IMPLEMENTASI PENUTUPAN CELAH KEAMANAN PADA APLIKASI WEB BERBASIS JOOMLA 1.5.5 SERTA SERVER BERBASIS UBUNTU 8.04 DENGAN KERNEL 2.6.24

Nur Arifin Akbar[*)], Maman Somantri, and R. Rizal Isnanto

Jurusan Teknik Elektro, Fakultas Teknik, Universitas Diponegoro,
Jln. Prof. Sudharto, Tembalang, Semarang, Indonesia

[*)] *email : hackshields@gmail.com*

Abstrak

Pertumbuhan bisnis yang pesat pada saat ini diimbangi dengan peningkatan kebutuhan terhadap aplikasi berbasis web akan membuat aplikasi web itu sendiri akan lebih sulit untuk diamankan. Sebagian besar kalangan korporat menggunakan firewall, SSL, serta metode pengamanan dari segi jaringan maupun host pada web site mereka, namun kebanyakan serangan itu sendiri berasal dari tingkatan aplikasi sehingga pengamanan semacam ini tidak dapat mencegah serangan tersebut. Pada pemodelan Open System Interconnection (OSI), setiap pesan melalui tujuh lapisan dari protocol jaringan, termasuk di dalam lapisan aplikasi yang didalamnya terdapat HTTP serta protokol lain yang berhubungan dengan pertukaran konten, seperti HTML, XML, Simple Object Access Protocol (SOAP), serta Web service. Banyak peretas tahu bagaimana cara membuat HTTP requests terlihat tidak berbahaya dari segi jaringan, akan tetapi tidak dengan data yang ada di dalamnya. Penyerangan dengan karier berupa HTTP dapat mengakses data ke database, menjalankan perintah dari sistem, serta mengubah konten dari Web site. Metode penetration test adalah salah satu cara yang paling efektif untuk mengidentifikasi kelemahan sistem dan kekurangan program. Dengan menerobos mekanisme pertahanan serta lolos dari control keamanan, seorang penetration tester dapat mengidentifikasi kemungkinan cara yang dilakukan peretas untuk membahayakan serta merusak keamanan dari organisasi secara keseluruhan. Oleh karena itu, tujuan dari penetration test ini adalah untuk menunjukkan bagaimana peretas dapat mengakibatkan bahaya yang serius kepada organisasi serta dampaknya dalam hal lain seperti pengaruh pendapatan, reputasi, serta perlindungan konsumen.

Kata Kunci : aplikasi web, peretas, penetration test, keamanan, konten, HTTP, firewall, SSL, jaringan.

Abstract

As businesses grow increasingly dependent upon Web applications, these complex entities grow more difficult to secure. Most companies equip their Web sites with firewalls, Secure Sockets Layer (SSL), and network and host security, but the majority of attacks are on applications themselves, so these technologies cannot prevent them. In the Open System Interconnection (OSI) reference model, every message travels through seven network protocol layers. The application layer at the top includes HTTP and other protocols that transport messages with content, including HTML, XML, Simple Object Access Protocol (SOAP) and Web services. Many hackers know how to make HTTP requests look benign at the network level, but the data within them is potentially harmful. HTTP-carried attacks can allow unrestricted access to databases, execute arbitrary system commands and even alter Web site content. A penetration test is one of the most effective ways to identify systemic weaknesses and deficiencies in these programs. By attempting to circumvent security controls and bypass security mechanisms, a penetration tester able to identify ways in which a hacker might be able to compromise an organization's security and damage the organization as a whole. So that the goal is to show, in a safe and controlled manner, how an attacker might be able to cause serious harm to an organization and impact its ability to, among other things, generate revenue, maintain its reputation, and protect its customers.

Keyword: web applications, hacker, penetration test, security, content, HTTP, firewall, SSL, network.

1. Pendahuluan

Web Server seringkali menjadi target dari berbagai jenis serangan baik yang sifatnya minor maupun major sehingga berakibat fatal. Hal ini dapat terjadi karena aspek keamanan *web server* kurang diperhatikan atau tidak diterapkan secara optimal, sehingga memungkinkan terjadinya resiko yang cukup signifikan.

