ANALISIS CELAH KEAMANAN APLIKASI WEBSITE MENGGUNAKAN METODE *CRAWLING* TERHADAP SERANGAN *SQL INJECTION*

[1,*]**Ari Dimas Yudistiawan,** [2]**Nuril Anwar**

[1,2] Program Studi Informatika, Universitas Ahmad Dahlan, Jl. RingRoad Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166, Indonesia
[1,*]aridimas145@gmail.com, [2]nuril.anwar@tif.uad.ac.id

Abstract

The internet is a global communication network that can be used as the latest media and information source. The use of the internet as a medium for providing information is widely used in the fields of business and marketing because it is considered more effective and efficient, codelatte is one of them. The application of https that is evenly distributed to users on the Codelatte website makes communication and servers safer, but unlike the implementation of https, the use of WordPress security is not applied to every page owned by the Codelatte website which causes SQL Injection loopholes to be found on old pages that have not been updated by developer. The open SQL Injection gap, of course, invites danger to all data contained on the website.

This study aims to analyze the security gaps found on the codelatte website against SQL Injection attacks using the crawling method to hide data. The steps include filling in the URL, processing the URL, crawling data, making a request to the website then proceeding to the web server, then producing an output in the form of an html file, the html file is used for vulnerability testing, and the test results are displayed on the main interface of the Acunetix web vulnerability application. scanner. Data collection through Observation and Crawling. The results of the crawling will be used as material in conducting penetration testing using Sqlmap and manual testing.

The result of the research is to reduce security vulnerabilities detected by crawling to the informational, low, and medium levels, while the results of the crawling carried out produce information on the location of security vulnerabilities at informational, low, medium, and high and 4 levels. The information that has been collected is used as data in conducting tests to find out how far the website's security is currently able to limit intruders from accessing the database from outside the system. The results of these tests are poured into a penetration testing report which will later be used as a basis for improving Codelatte's website.

Keywords *: Security Gap Analysis, Crawling, Information Security, Website Security, Sql Injection.*

PENDAHULUAN

Internet sebagai jaringan komunikasi global dapat dijadikan sebagai media dan sumber informasi terkini, seperti ilmu pengetahuan, teknologi, hiburan, bisnis, dan sumber informasi lainnya. Kemudahan serta kenyamanan seperti ini menyebabkan internet selalu digunakan dan dibutuhkan. Namun dibalik kemudahan dan kenyamanan yang diberikan, ternyata terdapat salah satu aspek yang saat ini masih kurang diperhatikan oleh pengguna internet, yaitu *security. Security* merupakan salah satu aspek penting pada *media* yang menghubungkan pengguna dengan internet.

Teknologi tidak dapat dipungkiri lagi bahwasannya dapat membawa dampak negatif yang tidak sedikit. Perkembangan internet membuat kejahatan yang sebelumnya bersifat konvensional seperti ancaman, pencurian, penggelapan, pemalsuan dan penipuan kini dapat dilakukan secara daring melalui media internet yang memiliki keuntungan berupa minimnya resiko tertangkat oleh individu maupun kelompok .

Dampak negatif yang merugikan dan berkembang secara pesat tersebut, menghasilkan suatu pemikiran bahwa tidak ada komputer dan jaringan yang benar-benar aman hingga saat ini. Hal ini terbukti dari banyaknya *hacker-hacker* pemula yang berseliweran muncul di berbagai platform internet untuk melakukan tindak kriminal dunia maya.

Seiring berkembang pesatnya teknologi dari waktu ke waktu, banyak *hacker-hacker* muda bermunculan untuk melakukan kejahatan dengan berbagai macam metode-metode serangan baru yang mereka gunakan untuk malancarkan aksinya, seperti *SQL Injection, Directory Traversal Attack, Cross Site Scripting* dan lain sebagainya. *SQL Injection* merupakan salah satu contoh metode paling berbahaya ketika berhasil dimanfaatkan dengan baik oleh *hacker* atau *cracker*.

SQL Injection sudah populer sejak lama dalam dunia per-*hacking*an sebagai salah satu teknik *web application hacking*, walaupun teknik ini sudah terbilang lama akan tetapi teknik ini masih menjadi salah satu teknik andalan karena sifatnya yang dapat merusak database dari suatu situs. Salah satu metode dalam teknik *SQL Injection* adalah dengan melakukan penginputan perintah-perintah standar dalam *SQL* seperti *insert, create, update, union, select, view*, beserta perintah standar lainnya yang tak asing lagi bagi yang sudah mempelajari *SQL* secara mendalam maupun yang baru belajar[1].

SQL Injection adalah suatu metode penyerangan terhadap kerentanan yang terjadi ketika penyerang memiliki kemampuan untuk mempengaruhi *Structured Query Language (SQL) Query* yang melewati suatu aplikasi ke database *back-end* yang mampu memanipulasi apa yang akan diteruskan ke database[2]. Penyerang memanfaatkan sintaks dan kemampuan dari *SQL* itu sendiri, serta kekuatan dan fleksibilitasnya untuk mendukung operasi dan fungsionalitas sistem yang tersedia ke database. *SQL Injection* tersebut diakibatkan oleh tidak adanya filter terhadap bahasa *SQL* sehingga penyerang dapat memanfaatkan kerentanan tersebut dengan tujuan meraih akses pada sistem database yang berbasis *SQL* dan menyusup ke dalamnya[3].

