Uji Kelayakan Implementasi SSH sebagai Pengaman FTP Server dengan Penetration Testing

Batara Sakti
Jurusan Informatika
Fakultas MIPA, UNS
Jl. Ir. Sutami 36 A Kentingan Surakarta
batara.sakti.gl@gmail.com

Abdul Aziz
Jurusan Informatika
Fakultas MIPA, UNS
Jl. Ir. Sutami 36 A Kentingan Surakarta
Abdul_7773@yahoo.com

Afrizal Doewes
Informatika
Fakultas MIPA UNS
Jl.Ir Sutami No.36 A Kentingan Surakarta
afrizal.doewes@googlemail.com

ABSTRACT
Penelitian ini ditujukan untuk mengukur tingkat keamanan SSH dalam mengamankan transmisi FTP. Penelitian dilakukan dengan menerapkan penetration testing pada sistem yang menggunakan service SSH dan FTP. Penetration testing adalah teknik pengujian untuk mencari kelemahan dan kekurangan suatu sistem dengan menghindari atau menerobos mekanisme keamanan yang ada untuk mensimulasikan teknik penyerangan yang mungkin untuk dilakukan. Tingkat keamanan SSH akan diukur dari ketahanannya dalam menghadapi setiap proses serangan yang disimulasikan dalam penetration testing.
Hasil penelitian menunjukkan bahwa implementasi SSH secara standar tidak cukup efektif untuk mengamankan FTP. Sebab SSH hanya mampu melindungi transmisi FTP dari penyadapan data. SSH terbukti rentan terhadap serangan brute force dan fitur keamanannya tidak berpengaruh pada pengaksesan backdoor yang telah dipasang pada FTP. SSH memerlukan konfigurasi khusus dan piranti pendukung lain untuk mengoptimalkan fungsi keamanannya.

Keywords
SSH, FTP. keamanan, penetration testing

1. PENDAHULUAN
File Transfer Protocol (FTP) merupakan salah satu sarana untuk melakukan *sharing* data dimana data tersebut tersimpan pada *directory* sebuah komputer *server* sehingga dapat diakses oleh sejumlah besar komputer secara bersamaan. FTP menggunakan autentikasi dengan *username* dan *password* untuk menambahprivasi pada data yang di-*sharing*. Sayangnya, *username* dan *password* yang digunakan terkirim dalam bentuk tidak terenkripsi [1].
Secure Shell (SSH) sebagai salah satu protokol keamanan jaringan, sering digunakan untuk mengamankan transmisi FTP dengan memanfaatkan fungsi enkripsi pada SSH. Namun dengan semakin berkembangnya teknik-teknik serangan pada keamanan jaringan komputer, perlu dilakukan pengujian untuk mengukur tingkat keamanan SSH dalam mengamankan transmisi FTP.
Salah satu metode pengujian yang dapat dilakukan adalah dengan menggunakan *penetration testing*. *Penetration testing* adalah metode pengujian keamanan jaringan dengan cara mensimulasikan serangan-serangan yang mungkin bisa dilakukan pada suatu sistem.
Beberapa penelitian yang berkaitan dengan *penetration testing* dan FTP *Server* telah dilakukan diantaranya penelitian [2] yang membahas analisa keamanan FileZilla Server sebagai salah satu server FTP berbasis Open Source. Selain itu, penelitian[3] yang membahas tentang bentuk-bentuk *penetration testing* yang dilakukan pada jaringan komputer yang memiliki mekanisme protokol kemananan tinggi. Lebih lanjut lagi penelitian [4] yang mengulas tentang *penetration testing* pada VPN.
Penelitian ini akan berfokus pada cara menerapkan *penetration testing* pada sistem sesuai dengan kaidahnya dan cara mengukur tingkat keamanan implementasi SSH berdasarkan hasil *penetration testing* yang telah dilakukan.
Pengujian dilakukan pada sistem yang dibangun sebagai objek penelitian dimana sistem tersebut terisolasi sehingga tidak memerlukan manajemen *user* secara mendetail. Sistem dibangun di Lab Komputer Jurusan Informatika UNS. Pengujian akan berfokus pada fungsi SSH dalam mengamankan transmisi FTP namun tidak mengikutsertakan teknik enkripsi yang digunakan pada SSH kedalam topik penelitian. Metode yang digunakan adalah *white-box testing* dimana pengujian dilakukan dari dalam jaringan dengan tanpa melakukan pengumpulan informasi terlebih dahulu.
Tujuan dari penelitian ini adalah untuk menguji tingkat keamanan pada implementasi SSH dalam melindungi jalur FTP dari berbagai kemungkinan serangan.
Penelitian ini akan menghasilkan dokumentasi dari data-data pengujian keamanan pada implementasi SSH untuk FTP. Data-data tersebut akan berguna sebagai parameter untuk menentukan solusi keamanan pada sistem jaringan komputer lain yang serupa.

2. Landasan Teori
2.1. FTP
FTP didefinisikan sebagai sebuah protokol untuk mengirim dan menerima *file* antara *host* (dalam ARPANET), dengan fungsi utama dari FTP adalah mengirim dan menerima *file* dengan efisien dan handal antara *host* dan mengijinkan penggunaan yang nyaman dari kemampuan untuk penyimpanan *file* secara *remote.*
FTP menggunakan autentikasi dengan *username* dan *password* untuk menambah privasi pada data yang di-*sharing*. Sayangnya, *username* dan *password* yang digunakan terkirim dalam bentuk tidak terenkripsi [1]. Hal tersebut menyebabkan transmisi FTP rentan terhadap penyadapan data. Jika transmisi FTP berhasil disadap, pesan yang ada didalamnya dapat dengan mudah dibaca oleh pihak luar.