Adakalanya *Web Server* dikelola oleh individu yang memiliki pengalaman minim dalam pengelolaan suatu

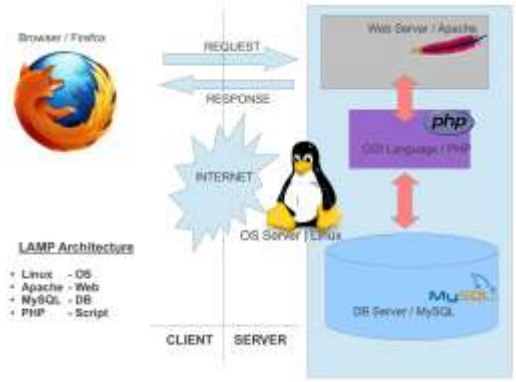
web server. Meskipun umumnya serangan yang terjadi hanya menimbulkan kesan negatif, memalukan atau ketidaknyamanan, namun tidak tertutup kemungkinan penyerang dapat membuat masalah yang lebih serius atau bahkan sangat merugikan.

Sehubungan dengan itu, kiranya perlu bagi pemilik *webserver*, untuk memahami dan menerapkan hal-hal mendasar (*minimum requirement*) dalam pengamanan suatu *webserver.* Di samping itu pemilik juga dapat menerapkan program keamanan yang komprehensif untuk meminimalisasi risiko keamanan *webserver*-nya pada tingkat yang dapat diterima.

2. Metode

Konsep LAMP

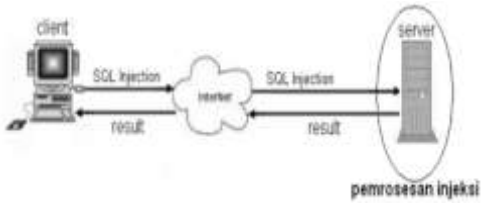
LAMP adalah sebuah *bundle software* untuk *webserver* yang terdiri atas Linux, *Apache HTTPD Server*, *MySQL*, serta *PHP/Python/Perl*. Paket yang terdapat pada LAMP dapat bervariasi, sehingga *compiler PHP* dapat juga diintegrasikan dengan Python atau Perl sesuai dengan kebutuhan. Gambar 1 menjelaskan arsitektur dari LAMP :



Gambar 1. Arsiterktur LAMP

SQL Injection

SQL *injection* adalah kegiatan menyisipkan perintah SQL kepada suatu statement SQL yang ada pada aplikasi yang sedang berjalan. Dengan kata lain SQL *injection* ini merupakan suatu tehnik pengeksploitasi pada web apilikasi yang didalamnya menggunakan database untuk penyimpanan datanya.

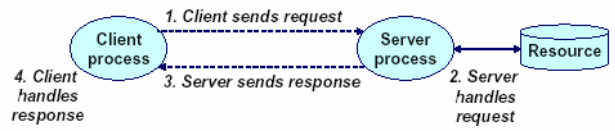


Gambar 2. Topologi Penyerangan SQL *Injection*

Terjadinya SQL *injection* tersebut dikarenakan *security* atau keamanan pada level aplikasi (dalam hal ini aplikasi web) masih kurang sempurna. Kurang sempurnanya adalah pada cara aplikasi meng-*handle* inputan yang boleh di proses ke dalam database. Misalnya pada suatu web yang terdapat fasilitas *login*, terdapat dua buah inputan pada umumnya, yaitu *username* dan *password*. Jika karakter yang masuk melalui dua buah inputan tersebut tidak difilter (disaring) dengan baik maka bisa menimbulkan efek SQL *injection*, ini dikarenakan biasanya inputan tersebut secara sistem akan menjadi bagian dari kriteria dari suatu perintah SQL di dalam aplikasi web-nya.

