan Kamaly
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
dzihankamaly@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.ac.id

Abstrak—Pada proses pembuatan *website* Akademik Penunjang Administrasi di Institusi XYZ, Perlu dilakukan *Security Testing* terhadap celah keamanan *Cross Site Scripting* dikarenakan celah ini memiliki potensi kerentanan yang tinggi dan berpotensi untuk pencurian data sensitif sehingga dapat dilakukan manipulasi data, jika hal tersebut terjadi dapat menghilangkan kredibilitas institusi terkait. Penelitian ini dilakukan untuk mengetahui celah *Cross Site Scripting* terhadap *Website* Administrasi Penunjang Administrasi di Institusi XYZ dan melakukan mitigasi celah tersebut dengan cara pengujian celah keamanan. Identifikasi celah *Cross Site Scripting* dilakukan dengan hasil celah tersebut teridentifikasi pada parameter *"warn"*. Proses eksploitasi dilakukan untuk membuktikan celah *Cross Site Scripting* dengan hasil keberhasil mengeksploitasi website target mengunakan beberapa *payload* yang diinjeksikan. Dapat disimpulkan bahwa pada parameter *"warn"* tidak menerapkan filter dan validasi input sebagai pencegah XSS dengan baik sehingga proses mitigasi dapat dilakukan terhadap *website* tersebut. Proses mitigasi dilakukan dengan menambahkan *header* CSP, penggunaan fungsi htmlspecialchars(), dan menggunaan SSL. Hasil yang didapatkan pasca mitigasi adalah *popup* dialog konfirmasi sudah tidak terdeteksi dan *alert error* masih dapat di modifikasi sehingga proses mitigasi berhasil mengurangi celah keamanan tersebut.

Kata kunci— Cross Site Scripting, Website, Mitigation, Testing

I. PENDAHULUAN

Berbicara mengenai celah keamanan, menurut OWASP *Top* 10 *Most Critical Web Application Security Risks*, menempatkan *Cross Site Scripting* berada diposisi ketiga [2]. *Cross Site Scripting* merupakan jenis serangan dengan tipe injeksi, penyerang dapat menyisipkan *script* berbahaya ke halaman *website*. Serangan ini membuat seolah-olah hal berbahaya tersebut berasal dari *website* itu sendiri. Akibatnya penyerang dapat mengendalikan browser secara tidak sah, penyerang dapat mencuri data sensitif dari target melalui *cookie* atau *form* data yang telah dimodifikasi yang kemudian dikirim kepada penyerang, dan dapat menyisipkan file berbahaya ke halaman *website* [3]. Maka dari itu keamanan *website* adalah salah satu faktor penting yang harus diperhatikan dalam pengembangan sebuah *website*. Keamanan *website* dapat menjamin bahwa informasi yang disimpan dan dikirim melalui *website* terlindungi dari pihak yang tidak berwenang. Institusi XYZ merupakan sebuah institusi pendidikan yang menggunakan bantuan sistem informasi untuk mengatur dan mengendalikan pengelolaan proses bisnisnya. Salah satu produk dari bantuan itu adalah sebuah *Website* Akademik Penunjang Administrasi. Pada proses pembuatan *website*, pihak terkait tidak melakukan proses *security testing*. Sehingga ketika *website* tersebut telah terpublikasi (*go live*) akan banyak celah keamanan yang masuk ke dalam *Website* tersebut. Perlu dilakukan *Security Testing* terhadap celah keamanan *Cross Site Scripting* dikarenakan celah ini memiliki potensi kerentanan yang tinggi, dapat merusak integritas dan kepercayaan pengguna, dan berpotensi untuk pencurian data sensitif. Maka dari itu, pada penelitian ini melakukan sebuah project *Security Mitigation* terhadap *Website* Akademik Penunjang Administrasi di Institusi XYZ dengan menggunakan kerangka kerja PTES (*Penetration Testing Execution Standard)* dengan tujuan mengurangi risiko keamanan sistem atau data sehingga dapat terhindar dari ancaman yang ada.

II. KAJIAN TEORI

A. Keamanan Sistem Informasi

Keamanan informasi melibatkan tindakan untuk melindungi informasi dan sistem informasi dari akses, penggunaan, pengungkapan, gangguan, modifikasi, atau penghancuran yang tidak sah. Dalam pengelolaan keamanan informasi, tujuan utamanya adalah memastikan kelangsungan operasional bisnis dan mengurangi dampak insiden keamanan yang mungkin terjadi. Dengan menerapkan langkah-langkah keamanan yang tepat, organisasi dapat menjaga kerahasiaan, integritas, dan ketersediaan informasi, serta melindungi diri dari ancaman dan risiko yang berpotensi merugikan [4]. Prinsip keamanan informasi mencakup tiga aspek yang paling umum disebut adalah C.I.A TRIAD yaitu, Confidentiality, Integrity, dan Availability.

B. *Security Mitigation*

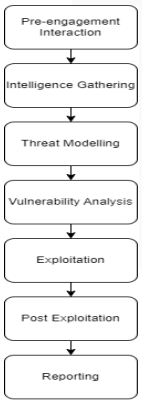
Security Mitigation merupakan sebuah langkah/tindakan yang dilakukan untuk mengurangi risiko dan melindungi sistem, jaringan, atau aplikasi dari potensi ancaman keamanan. Tujuan utama dari mitigasi keamanan adalah mencegah serangan atau pelanggaran keamanan dan mengurangi konsekuensi yang mungkin timbul jika serangan tersebut berhasil. Tindakan dan strategi mitigasi bertujuan untuk memitigasi atau mengurangi kemungkinan terjadinya kerentanan atau celah keamanan yang dapat dimanfaatkan oleh pihak yang tidak berwenang.

C. *Website*

Website merupakan kumpulan halaman web terkait berisi informasi yang dapat terdiri dari informasi yang dapat berupa gambar, suara, video, teks, animasi atau kombinasinya, dan dapat diakses melalui browser dan Internet. *Website* dapat menjadi sumber informasi yang sangat berguna bagi para pengguna internet. Selain sebagai sumber informasi, *website* juga dapat digunakan sebagai platform untuk berinteraksi dan berkomunikasi dengan orang lain, baik secara pribadi maupun di dalam suatu komunitas.