2.2. SSH
SSH adalah protokol jaringan yang memungkinkan pertukaran data melalui saluran aman antara dua perangkat jaringan [5]. Terutama banyak digunakan pada sistem *berbasis linux (*UNIX *like)* dan Unix untuk mengakses akun shell. SSH dirancang sebagai pengganti Telnet dan shell remote tak aman lainnya, yang mengirim informasi, terutama kata sandi, dalam bentuk *plain text* yang membuatnya mudah untuk disadap. Enkripsi

yang digunakan oleh SSH menyediakan kerahasiaan dan integritas data melalui jaringan yang tidak aman seperti Internet.SSH menggunakan kriptografi kunci publik untuk mengautentikasi komputer *remote* dan membiarkan komputer *remote* untuk mengautentikasi pengguna, jika perlu. SSH biasanya digunakan untuk *login* ke mesin *remote* dan mengeksekusi berbagai perintah. SSH dapat digunakan untuk mentransfer file melalui SFTP atau SCP. SSH menggunakan *client-server* model. Standar TCP *port* 22 telah ditetapkan sebagai jalur untuk server SSH. Sebuah klien SSH biasanya digunakan untuk membangun koneksi ke SSH *daemon* supaya dapat dikendalikan via *remote*.

SSH memiliki kelemahan diantaranya sebagai berikut.

1. Root Login

Kelemahan protokol SSH yang ada pada berbagai sistem operasi (dalam bentuk openSSH) adalah diperbolehkannya *login root* langsung [6]. Hal ini menyebabkan apabila *password root* diketahui, penyerang bisa memegang penuh kendali *server*. Salah satu cara penyerangan bisa dilakukan dengan tool thc-hydra yang telah dikompilasi sebelumnya. Hydra akan melakukan *brute-force* pada kombinasi *password* dan hasil pencarian akan didapat dengan waktu yang bergantung keunikan kunci.

2. Brute Force User Login

SSH yang ada tanpa konfigurasi biasanya tidak punya keamanan batasan untuk mencegah penyerangan autentifikasi login secara *brute force*[6].Dengan cara yang cukup sama, hydra bisa melakukan kombinasi brute force terhadap pasangan username (masukan file) dan password. Cara ini biasanya akan memakan waktu cukup lama karena juga perlu mencari user yang ada pada sistem.

2.3. Tipe Pengujian White-box

*White-box testing*adalah tipe pengujian dimana ketika melakukan pengujian atau evaluasi keamanan jaringan, pihak penguji mempunyai informasi dan data-data lengkap mengenai infrastruktur jaringan sama seperti yang dimiliki oleh *administrator* jaringan tersebut. Penguji dapat langsung memasuki fase penyerangan karena tidak perlu melakukan pengumpulan informasi, peninjauan, dan pemindaian infrastruktur jaringan [7].

2.4. Penetration Testing

Penetration testing adalah salah satu cara untuk mensimulasikan metode yang mungkin akan digunakan oleh penyerang untuk menghindari atau menerobos mekanisme keamanan dan mendapatkan akses secara ilegal ke dalam suatu sistem [7].

Dewasa ini, terdapat perubahan pada cara pandang dan definisi pada *penetration testing* dalam industri keamanan. Terdapat standar baru yang disebut *Penetration Testing Execution Standart* (PTES). PTES mendefinisikan ulang *penetration test* dalam berbagai cara yang akan mempengaruhi *penetration tester* yang berpengalaman maupun yang masih baru. PTES telah diadopsi oleh berbagai anggota terkemuka dari komunitas internasional keamanan jaringan. PTES menegaskan dan membangkitkan esensi dari *penetration testing* dengan menetapkan garis dasar dari prinsip fundamental tentang bagaimana melakukan *penetration test*.

Tahapan PTES didesain untuk menegaskan langkah-langkah *penetration test* dan meyakinkan *client* mengenai standarisasi yang digunakan saat melakukan *penetration test*. Tahapan PTES terdiri dari tujuh kategori dengan masing-masing tingkat pengerjaan yang berbeda tergantung dari sistem yang akan diserang [7].

1. Interaksi pra-pengujian

Interaksi Pra-pengujian biasanya terjadi ketika tester mendiskusikan ruang lingkup dan terminologi dari penetration testing dengan client. Sangat penting ketika masa pra-pengujian, tester menyampaikan target dan tujuan dari pengujian. Tahap ini juga berguna sebagai kesempatan untuk menginformasikan kepada customer tentang apa yang harus dicapai dan ruang lingkup penuh selama pengujian berlangsung.

2. Pengumpulan Informasi

Pada tahap pengumpulan informasi, penguji akan mengumpulkan informasi tentang sistem sebanyak mungkin. Salah satu keterampilan yang terpenting dari seorang *penetration tester* adalah kemampuan untuk mempelajari target, temasuk bagaimana perilaku target, bagaimana cara kerja target, dan pada akhirnya bagaimana target dapat mendapat serangan. Informasi tentang target yang telah dikumpulkan akan memberikan gambaran mengenai tipe kendali keamanan yang digunakan. Selama pengumpulan informasi, penguji akan mencoba untuk mengidentifikasi mekanisme keamanan yang digunakan pada target dengan melakukan penyelidikan pada sistemnya.

3. Pemodelan Ancaman

Pemodelan ancaman menggunakan semua informasi yang didapat dari tahap pengumpulan informasi untuk mengidentifikasi semua kerentanan yang ada pada sistem. Saat melakukan pemodelan ancaman, penguji akan menentukan metode serangan yang paling efektif, jenis informasi yang akan dicari, dan bagaimana sistem tersebut dapat terkena serangan. Pemodelan ancaman melibatkan pandangan akan sistem sebagai mangsa dan mencoba mengeksploitasi kelemahan seperti yang akan dilakukan penyerang.

4. Analisis Kerentanan

Penguji perlu mempertimbangkan cara untuk mengakses sistem target dengan menentukan metode serangan yang paling tepat. Selama proses analisis kerentanan, penguji mengkombinasikan informasi yang didapat dari tahap sebelumnya dan menggunakanya untuk mempelajari cara apa yang paling tepat untuk digunakan sebagai serangan. Di antara hal-hal lainnya, tahap analisis kerentanan mengantar penguji kepada *account port*, pengumpulan data dengan *banner grabbing*, dan pengumpulan informasi.

5. Eksploitasi

Eksploitasi mungkin adalah salah satu tahap paling penting dalam *penetration test*. Namun hal tersebut sering dilakukan dengan *brute force* daripada serangan yang lebih presisi. Eksploitasi harus dilakukan ketika penguji yakin bahwa eksploitasi tersebut akan berhasil. Mungkin halangan-halangan tak terduga yang disebabkan oleh keamanan sistem dapat mencegah eksploitasi berhasil, namun sebelum melakukan eksploitasi terhadap kerentanan sistem, seorang penguji harus tahu bahwa sistem tersebut benar-benar rentan. Serangan yang membabi buta tidak akan efektif dan hanya menghasilkan sedikit manfaat untuk penguji maupun client.