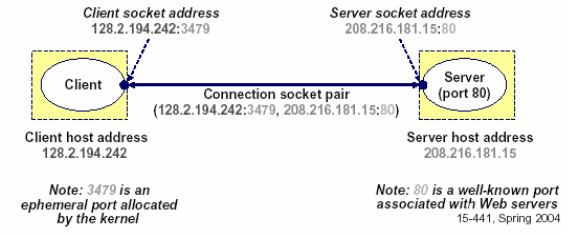
Socket Programming

Setiap aplikasi di jaringan, transaksinya didasarkan pada konsep *client-server*. Sebuah *server* dan sebuah atau beberapa *client* yang meminta/*request* pelayanan ke *server*. Fungsi *server* sebagai pengatur *resource* yang ada, yang menyediakan pelayanan dengan memanfaatkan resource yang untuk kebutuhan *client*. Proses ini (proses *client-server*) bisa dijalankan pada sebuah komputer (komputer tunggal) atau bisa juga satu komputer berfungsi sebagai *server* dan sebuah atau beberapa komputer berfungsi sebagai *client*.



Gambar 3 Transaksi *Client – server*

Pada saat suatu aplikasi berkomunikasi, awalnya aplikasi membuat *socket* baru, maka pada aplikasi tersebut akan diberikan nomer yang digunakan sebagai referensi *socket*. Jika ada suatu sistem yang menggunakan nomer referensi *socket* tersebut, maka akan terjalin suatu jaringan komunikasi antar komputer sebaik transfer data lokal.



Gambar 4. *Connection Socket Pair*

Untuk berkomunikasi dengan *server*, *client* harus tahu nomor *IP server* begitu juga nomor *port* yang dituju, nomor port menunjukkan service yang dijalankan. Contoh port 23 untuk *Telnet Server*, port 25 untuk *Mail Server* dan port 80 untuk *Web server*. Dalam hal

ini aplikasi di *client* sudah mengetahui port yang akan dituju.

Shellcode

Dalam dunia computer *security*, kata '*exploit'* merupakan salah satu kata yang paling sering digunakan dan dijumpai. *Exploit* seakan-akan seperti senjata yang dapat digunakan untuk membobol suatu sistem. Saat ini, ada cukup banyak jenis *exploit* yang dapat digunakan. Ada *exploit* yang menyerang suatus sistem secara langsung dengan memanfaatkan *security hole* pada sistem tersebut (*system based*), dan ada juga *exploit* yang digunakan untuk menyerang *security hole* pada suatu aplikasi web (*web based exploit*). Ada sangat banyak cara menemukan *hole security*, dan ada banyak juga jenis *hole* yang dapat di manfaatkan.

Diantara jenis *hole* tersebut, ada 1 jenis *hole* yang dimana jenis *hole* tersebut digunakan sebagai dasar pembuatan *exploit* bahkan hingga saat ini. Jenis *hole* tersebut adalah '*buffer overflow*', yang sampai hingga ini masih menjadi salah satu jenis *hole* tertua namun banyak di gunakan sebagai dasar menemukan kelemahan suatu sistem maupun aplikasi yang sifat nya *system based*.

Exploit akan membuat suatu aplikasi/sistem yang memiliki kelemahan tertentu 'crash' dan memboikot aplikasi tersebut sehingga prosesor yang seharus nya menjalankan kode-kode program aplikasi tadi akan diarahkan untuk menjalankan kode-kode lain yang telah di persiapkan oleh *exploit*, atau dengan nama lain ini adalah *shellcode*.

Saat *exploit* merusak suatu aplikasi/sistem, *exploit* akan menempatkan *shellcode* di suatu lokasi memori pada mesin target, selanjutnya *exploit* akan membuat prosesor menuju lokasi memori tempat *shellcode* kemudian akan mengeksekusi-nya. Mekanisme ini berhubungan langsung dengan prosesor dan memory, itu sebab nya *shellcode* dibuat agar dapat langsung di eksekusi pada memori sehingga bentuk nya cukup aneh. Berikut adalah contoh program *shellcode* :

```
char shellcode[] =
"\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xeb\x0f\x59\xb
3"

"\x01\xb2\x0d\xb0\x04\xcd\x80\xfe\xcb\xb0\x01\xc
d"

"\x80\xe8\xec\xff\xff\xff\x48\x65\x6c\x6c\x6f\x2
c"

"\x20\x77\x6f\x72\x6c\x64\x21";
```