Penelitian ini melakukan uji coba pencarian celah keamanan SQL Injection terhadap 5 website diantaranya ialah uad.ac.id, codelatte.id, pt-jayapura.go.id, mk.mercubuana-yogya.ac.id, jateng.polri.go.id. Dari kelima website tersebut tiga diantaranya ditemukan celah keamanan SQL Injection di dalamnya, ketiga website tersebut ialah codelatte.id, pt-jayapura.go.id, mk.mercubuana-yogya.ac.id, dan pt-jayapura.go.id. Namun izin pemerbaikan hanya dikirimkan ke website codelatte.id serta pt-jayapura.go.id, dan website codelatte.id lah yang mengizinkan pemerbaikan di dalam websitenya, oleh karena itu website codelatte.id terpilih menjadi objek pada penelitian ini.

Codelatte merupakan sebuah organisasi yang berfokus dalam jasa pengembangan web (*web developer*) dan pengujian penetrasi (*penetration testing)*. *Codelatte* memiliki *website* sebagai sarana periklanan, pembrandingan, dan peningkatan eksistensi. *Website* tersebut menjadi sasaran menggunakan protokol yang bertugas untuk mengirimkan data dari *web server* ke browser menggunakan HTTPS (*Hypertext Transfer Protocol Secure*) pada tiap tiap sub domainnya. Setelah melakukan *Lookup* diketahui bahwa pada *website* tersebut menggunakan PHP dan Wordpress sebagai *Content Management Sistemnya* (CMS)nya.

Wordpress merupakan CMS (*Content Management System*) dengan pengguna paling banyak saat ini dalam pengembangan situs *website*. Kemudahan dan sistem keamanan yang diberikan *Wordpress* selaku penyedia layanan sangat memuaskan penggunanya, sehingga mayoritas penggunanya enggan berpaling ke CMS lainnya[4].

Penerapan *Wordpress* berbanding terbalik dengan penerapan HTTPS, penggunaan *Wordpress* tidak diterapkan pada tiap-tiap subdomain yang dimiliki oleh *website Codelatte*. Sisa file *website* lama yang berada dalam file manager menyebabkan ditemukannya celah pada subdomain lainnya dari *website* tersebut, contohnya saja setelah melakukan *crawling* dan *scanning* yang berfungsi untuk menemukan celah keamanan

apa saja yang terdapat di dalam situs *web*, ditemukan bahwa terdapat celah *SQL Injection* terbuka pada *sub-domain web Codelatte*[5].

Crawling tidak hanya mengakibatkan terekspos enya suatu kelemahan situs *website*, *crawling* juga dapat dimanfaatkan oleh *pentester* untuk mencari letak kelemahan dan memperbaikinya sehingga tingkat keamanan dari suatu *website* meningkat, *tools acunetix* memiliki *engine* untuk melakukan *crawling* manual jika pengguna tidak ingin menggunakan automatic testing pada *acunetix web vulnerability scanner*, *tools* ini sangat berguna bagi kedua pihak, baik dari sisi penyerang maupun pentester guna menemukan kelemahan-kelemahan situs untuk menyusup kedalam sistem ataupun memperbaikinya[6].

Dirsearch merupakan alat yang mengandalkan metode *bruteforce* dan pengecekan http respond code pada folder yang sering digunakan pada website dalam proses *crawling*nya. Dengan memanfaatkan tools *dirsearch* ini seseorang dapat dengan mudah Mengetahui *path directory*, Mencari data sensitif, Mencari *hidden directory*, dan lain sebagainya.

Penyerang yang berhasil menemukan dan memanfaatkan celah *SQL Injection* pada sub-domain situs tersebut dan jika telah berhasil mencuri akses masuk ke dalam *database*, maka penyerang akan memiliki keleluasaan untuk memanipulasi data-data yang terdapat didalam *website*. Tidak berhenti di situ saja, penyerang juga mampu melakukan *jumping* antar sub-domain bahkan hingga domain utama untuk memasang backdoor yang mengakibatkan penyerang mampu mendapatkan akses penuh dari dalam domain tersebut.

Hal-hal tidak bertanggung jawab yang mampu dilakukan oleh penyerang dari dalam database *website* tentu dapat menimbulkan resiko bocornya *Confidentiality* organisasi, berkurangnya *integrity* dari informasi dan *availability* data ketika dibutuhkan menjadi tidak pasti[7].

Serangan *SQL Injection* merupakan metode penyerangan dengan dampak yang sangat berbahaya serta dapat dilakukan dari jarak jauh oleh pihak-pihak tidak bertanggung jawab dan memperbaikinya dapat membantu menjaga privasi *website Codelatte*, maka penulis memutuskan untuk mengambil tema ini dengan menggunakan judul "**ANALISIS CELAH KEAMANAN *WEBSITE* MENGGUNAKAN METODE CRAWLING TERHADAP SERANGAN SQL INJECTION"**

METODE PENELITIAN

Objek Penelitian

Objek dari penelitian ini ialah sebuah *website* yang bergerak di bidang *web development* dan *penetration testing, website* orgnasisasi tersebut beralamatkan di codelatte.org, yang saat ini berdomain utama di codelatte.id.

Metode Pengumpulan Data

Metode pengumpulan data pada penelitian ini antara lain :

A. Metode Observasi

Melakukan pengamatan cara kerja perangkat lunak dan jaringan yang digunakan. Pengamatan dilakukan pada *website* yang dapat diakses dan melakukan percobaan pada fitur-fitur di tiap-tiap halaman yang memiliki kemungkinan terdapatnya celah keamanan. Dengan melakukan observasi dapat membantu untuk menghemat waktu dalam melakukan testing terhadap *website* target.

