PENGUJIAN KEAMANAN SISTEM INFORMASI AKADEMIK MENGGUNAKAN METODE PENETRATION TESTING

(Studi Kasus: Institut Pertanian Stiper Yogyakarta)

PROYEK TUGAS AKHIR

Disusun oleh

Rum Haidar Fauzan

5140411327

Kepada

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN ELEKTRO

UNIVERSITAS TEKNOLOGI YOGYAKARTA

2019

NASKAH PUBLIKASI

PENGUJIAN KEAMANAN SISTEM INFORMASI AKADEMIK MENGGUNAKAN METODE PENETRATION TESTING

(Studi Kasus: Institut Pertanian Stiper Yogyakarta)

Disusun oleh:

Rum Haidar Fauzan

5140411327

Pembimbing

Rianto, S.Kom., M.Eng.

Tanggal 22/8 19

Pengujian Keamanan Sistem Informasi Akademik Menggunakan Metode Penetration Testing (Studi Kasus: Institut Pertanian Stiper Yogyakarta)

Rum Haidar Fauzan
Program Studi Teknik Informatika, Fakultas Teknologi Informasi dan Elektro
Universitas Teknologi Yogykarta
Jl. Ringroad Utara Jombor Sleman Yogyakarta
E-mail : rum.haidar@hotmail.com

ABSTRAK

SIA (Sistem Informasi Akademik) merupakan aplikasi yang dirancang untuk mengelola semua informasi akademik. Obyek penelitian pada penelitian ini adalah SIA yang digunakan pada Institut Pertanian Stiper Yogyakarta. Kelebihan SIA yaitu dapat memperbarui data yang tersedia secara realtime dan meminimalkan terjadinya duplikasi data yang lazim ditemui pada sistem konvensional. Hal lain yang harus diperhatikan adalah keamanan sistem. Jika terdapat celah pada keamanan SIA akan berakibat buruk bagi data yang bersifat krusial seperti data keuangan dan data KRS menjadi tidak sinkron. Oleh karena itu, pada penelitian ini dilakukan pengujian keamanan SIA dengan menggunakan metode Penetration Testing untuk mengetahui celah-celah pada sistem. Tahapan pada penelitian ini adalah Planning, Information Gathering, Vulnerability Assesment, Exploiting dan Reporting. Pengujian membutuhkan tools pendukung yaitu Acunetix v10.5, Shell AlfaTeam v3 dan Sublime Text 3. Acunetix v10.5 berfungsi sebagai pemindai bagian sistem yang rentan, Shell AlfaTeam v3 digunakan untuk mengeksploitasi sistem serta Sublime Text 3 digunakan untuk membuat atau mengubah script. Dari penelitian ini, diperoleh hasil bahwa terdapat kelemahan pada SIA Institut Pertanian Stiper Yogyakarta yang diklasifikasikan menjadi tiga kategori yaitu high risk berupa serangan SQL Injection dan Cross Site Scripting, medium risk berupa serangan shell backdoor dan low risk berupa login-guessing attack..

Kata kunci : *Penetration Testing*, *Vulnerability*, Keamanan Sistem

1. PENDAHULUAN

Sistem Informasi Akademik memiliki peran yang sangat penting demi berlangsungnya proses lalu lintas data administrasi yang saling terintegrasi pada suatu pendidikan tinggi. Keamanan sistem adalah salah satu hal yang perlu diperhatikan, karena jika terdapat celah sekecil apapun akan mngakibatkan ancaman bagi sistem seperti manipulasi input, perubahan program, perubahan file secara langsung, pencurian data, sabotase bahkan yang *high risk* yaitu penyalahgunaan atau pencurian sumber daya informasi.

Institut Pertanian Stiper Yogyakarta memiliki Sistem Informasi Akademik yang tujuannya tidak lain adalah sebagai pengelola seluruh informasi akademik seperti dosen, mahasiswa, keuangan dan rencana studi. Penulis yang juga sebagai mahasiswa aktif Universitas Teknologi Yogyakarta merasakan manfaat dari SIA tersebut. Namun hal negatif yang sering terjadi adalah *server* tidak kuat untuk menampung banyaknya *client* SIA pada saat diakses secara bersamaan atau dengan kata lain *server* mengalami *down.* Hal ini tentu akan menghambat proses pengolahan data akademik seperti yang disebutkan di atas. Akan lebih fatal jika ada seseorang yang tidak bertanggungjawab mencoba untuk melakukan serangan *website* jenis DDOS (*Distributed Denial Of Service*) secara terus menerus maka SIA tidak dapat bekerja sebagaimana mestinya, belum lagi serangan-serangan *website* jenis lain.

Seiring perkembangan internet yang semakin cepat dalam penggunaan internet yang semakin banyak, informasi dan data yang sangat penting sangat perlu dijaga. Di sisi lain terdapat banyak resiko terhadap keamanan dalam sistem jaringan terutama *web server*, sementara *asset* yang ada dalam informasi tersebut perlu dilindungi karena banyak cara yang dilakukan oleh seorang *hacker* untuk mendpatkan informasi atau data yang penting karena semakin terbuka dalam pengetahuan *hacking* dan *cracking*, sehingga banyak beberapa pihak yang tidak bertanggungjawab mencoba untuk mencuri atau mengambil informasi. Didukung banyaknya *tools* yang *free*, sehingga mempermudah para *hacker* dan *attacker* mencoba untuk melakukan aksi penyusupan maupun serangan.

Dari uraian tersebut disimpulkan bahwa serangan dapat dilakukan dimana pun dan kapan pun tanpa diketahui oleh pemilik sistem. Oleh karena itu, penulis berusaha agar penelitian yang akan dilakukan dapat memberikan solusi jika terdapat kelemahan pada SIA Institut Pertanian Stiper Yogyakarta.

2. LANDASAN TEORI

2.1 Hacking

[13] Peretasan (*hacking*) adalah suatu perbuatan penyambungan dengan cara menambah terminal komputer baru pada sistem jaringan komputer tanpa izin/secara melawan hukum dari pemilik sah jaringan komputer tersebut dan pelakunya disebut dengan istilah *hacker*. Namun bagi komunitas *hacker*, istilah penjahat komputer disebut dengan *cracker*. Bedanya, *hacking* membuat sesuatu, sedangkan *cracking* menghancurkan/ merusaknya..

[2] *Hacking* dibagi menjadi 3 tipe yaitu *black hat hacking*, *white hat hacking*, dan *grey hat hacking*.

a. Black Hat Hacking

Salah satu jenis *hacking* yang menggunakan kemampuan mereka untuk bertindak yang seringkall dianggap melanggar hukum dan merusak. Tipe peretasan sebagai gambaran *hacking* seperti ini dan selalu jahat karena mendapatkan akses ke komputer orang lain tanpa diketahui sistem dan berusaha menguasai data rahasia milik korban.

Kelompok *hacking* ini juga bisa membuat virus, *worm*, *trojan*, dan pandai memasang *backdoor*. Apabila diibaratkan, tanpa mereka dunia tidak akan memerlukan antivirus dan para *hacker* pun mungkin tidak akan terlalu diperlukan untuk menganalisis keamanan. Nama lain dari *black hat hacking* sendiri adalah *Cracking*.

b. White Hat Hacking

White hat hacking adalah jenis *hacking* yang menggunakan kemam-puan untuk menghadapi Black Hat Hacker. Pelakunya juga sekelompok profesional yang bekerja pada perusahaan keamanan yang berperan sebagai *Security Analysis*, *Security Consultant*, dan sebagainya.

c. Grey Hat Hacking

Grey hat hacking merupakan jenis *hacking* yang bergerak di wilayah abu-abu, kadang-kadang bisa menjadi Black Hat Hacker tetapi suatu saat juga bisa menjadi Hhite Hat Hacker.

2.2 Penetration Testing

[8] *Penetration testing* adalah layanan yang dilakukan oleh profesional keamanan informasi di mana individu atau tim mencoba untuk menyerang atau menghancurkan mekanisme kontrol akses. Kelas keamanan ditawarkan untuk mengajarkan orang (yang membutuhkan pengujian) bagaimana serangan ini terjadi sehingga mereka dapat memahami penyebab dan tindakan penanggulangan yang sesuai.

