VOL. 16 NO. 3 SEPTEMBER 2015

ISSN : 1411-3201

Jurnal Ilmiah DASI

DATA MANAJEMEN DAN TEKNOLOGI INFORMASI

STMIK AMIKOM
YOGYAKARTA

VOL. 16 NO. 3 SEPTEMBER 2015
JURNAL ILMIAH
Data Manajemen Dan Teknologi Informasi

Terbit empat kali setahun pada bulan Maret, Juni, September dan Desember berisi artikel hasil penelitian dan kajian analitis kritis di dalam bidang manajemen informatika dan teknologi informatika. ISSN 1411-3201, diterbitkan pertama kali pada tahun 2000.

PENANGGUNG JAWAB :
Ketua STMIK AMIKOM Yogyakarta, Prof. Dr. M. Suyanto, M.M.

ALAMAT PENYUNTING & TATA USAHA
STMIK AMIKOM Yogyakarta, Jl. Ring Road Utara Condong Catur Yogyakarta, Telp. (0274) 884201 Fax. (0274) 884208, Email : jurnal@amikom.ac.id

BERLANGGANAN
Langganan dapat dilakukan dengan pemesanan untuk minimal 4 edisi (1 tahun) pulau jawa Rp. 50.000 x 4 = Rp. 200.000,00 untuk luar jawa ditambah ongkos kirim.

DAFTAR ISI

PERLINDUNGAN DATA TERHADAP SERANGAN MENGGUNAKAN METODA TEBAKAN PADA SISTEM OPERASI LINUX

Akhmad Dahlan

Teknik Informatika STMIK AMIKOM Yogyakarta
email: alland@amikom.ac.id

Abstract

In a security system that allows users to use their password, users usually choose a password that is easy to remember and are usually easy to guess the password. This weakness is often the case in almost all of the existing system. Rather than forcing the user choose a good password, which means it's difficult to remember or long terralu.

Li Gong propose a solution that will preserve user comfort while delivering high security guarantees. The basic idea proposed is to ensure that data can be stolen by the attacker can not be predicted and tested in a way to guess. This can be achieved by using a random number generator, ie by performing XOR operations on each message contains confidential information with a value generated by a random number generator.

By utilizing a random number generator is expected that each message sent at different times have different shapes. This way every effort to keep it guesses attack more difficult, where the attackers have to guess on the random number is used as to disguise the message, before making a guess on the key of the message.

In this study will be discussed about the methods above in more detail and try. To be implemented on the Linux operating system. By utilizing the technique as mentioned above, studied authenticity protocol used in Linux to test the authenticity of the wearer, in order to create a program that utilizes the above method to exchange messages to verify the authenticity of the user.

***Keywords*:**
Information Systems, Linux, Kriptogragrafi

Pendahuluan

Pada suatu sistem operasi yang menerapkan sistem keamanan, biasanya diterapkan sistem keamanan yang memanfaatkan fungsi hash satu arah *f()* (fungsi hash disini dapat juga dianggap sebagai suatu fungsi pengacak), untuk menghasilkan suatu nilai dari kata sandi yang dipergunakan pemakai. Sebagai ilustrasi digambarkan sebagai berikut jika mempunyai kata sandi *p*, dan sistem menerapkan suatu fungsi *f()* untuk menghasilkan nilai, *f(p)* yang akan disimpan guna mengenali pemakai tersebut. Penyerang akan menerapkan prosedur yang sama dengan pembentukan nilai tersebut. Penyerang memasukkan kandidat kata sandi *p*' dan menerapkan fungsi hashing *h()*. Jika nilai *h(p')* sama dengan nilai *h(p)* maka dianggap bahwa *p' = p*, jika tidak maka penyerang akan mencoba dengan kandidat kata sandi berikutnya dari daftar yang dimilikinya.

Melihat hal tersebut tampak bahwa kelemahan sistem, terletak pada pemilihan kata sandi yang dipergunakan pemakai. Jika pemakai mempergunakan kata sandi yang berkaitan dengan dirinya dan penyerang mengenai pemakai, maka penyerang dapat memanfaatkan hal-hal yang berkaitan dengan pemakai sebagai kandidat kata sandi. Kelemahan ini hanya dapat ditutupi jika pemakai sistem dipaksa untuk mempergunakan kata sandi yang tidak berkaitan dengan dirinya.