B. Crawling untuk mencari *vulnerability*

*C*rawling sebagai algoritma pengumpul informasi, celah keamanan yang terdapat didalam

website codelatte.org akan terekspose dan terkumpul berdasarkan urutan *low* atau *high*nya suatu *vulnerability*. Metode ini dilakukan menggunakan bantuan engine *crawler* milik *software acunetix web scanner.*

Tahap Penelitian

Metode *crawling* pada penilitian ini menggunakan engine crawling pada *tool acunetix web vulnerability scanner* untuk mencari letak kerentanan pada situs codelatte.org yang tentunya memiliki skenario tersendiri. Sebelum memasuki tahap penelitian akan dijabarkan scenario crawling, yaitu

A. Memasukkan URL

Website Codelatte tentunya memiliki alamat URL sebagai alat untuk mengidentifikasi dan mengakses situs *website*nya. URL tersebut dapat digunakan sebagai langkah awal untuk memulai *crawling.* Pada skenario pertama ini URL dimasukkan ke dalam *tool acunetix.*

B. Pemrosesan URL

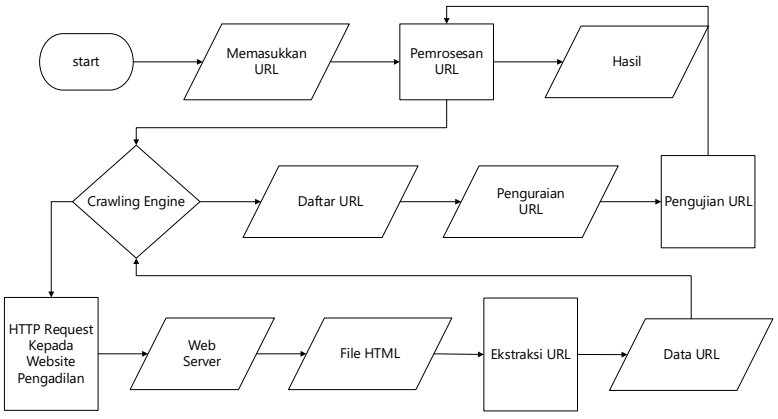
Pada tahap ini *tool* acunetix akan memasukkan URL yang akan di proses ke dalam engine crawling miliknya.

Pemrosesan URL dimulai ketika URL *website* telah dimasukkan ke dalam kolom inputan. URL tersebut dibawa menuju *crawling engine* dan sistem akan mengambil keputusan, jika URL belum dilakukan *crawling* maka menuju step berikutnya yaitu melakukan *http request* ke dalam *web server* milik *website Codelatte*, yang selanjutnya *web server* akan menghasilkan keluaran file HTML.

File HTML hasil keluaran *web server* digunakan untuk mengambil dan mengekstraksi URL lain yang terdapat di dalamnya, kemudian dimasukkan ke dalam data URL. Data URL selanjutnya mengirimkan URL hasil ekstraksi ke dalam *crawling engine* untuk melakukan *crawling* terhadap URL baru dari hasil ekstraksi file HTML, dan mengirimkan URL awal ke dalam Daftar URL sebagai URL yang sudah melewati tahapan *crawling*.

Penguraian URL dilakukan dengan tujuan menguraikan URL dalam daftar URL untuk membawanya ke dalam tahapan pengujian URL. Tahap pengujian URL ini memiliki banyak macam proses *testing*, terkhususnya *SQL Injection*,

Hasil dari Pengujian URL akan dibawa kembali menuju proses awal yaitu pada saat Pemproses URL, untuk menghasilkan keluaran berupa hasil pengujian. Selanjutnya URL baru yang ditemukan pada saat ekstraksi akan melalui tahap yang sama dengan URL pertama tadi, Sehingga Diagram Alur Skenario Lengkapnya akan terlihat seperti pada Gambar 1.



Gambar 1 Diagram Alur Skenario

Diagram pada Gambar 1. menunjukkan alur keseluruhan selama aplikasi memproses *crawling* dari awal mula URL dimasukkan, hingga mendapatkan banyak uraian hasil *vulnerability* dalam bentuk urutan dari level *threat low* hingga *level threat high.*

C. Hasil Crawling

Hasil dari seluruh rangkaian pemrosesan URL akan ditampilkan secara mendetail ke sistem antarmuka, sehingga kerentanan terendah dan tertinggi dari hasil *scanning* menggunakan *engine crawling* yang dimiliki oleh *tools acunetix web vulnerability scanner* yang dimiliki situs *Codelatte* dapat terlihat, dari hasil *scanning* menggunakan *engine crawling* yang dimiliki oleh *tools acunetix web vulnerability scanner.*