[11] *Penetration testing* adalah semua tentang mencoba alat yang berbeda, teknik, dan taktik untuk menemukan apa yang bekerja di lingkungan tertentu. Setiap tes sama sekali berbeda dan perlu untuk dapat beradaptasi dengan lingkungan yang berubah. Tanpa latihan yang memadai, mencoba beberapa alat yang berbeda, dan mengeksploitasi sistem menggunakan muatan yang berbeda, tidak akan dapat beradaptasi menabrak sebuah *firewall*.

[12] *Penetration testing* adalah pengujian yang lebih ketat dan metodis dari jaringan, aplikasi, perangkat keras, dan lain-lain. Pengujian yang dimaksud dilakukan melalui tahapan *Scoping*, *Intel Gathering*, *Vulnerability Analysis*, *Exploitation*, *Post Exploitation*, dan *Reporting*.

[19] *Penetration testing* atau *pentesting* melibatkan simulasi serangan nyata untuk menilai risiko yang terkait dengan potensi pelanggaran keamanan. Pada *pentesting*, para penguji tidak hanya menemukan kerentanan yang dapat digunakan oleh penyerang tetapi juga mengeksploitasi kerentanan jika memungkinkan, untuk menilai apa yang mungkin diperoleh penyerang setelah eksploitasi berhasil.

[14] Sebuah *penetration test* pada dasarnya adalah legal, yaitu ditugaskan meretas untuk menunjukkan kerentanan jaringan perusahaan dan sistem. Sebagai organisasi menjadi semakin sadar bahwa keamanan dan biaya pelanggaran keamanan meningkat pesat, banyak organisasi besar memulai kontrak layanan keamanan. Salah satu kunci layanan keamanan ini adalah *penetration test*. Umumnya, organisasi melakukan penilaian kerentanan terlebih dahulu untuk menemukan potensi kerentanan dalam jaringan, sistem operasi, dan layanan mereka.

[5] *Penetration testing* mensimulasikan serangan terhadap jaringan atau sistem oleh orang luar yang berbahaya atau orang dalam. Tidak seperti *vulnerability assessment*, *penetration testing* dirancang untuk mencakup fase eksploitasi.

[7] *Penetration testing* (terkadang disebut *pentest*) melibatkan penggunaan berbagai teknik manual dan

otomatis untuk mensimulasikan serangan terhadap pengaturan keamanan informasi organisasi baik dari pihak luar yang jahat atau staf organisasi sendiri.

2.3 Penetration Testing Life Cycle

[15] Secara umum dalam melakukan *penetration tests* memerlukan pendekatan yang direncanakan dengan baik dan metodologis. Berikut ini adalah proses multilangkah. Berikut ini adalah beberapa tahapan pengujian penetrasi:

a. Information gathering
   Pengumpulan informasi adalah fase terpenting dalam siklus pengujian penetrasi. Fase ini juga disebut sebagai pengintaian. Ini melibatkan penggunaan berbagai teknik pasif dan aktif untuk mengumpulkan informasi sebanyak mungkin tentang sistem target. Pengumpulan informasi rinci meletakkan dasar yang kokoh untuk fase lebih lanjut dalam siklus hidup pengujian penetrasi.

b. Enumeration
   Setelah memiliki informasi dasar tentang target, fase enumerasi menggunakan berbagai alat dan teknik untuk menyelidiki target secara rinci. Ini melibatkan mencari tahu versi layanan yang tepat yang berjalan pada sistem target.

c. Vulnerability assessment
   Fase penilaian kerentanan melibatkan penggunaan berbagai alat dan metodologi untuk menegaskan keberadaan kerentanan yang dikenal dalam sistem target.

d. Gaining access
   Dari tahap sebelumnya, telah dimiliki daftar kemungkinan kerentanan untuk target. Selanjutnya adalah mencoba untuk memanfaatkan kerentanan tersebut untuk mendapatkan akses ke sistem target.

e. Escalating privileges
   Dimungkinkan mendapatkan akses ke sistem target dengan mengeksploitasi kerentanan tertentu, hanya saja akses mungkin dibatasi. Untuk menyusup lebih dalam, perlu menggunakan berbagai teknik dan meningkatkan hak istimewa untuk yang tingkat tertinggi seperti *administrator*, *root*, dan sebagainya.

2.4 SQL Injection

[10] Serangan SQL *injection* adalah salah satu rutinitas di mana validasi input untuk program dilewati, memungkinkan seorang penyerang untuk memberikan perintah SQL terhadap program target untuk mengeksekusi. Ini adalah bentuk eksekusi perintah yang dapat menyebabkan masalah keamanan potensial.

[6] SQL *injection* adalah salah satu jenis yang paling populer dari serangan *online*, kadang disingkat sebagai SQLi. Serangan ini melibatkan penyisipan kode *database* menggunakan bahasa *query* terstruktur SQL (*Structured Query Language*), di mana penyerang dapat mengambil data dari database atau menimpa data yang ada.

2.5 XSS (Cross Site Scripting)

[16] XSS (*Cross Site Scripting*) adalah serangan mengeksploitasi kerentanan di halaman *web* yang dihasilkan secara dinamis, dan ini terjadi ketika data input yang tidak divalidasi disertakan dalam konten dinamis yang dikirim ke browser pengguna untuk *rendering*.

[17] Jenis serangan XSS (*Cross Site Scripting*) memungkinkan seorang penyerang untuk menyuntikkan JavaScript ke halaman. JavaScript adalah bahasa pemrograman, dengan menggunakan serangan ini, penyerang dapat mengeksekusi kode yang ditulis dalam JavaScript ke halaman tertentu, seperti situs *web*.

Cross Site Scripting terdiri dari tiga jenis, yaitu sebagai berikut:

a. Persistent/stored XSS
   Stored XSS akan disimpan dalam *database*. Kode yang disuntikkan akan disimpan dalam *database* atau halaman sehingga setiap kali orang melihat halaman tersebut, kode akan dieksekusi.

b. Nonpersistent/reflected XSS
   Dengan *reflected* XSS, kode hanya akan dieksekusi saat target pengguna menjalankan URL tertentu yang dibuat atau ditulis oleh penyerang. Sehingga akan memanipulasi semacam URL dan mengirimkannya ke target, dan ketika target menjalankan URL itu, kode akan dieksekusi.

c. DOM-based
   DOM-*based* XSS adalah hasil dari kode JavaScript yang ditulis pada klien, sehingga kode akan benar-benar ditafsirkan dan dijalankan pada sisi klien tanpa memiliki komunikasi dengan *server web*. Ini bisa sangat berbahaya (bagi penyerang) karena terkadang *server web* menerapkan tindakan keamanan dan filtrasi untuk memeriksa XSS, namun dengan XSS berbasis DOM, kode tidak pernah dikirim ke *server web*.

2.6 DDOS (Distributed Denial Of Service)

[4] Serangan DOS (*Denial of Service*) adalah salah satu serangan *hacker* yang paling umum. Dimuali dengan melakukan banyak *request* yang tidak valid ke *host* jaringan, di mana *host* menggunakan semua sumber daya menanggapi permintaan yang tidak valid dan mengabaikan permintaan yang sah.

DOS (*Denial of Service*) adalah jenis serangan terhadap sebuah komputer atau server di dalam jaringan *intemet* dengan cara menghabiskan sumber (*resource*) yang dimiliki oleh komputer tersebut sampai komputer tersebut tidak dapat menjalankan fungsinya dengan benar, sehingga secara tidak langsung mencegah pengguna lain untuk memperoleh akses layanan dari komputer yang diserang tersebut.

2.7 Shell Backdoor

[1] Proses *Backdoor Website* adalah membuat pintu masuk untuk seseorang tanpa harus melalu *Login Page Admin* dan autentifikasi lainnya, *Backdoor Website* sering di lakukan oleh para *Cracker* untuk membuat pintu *Website* tersendiri.

[9] *Shell Backdoor* adalah tipe serangan yang baru dalam aktifitas *deface*, merupakan sebuah kode-kode yang disusun menjadi *script* rahasia digunakan untuk mengendalikan sebuah *website*/*server*. Serangan ini salah satu upaya untuk melakukan *deface* pada sebuah *website*.

3. METODOLOGI PENELITIAN

3.1 Obyek Penelitian

Penelitian ini dilakukan pada SIA Institut Pertanian Stiper Yogyakarta dengan mengumpulkan sejumlah informasi penting dari sistem maka dapat membantu dalam pengujian kemanaan sistem.