D. *Penetration Testing Execution Standard* (PTES)

PTES adalah seperangkat pedoman yang menjelaskan cara melakukan uji penetrasi secara efektif dan efisien. PTES dikembangkan oleh sebuah komunitas ahli keamanan komputer yang bertujuan untuk menyediakan standar yang dapat digunakan oleh para profesional keamanan komputer dalam melakukan uji penetrasi. PTES dimulai pada awal tahun 2009 dan berawal dari pertemuan antara anggota pendiri disaat membicarakan tentang kepentingan atau kelemahan dalam *penetration testing* yang ada sekarang [1] Langkah-langkah dalam melakukan PTES dapat dilihat pada gambar II.1.



GAMBAR II. 1
Tahapan PTES

E. *Cross Site Scripting* (XSS)

Cross Site Scripting (XSS) merupakan jenis kejahatan terhadap keamanan aplikasi *website* yang menempati daftar OWASP *top ten* kerentanan paling berbahaya menurut organisasi keamanan web internasional *Open Web Application Security Project* (OWASP) [2]. OWASP *top ten* merupakan suatu panduan yang digunakan para pengembang aplikasi dan tim keamanan mengenai kelemahan terhadap aplikasi *website* yang rentan terhadap serangan dan harus segera diperbaiki salah satunya adalah *Cross Site Scripting* (XSS) [3]. *Cross Site Scripting* (XSS) bekerja dengan cara memanfaatkan input pada browser, serangan ini melibatkan penipuan target dengan mengarahkannya ke halaman tertentu yang telah disisipi dengan kode berbahaya oleh penyerang dengan tujuan antara lain mengendalikan browser secara tidak sah, mencuri data sensitif dari *form* data yang telah dimodifikasi yang kemudian dikirim kepada penerang, menyisipkan file berbahaya ke halaman *website* [3]. *Cross Site Scripting* (XSS) dapat diklasifikasikan ke dalam dua kategori utama antara lain:

1. *Stored Cross Site Scripting* (XSS)

Stored Cross Site Scripting (XSS) melibatkan penyerang dalam menyisipkan skrip berbahaya (disebut *payload*) ke dalam aplikasi target dengan cara yang permanen atau tetap. Contoh sederhana dari serangan XSS yang bersifat penyimpanan adalah saat penyerang menyisipkan skrip berbahaya ke dalam kolom komentar di sebuah *blog* atau pada postingan forum [5].

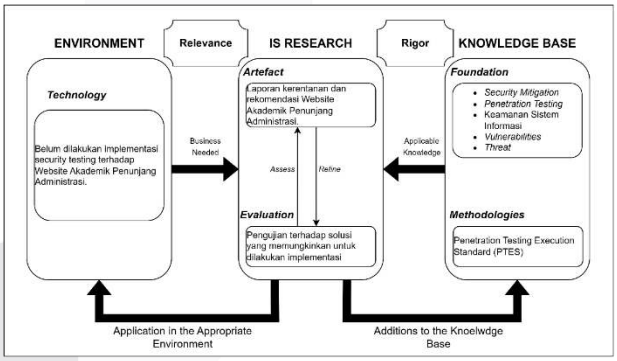
2. *Reflected Cross Site Scripting* (XSS)

Reflected *Cross Site Scripting* (XSS) adalah serangan oleh penyerang untuk mengirim *script payload* melibatkan bagian dari permintaan yang dikirim ke *server web*, kemudian direfleksikan kembali sedemikian rupa sehingga muatan permintaan HTTP juga termasuk di dalamnya [5].

III. METODE

A. Kerangka Berfikir

Dalam penelitian ini menggunakan model konseptual yang diadopsi dari framework penelitian *Design Science Research in Information System* oleh Hevner, gambaran mengenai model konseptual dalam penelitian ini dapat dilihat pada gambar III.1.



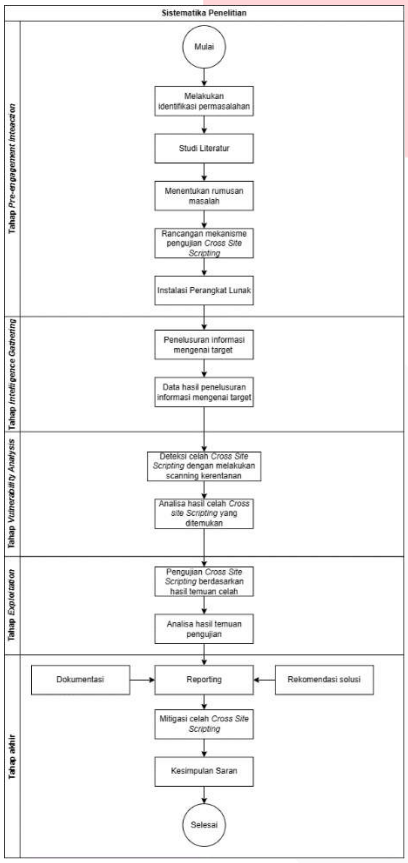
GAMBAR III. 1
Model Konseptual Penelitian

1. *Technology* adalah *environment* yang digunakan dalam melakukan penelitian ini, dikarenakan belum dilakukan implementasi terhadap *security testing* pada aplikasi yang ada dan untuk mengetahui kemampuan dari aplikasi untuk membatasi diri dari hak hak akses yang tidak tervalidasi.
2. *IS Research* adalah faktor yang memiliki fungsi untuk melakukan penyaringan dan penilaian. Dalam faktor penelitian memiliki dua bagian yaitu *artefact* dan *evaluation*. Artefak pada penelitian ini adalah Laporan kerentanan dan rekomendasi terhadap website terkait. Kemudian akan di evaluasi dengan pengujian terhadap solusi yang memungkinkan untuk dilakukan implementasi.

3. *Knowledge Base* merupakan teori-teori dasar yang dapat dijadikan acuan dalam menyelesaikan permasalahan dalam penelitian ini, dasar teori tersebut yaitu teori mengenai *Security Mitigation*, teori mengenai *Penetration Testing*, teori mengenai Keamanan Sistem Informasi, teori mengenai *Threat* atau Ancaman, dan teori mengenai *Vulnerabilities*. Pada bagian bawah *Knowledge Base* juga terdapat metodologi yang digunakan dalam melakukan penelitian ini adalah *Penetration Testing Execution Standard* (PTES).