HASIL DAN PEMBAHASAN

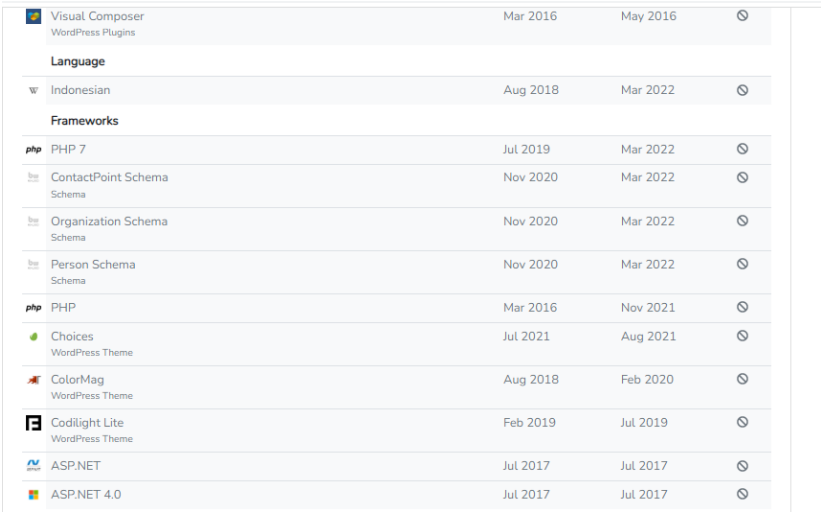
Alur proses yang dilakukan hingga membuat laporan meliputi *Scope, reconnaissance, vulnerability detection, information analysis & planning*, penetration testing, Izin dan pembenahan sistem. Proses *vulnerability* detection ini di dalamnya terdapat metode *crawling* dalam menemukan celah keamanan pada *website* Codelatte.org dengan memanfaatkan *crawling engine* milik *tools* Acunetix.

A. SCOPE

Scope merupakan langkah awal dalam melakukan *penetration testing*, menetapkan *scope penetration testing* dilakukan untuk mencegah melebarnya pengujian pada saat proses *penetration testing* berjalan. *Scope penetration testing* yang digunakan pada penelitian ini tentu saja *website* milik *Codelatte*.

B. Reconnaissance

Reconnaissance merupakan tahapan selanjutnya ketika *scope penetration testing* telah selesai ditentukan, *Reconnaissance* sendiri memiliki artian sebagai pengumpulan informasi sebanyak mungkin. Informasi yang dikumpulkan ialah informasi mengenai *website Codelatte* yang selanjutnya digunakan sebagai bahan untuk *penetration testing*. Pengumpulan informasi tersebut akan diawali dengan pemeriksaan arsitektur *website* menggunakan builtwith seperti pada Gambar 2.



Gambar 2 Arsitektur Web Codelatte.org

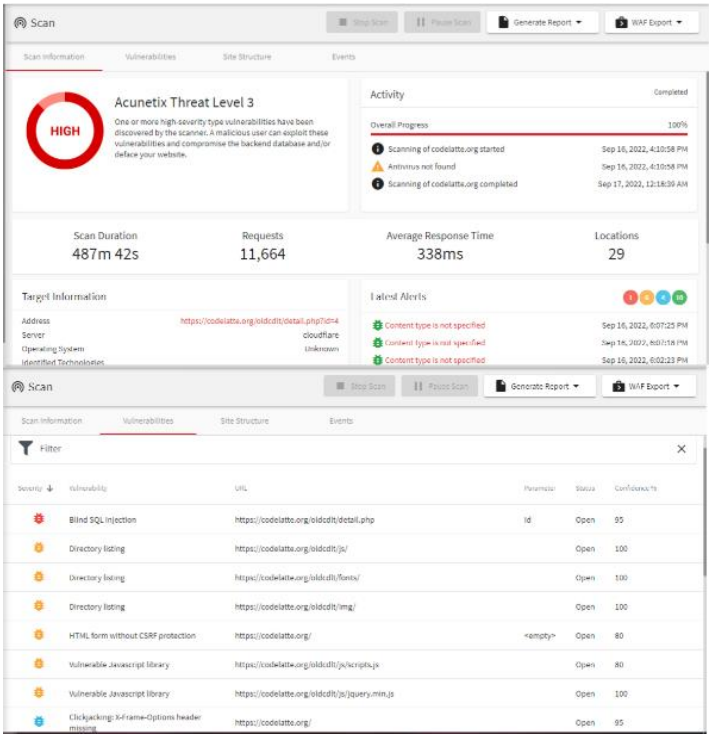
Gambar 2 menunjukkan informasi arsitektur *website* yang ditemukan oleh *tools* builtwith-lookup. *Tools* builtwith merupakan *tools* yang mampu mendeteksi arsitektur teknologi yang digunakan *website* dan memaparkannya dalam bentuk uraian menurun. Berdasarkan pencarian tersebut ditemukan bahwa *Codelatte* tidak menggunakan *Laravel* sebagai frameworknya, sehingga terdapat kemungkinan ditemukannya celah

keamanan *sql injection*. Selanjutnya setelah mengetahui informasi arsitektur *website* maka *reconnaissance* dilakukan secara manual dengan cara memeriksa langsung tiap halaman web target dengan *tools* pencari dir.

DirSearch merupakan alat yang dirancang untuk melakukan *crawling* direktori dan file pada sebuah *website*. Setelah prosesnya selesai, maka dirsearch akan menghasilkan keluaran list directory lengkap dengan alamat URLnya.