Pencegahan Injeksi SQL

Langkah yang dapat di tempuh untuk mengurangi penyusupan ke halaman web dengan SQL *Injection* dengan cara:

a. Memfilter dengan tidak membolehkan karakter seperti *single quote, double quote, slash, back slash, semi colon, extended character like NULL, carry return, new line,* dll dalam *string form:*
- Masukan dari from *user*s
- Parameters di URL
- Nilai dari *cookie*

b. Untuk nilai numeric, convert dulu sebelum melewati *statement* SQL dengan mengunakan ISNUMERIC untuk meyakinkan itu adalah integer.

c. Mengubah "Startup and *run* SQL *Server*" menggunakan *low privilege user* dalam SQL *Server Security* tab.

d. Ubah *stored procedure* – store procedure yang tidak terpakai, seperti: master..Xp_cmdshell, xp_startmail, xp_sendmail, sp_makewebtask

Pencegahan Eksekusi Webshell

Pencegahan *webshell* agar tidak dapat mengeksekusi perintah yang tedapat di Linux shell adalah dengan cara menonaktifkan sebagian fungsi yang terdapat pada php, berikut adalah fungsi yang dinonaktifkan :

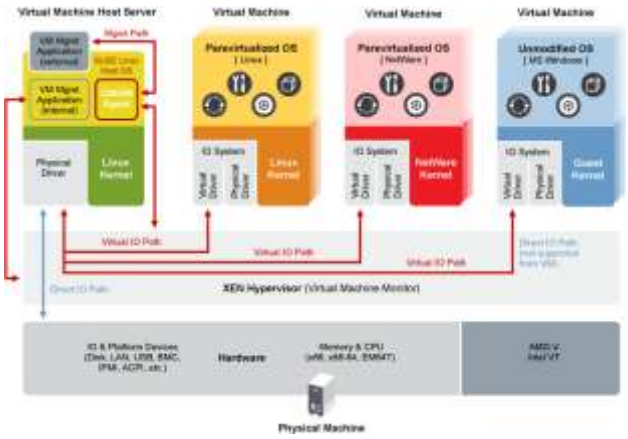
a. passthru
b. proc_open
c. proc_close
d. proc_get-status
e. proc_nice
f. proc_terminate
g. exec
h. system
i. suexec
j. popen
k. pclose
l. dl
m. ini_set
n. virtual
o. shell_exec

3. Perancangan Sistem

Perancangan pada *server host*

Perancangan sistem pada *host* diperuntukan untuk pembuatan *server* secara *virtual*. Pada *server host* akan dipasang teknologi *virtual*isasi untuk *guest* sehingga *guest* akan dipasang *server* yang *vulnerable*. Hal ini akan memudahkan dalam hal manajemen dan *maintenance server*.

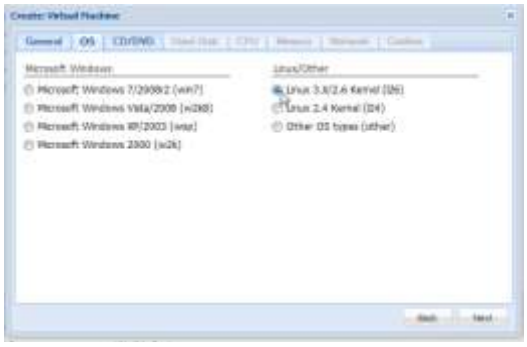
Teknologi virtualisasi yang digunakan adalah teknologi *full-virtualization* , sehingga hampir semua OS *guest* dapat disupport oleh teknologi ini. Hal ini memungkinkan OS *guest* memiliki kernel tersendiri, meskipun *resource* yang dibutuhkan jauh lebih besar.