3.2 Metode Penelitian

Untuk membangun suatu sistem diperlukan satu set langkah yang disebut Dengan metodologi pengembangan sistem. Adapun langkah-langkah dalam pengembangan sistem antara lain:

3.2.1 Pengumpulan Data

Tahap ini membahas mengenai pengumpulan data dan materi. Pengumpulan data dilakukan dengan mempelajari buku-buku karya ilmiah dan situs *web*, serta dokumentasi yang berkaitan dengan penelitian ini yang penulis pergunakan sebagai bahan acuan untuk mencari informasi dan teori-teori tentang perangkat lunak pendukung dalam pembuatan aplikasi sebagai referensi.

3.2.2 Analisis Sistem

Pengujian Keamanan Sistem Informasi Akademik Menggunakan Metode *Penetration Testing* pada Institut Pertanian Stiper Yogyakarta dilakukan dengan mengumpulkan informasi terkait SIA, menganalisa skema SIA, kemudian menentukan jenis serangan yang akan digunakan sesuai dengan *bugs* yang dihasilkan dari pemindaian celah keamanan sistem.

Langah-langkah yang dilakukan dalam pengujian keamanan sistem antara lain:

a. Perancangan Proses
   Perancangan proses merupakan penjabaran aktivitas maupun proses yang terjadi dalam keseluruhan pengujian sistem informasi akademik. Dalam prosesnya, perancangan sistem menggunakan *Fishbone Diagram* (Diagram Tulang Ikan).

b. Perancangan Alur Pengujian
   Perancangan alur pengujian adalah langkah-langkah yang akan diterapkan pada pengujian sistem secara terstruktur dan sistematis. Untuk perancangan alur pengujian sistem akan digunakan *flowchart* sistem.

c. Teknik Pengujian
   Teknik pengujian adalah jenis-jenis serangan yang dilakukan pada keamanan sistem. Jenis-kenis serangan ditentukan oleh hasil hasil pemindaian kelemahan keamanan sistem tersebut.

3.2.3 Implementasi

Pengujian ini dibantu menggunakan beberapa *tools* yaitu Acunetix versi 10 untuk memindai kelemahan sistem, *shell backdoor* AlfaTeam versi 3 untuk melakukan eksploitasi dan Adminer versi 4.7.2 sebagai SQL *manager*.

3.2.4 Pengujian

Pengujian pada sistem dilakukan dengan cara manual atau menggunakan *tools* pendukung. Proses ini melibatkan analisis aktif terhadap sistem untuk setiap kerentanan potensial yang diakibatkan oleh sistem yang lemah atau konfigurasi sistem yang tidak benar atau kelemahan operasional dalam proses teknis.

Masalah keamanan yang ditemukan akan disampaikan kepada pemilik sistem bersama dengan penilaian dampak dan mitigasi (solusi teknis) dari setiap kerentanan yang ditemukan. Terdapat tiga alternatif teknik pengujian yaitu sebagai berikut:

a. Passive Penetration Testing
Dalam hal ini yang dilakukan adalah kita melakukan pemetaan dan pengujian terhadap kontrol yang ada di dalam *web application*, *login* dan konfigurasinya, sehingga kita bisa memetakan target sistem.

b. Active Penetration Testing
Yaitu melakukan kegiatan aktif dalam pengujian terhadap keamanan sistem dengan melakukan manipulasi *input*, pengambilan akses, dan melakukan pengujian terhadap *vulnerabilities* yang sudah ada.

c. Aggressive Penetration Testing
Melakukan eksploitasi terhadap *vulnerability*, melakukan *reverse enginering* terhadap sistem, menanamkan *backdoor*, men-*download* kode, mencoba mengambil alih finansial dan informasi yang ada di *server*.

Setiap teknik pengujian keamanan sistem, dilakukan dua jenis pengujian yaitu pengujian yang dilakukan dengan cara yang benar, serta pengujian yang mencoba segala kemungkinan yang terjadi pada sistem.

3.3 Alat Penelitian
Alat yang digunakan dalam penelitian ini terdiri atas perangkat keras (*hardware*) dan perangkat lunak (*software*). Adapun alat-alat tersebut mempunyai detail sebagai berikut:

a. Perangkat Keras (*Hardware*)
Perangkat keras adalah perangkat yang digunakan sebagai alat bantu dalam pengujian keamanan sistem informasi akademik Institut Pertanian Stiper Yogyakarta adalah laptop dengan spesifikasi sebagai berikut:

Tabel 3.1 ***Perangkat Keras Penelitian***

No.	Nama Elemen	Nama Perangkat Keras
1.	Prosesor	Intel Core i5-8250U 1,6 GHz
2.	Memori	RAM 6 GB DDR 4
3.	Penyimpanan	*Hard-disk* 1 TB dan SSD 128 GB
4.	Kartu Grafis	NVIDIA GeForce MX130

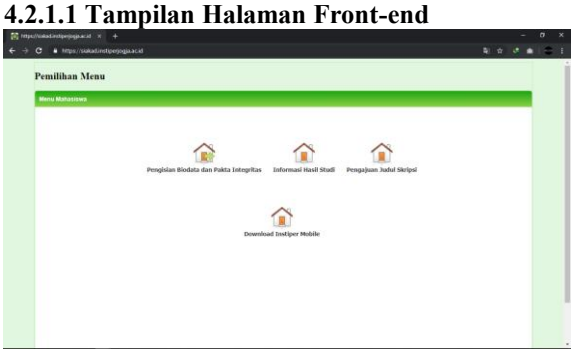
b. Perangkat Lunak (*Software*)
Perangkat lunak adalah sebuah program aplikasi yang digunakan oleh peneliti dalam membantu proses pengujian keamanan sistem informasi akademik Institut Pertanian Stiper Yogyakarta, yaitu sebagai berikut:

Tabel 3.2 ***Perangkat Lunak Penelitian***

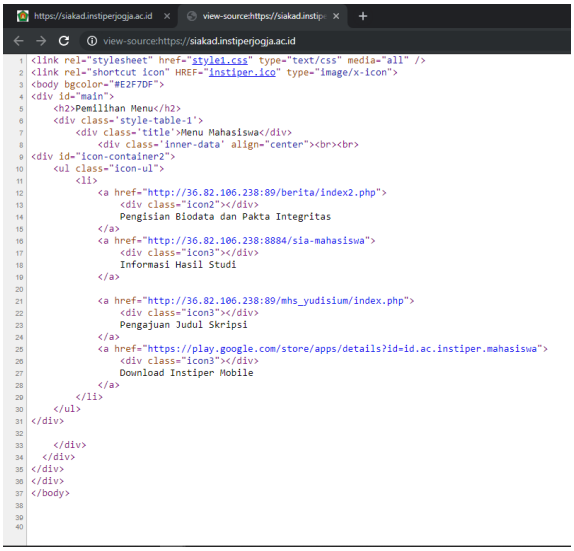
No.	Nama Elemen	Nama Perangkat Lunak
1.	Text Editor	Sublime Text versi 3.2.1
2.	Web Browser	Google Chrome
3.	Shell	AlfaTeam versi 3
4.	SQL Manager	Adminer very 4.7.2

4. HASIL DAN PEMBAHASAN

4.1 Pengumpulan Informasi
4.2.1 Informasi Front-end
Informasi *front-end* adalah informasi-informasi terkait elemen halaman, kode sumber halaman, kata kunci dan hal-hal yang berhubungan SIA Institut Pertanian Stiper Yogyakarta dari sisi depan *website*.