6. Pasca Eksploitasi

Fase pasca eksploitasi dimulai ketika penguji telah melakukan penetrasi ke dalam satu sistem atau lebih. Pasca eksploitasi adalah fase kritis dalam *penetration test*. Fase ini membedakan penguji dengan *hacker* dan menyediakan informasi dan intelijen yang berharga dari *penetration test yang* telah dilakukan. Pasca eksploitasi menargetkan pada *sistem specific*, mengidentifikasi infrastruktur kritis, dan data atau informasi yang paling diamankan pada sebuah sistem.

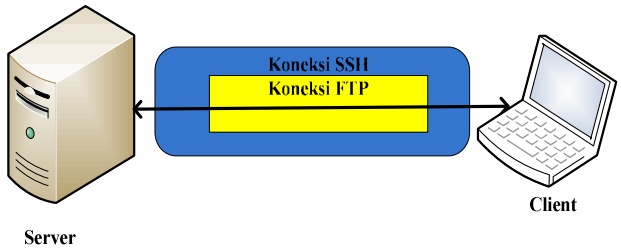
7. Pelaporan

Pelaporan adalah fase terakhir yang paling penting. Penguji menggunakan laporan untuk menunjukkan apa yang telah dilakukan, bagaimana melakukannya, dan yang terpenting adalah bagaimana cara memperbaiki atau menutup celah keamanan yang telah ditemukan selama *penetration test*.

3. Metode Penelitian

3.1. Pembangunan Sistem

Sistem akan dibangun pada Laboratorium Jaringan Komputer Jurusan Informatika FMIPA UNS. Gambar 1 menunjukkan sistem yang dibangun berupa *server* yang melayani transmisi FTP dengan *service* SSH. Kedua *service* tersebut akan dibangun secara terpisah namun masih berada pada satu *server* yang sama.



Gambar 1. Ilustrasi Sistem

Tabel 1 menunjukkan ringkasan dari informasi *server* dan sistem secara keseluruhan, yang digunakan sebagai objek penelitian, beserta konfigurasi *username* dan *password* FTP servernya:

Tabel 1. Ringkasan Informasi Server dan Sistem

Processor	2x Intel (R) Xeon (R) 3.65 GHz.
Memory	1028 MB.
Ipaddress	192.168.9.10
Sistem Operasi	Ubuntu Server 10.04.4 LTS.
Nama Komputer	Ubuntuserfinf.
Service (Daemon)	FTP (*Proftpd*), Telnet(*Telnetd*), SSH(*OpenSSH*).
Nama Pengguna (*password*)	Root(*informatika*),informatika(*informatika*), userftp alias sakti(*informatika*).

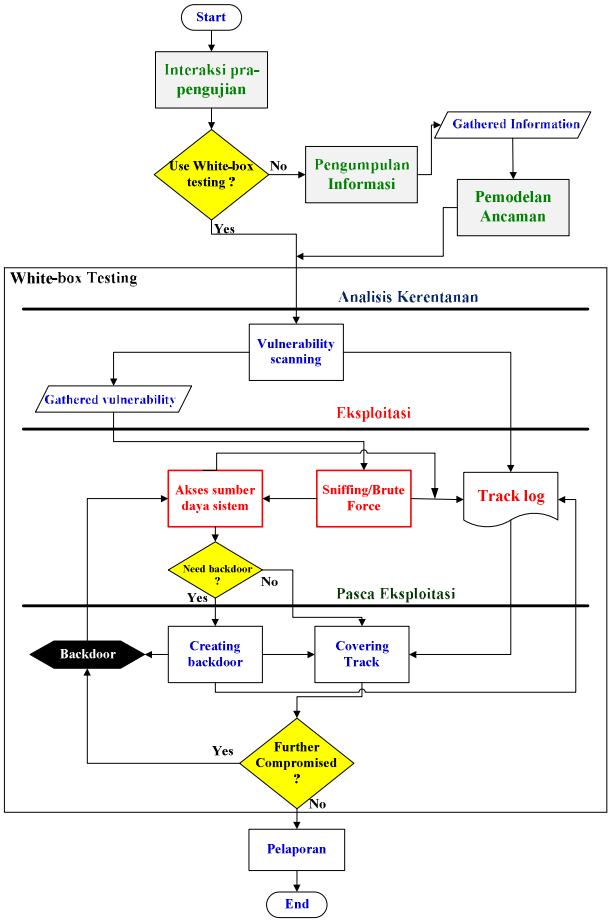
3.2. Perancangan Skenario Pengujian Sistem

Perancangan skenario pengujian mengadaptasi kaidah-kaidah pengujian dan standar *penetration testing* yang telah di bahas pada landasan Teori. Secara garis besar, proses *penetration testing* yang akan dilakukan dapat digambarkan dengan diagram alir sebagaimana ditunjukkan oleh Gambar 2.

Rincian dari skenario pengujiannya dijelaskan sebagai berikut :

1. Interaksi pra-pengujian, Pengumpulan Informasi, dan Pemodelan Ancaman

Pada penelitian ini, pengujian dilakukan pada *prototype* sistem dimana penguji tidak memiliki *client*. Pengujian dilakukan dengan metode *white-box* karena informasi tentang sistem sudah diketahui secara lengkap. Pengujian berfokus pada *service* FTP dan SSH dimana sudah terdapat model ancaman berdasarkan teori dari kelemahan masing-masing *service*. Dengan demikian, pada penelitian ini fase Interaksi pra-pengujian, Pengumpulan Informasi, dan Pemodelan Ancaman dapat dilewati.



Gambar 2. Diagram Alir Penetration Testing

Rincian dari skenario pengujiannya dijelaskan sebagai berikut :

2. Interaksi pra-pengujian, Pengumpulan Informasi, dan Pemodelan Ancaman

Pada penelitian ini, pengujian dilakukan pada *prototype* sistem dimana penguji tidak memiliki *client*. Pengujian dilakukan dengan metode *white-box* karena informasi tentang sistem sudah diketahui secara lengkap. Pengujian berfokus pada *service* FTP dan SSH dimana sudah terdapat model ancaman berdasarkan teori dari kelemahan masing-masing *service*. Dengan demikian, pada penelitian ini fase Interaksi pra-pengujian, Pengumpulan Informasi, dan Pemodelan Ancaman dapat dilewati.