Gambar 5. Skema Virtualisasi

Perancangan pada *server VPS*

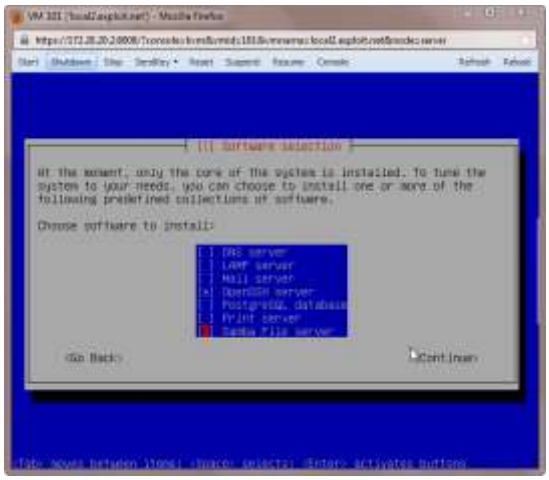
VPS akan diimplementasikan dengan peran *guest* OS terhadap *host*. Sebelum installasi OS terhadap VPS, maka perlu penyesuaian tipe kernel yang akan diimplementasikan. Pada *setting* di mesin *host*, VPS akan menggunakan sistem operasi Ubuntu *Server* 8.04 dengan kernel 2.6.24.



Gambar 6. Pemilihan tipe kernel

Konfigurasi NIC (*Network Interface* Card) pada VPS menggunakan sistem *bridge* agar VPS memiliki IP tersendiri yang dapat diakses dari luar. *Setting* IP dapat dilaksanakan ketika *guest* OS telah terinstall dalam VPS melalui *console* dari VNC melalui *web*.

Pada saat progress installasi *server*, paket SSH *server* diinstall pada *wizard* sehingga tidak perlu menginstall secara *manual* dari *repository*, sedangkan paket LAMP *server* diabaikan karena LAMP akan diinstall secara terpisah.



Gambar 7. *Software Selection*

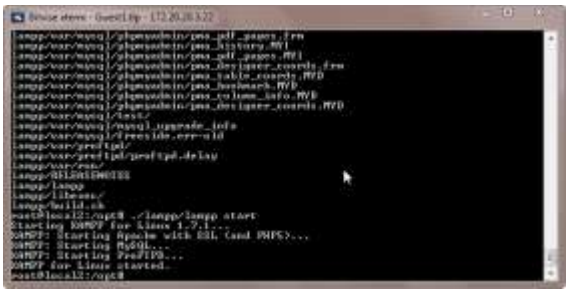
IP pada VPS dikonfigurasi menggunakan *console* dari *web browser*, perlu diingat bahwa JRE (*Java Runtime Environment*) diperlukan agar *console* dapat berjalan dengan baik. Pada ubuntu *server*, mengubah alamat IP dapat dilakukan dengan mengedit file /etc/*network*/*interfaces*.

Gambar 8. Konfigurasi IP

Perancangan Web Server dan aplikasi web pada VPS

Web server menggunakan aplikasi XAMPP 1.7.1 for Linux, adapun versi software yang digunakan adalah PHP versi 5.2.9, Mysql versi 5.1.3, Phpmyadmin versi 3.13. PHP versi 5.2.x digunakan untuk menghindari peringatan pada fungsi yang *deprecated* serta meningkatkan kompatibilitas pada aplikasi *web* yang lama. Mysql versi 5.x digunakan pada *server database* agar dapat menguji injeksi SQL menggunakan perintah UNION.

Installasi XAMPP pada Linux dapat dijalankan dengan meletakkan file .tar.gz pada direktori `/opt` , kemudian ekstrak paket pada direktori tersebut. Sedangkan untuk menjalankan *webserver* dapat menggunakan perintah `/opt/lamp/lampp start`
yang dijalankan sebagai *root*



Gambar 9. Menjalankan XAMPP

Gambar 10. Halaman XAMPP

Aplikasi *web* menggunakan software Joomla 1.5.5, aplikasi ini dilettakan pada folder htdocs dari aplikasi xampp pada *subfolder joomla*.