4.2.1.1 Tampilan Halaman Front-end


Gambar 4.1 Tampilan Halaman Front-end

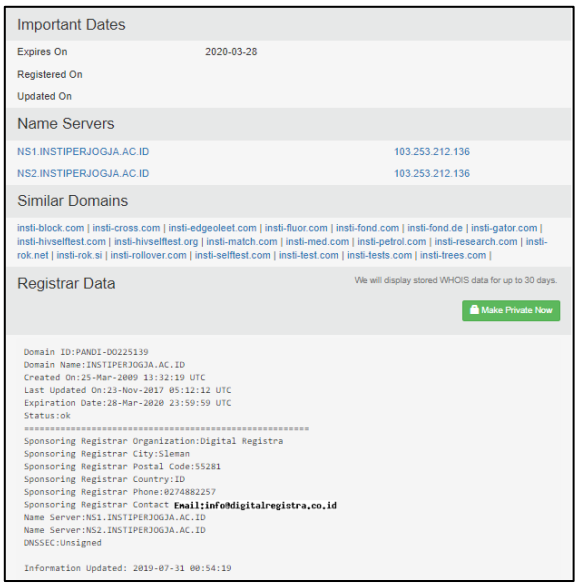
4.2.1.2 Kode Sumber Halaman Front-end


Gambar 4.2 Kode Sumber Halaman Front-end

4.2.2 Informasi Back-end
Informasi *back-end* adalah informasi-informasi terkait *whois*, DNS, diagnosa dan hal-hal yang berhubungan

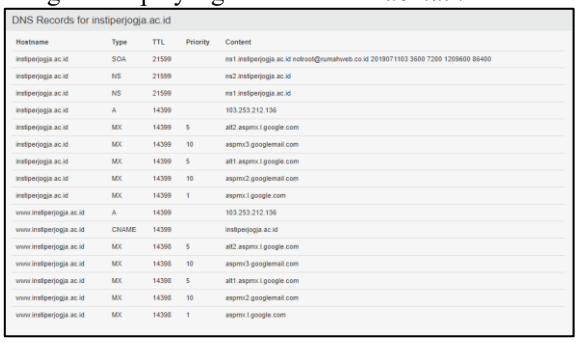
SIA Institut Pertanian Stiper Yogyakarta dari sisi belakang.

Dari hasil pengecekan melalui tautan https://who.is/ didapatkan informasi seperti yang terlihat pada gambar berikut:



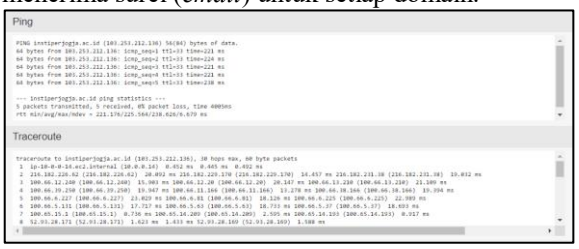
Gambar 4.3 Informasi Whois

Informasi Whois menjelaskan informasi mengenai *server* SIA yang digunakan oleh Institut Pertanian Stiper Yogyakarta. Terdapat informasi *Important Dates* yaitu informasi tanggal domain didaftarkan, tanggal domain diperbarui dan tanggal domain hangus; *Name Servers* yaitu *server* nama otoritatif yang mengasuh zona nama *domain* dari nama *domain* yang digunakan pada *server* Institut Pertanian Stiper Yogyakarta; *Registrar Data* menunjukkan informasi mengenai siapa yang mendaftarkan *domain* tersebut.



Gambar 4.4 Informasi DNS

Domain Name System adalah sebuah sistem yang menyimpan informasi tentang nama *host* ataupun nama *domain* dalam bentuk basis data tersebar (*distributed database*) di dalam *server* Institut Pertanian Stiper Yogyakarta. DNS menyediakan alamat IP untuk setiap nama *host* dan mendata setiap *server* transmisi surat (*mail exchange server*) yang menerima surel (*email*) untuk setiap domain.

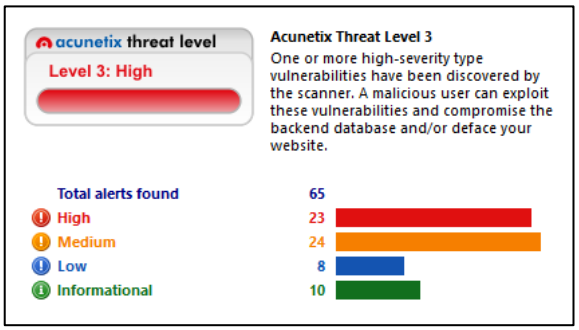


Gambar 4.5 Informasi Diagnosa

Gambar 4.5 menunjukkan bahwa *request* yang dilakukan pada *server* Institut Pertanian Stiper Yogyakarta baik *ping* dan *traceroute* berstatus normal dan berjalan dengan baik.

4.3 Pemindaian Kelemahan

Dari hasil pemindaian kelemahan menggunakan Acunetix versi 10, secara garis besar menghasilkan laporan bahwa terdapat kelemahan sistem yang terbagi atas jenis kelemahan sistem yaitu *high risk* (resiko tinggi), *medium risk* (resiko sedang) dan *low risk* (resiko rendah). Kelemahan dengan resiko tinggi diantaranya yaitu SQL *injection* dan *Cross Site Scripting*. Kelemahan dengan resiko sedang diantaranya yaitu *Denial Of Service, CSRF, parameter population* dan *clear text*. Sedangkan kelemahan dengan resiko rendah diantaranya yaitu *mod_negotiation filename bruteforcing, clickjacking, sensitive cookie* dan *password-guessing attack*.



Gambar 4.6 Thread Level

Salah satu komponen kunci dari hasil pemindaian adalah daftar semua kerentanan yang ditemukan dalam target selama pemindaian bergantung pada jenis pemindaian, bisa berupa Peringatan Web atau Peringatan Jaringan. Berdasarkan Gambar 4.6, peringatan tersebut dikategorikan menurut 4 tingkat resiko (dilansir dari https://www.acunetix.com/support/docs/wvs/analyzin

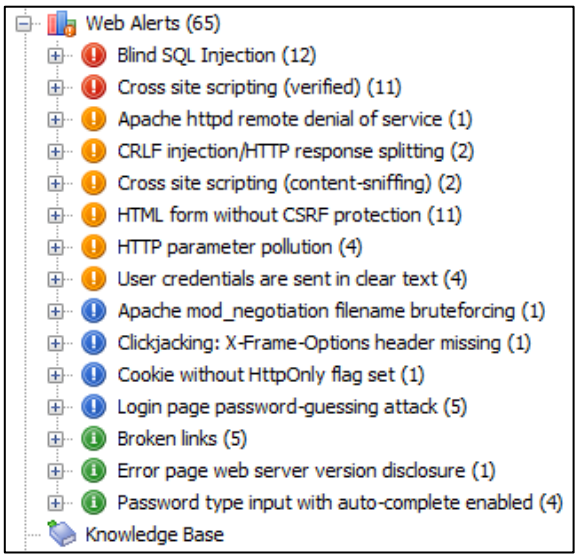
g-scan-results/) yang sudah menjadi ketentuan Acutnetix yaitu sebagai berikut:

a. High Risk Alert Level 3
Kerentanan dikategorikan sebagai yang paling berbahaya, yang menempatkan target pemindaian pada resiko maksimum untuk peretasan dan pencurian data.

b. Medium Risk Alert Level 2
Kerentanan yang disebabkan oleh kesalahan konfigurasi *server* dan cacat pengodean situs yang memfasilitasi gangguan dan intrusi *server*.

c. Low Risk Alert Level 1
Kerentanan yang berasal dari kurangnya *enkripsi* lalu lintas data atau pengungkapan jalur direktori.

d. Informational Alert
Ini adalah item yang telah ditemukan selama pemindaian dan yang dianggap menarik, sebagai contoh; kemungkinan pengungkapan alamat IP *internal* atau alamat *email*, kecocokan string pencarian yang ditemukan di Google Hacking Database, dan informasi tentang layanan yang telah ditemukan selama pemindaian.