2. Analisis Kerentanan

Analisis kerentanan akan dilakukan dengan *vulnerability scanning*. Tujuan dari *vulnerability scanning* adalah untuk mengidentifikasi kerentanan yang terpapar pada jaringan komputer. *Vulnerabilty scanning* dilakukan dengan menganalisis data-data, ciri-ciri, dan struktur jaringan yang sudah diketahui sebelumnya. Analisis dilakukan dengan menggunakan perangkat lunak **Nessus 4.4.1** sebagai *vulnerability scanner*.

3. Eksploitasi

Pada penelitian ini, eksploitasi akan dilakukan dengan cara sebagai berikut.

- *Sniffing*

Melakukan penyadapan pada aliran data antara *server* dan *client*. *Sniffing* dilakukan untuk mencari informasi penting yang

tidak terlindungi selama aliran data berlangsung. *Tool* yang digunakan adalah **wireshark**.

- *Brute Force*

Melakukan *login* dengan mencoba segala kemungkinan dan kombinasi dari *username* dan *password* hingga ditemukan kombinasi yang benar. Serangan *Brute Force* dilakukan untuk membuktikan teori dari kelemahan protokol SSH yang telah disebutkan pada Landasa Teori. *Tool* yang digunakan adalah **Hydra**.

- Akses sumber daya sistem

Mengakses sumber daya sistem dengan menggunakan informasi yang didapat dari tahapan *penetration testing* yang telah dilakukan. Tujuannya adalah untuk mengetahui *severity* kerentanan yang terpapar pada sistem.

4. Pasca Eksploitasi

Pasca Ekspoitasi meliputi kegiatan sebagai berikut.

- *Creating Backdoors*

Menciptakan pintu belakang pada berbagai bagian dari sistem sebagai jalan pintas untuk masuk kembali ke dalam sistem. *Creatingbackdoors* dapat dilakukan dengan cara membuat *account* palsu, menjadwalkan *batch job*, mengubah *startup file*, memasang servis kendali jarak jauh dan *monitoring tool* atau menggantikan aplikasi pada sistem dengan *trojan.Tools* yang digunakan adalah ***Metasploit***.

- *Covering Tracks*

Menutup jejak dan menghilangkan bekas-bekas yang timbul selama tahapan *penetration testing* dilakukan. *Covering tracks* meliputi pembersihan *network log* dan penggunaan *hide tools* seperti berbagai macam *rootkit* dan *file streaming.Tools* yang digunakan adalah ***stealth zapper***.

5. Pelaporan

Mendokumentasikan segala kegiatan pengujian yang kemudian dikemas ke dalam laporan. Laporan juga berisi tentang *highlight* kelemahan-kelemahan sistem beserta solusinya.

4. Implementasi Penetration Testing

4.1. Vulnerability Scanning

Nessus 4.4.1 menggunakan antar muka berbasis *website* dan diakses melalui *localhost*.*Scanning* dijalankan dengan input ip 192.168.9.10 sebagai *scan target*. *Scanning* akan menjangkau pada semua *service* yang tersedia pada *scan target*. Pada penelitian kali ini, analisis difokuskan pada *service* FTP dan SSH.

Berikut ini adalah hasil dari *vulnerability scanning* dengan Nessus.

Port	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	icmp	general	1	0	0	1	0
21	tcp	ftp	5	0	0	3	2
22	tcp	ssh	6	0	0	4	2
53	tcp	dns	3	0	0	1	2
53	udp	dns	4	0	0	4	0

Gambar 3. Detail Kerentanan

Gambar 3 menunjukkan kerentanan yang ditemukan pada setiap *port*, protokol, dan *service* yang aktif. Karena penelitian ini berfokus pada *service* FTP dan SSH, maka pembahasan dan analisis juga berfokus pada kerentanan yang ditemukan pada kedua *service* tersebut. Dari gambar dapat dilihat bahwa telah ditemukan 3 kerentanan tingkat rendah pada FTP dan 4 kerentanan tingkat rendah pada SSH. Daftar dari setiap kerentanan tersebut dapat dilihat dengan melakukan *left mouse click* pada setiap baris.

Plugin ID	Name	Port	Severity
22964	Service Detection	ftp (21/tcp)	Low
10092	FTP Server Detection	ftp (21/tcp)	Low
34324	FTP Supports Clear Text Authentication	ftp (21/tcp)	Low

Gambar 4. Kerentanan Service FTP

Plugin ID	Name	Port	Severity
22964	Service Detection	ssh (22/tcp)	Low
10267	SSH Server Type and Version Information	ssh (22/tcp)	Low
10881	SSH Protocol Versions Supported	ssh (22/tcp)	Low
39520	Backported Security Patch Detection (SSH)	ssh (22/tcp)	Low

Gambar 5. Kerentanan Service SSH

Gambar 4 dan Gambar 5 adalah *screenshot* hasil *scanning* kerentanan yang ditemukan pada *service* FTP dan SSH. Penjelasan detail dari setiap kerentan dapat dilihat dengan melakukan *left mouse click* pada baris nama kerentanan. Berikut ini penjelasan dari masing-masing kerentanan yang ditemukan.

6. Service Detection.

Jenis service yang sedang berjalan dapat diketahui dari banner service atau error message yang muncul saat mengirimkan HTTP request. Error message tersebut akan mengatakan bahwa service yang tersedia adalah FTP atau SSH. Kerentanan ini tidak beresiko apapun dan dapat diabaikan, sehingga tidak diperlukan solusi untuk memperbaikinya.

7. FTP Server Detection.

Jenis FTP server dan juga perangkat lunaknya dapat diketahui hanya dengan melihat banner yang muncul saat server sedang diakses dari pihak client. Isi dari banner tersebut adalah "220 ProFTPD1.3.2c Server ready". Informasi ini dapat digunakan untuk menentukan langkah-langkah penyerangan yang menyesuaikan ciri khas dan kelemahan ProFTPD. Dalam kasus ini, hal tersebut dimanfaatkan untuk menentukan teknik pemasangan backdoor.