Hal tersebut sulit untuk. dilaksanakan karena kata sandi yang tidak berkaitan dengan pemakainya akan mudah dilupakan. Pada Penelitian ini akan digambarkan suatu metode perlindungan pada rahasia yang dimiliki pemakai, dengan menerima kelemahan sebagai suatu kenyataan yang tidak dapat diubah.

Tinjauan Pustaka

Teknik Dasar Perlindungan dengan Model Li Gong

Dalam naskahnya, Li Gong memberikan teknik dasar dari perlindungannya dengan membe-rikan ilustrasi sebagai berikut.. A membangkitkan sebuah bilangan random *n* dan mengacakya dengan kunci bersama *k*. B membangun kembali pesan yang dikirim A tersebut dan menerapkan fungsi yang telah disepakati *f()* untuk menghasilkan *f(n)*. Kedua pesan tersebut jika berdiri sendiri tidak akan mengandung informasi yang dapat diprediksi atau dipakai sebagai acuan dari penyerangan tebakan, karena *n* dan *f(n)* adalah bilangan random, yang dapat berisi bilangan apa saja. Akan tetapi jika kedua pesan tersebut dapat di sadap oleh penyerang, serta penyerang cukup mempunyai pengetahuan tentang pertukaran pesan yang terjadi, maka penyerangan secara tebakan memungkinkan untuk dilakukan. Karena fungsi *f()* tidak dianggap sebagai rahasia maka penyerang dapat menebak nilai *n'*, menghitung *f(n')*. Setelah itu penyerang dapat membandingkan hasil tebakamya (*n'* dan *f(n')*) dengan pesan yang telah berhasil dia sadap dan telah dibentuk kembali (*n* dan *f(n)*), jika

kedua pesan tersebut sama maka kemungkinan kunci telah ditemukan oleh penyerang.[1]

Pertukaran tersebut dapat diperbaiki dengan menerapkan dua buah kunci yang berbeda. Lihat kembali penggambaran pertukaran pada gambar dengan anggapan bahwa kunci k dari tiap pesan adalah kunci yang berbeda. Dengan melihat pertukaran pesan tersebut, walaupun penyerang dapat menyadap semua pesan tersebut. Penyerang akan semakin sulit memprediksi tebakan kata sandi yang dipakai. Karena pesan pertama dapat berupa nilai sembarang dari pesan kedua juga bernilai sambarang yang keduanya diacak dengan kunci yang berbeda. Akan tetapi kelemahan cara ini adalah pemakai harus mengingat dua buah kunci yang berbeda. Namun demikian dapat saja anggap B adalah sebuah komputer yang dapat mengingat kedua kunci tersebut atau salah satu kunci tersebut adalah sebuah kunci publik.[1][2]

Pemakaian kunci publik dalam pertukaran tersebut dapat membuka kemungkinan baru bagi penyerang untuk melancarkan serangannya. Anggaplah k1 adalah kunci publik, sedangkan k2 adalah kunci pribadi milik pemakai A, sedangkan B adalah sebuah komputer yang mengetahui k1 dan k2. Penyerang akan melakukan tebakan terhadap kunci k2 dengan memakai kunci tebakan k2' membentuk kembali pesan Ek2[*f(n)*], untuk menda-patkan *n'*. Setelah itu dia akan menguji kebenaran tebakannya tersebut dengan membandingkan antara Ek1[*n'*] dengan Ek1[*n*] jika keduanya sama, maka hal ini berarti k2 = k2' sehingga tebakan dianggap benar.[2]

Metode penyerangan diatas dapat diatasi dengan menambahkan sebuah bilangan random yang cukup besar *(confounder),* ke dalam data pesan yang mempunyai kemungkinan akan dijadi-kan acuan kebenaran tebakan, terutama sekali pada pesan yang memanfaatkan kunci publik. Fungsi *confounder* ini tidak ada lain hanya untuk menyem-bunyikan isi pesan dari penyerang, sehingga dapat diabaikan oleh penerima pesan tersebut.[3]

1. ***A à B: Ek1[c, n]***
2. ***B à A: Ek2[f(n)]***

Dalam pertukaran pesan tersebut *c* adalah *confounder*. Sekarang penyerangan akan semakin sulit karena penyerang harus menebak k1, k2, dan *c* untuk membentuk kembali pesan pertama. Tebakan terhadap c tidak akan berguna apabila k1 tidak diketahui oleh penyerang. [4]