Gambar 4.7 Web Alert

Informasi detail mengenai hasil pemindaian kelemahan sistem dijelaskan pada tabel berikut:

Tabel 4.1 Hasil Pemindaian Kelemahan

No.	Tingkat Resiko	Jenis Kelemahan	Laman/sektor yang rentan	Jumlah bagian yang rentan
1.	*High Risk Alert Level 3*	*Blind SQL Injection*	/berita/proseslogin.php /feeder/ambilkota.php /feeder/excel_transaksi.php /feeder/login.php /feeder/matkul.php /feeder/transaksi.php /spp/ /yudisium/login.php	2 1 2 2 1 1 1 2
		Cross site scripting (verified)	/cetak/index.php /feeder/lihat_nilai.php /feeder/lihat_nilai2.php /feeder/transaksi.php /spp/ /spp/index.php /spp/semester.php	1 1 1 4 1 1 2
2.	*Medium Risk Alert Level 2*	*Apache httpd remote denial of service*	*Web Server*	1
		CRLF injection/HTTP response splitting	/spp/exportsemester.php	1
		Cross site scripting (content-sniffing)	/spp/export.php /spp/exportsemester.php	1 1
		HTML form without CSRF protection	/berita /cetak /feeder /feeder/lihat_nilai.php /feeder/lihat_nilai2.php /feeder/transaksi.php /medis /spp /spp/semester.php /transaksi/login.php	1 1 1 1 1 2 1 1 1 1
		HTTP parameter pollution	/spp/ /spp/index.php /spp/semester.php	1 1 2
		User credentials are sent in clear text	/berita /feeder /medis /transaksi/login.php	1 1 1 1
3.	*Low Risk Alert Level 1*	*Apache mod_negotiation filename*	*Web Server*	1

		bruteforcing		
		Clickjacking: X-Frame-Options header missing	*Web Server*	1
		Cookie without HttpOnly flag set	/	1
		Login page password-guessing attack	/berita/proseslogin.php /feeder/login.php /medis/ /transaksi/login.php /yudisium/login.php	1 1 1 1 1

4.4 Eksploitasi

Eksploitasi adalah inti dari pengujian keamanan SIA Institut Pertanian Stiper Yogyakarta. Langkah eksploitasi mengambil sampel dari jenis kelemahan yang ditemukan dari pemindaian kelamahan sistem dan serangan yang akan dilakukan berfokus pada jenis kelemahan resiko tinggi dan kelemahan resiko sedang.

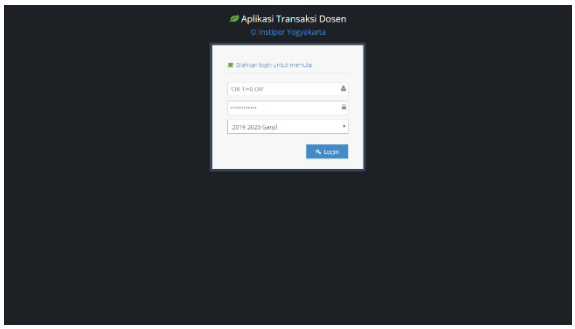
4.4.1 Implementasi SQL Injection

Logika dalam penerapan serangan SQL *injection* berhubungan dengan tipe data *Boolean* yaitu memiliki simbol kebenaran *TRUE* dan *FALSE*. Serangan SQL *injection* memanfaatkan *database query* dengan yaitu mem-*bypass* perintah *query* menjadi nilai yang *TRUE* (benar). Sebagai contoh kalimat dalam tipe data *integer* yaitu 1=1, 6/3=2, 1!=0 dan lain sebagainya. Sedangkan kalimat dalam bentuk tipe data *string* contohnya adalah 'hitam' = 'hitam', 'hitam' != 'putih' dan lain seabagainya Serangan ini akan berjalan hanya jika dalam sintaks kode yang terdapat di dalam sistem tidak memiliki vaidasi input karakter khusus.

Penulis mengambil sampel kelemahan sistem jenis SQL *injection* yang ditemukan pada halaman /transaksi/login.php seperti pada Gambar 4.8.

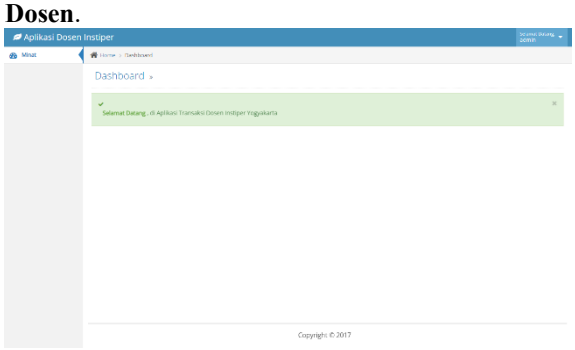
Gambar 4.8 Scan Thread Blind SQL Injection

Pada halaman *login* seperti Gambar 4.9 dicontohkan untuk menggunakan *value* dengan tipe data *integer* untuk *fileds username* dan *password* yang dikombinasikan dengan kalimat *or* (atau) dan *not* (tidak/bukan) sehingga menghasilkan notasi **'OR 1!=0 OR'**.



Gambar 4.9 Halaman Login Aplikasi Transaksi Dosen

Dari hasil *bypass* dengan serangan SQL *injection*, didapatkan hasil bahwa halaman /transaksi/login.php dapat ditembus. Dibuktikan pada Gambar 4.10. yang menampilkan halaman *dashboard* **Aplikasi Transaksi Dosen**.



Gambar 4.10 Halaman Dashboard Aplikasi Transaksi Dosen

[3] SQL *injection* adalah salah satu metode pertama yang harus diuji terhadap bentuk *login form*. Kerentanan terjadi karena kurangnya validasi input/penyaringan. Masukan dibuat bagian dari *query* SQL, yang memungkinkan untuk melakukan beberapa hal seperti pengambilan data dan membaca *file* sistem seperti /etc/passwd. Namun, di sini hanya fokus pada penggunaan SQL *injection* untuk memotong mekanisme otentikasi. Jika diasumsikan pada kode yang berpotensi rentan yang akan menghasilkan SQL *injection* yaitu sebagai berikut:

<?php $query="SELECT * FROM users WHERE username='".$_POST['username']. "' AND password='". $POST_['password']."'" response=mysql_query($query); ?>

Seperti yang terlihat dari kode di atas, sintaks *query* telah menerima dua masukan yaitu *username* dan *password*. Kemudian tanpa validasi apapun masukan *username* dan *password* dimasukkan sebagai *query* SQL dan kemudian dieksekusi. *Username* dan *password* kemudian akan dibandingkan dengan

database untuk melihat apakah kedua masukan tersebut cocok. Jika terdapat kecocokan, pengguna akan dikonfirmasi, jika tidak, kesalahan akan muncul. Berikut ini adalah bagaimana *query* tersebut dieksekusi:

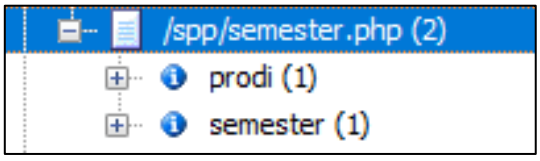
SELECT * FROM users WHERE username = 'administrator' AND password = 'mypass'

Query ini akan mengambil rincian *username* "*administrator*" dengan *password* "*mypass*" dari tabel *users*.

4.4.2 Implementasi Serangan XSS (Cross Site Scripting)

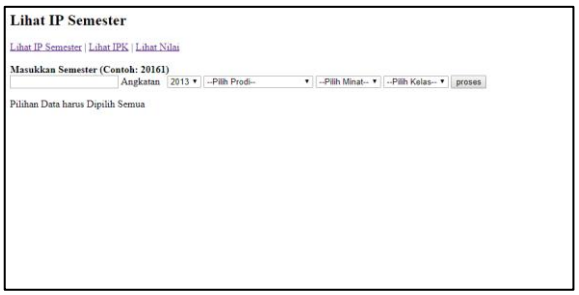
[3] Karena XSS adalah masalah validasi input, perlu menyelidiki semua input dan mencoba mencari tahu input apa pun yang tidak dibersihkan seperti parameter URL, formulir, *cookie*, dan *file uploads* sebelum respon dikembalikan ke pengguna. Tes dasar untuk menemukan jika sebuah situs *web* yang rentan terhadap kerentanan XSS adalah untuk menyuntikkan potongan kode berikut, yang merupakan variasi kecil dari kode *Locator* XSS (ditemukan pada "OWASP XSS *filter Cheat Sheet*") diantaranya adalah '"<>(); [] {}.

Setelah menyuntikkan muatan tersebut ke setiap input yang memungkinkan, akan terlihat sumber halaman yang diberikan kembali. Jika ditemukan kata "XSS" di sumbernya dan salah satu dari karakter ini tidak lolos, maka situs *web* tersebut mungkin rentan terhadap XSS. Implementasi serangan *Cross Site Scripting* dilakukan pada sampel dari hasil *scan tread* yang dalam hal ini adalah halaman /feeder/transaksi.php.