Gambar 11. Halaman Utama Joomla

4. Pengujian dan Patching Sistem

Pengujian Sistem
Pengujian Injeksi SQL

Joomla memiliki fitur apabila administrator atau *user* lupa akan *password*-nya. *User* akan *request password* dari halaman *web*, kemudian aplikasi akan mengirimkan *token* ke email *user* yang bersangkutan, lalu *user* diwajibkan untuk menginputkan nilai *token* yang telah dikirim ke email kepada *user*

Pada file `/components/com_user/controller.php` baris ke 379-399 berisi :

```
…
// Input
 $token = JRequest::getVar('token', null,
 'post', 'alnum'); //< --- {1}
…
…
// Verifikasi token
if ($model->confirmReset($token) === false)// <-
--{2}
…
```

serta pada file `/components/com_user/models/reset.php` pada baris 111-130 yang berisi :

```
…
$db->setQuery('SELECT id FROM #__users WHERE
block = 0 AND activation = '.$db-
>Quote($token)); //<---- {3}
…
```

Maka dari point 1,2, dan 3 yang ditunjukkan dengan *sintaks* `//<----` `{nomer}` terdapat celah yang dapat dimanfaatkan peretas untuk mengganti *password administrator* tanpa *token* yang *valid*. Pada halaman `index.php?option=com_user&view=reset&layout=confirm` di URL installasi *Joomla* muncul halaman yang meminta *token* dari email yang menggunakan URL installasi *Joomla* pada `http://172.20.20.3/Joomla/index.php?option=com_user&view=reset&layout=confirm`

Pada halaman *valid*asi *token*, diinput dengan menggunakan *single quote* (') sehingga perintah pada SQL berubah menjadi `"SELECT id FROM jos_users WHERE block = 0 AND activation = '' "`, kemudian pada halaman akan memasukkan *password* baru untuk administrator

Gambar 12. *New admin password*

Pengujian *upload Malicious Code*

Upload Malicious code dilakukan melalui *installer component* dalam Joomla. *Component* yang digunakan dalam *upload Malicious code* adalah *comextplorer* berupa file *browser*. *Permission* pada direktori *component* adalah *world-writeable*, seperti pada perancangan sistem yang telah dibahas sebelumnya.

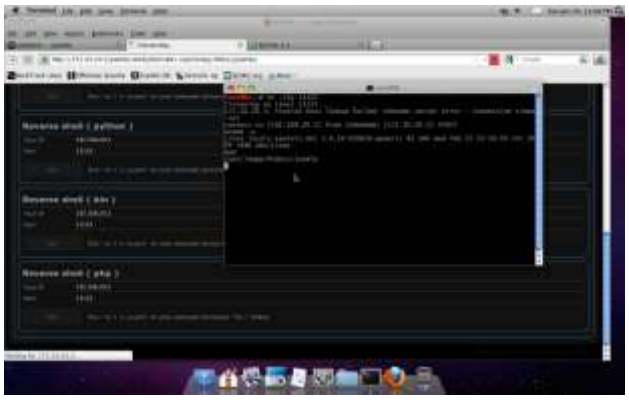
Component extplorer berfungsi sebagai manager file berbasis *web* pada aplikasi *Joomla*. *Extplorer* mengunggah file berupa *webshell* yang dapat melakukan *back-connection* melalui fungsi PHP pada folder yang *world-writable*. Installasi folder *Joomla* memiliki *Permission* yang *world-writeable*, sehingga *shell* akan diletakkan pada folder *root* dari *Joomla*. Versi *webshell* yang digunakan adalah b374k versi 2.2

Pengujian *Root Exploitation*

Pengujian exploitasi *root* pada kasus ini menggunakan sistem *client-server*, dimana *client* berperan sebagai peretas dan *server* sebagai target. Pada sisi *client* menggunakan sistem operasi *Back*track 5 R1, sedangkan pada sisi target menggunakan sistem operasi Ubuntu 8.04 *server*.

Setelah *webshell* sudah terunggah pada sisi *server*, sebagian besar peretas menggunakan metode *bind connection* untuk memulai teknik *remote shell* pada sisi *server*. Namun hampir semua *server* pada saat ini telah memiliki *firewall* sehingga tidak dimungkinkan untuk melakukan *bind connection* karena membuka *port* yang pada sisi *server*.