8. FTP Support Clear Text Authentication.

Username dan *password* dikirimkan dalam bentuk yang tidak terenkripsi. Kerentanan ini dapat dimanfaatkan untuk mengetahui *username* dan *password* dengan melakukan penyadapan pada aliran data.

9. SSH Server Type, Version Information, dan Protocol Supported

Profile perangkat lunak *daemon* SSH dan jenis protokol yang digunakan dapat diketahui dari banner yang muncul ketika *server* diakses. Informasi tersebut ditampilkan dalam banner dengan isi sebagai berikut.

```
SSH version : SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7
SSH supported authentication : publickey,password
```

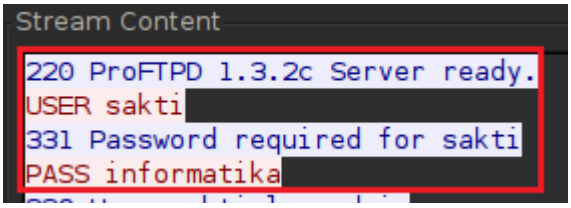
Kerentanan ini tidak memiliki resiko dan dapat diabaikan.

10. Backported security patch detection

Saat pemindaian kerentanan berlangsung, telah terdeteksi perubahan pada *security patch*, namun tidak terdapat informasi atau perubahan pada versi perangkat lunak. Hasil dari pemindaian ini hanya bersifat sebagai informasi dan tidak menyebabkan masalah keamanan.

4.2. Sniffing

*Sniffing*dilakukan dengan perangkat lunak *Wireshark* sebagai *data packet sniffer*. *Sniffing* akan dilakukan dua kali yaitu pada *service* FTP *over* SSH dan pada *service* FTP standar. Tujuannya adalah untuk membuktikan teori tentang kelemahan FTP pada transmisi data yang tidak terenkripsi dengan melihat perbandingan dari hasil yang didapat. Berikut ini adalah hasil yang didapatkan.



Gambar 6. Hasil Sniffing pada FTP Standar

Gambar 7. Hasil sniffing pada FTP dengan SSH

Gambar 6 dan Gambar 7 menunjukkan isi dari salah satu paket data yang berhasil disadap melalui kegiatan *sniffing*. Dapat dilihat bahwa isi dari paket data pada transmisi FTP dikirimkan dalam bentuk *plain text* sedangkan pada transmisi FTP dengan SSH, pesan dalam paket data dikirimkan dalam bentuk *chiper text* yang tidak bisa dibaca.

4.3. Brute Force Attack

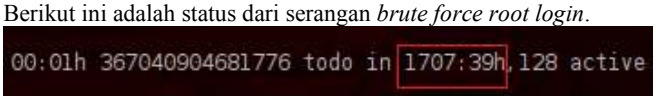
Serangan *brute force*akan dilakukan dengan *tool Hydra*dan dibagi menjadi dua tahap, yaitu serangan pada *root login* dan pada *user login*. Serangan akan dimulai dengan *root login* terlebih dahulu.

11. Root login brute force attack.

Hydra dapat langsung diinisiasikan sebagai berikut untuk memulai serangan *brute force*.

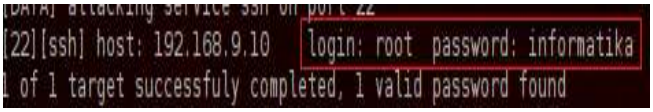
```
hydra -l root -p -x 11:11:a -e nsr -v -V -t 128 -s
22 192.168.9.10 ssh
```

Karena keterbasan *memory* pada komputer penyerang, serangan dilakukan dengan langsung menargetkan *profilepassword root login* yaitu *informatika* yang terdiri dari 11 karakter dan huruf kecil.Inisiasi *hydra* dengan parameter diatas akan menampilkan status dari serangan *brute force* yang sedang berlangsung. Berikut ini adalah status dari serangan *brute force root login*.



Gambar 8. Status Brute Force Root Login

Gambar 8 menunjukkan bahwa dibutuhkan 1707 jam dan 40 menit untuk mencoba semua kemungkinan yang ada. Namun, pada penelitian kali ini, *password* dari *root login* dapat ditemukan dalam waktu kurang dari 2 menit. Berikut ini adalah *screenshot* dari hasil *brute force root login*.



Gambar 9. Hasil Brute Force Root Login

Gambar 9 menunjukkan bahwa Hydra berhasil menemukan 1 *valid password* yang digunakan untuk *root* login. Hasil di atas menunjukkan bahwa teori tentang kelemahan SSH pada *brute force* akses langsung *root* terbukti benar.

12. User login brute force attack

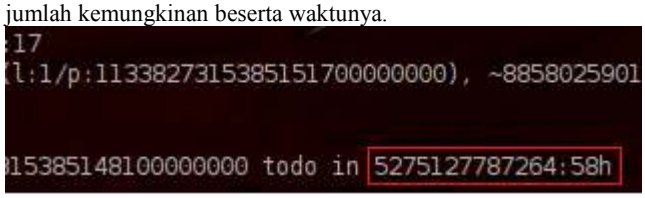
Selain untuk menebak *password* pada *rootlogin* SSH, serangan *brute force*juga akan digunakan untuk menebak kombinasi *username* dan *password* untuk akun pengguna SSH yang lain. Berikut ini adalah inisiasi beserta parameter yang digunakan.

```
hydra -l -x 6:6:a -p -x 11:11:a -v -V -t 128 -s 22
192.168.9.10 ssh
```

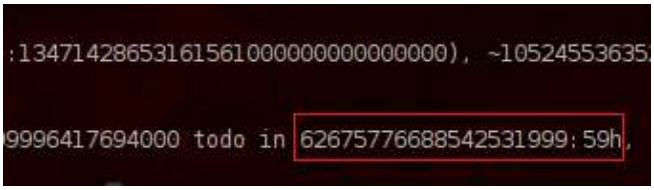
```
hydra -l -x 11:11:a -p -x 11:11:a -v -V -t 128 -s
22 192.168.9.10 ssh
```

Karena keterbatasan *memory*, serangan *brute force*pada dua akun yang ada akan dilakukan secara terpisah dengan langsung menargetkan pada profile akun pengguna beserta passwordnya, yaitu *userftp* dan *informatika*.