Alternatif yang lain adalah dengan membuat fungsi *host f()* sedemikian rupa sehingga walaupun telah diberikan *f(n)* penyerang tetap tidak dapat menentukan nilai dari menentukan nilai dari *n*. Resiko yang dihadapi dari alternatif adalah kemungkinan *n* tidak benar-benar berisi bilangan acak, ada kemungkinan pula *n* diisi oleh penanda waktu *(timestamp)* sehingga data *n* mengandung suatu data yang dapat dikenali oleh penyerang *(recognizable data).* Sehingga penyerang tetap dapat memanfaatkan data tersebut sebagai acuan untuk menentukan kebenaran tebakannya. Untuk mena-ngani hal tersebut dapat dilakukan operasi bit wise *exclusive or,* yang dinotasikan sebagai Å , sehingga dapat menggantikan *f(n)* dengan (*c* Å *f(n)*) dalam pesan kedua. Namun penjagaan agar penyerang benar-benar tidak dapat menebak k2 metode diatas perlu juga dilakukan pada pesan ke 1. Sehingga keseluruhan pertukaran pesan yang terjadi adalah sebagai berikut :[5]

1. **A à B : Ek1[c1,c2,*n*]**
2. **B à A : Ek2[c2 Å *f(n)*]**

Protokol Needham and Schroeder

Ciri utama dari protokol pengujian keaslian yang dideskripsikan oleh Needham-Schroeder adalah adanya pihak ketiga yang dianggap dapat dipercaya sebagai penguji keaslian pemakai sebelum pemakai tersebut dapat meminta pelayanan dari *server* yang ditujunya. Dalam protokol Needham-Schroeder yang digunakan dalam sistem kerberos, informasi rahasia yang digunakan verifikasi tidak pernah ditransinisikan dengan jelas dan tidak pernah dikirimkan kepada penerima. Malahan, "Server Keotentikan membuat sekumpulan "sesi rahasia" (dihasilkan dari pengetahuan dari rahasia-rahasia dari pengirim dan penerima dalam proses pertukaraan itu) yang akan digunakan oleh pengirim dan penerima untuk pembuktian keaslian dalam interaksi tersebut. Informasi ini hanya boleh diberikan pada interaksi sesi tersebut, dan dapat diberi penanda waktu(timestamp) untuk mencegah pemutar ulangan pesan.[5]

Setiap interaksi baru (termasuk interaksi antara klien dan server) memerlukan kunci sesi yang baru. Algoritma dasarnya adalah seperti di-bawah ini: Setiap peserta dalam proses pembuktian keaslian memiliki kunci rahasia pengacak yang hanya diketahui oleh dirinya sendiri dan server keotentikan. Kunci ini digunakan hanya untuk berkomunikasi dengan server keotentikan, yang di dapat dipercaya dan aman (dianggap tidak akan menyalah gunakan atau membuka kunci tersebut).[3]

Interaksi antara klien dengan server dimulai dengan permintaan klien kepada server keotentikan akan sebuah "kunci enkripsi sesi" dan sebuah "tiket keotentikan" yang akan digunakan dalam interaksi klien/server. Kunci sesi akan digunakan dalam pertukaran pesan-pesan antara klien dan server hanya pada saat itu. [3]

Saat menerima pesan balasan , *server* membaca isi tiket. Didalamnya dapat ditemukan kunci sesi, yang dapat digunakan membaca pesan, serta informasi keotentikan yang diletakkan disana oleh server keotentikan sebagai pernyataan bahwa tiket tersebut sah untuk saat dan klien teersebut. Penanda waktu juga dimuat dalam tiket untuk, membatasi umur dari kunci sesi tersebut dapat dianggap sah. Karena kunci-kunci sesi hanya di-ketahui oleh klien dan server yang terlibat dalam sesi tersebut, pengacak konvensional dapat digunakan. Hal ini menguntung-