Teknik yang akan dipakai pada kasus ini menggunakan *reverse shell* atau disebut juga dengan *back connection*, pada sisi *client* di buka *port* untuk *listening* pada *port* tertentu. Pada kasus ini digunakan *port* 31123 sebagai *listening port*, ketika *client* menjalankan perintah `nc -vlp 31123` , kemudian *webshell* akan mengeksekusi perintah untuk *back connection.*



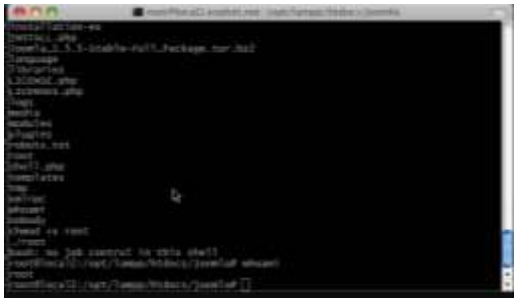
Gambar 13. *Reverse shell console*

Perintah `uname -a` digunakan untuk mengetahui versi kernel yang berjalan, pada gambar 4.10 ditunjukkan bahwa versi kernel yang berjalan adalah versi 2.6.24. Kernel versi 2.6.24 memiliki kelemahan pada fungsi `get_iovec_page_array()` yang terdapat pada file `fs/splice.c` yang dapat dilihat dari *source code* kernel versi 2.6.24. Terdapat *bug* berupa *buffer overflow* sehingga peretas dapat mengeksekusi perintah untuk mendapatkan perintah *root.*

Shellcode yang digunakan untuk mendapatkan *user root* dari kernel 2.6.24 adalah vmsplice, *source code shellcode* ini harus di kompilasi terlebih dahulu menggunakan gcc. Kompilasi dari *source code* vmsplice dapat dilakukan di *server* jika terinstall gcc maupun *client*.

Pada *reverse shell*, *user* yang menjalankan *shell* adalah *user* yang memiliki direktori *webserver*. Secara default, folder htdocs pada xampp dimiliki oleh *nobody*. Namun demikian *shell* masih dapat dijalankan asalkan memiliki *permisson* agar dapat di eksekusi dari *user* manapun. Untuk mengetahui *user* yang berjalan pada *reverse shell* dapat menggunakan perintah `whoami` .

Setelah perintah whoami dijalankan, maka akan muncul pesan `nobody` yang berarti *shell* berjalan sebagai *user nobody*. Kemudian *binary code* untuk *root*ing dijalankan sehingga *user* dapat berjalan sebagai *root.*



Gambar 14. *Root Privelleges*

Patching Sistem
Patch Injeksi SQL

Joomla 1.5.5 memiliki kelemahan pada *token valid*ation yang memungkinkan peretas dapat mengganti *password admini*strator dengan metode Injeksi SQL. *Token* di *generate* melalui sistem *Joomla* ketika ada permintaan *forget password*, panjang *token* adalah 32 karakter. Solusi untuk menutup celah ini selain *update* adalah menambahkan kode pada file `/components/com_user/models/reset.php` di baris 113 setelah kode `$mainframe` dengan kode sebagai berikut :

```
if(strlen($token) != 32) {
$this->setError(JText::_('INVALID_TOKEN'));
return false;}
```

Patch Upload Malicious Code

Upload Malicious code seperti *webshell* dapat dilakukan melalui *web* ketika folder memiliki *Permission* pada mode *world-writeable*. Teknik *upload shell* dapat dilakukan melalui *admin panel* yang biasanya

menyediakan fasilitas *upload* baik untuk *template*, gambar, maupun modul. Pada *web server*, hal yang dilakukan untuk meningkatkan keamanan pada *web server*, *setting Permission* pada folder tidak lebih dari 755 dan pada file tidak lebih dari 644. Perintah yang digunakan untuk chmod folder pada mode 755 secara rekrusif adalah :

```
find . -type d -exec chmod 755 {}\;
```

sedangkan perintah untuk chmod file pada mode 644 adalah :

```
find . -type f -exec chmod 644 {}\;
```