B. Sistematika Penelitian

Sistematika peneletian ini terbagi atas tahap awal, tahap pengujian, tahap analisis, dam tahap akhir. Ilustrasi sistematika penelitian ini digambarkan dalam bentuk bagan seperti pada gambar III.2 Sistematika Penelitian.



Gambar III. 2
Sistematika Penelitian

a. Tahap *Pre-engagement Interaction*
Tahap *Pre-engagement Inteaction* merupakan tahap awal dalam penelitian ini. Tahap ini melakukan identifikasi masalah yang mengacu pada studi litaratur. Studi literatur berguna untuk memperdalam teori mengenai celah keamanan XSS terhadap *website* target. Selanjutnya menentukan rumusan masalah yang akan menjadi inti permasalahan dari identifikasi masalah tersebut dan menentukan batasan masalah agar penelitian fokus terhadap permasalahan yang akan diselesaikan. Selanjutnya menentukan rancangan/gambaran kasar mengenai mekanisme penyerangan celah XSS yang akan dilakukan dan melakukan instalasi perangkat lunak yang akan digunakan.

b. Tahap Hipotesis
Pada Tahap Hipotesis melakukan hipotesis berupa praduga sementara terhadap hipotesis mengenai penggunaan kerangka kerja *Penetration Testing Execution Standard* (PTES) terhadap identifikasi celah keamanan Cross Site Scripting.

c. Tahap *Intelligence Gathering*
Tahap *Intelligence Gathering* merupakan tahap pengumpulan informasi mengenai *website* target yang didapatkan dengan berbagai *tools* yang telah terinstall antara lain Nslookup dan Zenmap.

d. Tahap *Vulnerability Analysis*
Tahap *Vulnerability Analysis* merupakan tahap identifikasi celah kerentanan *Cross Site Scripting* yang ada pada *website* target menggunakan OWASP ZAP (*Zed Attack Proxy*). Kemudian dari hasil yang telah dideteksi tersebut di analisa dan di data sebagai acuan untuk melakukan proses pengujian celah keamanan *Cross Site Scripting.*

e. Tahap *Exploitation*
Tahap *Exploitation* merupakan tahap uji celah kerentanan *Cross Site Scripting* berdasarkan hasil temuan yang di lakukan pada tahap *Vulnerability Analysis.* Dari hasil temuan tersebut akan diuji menggunakan *tools* XSStrike dan Burp Suite. Kemudian hasil pengujian di analisis dampak dan resiko yang terjadi ketika celah tersebut di lakukan oleh orang yang tidak bertanggung jawab. Dari analisa tersebut akan dijadikan acuan untuk melakukan proses selanjutnya.

f. Tahap Akhir
Tahap Akhir merupakan tahap *reporting* yang berupa dokumentasi dan rekomendasi solusi dari hasil pengujian celah keamanan tersebut dan akan diberikan kepada pihak developer. Dari *reporting* ini akan dikaji ulang mengenai kemungkinannya untuk di mitigasi. Jika rekomendasi dan solusi yang diberikan memungkinkan dilakukan mitigasi, maka akan melakukan mitigasi sesuai dengan rekomendasi dan solusi yang diberikan. Jika rekomendasi dan solusi tidak memungkinkan untuk dilakukan mitigasi dengan alasan tertentu maka tahap mitigasi tidak dilakukan dan berhenti di tahap *reporting* dan kesimpulan dan saran.

IV. HASIL DAN PEMBAHASAN

A. *Pre-Engagement Interaction*

1. Perancangan Sistem

Dalam penelitian ini, diperlukan suatu rencana sistem agar penelitian ini dapat dilaksanakan. Perancangan sistem ini berfungsi untuk merancang atau menciptakan suatu gambaran mengenai langkah-langkah yang akan dilakukan dalam penelitian. Perancangan sistem ini penting untuk melakukan pengujian sistem dan diperlukan perangkat yang memadai. Terdapat dua jenis perangkat, yaitu perangkat keras dan perangkat lunak.

TABEL IV. 1
Spesifikasi Perangkat Keras

Komponen	Spesifikasi	
Perangkat Keras (*Main OS*)	*Processor*	*AMD Athlon Silver 3050U with Radeon Graphics (2CPUs), ~2.3Ghz*
	Memory	8 GB
	SSD	728 GB
	System Type	64-bit
	Operating System	*Windows 11 Home Single Language 64-bit*
Perangkat Keras (*Virtual Machine*)	*Processor*	*AMD Athlon Silver 3050U with Radeon Graphics (2CPUs), ~2.3Ghz*
	Memory	4 GB
	Hard Disk	30 GB
	System Type	64-bit
	Operating System	*Kali Linux*

TABEL IV. 2
Spesifikasi Perangkat Lunak

Tipe	Perangkat Lunak
Virtual Machine	VMware Workstation 16 Pro
Operating System	*Kali Linux* 2023.1
Scanning Tools	Nslookup
Vulnerability Analysis	Zenmap 7.92
	OWASP ZAP 2.12.0
Exploitation Tools	Burp Suite 2023.1.2
	XSStrike 3.1.5

Berdasarkan tabel IV.2, penelitian ini menggunakan beberapa perangkat lunak. Berikut akan dijelaskan fungsi dari perangkat lunak yang digunakan pada penelitian ini.

1. VMware Workstation

VMware Workstation adalah sebuah perangkat lunak virtualisasi yang memungkinkan pengguna untuk menjalankan beberapa sistem operasi yang berbeda di dalam satu komputer fisik. Pada penelitian ini VMware Workstation berfungsi untuk menjalankan Operationg System Kali Linux.

2. Kali Linux

Kali Linux adalah *Operating system* berbasis *open-source* yang digunakan untuk melakukan *Penetration Testing*. Pada penelitian ini Kali Linux berfungsi untuk proses *Intelligence Gathering*, *Vulnerability Analysis*, *Exploitation,* dan *Mitigation.*

3. Nslookup

Nslookup adalah sebuah *tool* yang tersedia di *Kali Linux* yang berfungsi untuk *information gathering* seperti mendeteksi *IP Address* dan Name Server

4. Zenmap

Zenmap adalah antarmuka grafis (GUI) yang digunakan untuk mengendalikan Nmap, alat pemindaian keamanan jaringan yang populer. Nmap (*Network Mapper*) sendiri adalah alat sumber terbuka yang digunakan oleh para profesional keamanan jaringan untuk melakukan pemindaian jaringan, penilaian kerentanan, dan penemuan perangkat terhubung dalam jaringan. Pada penelitian ini Zenmap berfungsi untuk pemindaian jaringan.