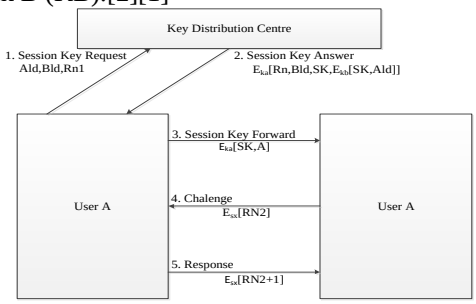
kan karena pengacak konvensional relatif lebih cepat dibanding enkripsi kunci publik. Kerugian dari Needham/Schroeder adalah kebutuhan akan interaksi dengan server keotentikan dan kebutuhan untuk mempercayai server keotentikan.[3]

Arsitektur Protokol Keontentikan Needham-Schroeder

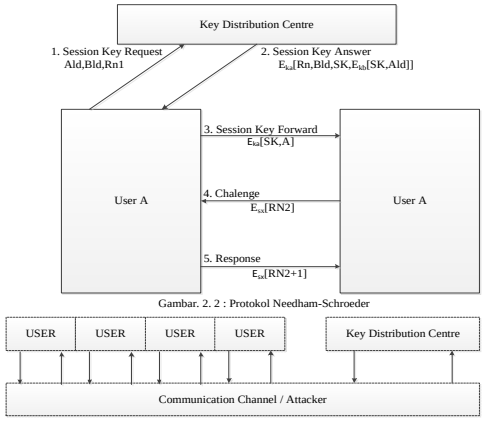
Protokol keotentikan Needham-Schroeder (N-S) menggunakan Pusat pendistribusian kunci (key distribution centre) atau disingkat dengan KDC yang dapat dipercaya dan kunci kriptografi privat yang simetris untuk membentuk hubungan yang aman antar pemakai. [2]

Saat pemakai ingin berkomunikasi dengan yang lain, katakanlah pemakai A dan pemakai B. A pertama-tama mengirim pesan ke KDC meminta kunci sesi baru. Permintaan berisi identifier dari A dan B (Aid, Bid) serta bilangan random (Rn1) yang dihasilkan oleh A. [2]

KDC membalasnya dengan sebuah pesan yang diacak dengan kunci privat milik A (KA). Pesan berisi bilangan random (Rn1), identifier B (Bid), kunci sesi yang baru (SK) dan subpesan (EKB[SK,Aid]) yang diacak dengan kunci privat milik B (KB).[2][1]



Gambar 3. Protokol Needham-Schroeder



Gambar. 2. 2 : Protokol Needham-Schroeder

Gambar 1. level atas model dari protokol Needham-Schroeder

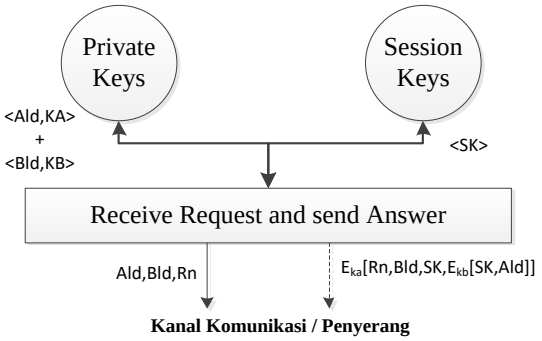
Jika bilangan random dan identifier B cocok dengan permintaan sebelumnya, menerima kunci sesi baru dan meneruskan subpesan kepada B. Ketika B menerima subpesan tersebut, B akan menemukan kuinci sesi (SK) dan identifier A (Aid). B akan menguji pengetahuan A tentang kunci sesi dengan membuat bilangan random (Rn2), mengacaknya dengan kunci sesi dan mengirimkannya kepada A. A kemudian dapat membaca bilangan random dengan kunci sesi. Menambahnya dengan satu, mengacaknya dengan kunci sesi dan mengirimkannya kembali kepada B. Jika B menerima bilangan random yang dengan tepat bertambah (Rn2+1) dan mengacaknya, dia menerima kunci sesi sebagai otentik. Kemudian A dan B percaya bahwa hubungan yang aman dengan kunci sesi yang baru telah tercapai dan telah menguji keaslian masing-masing entitas[1][5]

Entitas dari N-S Protokol

Dalam protokol Needham/Schroeder, tipe entitas adalah pemakai dan KDC. Pemakai dapat berlaku sebagai pemakai A yang menginisialisasi komunikasi atau pemakai B. User dapat bergantian antara A dan B data beberapa kali menjalankan protokol. [5]