Patch Exploitasi Root

Exploit Root yang dijalankan berupa *binary code* dari vmsplice yang sudah terkompilasi akan menyerang *system call* dari kernel sehingga terjadi *buffer overflow*. Salah satu cara agar celah tertutup adalah dengan cara *update* kernel, baik secara *manual* dengan cara kompilasi dari *source* kernelnya, maupun installasi secara *automatis* dari *repository*. Kompilasi dari *source* kernel membutuhkan waktu yang lama untuk kompilasinya, sehingga pada kasus ini digunakan cara *update* kernel dari *repository*. Perlu diingat bahwa *repository* harus selalu sinkron dengan *server repo* agar selalu *update*. Perintah untuk *update repository* adalah :

```
apt-get update
```

sedangkan cara untuk *upgrade* kernel linux pada *ubuntu server* menggunakan *repository* adalah :

```
apt-get install linux-image-server
```

5. Penutup

Kesimpulan

Dari hasil pengujian dan *patching* dapat disimpulkan bahwa:

1. Metode pengujian celah keamanan yang dilakukan oleh penulis adalah rangkaian dari metode *SQL Injection*, *Reverse Shell*, serta Root *Exploitation.*
2. Penutupan celah keamanan dilakukan berdasarkan rangkaian metode yang digunakan penulis untuk melakukan penetration test terhadap server .
3. Pengaturan permission pada direktori web server tidak lebih dari 755, sedangkan permission pada file tidak lebih dari 644 dilakukan untuk mencegah shellcode dapat terunggah di sisi server.
4. Form pada aplikasi web memiliki filter pada bagian input agar dapat terhindar dari serangan injeksi SQL.

Saran

Adapun saran yang dapat diberikan untuk menjadi masukan pada penelitian lebih lanjut adalah:

1. Perlu dilakukan penelitian untuk teknik lain untuk penyerangan melalui *web* seperti LFI (*Local File Inclusion*), RFI (*Remote File Inclusion*) serta cara penanggulangannya.
2. Kernel pada server harus memiliki versi terbaru yang stabil dan kompatibel, baik update menggunakan repository maupun kompilasi secara manual.
3. Perlu dilakukan penelitian untuk pengamanan melalui konfigurasi modul pada *web server* apache serta konfigurasi pada *interpreter* PHP.
4. Perlu diketahui penelitian untuk penggunaan program keamanan seperti *rootkit hunter*, *webshell detector*.

Daftar Pustaka

[1]. John D. Howard, *An Analysis Of Security Incidents On The Internet 1989 - 1995*, PhD thesis, Engineering and Public Policy, Carnegie Mellon University, 1997.

[2]. Stuttard, Dafydd and Marcus Pinto. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, Second Edition*. John Wiley & Sons, Inc.. 2011.

[3]. Rahardjo, Budi. *Keamanan Sistem Informasi Berbasis Internet*. PT Insan Infonesia & PT INDOCISC.2005.

[4]. Sulistyo, Anom. *Membuat Distro Linux untuk Security*. Politeknik Elektronika Negeri Surabaya. 2010.

[5]. Justin, Clarke. *SQL Injection Attack and Defense*. "http://www.flazx.com/ebook13843.php". 2009

[6]. McClure. Stuart, Shah. Saumil. *Web Hacking: Attacks and Defense*. "http://www.flazx.com/ebook982.php". 2002

[7]. Utdirartatmo. Firrar. *Ancaman Internet Hacking dan Trik Menanganinya*. Yogyakarta: ANDI-Offset. 2006.

[8]. Shulman. Amichai. *Traditional SQL Injection Injection*: *The Wrong Solution for the Right Problem*. "http://www.imperva.com/resources/Adc/pdfs/Traditional_SQL_InjectionProtection_The_Wrong_Solution_for_the_Right_Problem.pdf". 2010

[9]. Tim Berners-Lee, *Weaving the Web: the past, present and future of the world wide web by its inventor*. Texere. 2000.

[10]. Steven M. Bellovin, "*Security Problems in TCP/IP Protocol Suite*," Computer Communication Review, Vol. 19, No. 2, pp. 32-48, 1989.