5. OWASP ZAP (Zed Attack *Proxy*)

OWASP ZAP merupakan *tool open-source* yang dikembangkan oleh *Organisasi Open Web Application Security Project* (OWASP) yang digunakan untuk melakukan pengujian keamanan *website*. Pada penelitian ini OWASP ZAP berfungsi untuk mendeteksi celah keamanan.

6. XSStrike

XSStrike merupakan salah satu *tool* berbahasa *python* yang dirancang untuk mendeteksi dan mengeksploitasi celah *Cross Site Scripting* dengan cara mencoba berbagai teknik dan *payload* secara otomatis untuk mendeteksi dan mengeksploitasi kerentanan pada target.

7. Burp Suite

Burp Suite adalah *platform* yang digunakan untuk melakukan pengujian keamanan terhadap aplikasi web. Platform ini memiliki berbagai macam alat dan fitur yang dirancang khusus untuk mendeteksi kerentanan keamanan, melakukan uji penetrasi, dan melakukan analisis terhadap aplikasi web. Pada penelitian ini Burp Suite berfungsi untuk melakukan eksploitasi terhadap celah keamanan yang ditemukan menggunakan fitur *intercept Proxy* dan *repeater*.

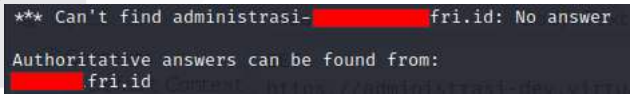
B. *Intelligence Gathering*

Pencarian informasi mengenai *IP Address* dilakukan dua kali. Pengujian pertama menggunakan subdomain yaitu administrasi-xxx.xxxxxxxfri.id dan yang kedua menggunakan domain yaitu xxxxxxxfri.id. Proses pencarian *IP Address* dapat dilihat pada gambar IV.1 dan gambar IV.2.

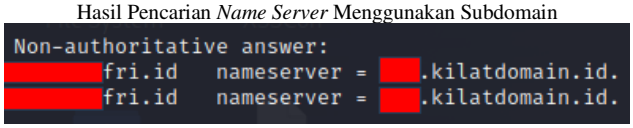
GAMBAR IV. 1
IP *Address subdomain*

GAMBAR IV. 2
Name Server domain

Pada gambar IV.3 dan gambar IV.4 nslookup juga dilakukan pencarian *Name Server* yang dilakukan dua kali pengujian. Pertama menggunakan subdomain yaitu administrasi-xxx.xxxxxxxfri.id dan yang kedua menggunakan domain yaitu xxxxxxxfri.id.



GAMBAR IV. 3
Hasil Pencarian *Name Server* Menggunakan Subdomain



GAMBAR IV. 4
Hasil pencarian Name Server menggunakan domain

Penggunaan *IP Address* subdomain dan domain sama dikarenakan keduanya di *hosting* pada server yang sama yaitu Kilat Domain. Kilat domain merupakan salah satu perusahaan yang menyediakan layanan *hosting* dan *Website* Akademik Penunjang Administrasi ini menggunakan layanan tersebut untuk *hosting* di Kilat Domain.

C. Pengujian *Vulnerability Analysis* Menggunakan OWASP ZAP

Hasil pengujian *Vulnerability Analysis* menggunakan OWASP ZAP yang dilakukan sesuai dengan langkah pengujian mendapatkan hasil yang dapat dilihat pada gambar IV.5. Celah *Cross Site Scripting* dengan jenis *reflected* terdeteksi oleh OWASP ZAP dengan tingkat resiko *high* dan tingkat *confidence* atau keyakinan terhadap temuan celah *Cross Site Scripting* ini dalam tingkatan *medium* pada parameter *"warn"*.

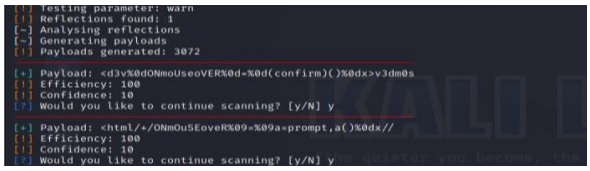
GAMBAR IV. 5
Output OWASP ZAP

pengujian celah keamanan *Cross Site Scripting* yang dilakukan pada URL administrasi-xxx.xxxxxxxfri.id memperoleh hasil sebagai berikut:

1. Celah *Cross Site Scripting* terdeteksi dengan jenis *Reflected* yang terdeteksi pada URL http://administrasi-xxx.xxxxxxxfri.id/signin.php?*"warn"*=%3C%2Fh4%3E%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E%3Ch4%3E
2. Celah yang terdeteksi memiliki resiko dengan tingkat *high*. Dapat dijelaskan bahwa tingkat resiko ini memiliki dampak yang sangat serius sehingga celah tersebut dapat dieksploitasi dengan mudah dan dapat menyebabkan kerusakan sistem yang signifikan.
3. Hal yang dijelaskan pada point dua belum dapat dibuktikan karena OWASP ZAP mendeteksi tingkat *confidence* atau keyakinan OWASP ZAP terhadap temuan celah *Cross Site Scripting* ini dalam tingkatan *medium* yang dapat dijelaskan bahwa tingkat *confidence* ini memiliki bukti untuk mendukung keberadaan kerentanan tersebut, tetapi masih ada ketidakpastian dan memerlukan verifikasi tambahan serta analisis lebih lanjut. Tingkat kepercayaan ini mengindikasikan bahwa ada keraguan terhadap keparahan kerentanan.
4. Celah tersebut terdeteksi pada parameter *"warn"*, parameter ini berada pada halaman sign in yang digunakan untuk memberikan *alert error* jika pengguna salah memasukkan username atau password.
5. OWASP ZAP memberikan solusi secara general terhadap celah *Cross Site Scripting*. Dapat dijelaskan bahwa celah tersebut secara umum terjadi ketika *website* tidak menggunakan library atau framework yang berguna untuk menghindari celah *Cross Site Scripting*, *website* tidak menerapkan karakter *encoding* seperti ISO-8859-1 atau UTF-8, *website* tidak mengatur *session cookie* sebagai *HttpOnly*, dan *website* tidak menerapkan strategi validasi input yang aman dan kuat dalam pengembangannya.