Mememodelkan pusat pendistribusian kunci dengan sederhana. (Gambar 2.) KDC menjawab permintaan selalu dengan cara yang sama. Pembedanya hanyalah bahwa ia mempunyai persediaan kunci publik yang tidak akan pernah digunakan untuk yang kedua kalinya.[2]

KDC (Key Distribution Centre)



Gambar 2. Model pusat pendistribustan kunci

Enkripsi Kunci Publik

Dalam enkripsi kunci publik, tiap pemakai, 1, mempunyai kunci publik, Ei, dan kunci pribadi, Di, yang hanya diketahui oleh pemakai i. [4]

Kunci publik dan kunci private adalah kebalikan dan simetris, deism hal jika diberikan pesan m, Ei(Di(m)) = Di(Ei(m)) = m. Untuk mempertahankan privasi, pemakai X akan menerima kunci publik Ey mtuk pemakai Y dan menghitung Ey(m). karena hanya Y yang mengetahui Dy, maka hanya Y saja yang bisa mendekrip. Identifier yang lain dapat ditambahkan pada m sehingga hasil dekripsi dapat diuji kebenarannya.[2]

Tanda tangan digital bekerja dengan cara yang serupa, kecuali pada saat X ingin menandai pesan pada Y, X menggunakan kunci privat miliknya Dx dan menghitung Dx(m). saat menerima, Y menghitimg Ex(Dx(m)) = m. karena hanya X yang mengetahui Dx, hanya X yang dapat menandai pesan tersebut. Enkripsi privat dan tanda tangan digital dapat digabungkan dengan menghitung Ey(Dx(m)),

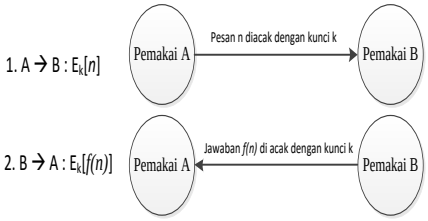
yang akan didekrip dengan Ex(Dy(Ey(Dx(m)))) = m. Pendaftaran kunci publik Ei tidak perlu dibaca dalam kondisi aman, karena Ei diberikan dengan bebas.[4]

Pendaftaran hanya perlu melindungi dari perubahan, karena hal tersebut berarti mengijinkan kecurangan pemberian kunci. Kerugian dari enkripsi kunci publik adalah dalam beberapa hal lebih lambat dibanding enkripsi konvensional karena sifat-sifat algoritma pengacaknya.[4]

Teknik Dasar Perlindungan dari Serangan Secara Tebakan

Teknik dasar dari perlindungm dalam jaringan dapat digambarkan melalui pertukaran pesan sederhana yang biasanya ada dalam beberapa protokol komunikasi. Pesan ini dapat dibagi menjadi 2 bagian yaitu isi pesan dan atribut pesan. Pesan adalah satu nilai atau data yang dikirimkan dari seormg pengirim kepada penerima, sedangkan atribut pesan adalah atribut yang menyertai pesan yang dikirimkan baik berupa suatu metode pengacakan pesan maupun sualu fungsi yarg mentransformasi pesan menjadi nilai tertentu. [2]

Gunakan notasi A □□B : *m* untuk menandakan bahwa A mengirim pesan *m* kepada B. Penjelasan notasi yang dipakai dapat dilihat secara jelas pada lampiran tabel notasi. Dalam hal ini B bisa dianggap sebagai komputer ataupun manusia. Pertukaran pesan tersebut dapat digambarkan dalam gambar 4. berikut[4]



Gambar 4. Pertukaran Pesan Sederhana

A memberikan bilangan random *n* dan mengacaknya dengan kunci *k* yang akan digunakan secara bersama dengan B. B menterjemahkan pesan, dengan menerapkan fungsi yang telah disepakati *f()* untuk menghasilkan *f(n)*, dan B mengacak *f(n)* sebelum mengirimkan kembali kepada A. Fungsi *f()* untuk memastikan bahwa kedua pesan tidak identik dan menyatakan bahwa pesan dikirim adalah baru. Kedua pesan yaitu *n* dan *f(n)* ini tidak akan diketahui penyerang karena keduanya adalah suatu nilai acak, sedangkan fungsi *f()* biasanya bukan merupakan rahasia. Sehingga jika penyerang dapat memperoleh kedua pesan di atas yaitu Ek[*n*] dan Ek[*f(n)*] maka memudahkan penyerang untuk menebak apakah k tersebut. Penyerang akan melakukan tebakan terhadap k, misalnya dengan memberikan k'. Hasil rekonstruksi pesan adalah n' dan *f'(n)*. Penyerang akan melihat hubungan antara n' dengan *f(n)*. Jika *f(n') = f (n)* maka dapat diambil kesimpulan bahwa k = k'. [4]