Pada *brute force user login*, Hydra tidak hanya mencari *valid password*, namun juga mencari *valid username* yang sesuai dengan *valid password.* Hal ini menyebabkan jumlah kemungkinan untuk percobaan *login* meningkat drastis dibandingkan dengan *brute force* pada *root login*. Berikut ini adalah *screenshot* dari masing-masing status *brute force* untuk jumlah kemungkinan beserta waktunya.



Gambar 10. Userftp brute force status



Gambar 11. Informatika brute force status

Gambar 10 dan Gambar 11 menunjukkan bahwa waktu yang dibutuhkan untuk mencoba semua kemungkinan dari kombinasi *username* dan *password* sangatlah lama. Dalam penelitian ini, komputer yang digunakan tidak cukup cepat ketika melakukan percobaan untuk setiap kombinasi. Walaupun proses serangan *brute force* tidak memerlukan percobaan pada semua kemungkinan yang ada untuk menemukan pasangan *username* dan *password* yang tepat, waktu yang dibutuhkan untuk menemukan pasangan yang tepat masih terlalu lama dan tidak mungkin untuk ditunggu. Oleh karena itu pada penelitian kali ini, teori kelemahan SSH untuk serangan *brute force*pada akun pengguna tidak dapat dibuktikan. Namun secara teori, hal tersebut dapat dilakukan.

4.4. Akses Sumber Daya Sistem

Akses ke sumber daya sistem dilakukan dengan menggunakan informasi yang didapat dari proses *penetration testing* yang telah dilakukan. Dalam penelitian kali ini, proses *sniffing* dan serangan *brute force* menghasilkan informasi tentang *login* FTP, *login remote access,* dan *password* untuk *root login*.

Informasi-informasi tersebut cukup untuk melakukan eksploitasi pada sistem. Akses *user login* (FTP dan *remote*) memberikan hak akses penuh kepada penyerang terhadap sumber daya sistem, dan bahkan dengan *root login,* pihak penyerang mampu untuk memanipulasi sistem. Dalam penelitian ini, akses *root login* digunakan untuk memasang *backdoor* dan melakukan *covering tracks*.
Melihat pada dampak dari kelemahan yang ditemukan, dapat disimpulkan bahwa kelemahan yang paling fatal terletak pada *root compromised* dimana akses sebagai root bisa didapatkan oleh pihak penyerang.

4.5. Creating Backdoor

Teknik dalam menciptakan *backdoor* tidak terbatas. Berbagai macam upaya sah untuk dilakukan untuk mencapai tujuan utama dari menciptakan *backdoor*. Dalam penelitian kali ini, *creating backdoor* dilakukan dengan memanfaat kelemahan dari Proftpd. Pada tanggal 28 Oktober 2010, *server* utama untuk pendistribusian proyek Proftpd telah ditembus oleh pihak yang tidak dikenal. Penyerang tersebut menggantikan *source file* Proftpd versi terbaru saat itu yaitu 1.3.3c dengan versi yang mengandung *backdoor*. Berita secara lengkapnya terdapat pada halaman https://forums.
proftpd.org/smf/index.php?topic=5206.0[11]
Pemasangan *backdoor* dalam penelitian kali ini akan menggunakan *backdoor* khusus untuk Proftpd 1.3.3c.
Source code untuk *backdoor* tersebut telah disebarluaskan untuk kepentingan pengembangan keamanan. Peneliti mendapatkan *source code* tersebut dari buletinilmiah **Actu Sectu** edisi 27 terbitan XMCO [8].*Source code* tersebut kemudian disimpan dengan nama *backdoor.patch*. Berikut ini adalah langkah-langkah yang digunakan dalam *creating backdoor*.

13. Memastikan *server* berjalan dengan Proftpd 1.3.3c

Backdoor yang akan dipasang khusus hanya dapat berfungsi di Proftpd 1.3.3c. Jika *server* tidak berjalan dengan Proftpd versi tersebut, maka versi yang ada harus digantikan dengan 1.3.3c namun tetap menggunakan konfigurasi yang sama dengan yang sebelumnya. Tujuannya adalah agar pihak *administrator sistem* tidak mengetahui jika *daemonserver* telah digantikan. Pada penelitian ini, *server* berjalan dengan Profrpd 1.3.2c sehingga *upgrade* versi perlu dilakukan.

14. Mengunduh Proftpd 1.3.3c

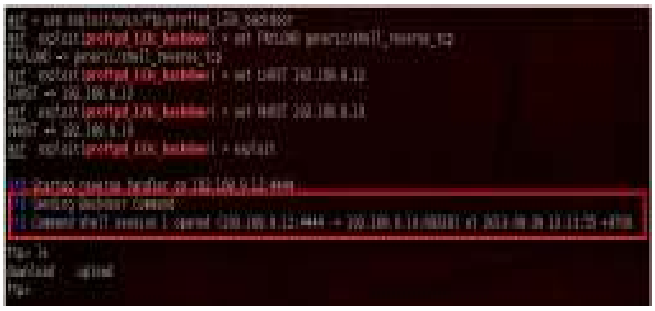
Proftpd 1.3.3c perlu diunduh secara manual untuk dapat dimodifikasi karena fungsi *apt-get install* pada *server* akan mengunduh dan meng-*installproftpd* secara otomatis.

15. Menambahkan *backdoor* ke dalam *source* Proftpd133c

Versi original dari Proftpd 1.3.3c perlu dimodifikasi menjadi Proftpd 1.3.3c *backdoored version*. Modifikasi dilakukan dengan menambah *backdoor.patch* ke dalam *source file*Proftpd 1.3.3c.

16. Install Proftpd 1.3.3c *backdooredversion* ke dalam *server*.

Langkah terakhir adalah menggantikan Proftpd yang ada sebelumnya, yaitu versi 1.3.2c dengan 1.3.3c *backdoored version*. Caranya adalah dengan mengunggah *package* Proftpd 1.3.3c *backdoored version* ke dalam *server* lalu kemudian melakukan instalasi.
Setelah *backdoor* selesai dipasang, *backdoor* tersebut akan aktif secara otomatis ketika *service* FTP diaktifkan. *Backdoor* dapat diakses dengan menggunakan *tool Metasploit*. Berikut ini adalah *screenshot* dari hasil pengaksesan *backdoor* tersebut.