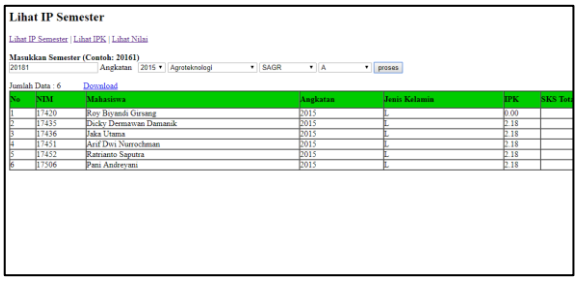
Gambar 4.11 Scan Thread Cross Site Scripting

Feeder pada SIA Institut Pertanian Stiper Yogyakarta berisi proses pengecekan IP Semester, IPK dan Nilai. Halaman awal **Lihat IP Semester** dapat dilihat pada Gambar 4.12.



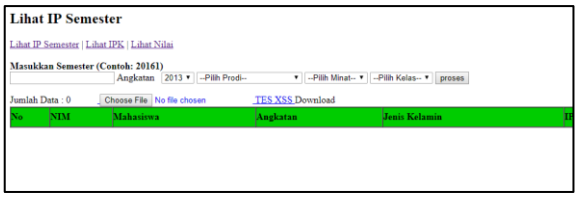
Gambar 4.12 Halaman Lihat IP Semester

Hasil dari pengisian *field* secara normal akan menampilkan *datatable* yang berisi informasi yang dimaksud. Seperti yang terlihat pada Gambar 4.13.



Gambar 4.13 Hasil Normal Proses Lihat IP Semester

Sekarang percobaan serangan *Cross Site Scripting* dengan menginputkan *value* **"> <input type="file"> TES XSS <a "**. Maka terdapat perubahan pada tombol **Download** menjadi sebuah teks, dan muncul *field* baru berupa *file upload* seperti yang terlihat pada Gambar 4.14.



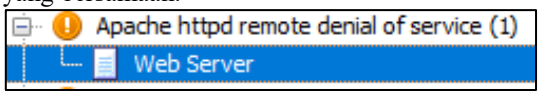
Gambar 4.14 Hasil Serangan Serangan Cross Site Scripting

Dengan demikian, maka serangan *Cross Site Scripting* berhasil dilakukan. Sehingga terdapat kemungkinan bahwa target serangan terhadap sistem bisa lebih dalam lagi.

4.4.3 Implementasi Serangan DDOS (Distributed Denial Of Service)

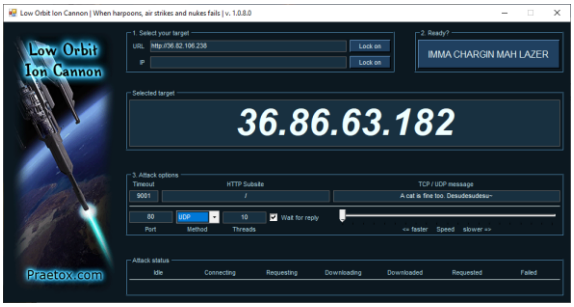
Serangan *Distributed Denial Of Service* dilakukan secara langsung menuju *server* SIA Institut Pertanian Stiper Yogyakarta. Dari hasil *scan thread*, terapat kemungkinan bahwa *server* akan mengalami masalah

jika paket data dikirim secara masal dan dalam waktu yang bersamaan.



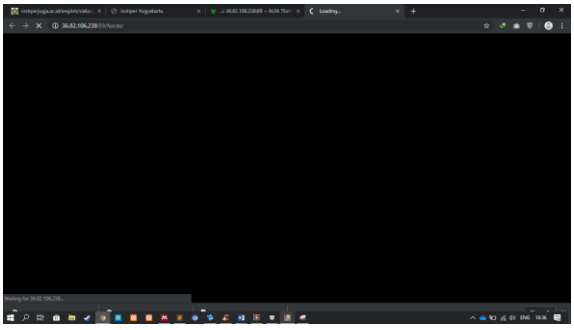
Gambar 4.15 Scan Thread Distributed Denial Of Service

Pengujian dilakukan menggunakan metode penyerangan UDP seperti yang terlihat pada Gambar 4.16.

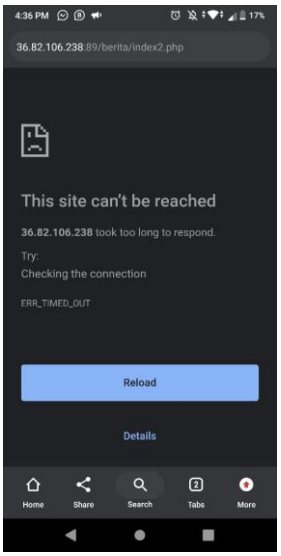


Gambar 4.16 Tampilan Konfigurasi Serangan DDOS

Saat *flooding* dijalankan, halaman situs SIA Institut Pertanian Stiper Yogyakarta mengalami *down.* Ini menunjukkan bahwa serangan *Distributed Denial Of Service* berhasil dilakukan. Seperti yang terlihat pada Gambar 4.17 dan Gambar 4.18.



Gambar 4.17 Tampilan Halaman SIA pada Perangkat Desktop



Gambar 4.18 Tampilan Halaman SIA pada Perangkat Mobile

[3] Proses serangan *Denial Of Service* bekerja dengan menghubungkan IP *router* korban ke IP yang tidak ada, sehingga menyangkal korban akses ke *internet*. Ketika korban mencoba untuk terhubung ke *internet*, ia akan mencapai tempat yang tidak ada (biasanya muncul pesan *error* "*This site can't be reached*"). Serangan ini dilakukan dengan mengirimkan balasan ARP palsu ke alamat *router's* MAC korban yang tidak ada. Sekali lagi, dalam lingkungan *penetration testing* nyata, *pentester* akan jarang melakukan jenis serangan ini dan akan lebih fokus pada peluncuran serangan *spoofing* ARP.

4.1.5 Implementasi Serangan Shell Backdoor

Aplikasi *web* biasanya menyediakan fitur untuk meng-*upload* gambar profil, avatar, CV, dan lain-lain. Namun, jika unggahan *file* tidak dibatasi dengan benar, penyerang dapat dengan mudah mengunggah *file* berbahaya sehingga mengorbankan keamanan aplikasi *web*. Kerentanan *file upload* mungkin tidak terbatas pada *file upload* berbahaya saja, namun juga dapat membuat penyerang menyebabkan *denial of service attacks*, *cross site scripting*, dan bahkan kerentanan traversal direktori [3].

Implementasi serangan *shell backdoor* merupakan serangan yang dikembangkan penulis dari jenis kelemahan yang ditemukan. Hal ini dilakukan agar menemukan kelemahan lain yang tidak teridentifikasi oleh *tools* pemindai kelemahan sistem yang dalam hal ini adalah Acunetix. Serangan yang akan dilakukan adalah menyisipkan *shell backdoor* AlfaTeam.

Shell backdoor AlfaTeam memiliki fungsi sebagai alat untuk membuat, memodifikasi, dan menghapus *file*

atau direktori yang terdapat di dalam sistem dengan fitur-fitur untuk mengeksekusi perintah pada *terminal server*.

Untuk menyisipkan *shell backdoor* diperlukan kelemahan seperti *file upload* yang tidak memiliki validasi format *file*, FTP yang dibiarkan terbuka atau memanfaatkan fitur *file manager* yang tidak memiliki hak akses.

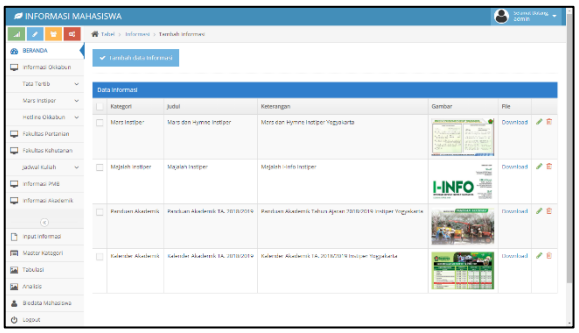
Serangan *shell backdoor* diterapkan pada halaman /berita.index.php. Halaman *login* subsistem berita SIA Instiper Pertanian Stiper Yogyakarta.

Gambar 4.19 Halaman Login Informasi Mahasiswa

Pada *fields username* dan *password* diisi menggunaan *value* SQL *injection* seperti yang telah dijelaskan pada sub bab 5.4.1 tentang Implementasi SQL *injection*. Dalam hal ini dicontohkan *value* dari *fileds* tersebut adalah **'OR "tes" = "tes" OR'** seperti yang terlihat pada Gambar 4.19.