C. *Vulnerability* Detection

Tahap *vulnerability detection* atau deteksi celah keamanan pada domain *Codelatte* bertujuan untuk menemukan semua celah keamanan berbahaya yang terdapat didalam tumpukan halaman *website.* Acunetix merupakan alat untuk pengujian keamanan yang diakses melalui virtual server. Sebagai alat yang berguna untuk pengujian keamanan, acunetix memeriksa berbagai macam kode, file dan keseluruhan website yang berpotensi mengandung kerentanan seperti SQL *injection*, xss, dan banyak kerentanan lainnya. Hasil crawling tersebut dipaparkan pada Gambar 4.9



Gambar 3 Hasil *Scanning* Acunetix

Gambar 3 menunjukkan bahwa terdapat satu celah *Blind SQL Injection* yang terdapat pada *website*, dua celah keamanan berlevel *medium threat*, empat celah keamanan berlevel *low threat*, dan tiga celah keamanan pada level *informational*. Setelah ditemukannya letak *vulnerability* pada *website* melalui metode *crawling*, selanjutnya akan dilakukan analisis dan planning terkait informasi yang baru saja ditemukan pada tahapan *information analysis & planning.*

D. INFORMATION ANALYSIS & PLANNING

Vulnerability detection pada tahap sebelumnya menghasilkan beberapa temuan celah keamanan pada *website* codelatte.org, sehingga pada tahap *information analysis & planning* akan memilih celah keamanan mana yang akan dieksploitasi dalam pengujian penetrasi, diantara celah keamanan tersebut celah keamanan yang digunakan untuk pengujian terdapat pada tabel 4.1.

Tabel 4 Celah Keamanan Terpilih

No	Jenis Kerentanan yang digunakan
1.	SQL Injection
2.	Directory listing

Celah keamanan pada table 4.1 dipilih berdasarkan analisa informasi yang dilakukan dimana *Sql Injection* menjadi fokus utama untuk melakukan *penetration testing* karena kerusakan saat penginjeksian terhadap sql dilakukan dapat meliputi seluruh data *website* yang dibutuhkan untuk mendapatkan akses masuk ke dalam sistem, sementara celah *directory listing* menjadi *support* atas berjalannya *sql injection* tersebut.

E. Penetration Testing

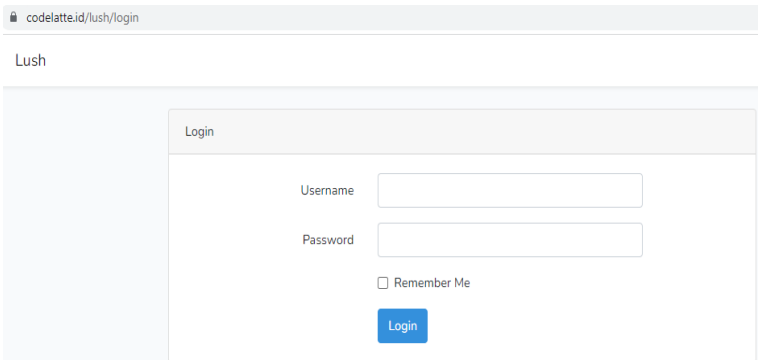
Penetration testing dilakukan guna memvalidasi atau membuktikan bahwasannya celah keamanan yang ditemukan benar adanya, sehingga celah keamanan yang tervalidasi selanjutnya digunakan sebagai sasaran untuk perbaikan. Pengujian tersebut dilakukan bertahap sebagai berikut :

1. SQL Injection

Vulnerability yang ditemukan dari hasil *crawling* pada tahap *vulnerability detection* kemudian digunakan sebagai target, guna memvalidasi atau membuktikan eksistensi celah keamanan tersebut. Identifikasi SQL Injection melalui URL pada domain codelatte.org, dilakukan dengan cara menambahkan *string* (') pada akhiran URL. Identifikasi tersebut akan menghasilkan temuan berupa ciri celah keamanan yang dapat digunakan untuk pengujian *SQL Injection*, alamat URL yang akan digunakan untuk pengujian adalah, http://codelatte.org/oldcdlt/detail.php?id=4, setelah string ditambahkan, halaman tersebut memperlihatkan *error* dengan menampilkan kekosongan.

Halaman yang memiliki ciri celah keamanan SQL Injection tadi setelah berhasil diidentifikasi, maka langkah selanjutnya yaitu melakukan uji coba dengan pengujian menggunakan SQL Map versi 1.6.8.2#dev. Pengujian melalui SQL Map membuahkan hasil berupa masuknya injeksi yang dilakukan oleh SQLMap ke dalam *database* dan membuka *tables* yang ada pada *database website*. Setelah membuka tables pada database website ditemukan bahwa salah satu satu *tables* yang dimiliki *database* lush yaitu *tables* users yang memiliki akses *login* ke dalam *login panel website* codelatte.org.

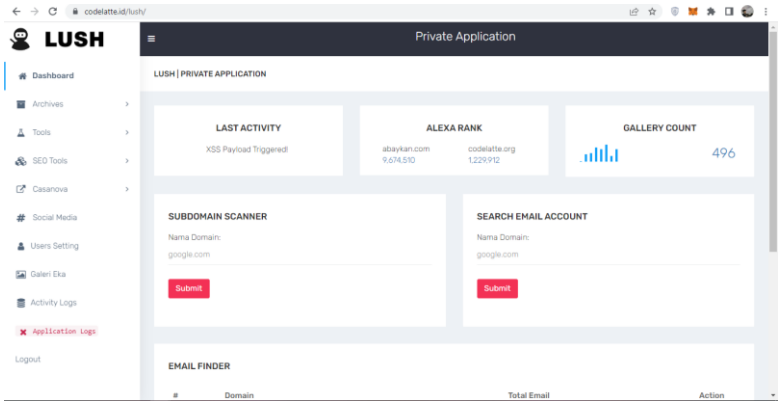
Database lush memperlihatkan *tables Username/password* pada saat *database* dibuka, ketika sudah melakukan *dump* dan menemukan *username/password* maka akses *administrator* telah di dapatkan yang kemudian dilakukan pengujian untuk memasukkan username/password tadi ke dalam *admin panel* yang ditemukan, *admin panel* tersebut terlihat pada Gambar 5



Gambar 5 *Login* Admin Page Lush

Gambar 5 menunjukkan tampilan dari *admin panel private application* milik *website Codelatte*.