Dari hal tersebut diketahui bahwa masalah utama dari sistem tersebut pada isi pesan itu sendiri dan kunci yang menyertainya. Secara lebih jelasnya pesan yang dikirim terlalu mudah diprediksi, karena hanya terdiri dari satu bagian yaitu pesan itu sendiri walaupun pesan tersebut sudah diacak. Sedangkan kunci akan merupakan masalah jika kunci tersebut mudah ditebak karena berasal dari pemakai, yang biasanya tidak mempergunakan kata sandi yang tidak terlalu panjang dan merupakan kata-kata yang dipakai setiap hari. [5]

Untuk melindungi sistem tersebut akan dilakukan proses modifikasi pesan dengan menambahkan suatu pesan tambahan (*c*) yang digabungkan dengan pesan asli agar pesan tersebut sulit dikenali baik itu pada pesan *n* maupun *f(n)*. Untuk itulah perlu dilakukan operast XOR(□□) antara pesan asli dengan pesan tambahan, untuk menyamarkan pesan sebenarnya dengan tambahan yang diberikan. [5]

Sehingga pertukaran pesan tersebut dapat dinyatakan sebagai berikut :

1A → B : Ek1[c,n]

2B → A : Ek2[c□□f(n)].

Disini dapat diperoleh kepastian bahwa pengguna adalah pemakai yang sebenarnya, dengan menguji pengetahuan terhadap c. Jika B seorang pemakai yang sah dia akan dapat memisahkan pesan n daripesan pertama dan kemudian akan dapat mengembalikan pesan balasan dengan mencampurkan c dengan hasil fungsi yang disepakati dengan operasi XOR. Teknik dasar inilah yang nantinya akan dipakai dalam menambah keamanan pada protokol keotentikan dalam Linux.[5]

Metode Penelitian

Protokol Keotentikan Standar

Setiap perintah untuk operasi jarak jauh memerlukan proses pengenalan pemakai perintah. Setiap perintah tersebut secara umum memakai protokol keotentikan standar dari Barkeley, jika tidak ada parameter yang menspesifikasikan. Protokol keotentikan tersebut adalah sebagai berikut :[5]

1. **A → S : ID_A | ID_{AS} | ID_S | KataSandi | Cmd | Port_Cli**
2. **S → A : ID_A | Port_serv**

Penjelasan tiap-tiap elemen dapat dilihat pada tabel 3.1. Setiap pemakai menginginkan suatu pelayanan dari *server*, pemakai mengirimkan identitas pemakai dalam sistem *client* (IDA), identitas pemakai dalam sistem *host* (IDAS), identitas *host* (IDS), nomor port untuk berhubungan (Port_Cli), dan pelayanan yang diinginkan (Cmd), serta kata sandi (KataSandi). Semua pesan tersebut dikirimkan secara sebenarnya *(plaintexxt). [5]*

Server akan membawa pesan dari pemakai jika kata sandi (KataSandi) benar. *Sever* juga memberikan nomor port (Port_serv) untuk berhubungan lebih lanjut sesuai dengan pelayanan yang diinginkan. Protokol ini dibuat dengan asumsi bahwa *server* dapat dipercaya.

Protokol ini sangat mengandalkan daftar kata sandi yang disimpan pada daftar yang ada pada

server. Pengujian keotentikan dilakukan dengan membandingkan hasil transformasi kata sandi (Kata Sandi) dengan daftar yang ada pada *server*.