Gambar 12. Akses Backdoor dengan Metasploit

Pada Gambar 12 dapat dilihat bahwa akses *backdoor* pada FTP *server* berhasil dibuka. FTP *server* dapat diakses sepenuhnya dengan menggunakan *command* untuk FTP *operation*. Dari hasil tersebut dapat disimpulkan bahwa pemasangan *backdoor* langsung pada FTP dapat menghindari mekanisme keamanan SSH. Sebab *backdoor* pada FTP dapat diakses langsung tanpa melalui autentikasi.

4.6. Covering Tracks

Semua aktifitas penyerangan yang telah disimulasikan dalam *penetration testing* akan meninggalkan jejak pada sistem. Jejak tersebut diantara lain : *system log files*, *foreign files*, dan *command lines history*. Berikut ini adalah penjabaran dari kegiatan *covering tracks* yang dikategorikan sesuai dengan jenis jejak yang telah disebutkan diatas.

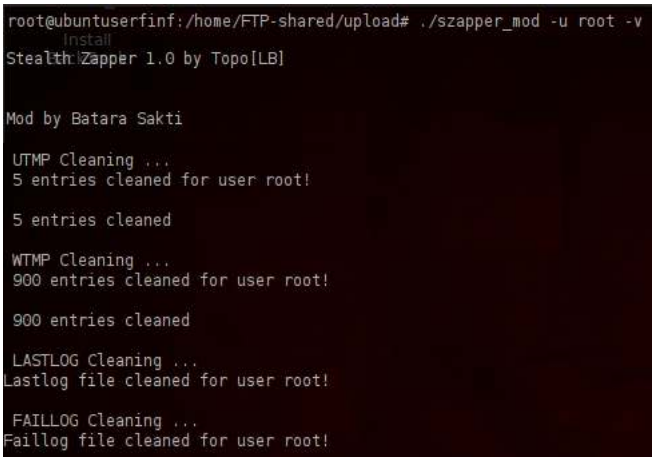
4.6.1. Menghapus Sistem Log Entries

Aktifitas yang terjadi pada *server* akan tercatat dan tersimpan dalam log sistem, termasuk segala aktifitas serangan yang pada penelitian kali ini disimulasikan dengan *penetration testing*. Pada *server* yang menggunakan sistem operasi *berbasis linux (*UNIX *like)*, log sistem secara *default* tersimpan pada *directory var/log*. Tahapan *penetration testing* yang telah dilakukan sampai saat ini adalah *vulnerability scanning*, *sniffing*, serangan *brute force*, eksploitasi, dan *creating backdoor*. Aktifitas-aktifitas tersebut akan meninggalkan jejak entries pada sistem log berikut ini.

- */var/log/messages*
- */var/log/proftpd/*
- */var/log/auth.*
- */var/log/faillog*
- */var/log/lastlog*
- */var/log/wtmp*
- */var/log/utmp*

Entries pada *logmessage, proftpd log, dan auth* dapat dihapus dengan menggunakan *text editor*. Pihak penyerang cukup menghapus log *entries* yang berkaitan dengan kegiatan penyerangan.
Sedangkan untuk menghapus pesan log pada *lastlog*, *faillog*, *wtmp*, dan *utmp* dibutuhkan cara khusus karena pesan log pada keempat sistem log tersebut tertulis dalam format *binary* dan tidak bisa ditampilkan ke dalam *plain text*. Sehingga, *record* pada keempat sistem log tersebut tidak dapat di-*edit* menggunakan teks editor. *Record* pada keempat sistem log tersebut hanya dapat diedit dengan menggunakan *tools* seperti *wtmped*, *marry*, *cloak*, *logwedit*, *wzap*, dan*zapper*.
Dalam penelitian kali ini, peneliti menggunakan program *stealth zapper* untuk menghapus pesan log pada *lastlog*, *faillog*, *wtmp*,dan *utmp*. *Stealth zapper* adalah program yang didesain khusus untuk membersihkan catatan log yang tidak dapat ditampilkan ke dalam bentuk *plain text*.

Peneliti mendapatkan *source codestealth zapper* dari halaman http://hostcode.sourceforge.net/dl/530 [9] dengan original *creator* yang tidak diketahui. Dalam penggunaanya, penulis sedikit memodifikasi *source code stealth zapper* agar dapat berfungsi terhadap *faillog*[10]. Sebab secara original, *stealth zapper* hanya ditujukan untuk menghapus *entries* dari *wtmp*, *umtp*, dan *lastlog*.
Dalam penggunaanya, *stealth zapper* yang sudah dimodifikasi di-*upload* kedalam *server* dan kemudian dikompilasi untuk menghasilkan *executable file*.Berikut ini adalah *screenshot* dari hasil penggunaan *stealth zapper*untuk menghapus jejak pada *root login*.



Gambar 13. Output Clean User Root

Stealth zapper akan menghapus semua *entries* untuk *username* tertentu pada hari dimana program tersebut dijalankan sebagaimana ditunjukkan oleh Gambar 13. Hal ini merupakan kelemahan dari penggunaan program tersebut. Penyerangan terhadap sistem dapat terdeteksi jika *administrator* sistem mengetahui kapan saja suatu *username* login ke dalam sistem. Sebab, *stealth zapper* akan menghapus semua *entries* dengan *username* yang sama. Sebagai contoh, *stealth zapper* akan menghapus *entries* root login dari*administrator* dan *root login* dari *brute force attack*yang terjadi pada hari yang sama.

4.7. Menghapus Foreign Files

Selama proses serangan berlangsung, pihak penyerang sering kali menambahkan file-fileke dalam *server* sistem untuk mendukung kegiatan-kegiatan yang dilakukan. Dalam penelitan kali ini, selama kegiatan *penetration testing* berlangsung, peneliti juga menambahkanbeberapa *file*untuk kepentingan tahapan *penetration testing*. Berikut ini adalah daftar *foreign files* yang telah ditambahkanke dalam *server*.

- proftpd-1.3.3c.tar.gz
- backdoor.patch
- *directory*proftpd-133c
- szapper_mod.c
- szapper_mod *executable file*
- faillog.h

File-file di atas akan dihapus melalui *remote access* dengan *privilege* sebagai *root*.