Dashboard admin panel private application akan terbuka ketika menggunakan *username* dan *password* yang benar, dan akan menampilkan tampilan seperti 6



Gambar 6 Index Admin Page Lush

Gambar 6 menunjukkan halaman *dashboard private application* dimana admin didalamnya memiliki akses untuk membuat, menggunakan, menghapus dokumentasi artikel dan *tools* yang ada di dalam *website* dan mengedit *username, password* milik *user* dan *admin*.

2. LISTING DIRECTORY

Celah Keamanan *Listing Directory* sebenarnya adalah fungsi web yang menampilkan list daftar isi semua file ketika tidak ada indeks di dalam direktori tersebut, seperti index.php, index.html dan default.asp. Biasanya, seorang pengguna yang membuka www.contoh.org/oldcdlt/js tanpa menuliskan nama file membuat server web memproses permintaan ini dan akan menampilkan file index yang berada di dalam direktori tersebut dan situs web yang sebenarnya akan muncul. Akan tetapi, jika file index tidak ada seperti pada kasus ini, web server akan mengembalikan *listing directory* atau daftar isi dari direktori tersebut. Fungsionalitas seperti ini dapat diparalelkan menggunakan *command* 'ls' pada unix dan linux serta 'dir' pada Windows. Ditemukan 4 celah keamanan bertipe medium ini di dalam website codelate.org yang masing-masing berisi informasi mengenai, JS, CSS, IMG dan Font yang dimana ke-empat direktori tersebut dapat diakses dan diunduh secara publik.

F. Izin Pembenahan Sistem

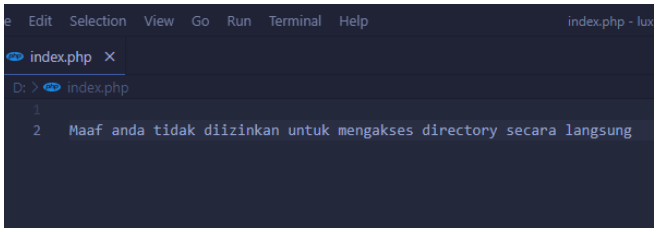
Pembenahan sistem yang dilakukan memerlukan izin sebagai kekuatan hukum atas eksploitasi yang dilakukan selama melakukan *penetration testing*. Oleh karena itu dilakukan permintaan izin *penetration testing* sekaligus perbaikan melalui email yang tertera di kolom contact us website *Codelatte. Email* perizinan tersebut dikirimkan kepada pengembang codelatte.org, dan ditanggapi dengan respon positif dari pihak pengembang berupa permintaan untuk tetap *keep in touch* pada saat proses penetration testing berjalan. Setelah mengirimkan email perizinan maka penelitian dilanjutkan ke tahap perbaikan.

PEMERBAIKAN

Tahap ini berisikan tentang perbaikan terhadap celah keamanan yang ditemukan dan telah selesai melalui proses *penetration testing* pada *website* codelatte.org. Kerentanan tersebut meliputi *Sql Injection* dan *Directory Listing* yang ditemukan pada salah satu dir lama milik codelatte.org.

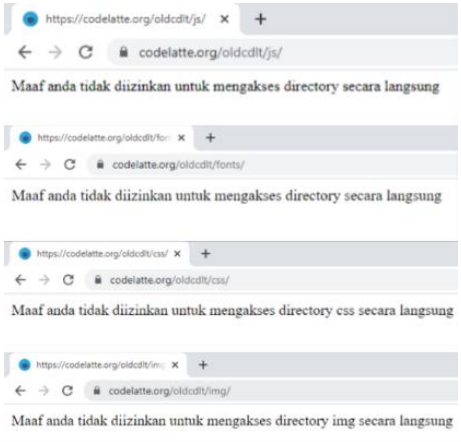
A. Pemerbaikkan Directory Listing

Perbaikan *directory listing* dapat dilakukan dengan metode yang sederhana sehingga untuk mengupayakan pemerbaikkan yang efektif dan efisien, akan dilakukan metode terbaik. Metode tersebut yaitu pembuatan index.php yang selanjutnya diletakkan ke dalam directori bercelah dirlisting. Proses metode pembuatan file index.php ini berfungsi untuk mengisi kekosongan file yang dipanggil web server saat user meminta untuk membuka folder direktori tanpa menyertakan file yang spesifik. Pembuatan file index.php ini dilakukan dengan cara membuka code editor dan memasukkan *plaintext* seperti pada Gambar 7



Gambar 7 Pembuatan Index.php

Gambar 7 menunjukkan *plaintext* yang tersimpan dengan nama index.php. Setelah itu upload file index.php yang baru saja dibuat ke dalam *directory* yang memiliki celah keamanan *directory listing*. Kemudian setelah *upload* berhasil tampilan *directory listing* tadi akan terlihat seperti pada Gambar 8