D. Hasil dan Analisis *Exploitation* Menggunakan *Tool* XSStrike

Pada gambar IV.6 merupakan hasil pengujian celah *Cross Site Scripting* menggunakan *tool* XSStrike. Dari hasil tersebut menjelaskan bahwa *tool* melakukan testing terhadap parameter *"warn"* dan menemukan celah reflection terhadap parameter tersebut. Kemudian *tool* tersebut melakukan generate sebanyak 3072 *payload*.



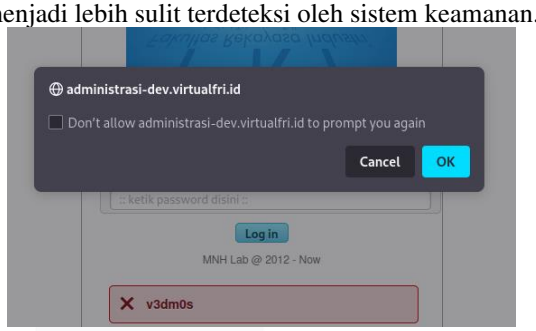
GAMBAR IV. 6 *Output* Pengujian Exploitation menggunakan XSStrike

Hasil pengujian *Exploitation* menggunakan *tool* XSStrike didapatkan *payload* sebanyak 3072 *payload* yang telah di generate. Pada penelitian ini mengambil sampel sebanyak dua dari 3072 *payload* yang dilakukan oleh XSStrike yang dilakukan pengujian, dapat dilihat pada tabel IV.3.

TABEL IV. 3
Payload XSStrike

Payload
<d3v%0dONmoUseoVER%0d=%0d(confirm)()%0dx>v 3dm0s
<html/+/ONmOuSEoveR%09=%09a=prompt,a()%0dx//

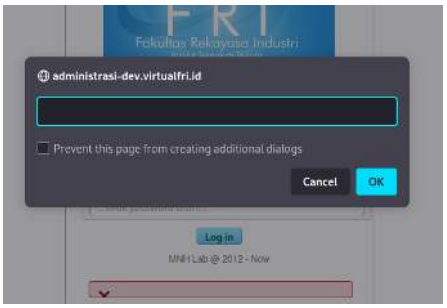
Kedua sampel *payload* tersebut dibentuk oleh XSStrike untuk menghindari filter yang mencegah serangan *Cross Site Scripting* dan menghindari *Web Application Firewall* (WAF) menggunakan teknik evasion dengan tujuan utama memungkinkan serangan dapat berhasil melewati pertahanan dan menjadi lebih sulit terdeteksi oleh sistem keamanan.



GAMBAR IV. 7
Hasil Pengujian Pertama Payload XSStrike

Pada gambar IV.7 dapat dilihat hasil pengujian dari *payload* pertama yang diinjeksikan kepada *website* target. Adapun hasil pengujian yang diperoleh sebagai berikut:

1. Pada tampilan *alert error* terdapat tulisan "v3dm0s" yang menunjukkan bahwa tampilan *alert error* dapat di modifikasi dan menampilkan pop up dengan dialog konfirmasi dan button "OK" dan "Cancel" yang dibentuk dari *payload* untuk menghindari filter yang mencegah serangan *Cross Site Scripting* dan menghindari WAF.
2. *Payload* berhasil di eksekusi terhadap parameter *"warn"* sehingga dapat disimpulkan bahwa parameter tersebut tidak menerapkan filter pencegah serangan *Cross Site Scripting*.



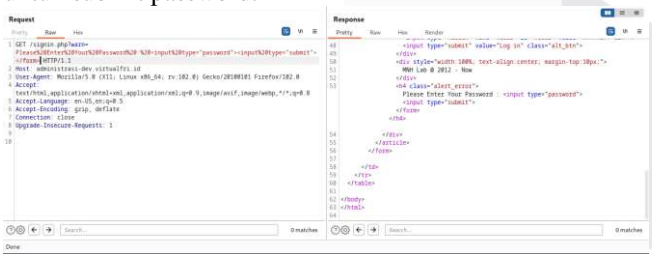
GAMBAR IV. 8
Hasil Pengujian Kedua Payload XSStrike

Pada gambar IV.8 dapat dilihat hasil serangan dari *payload* kedua yang diinjeksikan kepada *website* target. Adapun hasil pengujian yang diperoleh sebagai berikut:

1. Pada tampilan *alert error* tidak terdapat tulisan apapun dan menjadi lebih kecil dan menampilkan *alert error* berupa dialog input dengan button "OK" dan "Cancel" yang dibentuk dari *payload* untuk menghindari filter yang mencegah serangan *Cross Site Scripting* dan menghindari WAF.
2. *Payload* berhasil di eksekusi terhadap parameter *"warn"* sehingga dapat disimpulkan bahwa parameter tersebut tidak menerapkan filter pencegah serangan *Cross Site Scripting*.

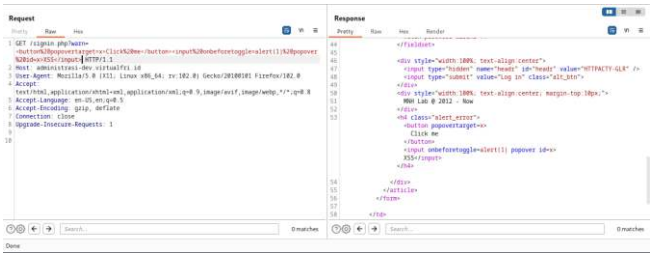
E. Hasil dan Analisis *Exploitation* Menggunakan *Tool* Burp Suite

Hasil pengujian *Cross Site Scripting* menggunakan *tool* Burp Suite pada penelitian ini menggunakan fitur *repeater*, fitur ini dapat mengirim ulang permintaan dan memodifikasi permintaan HTTP, dengan fitur ini pengujian dapat memudahkan pengguna untuk menyisipkan *payload* tambahan dalam permintaan. Pada penelitian ini, percobaan penyerangan dilakukan menggunakan dua *payload* yang bersumber dari XSS *payload* Cheat Sheet. Pada gambar IV.9 dapat dilihat *output* dari pengujian menggunakan Burp Suite. Pada parameter *"warn"* yang memberikan *alert error* telah berubah tidak hanya menampilkan *alert error* tetapi juga menampilkan form untuk mengisikan password dan element untuk submit password.