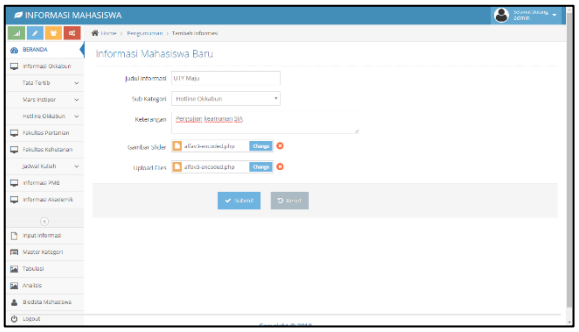
Gambar 4.20 Halaman Dashboard Informasi Mahasiswa

Selanjutnya adalah mencari sebuah *read data* yang di dalamnya terdapat menampilkan *file* berupa dokumen, gambar, audio atau video. Pada kasus ini ditemukan **Data Informasi** yang menampilkan *file* gambar seperti yang terlihat pada Gambar 4.21.



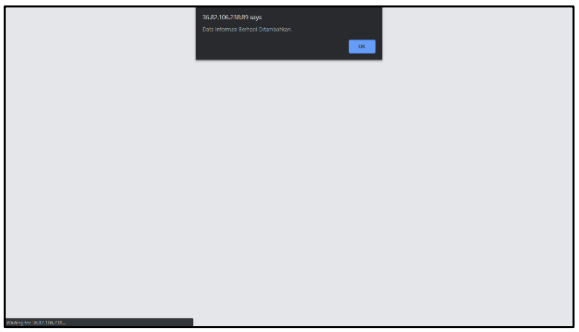
Gambar 4.21 Halaman Data Informasi Mahasiswa

Beralih ke proses **Tambah Data Informasi**, terdapat *fields* dengan tipe inputan *file* yaitu **Gambar Slider** dan **Upload Files** seperti yang terlihat pada Gambar 4.22.



Gambar 4.22 Halaman Form Input Informasi Mahasiswa

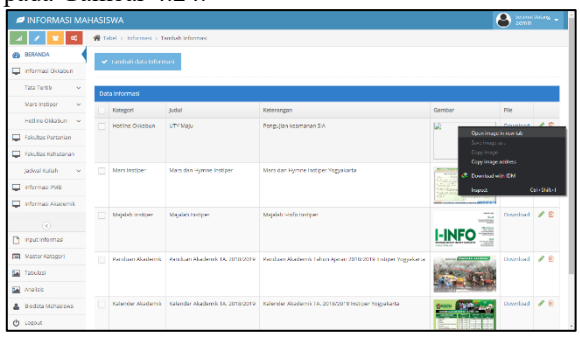
Gambar 4.23 menjelaskan proses **Tambah Data Informasi** berhasil dengan dibuktikan oleh munculnya *alert* "Data Informasi Berhasil Ditambahkan". Ini menunjukkan bahwa tidak terdapat validasi untuk proses **Tambah Data Informasi**. Data yang telah diinputkan dapat dilihat kemabali pada **Data Informasi**.



Gambar 4.23 Notifikasi Sukses Input Informasi Mahasiswa

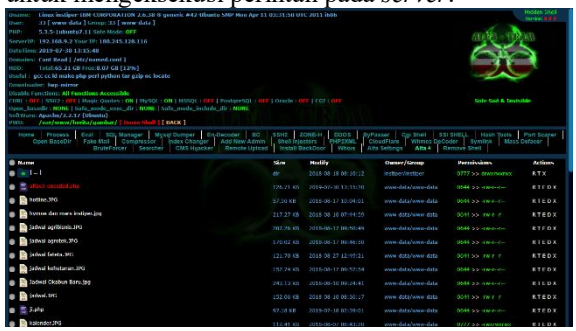
Untuk membuka *shell backdoor* tersebut yaitu dengan membuka gambar yang tampak rusak, karena *file* yang

telah diunggah bukan dalam format gambar seperti pada Gambar 4.24.



Gambar 4.24 Halaman Data Informasi Mahasiswa Terbaru

Proses implementasi serangan *shell backdoor* berhasil dilakukan. Letak dari *file* yang diunggah tadi berada di direktori /berita/gambar. Gambar 4.25 adalah tampilan dari *shell backdoor* AlfaTeam versi 3 yang yang memuat informasi *server*, modul-modul yang digunakan pada *server* dan fitur-fitur yang disediakan untuk mengeksekusi perintah pada *server*.



Gambar 4. 25 Halaman Shell Backdoor

4.5 Pelaporan

Dari hasil pengujian dismpulkan bahwa masing-masing serangan berdasarkan kelemahan yang dipindai menggunakan *tools* Acunetix versi 10 mendapatkan hasil *high risk* (beresiko tinggi), berikut pengembangan serangan oleh penulis. Hal ini dibuktikan oleh empat teknik serangan yang dilakukan, tiga diantaranya adalah SQL *injection, Cross Site Scipting* dan *Distributed Denial Of Service* yang dihasilkan oleh *scan thread* Acunetix dan satu sisanya adalah teknik pengembangan serangan yaitu menyisipkan *shell backdoor*.

Oleh karena itu, dapat disimpulkan bahwa:

a. SIA Institut Pertanian Stiper Yogyakarta **berhasil** ditembus menggunakan implementasi serangan SQL *Injection* pada sampel halaman situs http://36.82.106.238:89/transaksi/login.php.

b. SIA Institut Pertanian Stiper Yogyakarta **berhasil** ditembus menggunakan implementasi serangan *Cross Site Scripting* pada halaman situs http://36.82.106.238:89/feeder/transaksi.php.

c. SIA Institut Pertanian Stiper Yogyakarta **berhasil** ditembus menggunakan implementasi serangan *Distributed Denial Of Service* pada sampel halaman situs http://36.82.106.238:89.

d. SIA Institut Pertanian Stiper Yogyakarta **berhasil** ditembus menggunakan implementasi serangan *Shell Backdoor* pada sampel halaman situs http://36.82.106.238:89/berita/home.php?p=tambah_berita.

Tabel 4.2 Hasil Laporan Pengujian

No.	Halaman Situs Target	Jenis Serangan	Status Serangan
1.	http://36.82.106.238:89/transaksi/login.php	SQL *Injection*	Berhasil
2.	http://36.82.106.238:89/feeder/transaksi.php	*Cross Site Scripting*	Berhasil
3.	http://36.82.106.238:89	*Distributed Denial Of Service*	Berhasil
4.	http://36.82.106.238:89/berita/home.php?p=tambah_berita	*Shell Backdoor*	Berhasil

4.5.1 Solusi Perbaikan Serangan SQL *Injection*

Untuk mengatasi serangan SQL injection, dapat dilakukan dengan membuat *script* sederhana anti SQL *injection* yaitu memanfaatkan fungsi yang disediakan oleh PHP yaitu **mysql_real_escape** atau **mysql_real_escape_string**, sebagai contoh:
$id = mysql_real_escape_string($_GET['id']);
Selain cara tersebut hal yang dapat dilakukan adalah sebagai berikut:

a. Batasi panjang *input box* (jika memungkinkan), dengan cara membatasi kode program, sehingga *cracker* pemula akan melihat *input box* tidak dapat di-*inject* dengan perintah yang panjang.

b. Filter input yang dimasukkan oleh *user*, terutama penggunaan tanda kutip tunggal (*Input Validation*).

c. Matikan atau sembunyikan pesan-pesan *error* yang keluar.

d. Matikan fasilitas-fasilitas standar seperti *Stored Procedures*, *Extended Stored Procedures* jika memungkinkan.

e. Menggunakan *low privilege user*

4.5.2 Solusi Perbaikan Serangan Cross Site Scripting

Terdapat beberapa aspek target yang banyak digunakan untuk eksploitasi Cross Site Scripting diantaranya yaitu URL, HTTP referrer objects, Parameter GET, Parameter POST, Window.location, Document.referrer, document.URL, document.URLUnencoded, cookie data, headers data dan database data. Maka untuk menangani serangan tersebut adalah melakukan *encoding* terhadap karakter khusus seperti **<, >,'** dan **"**. Adapun solusi perbaikan dari serangan ini adalah dengan menggunakan fitur dari PHP yaitu **htmlspecialchars()** atau **str_replace().**