Gambar 8 Hasil Dari Pembenahan Directory Listing

Gambar 8 menampilkan isi daripada file index.php dalam *directory* js yang memiliki celah keamanan *directory listing*, sehingga tampilan daftar isi file atau *directory listing* tidak ditampilkan lagi. Pencegahan yang dilakukan yaitu dengan menambahkan *code* "Option All -Indexes" pada Htaccess sehingga sebelum memasuki file index.php, *web server* akan mengirimkan pesan error kepada *interface* pengguna

B. Pemerbaikkan SQL Injection

Penyebab adanya celah keamanan *SQL Injection* harus diketahui dan dipahami terlebih dahulu sebelum menuju ke tahap pemerbaikkan, dengan harapan pemerbaikkan yang dilakukan efektif serta tepat sasaran. Untuk mengetahui hal tersebut dilakukan pemeriksaan terhadap *source code* yang memiliki celah kelemahan dan ditemukkan bahwa koneksi dari *website* ke *database* masih menggunakan ekstensi mysqli tanpa menggunakan lapisan abstraksi *database* seperti PDO (*PHP Data Object*). PDO merupakan lapisan akses *database* yang menyediakan metode akses yang seragam ke beberapa *database*. Kode yang digunakan dalam penerapan PDO dapat menghindarkan proses *bad query* yang dilakukan penyerang untuk kasus *SQL Injection*. Dengan demikian untuk mengatasi penyebab celah keamanan *SQL Injection* tersebut maka dilakukanlah penerapan PDO pada *Source Code* yang digunakan pada halaman codelatte.org.

PDO diterapkan pada *source code* yang digunakan dengan mengganti query mysqli yang terkonek dengan *database* sesuai dengan code PDO seperti pada Gambar 9.

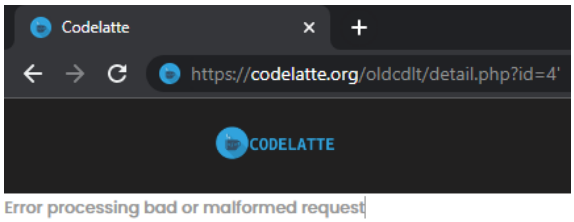
```
<?php
 if (isset($_GET['id'])){
   $id = $_GET['id'];
   if ( is_numeric($id) == true){
       try{
           $dbh = new PDO('mysql:host=localhost;dbname=names', "root", "");
           $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
           $q = "SELECT * FROM db_barang where id = " . $_GET['id'];
           $sth = $dbh->prepare($q);
           $sth->bindParam(':id', $id);
           $sth->execute();
           $sth->setFetchMode(PDO::FETCH_ASSOC);
           $result = $sth->fetch();
           $dbh = null;
       }
       catch(PDOException $e){
           error_log('PDOException - ' . $e->getMessage(), 0);
           http_response_code(500);
           die('Error establishing connection with database');
        }
       } else{
        http_response_code(400);
        die('Error processing bad or malformed request');
       }
      }

?>
```

Gambar 9 Pengaplikasian PDO pada Source Code

Gambar 9 menunjukkan *source code* codelatte.org yang sudah dilakukan perubahan berupa penerapan PDO pada codingan database.51-78 Baris 52 kode berfungsi untuk melakukan pengecekan apakah variabel GET 'id' sudah terpasang, baris 54 kode berfungsi sebagai validasi data sebelum masuk ke *database*. Dalam hal ini memeriksa apakah nilai parameter GET 'id' adalah numerik atau bukan, baris 56 berfungsi untuk melakukan setup koneksi ke *database*, baris 57 berfungsi untuk merekam kesalahan dan menuliskannya ke file log, baris 58 merupakan kode untuk menjalankan query database dan baris ini merupakan kode PDO yang memperbaiki celah *SQL Injection*, baris 59 berfungsi untuk mempersiapkan *string query* SQL, Baris 60 berfungsi untuk mengikat parameter ke *variabel statement*, baris 61 berfungsi untuk mengeksekusi pernyataan, baris 62 berfungsi untuk mengubah mode pengambilan ke FETCH_ASSOC untuk mengembalikan *array* yang diindeks menurut *column name*, baris 63 berfungsi untuk mengambil hasil, baris 64 berfungsi untuk menutup koneksi ke *database*, baris 66 - 67 berfungsi untuk mencatat pengecualian PDO ke dalam log sistem php menggunakan mesin log milik sistem operasi, baris 68-69 berfungsi untuk menghentikan eksekusi dan mengembalikan tampilan *error* 500 yaitu kesalahan sistem

internal, baris 71-73 dijalankan ketika nilai parameter GET ‘id’ bukanlah numerik, menghentikan eksekusi, dan mengembalikan a 'Bad request' HTTP status code (400). Ketika data telah diserahkan ke PDO seperti sekarang ini, maka dilakukan pemeriksaan dengan memasukkan karakter kutip (‘) pada akhiran URL seperti pada Gambar 10.



Gambar 10 Hasil Pengaplikasian PDO

Gambar 10 memperlihatkan *error* ketika isi dari parameter bukanlah angka melainkan karakter lain, hal ini telah diatur didalam PDO pada kodenya untuk mencegah penyerang yang berusaha menerobos *database* sistem lebih dalam lagi.