GAMBAR IV. 9
Output pengujian Pertama Menggunakan Burp Suite

Dan pada gambar IV.10 dapat dilihat *output* dari pengujian menggunakan Burp Suite, pada parameter *"warn"* yang memberikan *alert error* telah berubah. Tidak hanya menampilkan *alert error* tetapi juga menampilkan element "button" dan "input".



GAMBAR IV. 10 Output pengujian Kedua Menggunakan Burp Suite

Hasil pengujian *Exploitation* menggunakan *tool* Burp Suite dilakukan dengan injeksi *payload* yang diambil dari external, dapat dilihat pada tabel IV.1

TABEL IV. 1 *Payload* Untuk Pengujian Burp Suite

Payload
Please%20Enter%20Your%20Password%20:%20 <input%20type="password"><input%20type="submit"> </form>
<button%20popovertarget=x>Click%20me</button> <input%20onbeforetoggle=alert(1)%20popover%20id=x > XSS</input>

Kedua *payload* tersebut merupakan *payload* dengan script HTML yang dapat mengubah value pada pamameter *"warn"* menjadi form untuk pengisian password dengan button "Submit" dan mengubah *alert error* menjadi form inputan dengan button "Click me" dan menggunakan %20 sebagai pengganti spasi dikarenakan spasi merupakan karakter khusus dalam URL yang tidak diperbolehkan. Jika kedua *payload* tersebut berhasil tereksekusi menandakan bahwa parameter *"warn"* tidak memiliki filter untuk pencegah *Cross Site Scripting*.

```
GET /signin.php?warn=
Please%20Enter%20Your%20Password%20:%20<input%20type="password"><input%20type="submit">
</form> HTTP/1.1
```

GAMBAR IV. 11 Hasil Data Request Pertama Menggunakan Burp Suite

Pada gambar IV.11 dapat dilihat hasil mendapatkan data *request* pada halaman *sign in*. Data request pada parameter *"warn"* dapat dimodifikasi sesuai dengan *value* parameter yang diinginkan, di penelitian ini *value* tersebut dimodifikasi dengan *payload* pertama.

```
<h4 class="alert_error">
  Please Enter Your Password : <input type="password">
  <input type="submit">
 </form>
</h4>
```

GAMBAR IV. 12
Hasil Data Request Pertama Menggunakan Burp Suite

Pada gambar IV.12 merupakan hasil *response* ketika *data request* dikirimkan, adapun hasil pengujian yang diperoleh:

1. *Alert* yang dihasilkan berupa kolom input *password* yang dapat diisi dan di*submit* oleh pengguna *website*. Hal tersebut dapat terjadi dikarenakan pengguna merasa seolah-olah hal tersebut disediakan oleh *website* itu sendiri.
2. *Payload* tersebut berhasil tereksekusi pada parameter *"warn"* sehingga dapat disimpulkan bahwa parameter *"warn"* tidak menggunakan filter pencegah serangan *Cross Site Scripting*.

Pada gambar IV.13 merupakan hasil dari tampilan. Pada *tab render* sudah menyediakan hasil dari tampilan halaman

website setelah dilakukan modifikasi parameter sehingga tidak perlu lagi untuk membuka *payload* di browser.

GAMBAR IV. 13
Render Payload Pertama Burp Suite

Proses Verifikasi celah *Cross Site Scripting* terhadap *website* target dilakukan dua kali untuk memberikan tampilan yang berbeda. Pada gambar IV.14 dapat dilihat hasil data *request* dari halaman *sign in.* Data request pada parameter *"warn"* dapat dimodifikasi sesuai dengan *value* parameter yang diinginkan, di penelitian ini *value* tersebut dimodifikasi dengan *payload* kedua.

GAMBAR IV. 14
Hasil Data Request Kedua Menggunakan Burp Suite

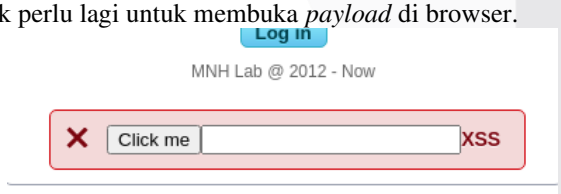
Pada gambar IV.15 merupakan hasil *response* ketika *data request* dikirimkan, adapun hasil pengujian yang diperoleh:

1. *Alert* yang dihasilkan berupa kolom inputan dengan button "Click me" yang ketika di click akan muncul *popup* alert dengan tulisan angka satu.
2. *Payload* tersebut berhasil tereksekusi pada parameter *"warn"* sehingga dapat disimpulkan bahwa parameter *"warn"* tidak menggunakan filter pencegah serangan *Cross Site Scripting*.

```
<h4 class="alert_error">
  <button popovertarget=x>
    Click me
  </button>
  <input onbeforetoggle=alert(1) popover id=x>
  XSS</input>
</h4>
```

GAMBAR IV. 15
Hasil Data Request Kedua Menggunakan Burp Suite

Pada gambar IV.16 merupakan hasil dari tampilan. Pada *tab render* sudah menyediakan hasil dari tampilan halaman *website* setelah dilakukan modifikasi parameter sehingga tidak perlu lagi untuk membuka *payload* di browser.



GAMBAR IV. 16
Render Payload Kedua Burp Suite

F. *Reporting*

Pada hasil pengujian menggunakan kedua *tools* eksploitasi dengan beberapa payload yang diuji cobakan menunjukkan adanya celah keamanan *Cross Site Scripting* pada *Website* Akademik Penunjang Administrasi di Institusi XYZ. Sehingga dapat berdampak buruk terhadap *website* dikarenakan tidak adanya filter pencegah serangan *Cross Site Scripting* terhadap parameter *"warn"* yang dapat dibuktikan dengan analisa pada proses pengujian eksploitasi. Proses tersebut menyatakan bahwa *payload* yang dibentuk untuk menghindari filter yang mencegah serangan *Cross Site Scripting* dan menghindari *Web Application Firewall* (WAF) menggunakan teknik *evasion* dengan tujuan utama keberhasilan melewati pertahanan sistem keamanan berhasil dilakukan. Pada tabel IV.4 adalah rekomendasi dari hasil pengujian celah keamanan *Cross Site Scripting* yang berguna untuk mengamankan *website* target dan mengurangi dampak dari serangan yang terjadi.