Tabel 1. Elemen pertukaran pesan

A	Sistem yang meminta pelayanan
S	Sistem yang memberikan pelayanan
ID_A	Identitas A pada sistem yang bertindak sebagai Client
ID_{AS}	Identitas A pada sistem Host
ID_S	Identitas Host S
Kata Sandi	Kunci Milik A
Cmd	Perintah atau pelayanan yang diinginkan Client
Port_Cli	Nomor Port Client yang di pakai untuk berhubungan dengan host
Port_serv	Nomor port yang dipakai server untuk berhubungan dengan A

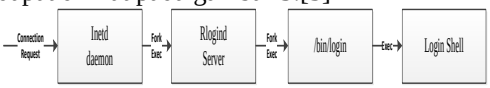
Proses Pelaksanaan Program
Client

Pemakai yang ingin mendapat pelayanan memasukkan data untuk pesan 1 seperti pada protokol keotentikan. Pesan tersebut akan dikirim-kan kepada program yang mengawasi pemakaian port jaringan yaitu inetd, program inilah yang menuliskan pesan kepada jaringan. Setelah itu *Client* akali menunggu balsan dari *server* yang di-hubunginya pada nomor port yang telah dikirimnya,[2]

Jika program inetd, menerima masukan dari jaringan, program tersebut segera mencari program yang menanti data tersebut. Setelah itu data diserahkan pada program yang menantinya.

Server

Jika daemon inetd, menerima masukan dari jaringan, progam tersebut segera mencari *server* yang dapat melayani permintaan itu. Setelah itu data diserahkan pada *server* tersebut. Contoh pelaksanaan permintaan pelayanan pada server rlogind dapat dilihat pada gambar 5.[3]



Gambar 5. langkah yang diperlukan untuk meminta pelayanan rlogind

Jika pelayanan yang diminta membutuhkan kata sandi khusus maka *server* akan mencari kata sandi tersebut pada data yang diterimanya. Jika tersedia, kata sandi dicocokkan dengan data yang dimiliki oleh *server* tersebut. Jika proses otentikasi tersebut berhasil dilalui maka merger akan menyatakan pemakai telah mendaftar pada system *host.*

Setelah proses tersebut selesai program mengirimkan balasan kepada *client* yang menyatakan pelayanan dapat diberikan atau tidak.

Kelemahan protokol keotentikan standar terhadap serangan secara tebakan.

Beberapa kelemahan dapat diperoleh dari gambaran protokol keotentikan diatas, antara lain:[2][1]

1. Protokol keotentikan diatas tidak dirancang untuk menghadapi penyerangan secara tebakan yang berdasarkan pada pengkopian pesan. Hal ini terlihat bahwa semua data yang dipakai dalam pertukaran pesan tidak diacak dan dibiarkan lain bentuk aslinya. Dengan demikian seseorang yang berhasil menyalin pesan 1 yang ditransmisikan akan mengetahui kata sandi dan pemakai yang memilikinya.
2. Protokol sangat mengandalkan kepercayaan bahwa semua pihak yang terkait dalam komunikasi dapat dipercaya, bai *client*, *server*, dan jaringan. Protokol tidak mengantisipasi penyerangan yang memanfaatkan kondisi jaringan yang terbuka untuk siapa saja. Sebagai contoh penyerang cukup merubali alamat port yang ada pada pesan untuk mendapatkan pelayanan dan mengirimkan pesan yang telah berubah tersebut kepada *server*.
3. Protokol hanya menguji kecocokan identitas pemakai dengan kata sandi diberikan terhadap pelayanan yang diinginkan, tanpa menguji keotentikan pengirim pesan tersebut.

Protokol Keotentikan Kerberos Versi 5.

Tabel 2. merupakan ringkasan dari dialog yang digunakan pada Kerberos versi 5. Pesan (1) adalah permintaan *client* untuk mendapatkan tiket yang diperlukan untuk meminta tiket ijin yang ditujukan kepada AS *(Autlientication Server).* Pesan ini terdiri dari *Options* yang merupakan parameter tambahan agar beberapa *flags* diset pada tiket yang diberikan, kemudian ID, pengenal dari pengirim pesan, dalam hal ini pemakai *C. Realmc* digunakan untuk mengidentifikasi lingkungan asal dari pengirim pesan, yaitu lingkungan asal dari pemakai C. IDtgs adalah pengenal dari sel-sel kerberos yang dituju. *Times* digunakan untuk meminta masa berlaku dari tiket yang diminta. Nonce1 adalah sebuah nilai yang harus dikirimkan kembali pada pesan balasan, sehingga dapat diidentifikasi bahwa pesan balasan tersebut benar-benar baru dan bukan hasil rekayasa penyerang.