PDO yang diterapkan sayangnya hanya salah satu garis pertahanan untuk mengatasi SQL Injection, perlindungan yang diberikan PDO tidak mencakup kerentanan terhadap validasi input lainnya, tetapi berfungsi dengan baik dalam melindungi *website* dari SQL injection. Dengan menggunakan PDO, strategi lain untuk pemaksimalan perlindungan yaitu melakukan filtering terhadap data, artinya tidak mengizinkan data atau karakter berbahaya untuk dimasukkan ke dalam query. Maka selanjutnya dilakukan filtering terhadap karakter berbahaya yang mampu memperlihatkan keluaran berupa isi dari database sistem milik codelatte.org Filtering yang dilakukan ialah :

- Filtering Validasi URL filtering Numeric
  Validasi URL ini berguna untuk memeriksa apakah URL tersebut valid atau tidak, dengan tujuan untuk memeriksa inputan yang dimasukkan oleh user, apakah telah sesuai dengan standar penulisan URL atau belum
- *Filtering* angka pada parameter
  Filter is_numeric mengecek tipe data yang dimasukkan *user* apakah berupa angka atau bukan. Tujuannya agar penyerang tidak dapat menggunakan *string* kutip (‘) untuk mencoba masuk ke dalam *database* sistem.
- Html *Entities* dan *XSS*
  Filter *string* tersebut juga mampu melindungi serangan *cross site scripting* (xss), sebab script html atau javascript akan dirubah menjadi *string* biasa. Sehingga menjadikan kode tersebut penting dan seharusnya diterapkan pada *website* codelatte.org ini.

HASIL

Pengujian ulang kemudian kembali dilakukan setelah pengaplikasian PDO dan filtering menggunakan sqlmap dengan *command* yang berbeda dari pengujian sebelumnya karena pada pengujian kali ini menggunakan *command* --flush-session, peningkatan level pengujian ke 1 hingga level 5.

Kelima pengujian berbeda *level* tersebut serentak menghasilkan *error* 404 dalam usaha menerobos hingga *database* sistem, serta usaha *re-crawling* menggunakan acunetix tidak lagi menunjukkan tanda-tanda keberadaan celah keamanan *SQL Injection* dan *Directory listing*, dengan demikian pencegahan *SQL Injection* menggunakan penerapan PDO dan dilengkapi dengan *filtering* terhadap karakter berbahaya dapat dikatakan berhasil.

KESIMPULAN

Kesimpulan dari penelitian yang dilakukan ialah, *crawling* yang dilakukan menggunakan bantuan crawling engine milik tools acunetix tersebut menghasilkan alamat URL serta diagnosa kerentanan dari tingkatan *low* hingga *high,* sehingga memanfaatkannya dapat membantu menemukan celah keamanan yang terdapat di dalam aplikasi *website codelatte*. *Vulnerability* hasil *crawling* juga dapat dibuktikan keberadaannya sehingga *Penetration testing* terhadap situs codelatte.org dapat dilakukan dan pemerbaikan serta pencegahan dapat diterapkan.

Pemerbaikkan yang dilakukan untuk celah keamanan *directory listing* berupa pembuatan index.php, sementara perbaikan *SQL Injection* yang dilakukan yaitu dengan penerapan PDO menggantikan query biasa di dalam *source code* halaman yang memiliki celah keamanan. *Vulnerability* belum dieksploitasi oleh pihak tidak bertanggung jawab sebelumnya sehingga kerusakan yang dialami hanya sebatas celah keamanan terbuka.

Pencegahan tak luput dilakukan demi mengantisipasi eksploit terhadap celah keamanan yang mungkin terjadi dikemudian hari. *Directory Listing* dicegah melalui pemasangan *code* "Option All -Indexes" pada Htaccess sehingga akan menampilkan halaman lain pada *interface user*. *SQL Injection* dicegah dengan cara menambahkan validasi URL, *filtering* pada angka, dan karakter campuran seperti *string, metakarakter* angka. Dengan demikian *vulnerability* yang terkandung pada *website* codelatte.org mampu diatasi sehingga menghapus resiko bocornya privasi dari *website* tersebut

DAFTAR PUSTAKA

[1] M. D. Prayoga, "Pengertian Dan Komponen Sql," *Osf.Io*, pp. 1–7, 2018, doi: 10.31219/osf.io/kj43y.

[2] Bangkit Wiguna, W. Adi Prabowo, and R. Ananda, "Implementasi Web Application Firewall Dalam Mencegah Serangan SQL Injection Pada Website," *Digital Zone: Jurnal Teknologi Informasi dan Komunikasi*, vol. 11, no. 2, pp. 245–256, 2020, doi: 10.31849/digitalzone.v11i2.4867.

[3] M. A. Z. Risky and Y. Yuhandri, "Optimalisasi dalam Penetrasi Testing Keamanan Website Menggunakan Teknik SQL Injection dan XSS," *Jurnal Sistim Informasi dan Teknologi*, vol. 3, pp. 215–220, 2021, doi: 10.37034/jsisfotek.v3i4.68.

[4] A. P. Adi, *Wordpress untuk Segala Kebutuhan*. Jakarta: Elex Media Komputindo, 2018.

[5] R. Azis and S. Yazid, "Pengujian Kerentanan Website WordPress Dengan Menggunakan Penetration Testing," vol. 3, no. 3, pp. 93–105, 2021.

[6] E. I. Alwi, H. Herdianti, and F. Umar, "Analisis Keamanan Website Menggunakan Teknik Footprinting dan Vulnerability Scanning," *INFORMAL: Informatics Journal*, vol. 5, no. 2, p. 43, 2020, doi: 10.19184/isj.v5i2.18941.

[7] A. Ramadhani, "Keamanan Informasi," *Nusantara - Journal of Information and Library Studies*, vol. 1, no. 1, p. 39, 2018, doi: 10.30999/n-jils.v1i1.249.