TABEL IV. 4
Rekomendasi Solusi

Kerentanan	Deskripsi	Rekomendasi solusi
Cross Site Scripting (XSS)	*Cross Site Scripting* merupakan kerentanan dengan level risiko *high*. Kerentanan ini dapat diserang dengan memberikan kode-kode berbahaya sehingga dapat dieksekusi.	Memberikan *header Content Security Policy* (CSP). Dengan menerapkan *header* CSP dalam respons HTTP *website* dapat memberikan informasi terhadap browser tentang sumber daya yang diizinkan untuk di akses. CSP memberikan kebijakan yang ketat untuk membatasi sumber daya yang masuk, oleh karena itu hal ini dapat mengurangi kemungkinan celah keamanan *Cross Site Scripting* terjadi. Menerapkan fungsi htmlspecialchars() pada parameter *"warn"* fungsi ini berfungsi untuk mengkonversi karakter-karakter khusus HTML menjadi entitas HTML. Penerapakan *Secure Socket Layer* (SSL). Penerapan ini berfungsi untuk mengenkripsi data yang di transmisikan antara *server* dan browser, mencegah pencurian data yang di transmisikan antara server dan browser, dan meningkatkan kepercayaan pengguna *website*. Menggunakan *library* atau *framework* yang telah terverifikasi untuk menutupi atau menghindari celah *Cross Site Scripting*.

Proses Mitigasi dilakukan dengan menerapkan rekomendasi solusi yang telah di buat terhadap *Website* Akademik Penunjang Administrasi. Proses mitigasi dilakukan dengan bekerja sama oleh pihak *developer*. Sebelum proses mitigasi dilakukan, pihak *developer* harus mengetahui langkah-langkah yang akan dilakukan untuk mitigasi dengan melihat laporan hasil akhir pengujian, jika disetujui pihak *developer* akan dberikan akses *git repository website* untuk di lakukan mitigasi dan jika tidak, mitigasi tidak dilakukan.

G. Perancangan *Mitigation*

Berdasarkan hasil analisis yang telah dilakukan, perancanga mitigasi dilakukan untuk meminimalisir atau menambal celah *Cross Site Scripting* pada *Website* Akademik Penunjang Administrasi pada celah yang telah terdeteksi. Berikut adalah proses mitigasi yang dilakukan dengan mengacu pada rekomendasi solusi yang telah dibuat.