4.8. Menghapus Command Lines History

Seorang penyerang tentu akan banyak menggunakan *commands* pada saat melakukan proses serangan. Pada sistem operasi *berbasis linux (*UNIX *like)*, semua *command lines* yang pernah diinputkan oleh *user* akan tersimpan pada file *.bash_history*.
File .bash_history dapat dibuka dan di-*edit*menggunakan *text editor* dengan *previlageroot*. Pihak penyerang cukup menghapus *command lines* yang terkait dengan serangan yang telah dilakukan.

4.9. Pelaporan

Laporan dibuat setelah kegiatan *penetration testing* selesai dilakukan. Laporan berisi tentang hasil dan pembahasan, serta *countermeasure* (penanggulangan)dari setiap tahapan *penetration testing*. *Countermeasure* yang disebutkan pada laporan *penetration testing* dapat dijadikan dasar untuk pemberian saran dalam penelitian kali ini.

5. Kesimpulan dan Saran

5.1. Kesimpulan

Hasil penelitian ini membuktikan bahwa implementasi SSH secara standart tidak cukup efektif dalam mengamankan transmisi FTP. SSH secara standar terbukti hanya dapat melindungi transmisi FTP dari penyadapan pada fase *sniffing*.
Teori tentang kelemahan SSH pada *brute force rootlogin* terbukti benar dengan hasil yang didapat pada fase *brute force* dengan Hydra. Password dari *root* dapat ditemukan dalam waktu kurang dari 2 menit walaupun estimasi total waktu yang dibutuhkan untuk mencoba semua kemungkinan *password* berjumlah 1707 jam dan 40 menit. Sedangkan teori kelemahan SSH untuk *brute force user login* tidak dapat dibuktikan pada penelitian ini. Sebab, keterbatasan *memory* pada komputer yang digunakan sebagai penyerang menghasilkan estimasi total waktu yang mencapai angka milyaran jam dan tidak mungkin untuk ditunggu.

5.2. Saran

Langkah-langkah pencegahan (*countermeasure*) untuk setiap tahapan *penetration testing* dapat dijadikan sebagai parameter keamanan untuk membangun sebuah sistem berbasis SSH. Tingkat keamanan yang dihasilkan akan semakin tinggi jika langkah-langkah pencegahan tersebut dikombinasikan dengan piranti keamanan lain untuk menciptakan keamanan berlapis. Secara garis besar, berikut ini adalah saran-saran yang sebaiknya dilakukan untuk mengoptimalkan fungsi pada SSH dalam mengamankan FTP.

- Menghilangkan *banner start up* program untuk menjaga kerahasiaan *profile service*.
- *Disable root login* pada *service* SSH untuk mencegah serangan *brute force root login*.
- Membatasi waktu percobaan *login* untuk memperlambat *serangan brute force*.
- Menggunakan *password* dengan tingkat kekuatan tinggi, yaitu dengan panjang minimal 6 karakter dan terdiri dari huruf dan angka.
- Melakukan verifikasi dengan *digital signature* pada piranti lunak yang digunakan untuk memastikan keaslian dan mencegah beredarnya *malicious program* pada sistem.
- Melakukan *update* piranti lunak secara rutin.
- Menkonfigurasikan ulang sistem log agar tersimpan pada direktori dan dengan nama yang berbeda, untuk menghambat proses *covering tracks*.
- Membuat backup sistem log secara rutin dengan *scheduller program*.
- Menambahkanpiranti *security sentry* untuk memonitor sistem log.
- Menambahkan jenis sistem log yang aktif.
- Mengimplementasi IDS dan *port sentry* untuk menambahkan informasi ekstra pada sistem log dan sebagai peringatan dini jika sistem mendapat serangan.

6. Referensi

[1] RICOH. 2010. Network Security White Paper.

[2] Sanford, M, Woodraska, S, Xu, D. 2011. Security Analysis of FileZilla Server Using Threat Models. National Center for the Protection of the Financial Infrastructure. Dakota State University

[3] Ali, Q dan Alabady, S. 2010. Applying Penetration Tests on a Highly Secured Cooperative Network. International Arab Journal of e-Technology, vol 1, no 3, January 2010. Computer Engineering Department, University of Mosul, Iraq

[4] Timothy K. Shih. 2010. Advance Penetration Testing Methodology for VPN. Journal of Security Engineering. Department of Computer Science and Information Engineering, Tamkang University, Taiwan.

[5] Daniel J. Barrett, Richard Silverman. 2001. SSH, The Secure Shell: The Definitive Guide. O'Reilly.

[6] Boss, E. 2012. Evaluating Implementations of SSH: A Model-Based Testing Approach. Bachelor Thesis. March 16, 2012. Cambridge University.

[7] EC-Council. Ethical Hacking and Countermeasures. http://www.eccouncil.org.

[8] Keyzer, J. 2012. Attaque Proftpd. Actu Sectu. Volume 27. Halaman 50. Penerbit XMCO. Paris, Perancis, January 2012.

[9] Hostcode. Source Code Stealth Zapper. http://hostcode.sourceforge.net/dl/530

[10] Opensource, Apple, Source Code Faillog.h. http://www/opensource.apple.com/source/pam/modules/pam_tally/faillog.h

[11] Proftpd. ftp.proftpd.org compromised. https://forums.proftpd.org/smf/index.php? topic=5206.0

[12] Chaudhary, H. 2011. Assuring Data Security through Penetration Testing. August, 2011, USA: QA Info Tech.

[13] Schuster, F. and Holz, T. 2013. Towards Reducing the Attack Surface of Software Backdoors. Horst Gortz Institute for IT-Security, Ruhr-University Bochum, Germany.

[14] Waksman, A. and Sethumadhavan, S. 2011. Silencing Hardware Backdoors. IEEE Symposium on Security and Privacy.

[15] Sang Oh, Spencer Kam, and Takagi, A. 2006. Chapter 25. File Transfer and Access (FTP, TFTP, NFS).

[16] Ruswanda, M. M., Ibnugraha, P.J., dan Zani, T. 2011. Implementasi FTP Server dengan Secure Sockets Layer dan Secure Shell untuk Keamanan Transfer Data. Program Studi Teknik Komputer, Politeknik Telkom Btandung, 2011.

[17] David Kennedy, Jim O'Gorman, Devon Kearns, and Mati Aharoni. 2011. Metasploit The Penetration Tester's Guide. Cruzenaldo