4.5.3 Solusi Perbaikan Serangan Distributed Denial Of Service

Penanganan serangan Distributed *Denial Of Service* dapat dibedakan berdasarkan jenis DDOS. Yaitu sebagai berikut:

a. *Syn Flooding*, gunakan *firewal* untuk tidak meneruskan paket data yang tidak diketahui dengan jelas asalnya.

b. *Remote Controled Attack*, *block* alamat IP dan portnya.

c. UDP *Flooding*, Menolak paket trafik yang datang dari luar jaringan dan mematikan semua layanan UDP.

d. *Smurf Attack*, *disable broadcast address* pada *router* atau *filtering* permintaan ICMP *echo request* pada *firewall* atau juga membatasi trafik ICMP.

e. *Upgrade server* ke versi yang lebih *up-to-date.*

4.5.4 Solusi Perbaikan Serangan Shell Backdoor

Untuk menangani serangan *shell backdoor*, *webmaster* harus selalu memantau *server* yang digunakan agar tidak terdapat *script-script* yang dapat membahayakan seperti halnya *shell backdor*. Karena jika *server* telah diinangi oleh *shell backdoor*, berkemungkinan seorang *cracker* memiliki hak akses penuh terhadap sistem. Berikut ini adalah cara mengatasi *shell backdoor*:

a. Mengatasi *backdoor* di *web hosting*
Pada *web hosting* ada beberapa hal yang bisa dilakukan untuk membersihkan *backdoor*. Berikut beberapa pengecekan yang bisa dilakukan untuk penanganan backdoor. Tema-tema yang tidak digunakan lebih baik dihapus dan gunakan tema yang original serta terpercaya. Lakukan pengecekan *file* **.htaccess**, pastikan tidak ada *script redirect*. Periksa *file* **wp-config.php** sehingga sama dengan *file* **wp-config-sample.php** pada konfigurasinya.

b. Mencari File yang Mengandung Backdoor
Langkah ini terkadang berguna untuk penanganan pasca terjadi serangan pada *server web hosting*. Tujuannya adalah untuk mencari jika masih ada *script* yang tertanam di suatu *file* di salah satu *folder*. Masalahnya adalah di sana banyak sekali *folder* dan *file* yang tersimpan. Mulai dari ratusan *megabytes* sampai dengan beberapa *gigabytes*.

Salah satu cara yang bisa di gunakan untuk mencari potongan kode yang ada di salah satu *file server* adalah dengan menggunakan teks perintah. Pada Linux terdapat perintah *grep* yang dapat digunakan untuk mencari potongan *script* yang ada di dalam *file*. *Grep* sudah ada di setiap Linux sehingga tidak perlu melakukan instalasi lagi. Sedangkan untuk *file website* yang ingin diperiksa harus di unduh terlebih dahulu.

c. Mencari Baris Perintah Passtrhu Menggunakan Linux
Baris perintah berikut digunakan untuk mencari seluruh folder dan file yang terdapat pada public_html yang mengandung *script passtrhu*.
$ grep -Rn "passthru *(" public_html/

Jika ingin menyimpan hasil dari pencarian ke dalam sebuah *file*, makan dapat menggunakan perintah berikut.

$ grep -Rn "passthru *(" public_html/ >> hasil.txt

d. Mencari Baris Perintah Passtrhu Menggunakan Windows

Jika menggunakan sistem operasi Windows juga terdapat perintah yang dapat digunakan untuk melakukan pengecekan baris kode. Jika di Linux menggunakan *Grep* maka di Windows menggunakan Findstr.

$ findstr /r /s /n /c:"passthru *(" *.*

Keempat cara tersebut adalah salah satu contoh dari beberapa hal yang bisa dilakukan untuk menangani masalah pada *backdoor*.

5. PENUTUP

5.1. Kesimpulan

Berdasarkan hasil penelitian yang dilakukan penulis dengan judul Pengujian Keamanan Sistem Informasi Akademik Menggunakan Metode *Penetration Testing*

(Studi kasus: Institut Pertanian Stiper Yogyakarta), maka didapat kesimpulan sebagai berikut:

a. Terdapat kelemahan pada SIA Institut Pertanian Stiper Yogyakarta.

b. Jenis kelemahan pada SIA Institut Pertanian Stiper Yogyakarta dikategorikan menjadi tiga kategori yaitu *high risk*, *medium risk* dan *low risk*.

c. Ditemukan jenis kelemahan yang tidak teridentifikasi oleh *tools* Acunetix yaitu *file upload bypass*.

d. Perbaikan yang dapat dilakukan secara langsung adalah jenis kelemahan SQL *injection* yaitu dengan menambahkan *function* terhadap karakter-karakter khusus, sedangkan kelamahan jenis *Denial of Service* hanya menghasilkan solusi yaitu konfigurasi server dan meng-*upgrade server* ke *space* yang lebih besar beserta teknologi *server* yang terbaru.

5.2. Saran

Pengujian keamanan sistem informasi akademik ini masih belum sempurna. Banyak hal yang dapat dilakukan untuk mengembangkan teknik pengujian sistem dan perbaikan sistem agar menjadi lebih baik lagi, antara lain:

a. Pengembangan terhadap *tools* yang digunakan agar dapat menutup celah kelemahan sistem secara otomatis.

b. Perlu dibangun sebuah *tools* yang dapat dimanfaatkan minimal membuat sebuah laporan *error log* dari kelemahan yang terdapat pada sistem.

DAFTAR PUSTAKA

[1] Abdurrohim, I. (2019), *Penetration Testing Sistem Keamanan Aplikasi Web Berbasis E-Commerce Pada Perusahaan Hptasik*, Universitas Nasional Pasim.

[2] Agency, B. (2014), *Hacking Attack!*, Elex Media Komputindo.

[3] Baloch, R. (2015), *Ethical Hacking And Penetration Testing Guide*, New York: CRC Pres.

[4] Beaver, K. (2016), *Hacking For Dummies*, ed. 5 New Jersey: John Wiley & Sons.

[5] Beggs, R.W. (2014), *Mastering Kali Linux for Advanced Penetration Testing*, Birmingham: Packt Publishing.

[6] Binnie, C. (2016), *Linux® Server Security: Hack and Defend*, Indianapolis: John Wiley & Sons.

[7] Creasey, J. dan Glover, I. (2017), *A guide for running an effective Penetration Testing programme*, CREST.

[8] Harris, S., Harper, A., Eagle, C. dan Ness, J. (2018), *Gray Hat Hacking*, ed. 5 McGraw-Hill Education.

[9] Hasibuan, M.S. dan Gultom, L.M. (2019), *Analisis Serangan Deface Menggunakan Backdoor Shell Pada Website*, *Techno.Com*, *17*(4), 415–423.

[10] Hertzog, R., O'Gorman, J. dan Aharoni, M. (2017), *Kali Linux Revealed*, Cornelius NC: Offsec Press.

[11] Kim, P. (2015), *The Hacker Playbook 2*, ed. 2 South Carolina: Secure Planet LLC.

[12] Kim, P. (2018), *The Hacker Playbook 3*, ed. 3 South Carolina: Secure Planet LLC.

[13] Napitupulu, M.R.T. (2018), *Pertanggungjawaban Pidana Terhadap Pelaku Tindak Pidana Peretasan (Hacking) Berdasarkan Undang-Undang Nomor 19 Tahun 2016Tentang Perubahan Atas Undang-Undang Nomor 11 Tahun 2008 Tentang Informasi dan Transaksi Elektronik (Putusan Nomor 253/PID.B/2013/PN.J*, Universitas Sumatera Utara.

[14] OccupyTheWeb (2018), *Linux Basics For Hackers*, ed. 1 San Francisco: No Starch Press.

[15] Rahalkar, S. (2019), *Quick Start Guide to Penetration Testing*, Maharashtra: Apress.

[16] Raj, M. (2018), *Python Penetration Testing Essentials*, ed. 2 Packt Publishing.

[17] Sabih, Z. (2018), *Learn Ethical Hacking from Scratch*, Birmingham: Packt Publishing.

[18] Syaifuddin, Risqiwati, D. dan Irawan, E.A. (2018), *Realtime Pencegahan Serangan Brute Force dan DDOS Pada Ubuntu Server*, *Techno.Com*, *17*(4), 347–354.

[19] Weidman, G. (2014), *Penetration Testing*, San Francisco: No Starch Press.