1. Penambahan Header *Content Security Policy* (CSP)

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self'; style-src 'self' 'unsafe-inline';">
```

GAMBAR IV. 17
Penambahan Header Content Security Policy (CSP)

Atribut *http-equiv* digunakan untuk memberi tahu browser bahwa tag <meta> tersebut memberikan informasi terkait kebijakan yang harus diterapkan oleh halaman web. Dalam hal ini, nilai atribut adalah "*Content-Security-Policy*", yang menunjukkan bahwa ini adalah tag meta untuk mengatur kebijakan keamanan konten. Dalam artibut tersebut terdapat nilai nlai yang memberikan kebijakan untuk membatasi sumber daya seperti sumber daya *default* menunjukkan sumber daya dapat diambil hanya dari domain yang sama dengan halaman web saat ini, sumber daya script yang menunjukkan bahwa skrip hanya diizinkan berasal dari domain yang sama dengan halaman web saat ini, dan sumber daya *style* yang menunjukkan *style* hanya diizinkan berasal dari domain yang sama dengan halaman web saat ini.

2. Menerapkan htmlspecialchars()

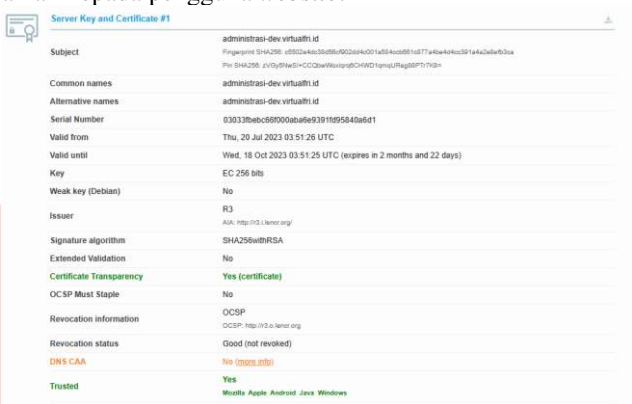
```
$warn = isset($_GET["warn"]) ? $_GET["warn"] : "";
$warn = htmlspecialchars($warn, ENT_QUOTES, 'UTF-8');
```

GAMBAR IV. 18
Penerapan htmlspecialchars()

$*"warn"* = isset($_GET[*"warn"*]) ? $_GET[*"warn"*] : ""; Baris ini digunakan untuk mengambil nilai dari parameter *"warn"* yang dikirim melalui URL. $_GET adalah variabel PHP yang digunakan untuk mengakses data yang dikirim melalui metode GET. Fungsi isset() digunakan untuk memeriksa apakah parameter *"warn"* ada dalam URL. Jika parameter tersebut ada, maka nilainya akan disimpan ke dalam variabel $*"warn"*. Jika tidak ada, maka nilai variabel $*"warn"* akan diatur menjadi string kosong ("").
$*"warn"* = htmlspecialchars($*"warn"*, ENT_QUOTES, 'UTF-8'); Baris ini digunakan untuk mengamankan nilai variabel $*"warn"* dengan menggunakan fungsi htmlspecialchars(). Fungsi ini digunakan untuk mencegah serangan Cross-Site Scripting dengan mengubah karakter-karakter khusus menjadi entitas HTML. Parameter pertama dalam fungsi htmlspecialchars() adalah string yang akan diubah. Parameter kedua (ENT_QUOTES) digunakan untuk mengonversi tanda kutip ganda dan tanda kutip tunggal menjadi entitas HTML juga. Parameter ketiga ('UTF-8') adalah kode penentu set karakter yang digunakan. Dengan menggunakan htmlspecialchars(), nilai variabel $*"warn"* akan diubah sehingga karakter-karakter khusus seperti <, >, &, dan tanda kutip akan diubah menjadi entitas HTML, sehingga mengurangi risiko serangan XSS.

3. Penerapan *Secure Socket Layer* (SSL)

Pada gambar IV.19 dapat dilihat sertifikat penerapan SSL telah dilakukan. SSL mengubaah protokol HTTP menjadi HTTP Secure dengan tujuan mengenkripsi data yang dikirimkan antara client dan server dan juga memberikan rasa aman kepada pengguna *website*.



GAMBAR IV. 19
Penerapan SSL

4. Menggunakan *Library* atau *Framework* yang terverifikasi untuk menutup celah *Cross Site Scripting*

Hal ini tidak dapat dilakukan karena pada *Website* Akademik Penunjang Administrasi di Institusi XYZ *framework* yang digunakan adalah *CodeIgniter* yang telah di *custom*. Perlu penjelasan mendalam dari sisi *source code* dan dapat merubah struktur dari *source code* yang telah tersedia.

H. Pengujian ulang Pasca *Mitigation*

Pengujian ulang terhadap *website* dilakukan dengan tujuan untuk mengetahui hasil implementasi yang telah dilakukan pada perancangan mitigasi dengan cara melakukan perbandingan sebelum dilakukan mitigasi dan sesudah dilakukan mitigasi. Dan juga dilakukan *scanning* ulang menggunakan OWASP ZAP untuk mendeteksi celah *Cross Site Scrpting* tersebut masih tersedia atau tidak.

1. Tampilan sebelum dilakukan pengujian ulang pasca Mitigasi menggunakan satu *payload* hasil pengujian dari XSStrike.

GAMBAR IV. 20
Tampilan Sebelum Mitigasi

Terlihat pada gambar IV.20 sebelum dilakukan mitigasi, *popup* berupa dialog input dengan button "OK" dan "Cancel", juga pada bagian *alert error* menunjukkan kerusakan terhadap source code muncul akibat dari injeksi

script yang mengandung kode untuk menghindari filter pencegah *Cross Site Scripting*.

2. Tampilan Setelah dilakukan pengujian ulang pasca Mitigasi menggunakan satu *payload* yang sama hasil pengujian dari XSStrike.

GAMBAR IV. 21
Tampilan Setelah Mitigasi

Terlihat pada gambar V.21 setelah dilakukan mitigasi, *popup* berupa dialog input dengan button "OK" dan "Cancel" sudah tidak muncul tetapi pada bagian *alert error* masih menunjukkan kerusakan terhadap source code.

3. *Vulneravility scan* Pasca Mitigasi menggunakan OWASP ZAP

Proses *scanning* ulang menggunakan OWASP ZAP yang telah dilakukan masih mendeteksi celah *Cross Site Scripting* dengan *risk* yang masih *high*.

Dapat disimpulkan bahwa kerentanan tersebut berada dari sisi *source code* yang digunakan Sehingga perlu adanya pengecekan ulang terhadap optimasi penggunaan *scripting* dari sisi *source code* dan perlu adanya *upgrade framework* dari sisi penggunaan *source code* yang lebih mendukung untuk mengatasi celah celah keamanan.

V. KESIMPULAN

Dari hasil pengujian Vulnerability Analysis menggunakan OWASP ZAP, celah Cross Site Scripting teridentifikasi dengan jenis reflected pada halaman sign in dengan parameter "warn" dengan tingkat risiko high dan tingkat confidence yang medium, sehingga dilakukan verifikasi terhadap celah Cross Site Scripting dengan melakukan uji eksploitasi menggunakan tools XSStrike dan Burp Suite, hasil dari uji eksploitasi celah tersebut dapat terverifikasi di halaman sign in pada parameter "warn" sehingga harus dilakukan proses mitigasi untuk mengurangi atau menutup celah tersebut.

Proses Mitigasi dilakukan menggunakan rekomendasi solusi dari hasil pengujian yang telah dilakukan dengan cara menambahkan header Content Security Policy (CSP) pada bagian header halaman log in, menggunakan fungsi htmlspecialchars() pada variabel "warn" yang memberikan alert error ketika pengguna salah melakukan input username dan password dan penerapan Secure Socker Layer (SSL). Dari hasil pengujian ulang menyatakan hasil scanning ulang menggunakan OWASP ZAP masih mendeteksi celah Cross Site Scripting dengan risk yang masih high. Tetapi dari hasil pengujian ulang menggunakan payload memperlihatkan bahwa script yang dapat memunculkan popup alert sudah tidak muncul dan script yang dapat memodifikasi alert error masih terdeteksi. Dapat disimpulkan bahwa kerentanan tersebut berada dari sisi source code yang digunakan Sehingga perlu adanya pengecekan ulang terhadap source code.

REFERENSI

[1] T. Revolino Syarif dan D. Andri Jatmiko, "ANALISIS PERBANDINGAN METODE WEB SECURITY PTES, ISSAF DAN OWASP DI DINAS KOMUNIKASI DAN INFORMASI KOTA BANDUNG," 2019. [Daring]. Tersedia pada: http://elibrary.unikom.ac.id/id/eprint/880

[2] M. I. Hany, A. Bhawiyuga, dan A. Kusyanti, "Implementasi Cross Site Scripting Vulnerability Assessment Tools berdasarkan OWASP Code Review," 2021. [Daring]. Tersedia pada: http://j-ptiik.ub.ac.id

[3] I. Riadi, R. Umar, dan T. Lestari, "Analisis Kerentanan Serangan Cross Site Scripting (XSS) pada Aplikasi Smart Payment Menggunakan Framework OWASP," *JISKA (Jurnal Informatika Sunan Kalijaga)*, vol. 5, no. 3, hlm. 146–152, Nov 2020, doi: 10.14421/jiska.2020.53-02.

[4] Y. Yulianingsih, "Melindungi Aplikasi dari Serangan Cross Site Scripting dengan Metode Metacharacter," *Jurnal Nasional Teknologi dan Sistem Informasi*, vol. 3, no. 1, hlm. 83–88, Apr 2017, doi: 10.25077/TEKNOSI.v3i1.2017.83-88.

[5] A. Kurniawan, "Penerapan Framework OWASP dan Network Forensics untuk Analisis, Deteksi, dan Pencegahan Serangan Injeksi di Sisi Host-Based," *Jurnal Telematika*, vol. 14, no. 1, 2019.