Pesan (2) merupakan balasan dari AS yang berisi tiket tanda ijin meminta tiket servis, tiket inilah yang berisi informasi identifikasi dari C sebagai peminta ijin. Pesan (2) ditambah lagi dengan suatu blok data yang diacak dengan memakai kunci dari *password* pemakai. Blok ini berisi kunci sesi yang akan digunakan secara ersama antara C (sebagai peminta ijin) dan TGS*(Ticket Granting Server)*, waktu yang telah dipesan dengan pesan (1), dan *nonce* yang dikirim dari pesan 1. Tiket sendiri juga merupakan suatu blok yang diacak dengan kunci yang diberikan oleh TGS, sehingga pemakai C

tidak perlu tahu apa isi sebenarnya dari informal yang ada dalam tiket itu. Informasi dalam tiket itu sendiri terdiri dari kunci sesi yang dipakai, informasi untuk mengidentifikasi pemakai C, nilai waktu yang diminta pemakai, dan bendera (*falgs*) yang menyatakan status dari tiket dan pilihan *(options)* yang diminta oleh pemakai. Penjelasan tentang macam-macam bendera yang dipakai dapat dilihat pada table 2.[2]

Tabel 2. Ringkasan pertukaran pesan pada Kerberos versi 5

(a) Pertukaran keotentikan servise : untuk mendapatkan tiket ijin meminta tiket

(1) C → AS : Option || ID_c || $Realm_c$ || ID_{tgs} || Times || $Nonce_1$

(2) K → C : $Realm_c$ || ID_c || $Ticket_{tgs}$ || E_{kx} [$K_{c,tgr}$ || Times || $Nonce_1$ || $Realm_{tgs}$|| ID_{tgs}]
$Ticket_{tgs} = E_{ktgs}$ [Flags || $K_{c,tgs}$ || $Realm_c$ || ID_c || AD_c || Times]

(b) Pertukaran servis ijin meminta tiket : untuk mendapatkan tiket ijin meinta servis

(3) C → TGS : Option || ID_V || Times || $Nonce_2$ || $Ticket_{tgs}$ || $Authenticator_c$

(4) TGS → C : $Realm_c$ || ID_c || $Ticket_V$ || $E_{kx,tgs}$ [$K_{c,V}$times || $Nonce_2$ || $Realm_V$ || ID_V]
$Ticket_{tgs} = E_{ktgs}$[Flags || $K_{c,Tgs}$ || $Realm_c$ || ID_c || AD_c || Times]
$Authenticator_c = E_{kx,tgs}$ [ID_c || $Realm_c$ || TS_1]

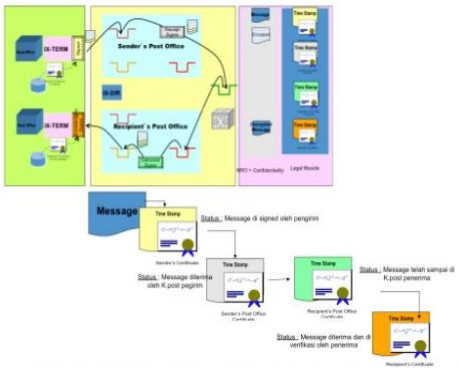
(c) Pertukaran Keotentikan Klien/Server : untuk mendapatkan servis.

(5) C → TGS : Option || $Ticket_V$ || $Authenticator_c$

(6) TGS → C : $E_{kc,v}$ [TS_2 || Subkey || Seq#]
$Ticket_v = E_{kv}$ [Flags || $K_{c,V}$ || $Realm_c$ || ID_c || AD_c || Times]
$Authenticator_c = E_{kc,v}$ [ID_c || $Realm_c$ || TS_2 || Subkey || Seq#]

Setelah pemakai meniiliki tiket untuk meminta tiket dari AS kemudian, pemakai mengirimkan pesan (3) kepada TGS untuk mendapatkan tiket servis. Pesan (3) ini terdiri dari *Options*, untuk meminta gara beberapa bendera tertentu di set, IDv sebagai server yang dituju untuk mendapatkan servis, *Times.*, lama pemakaian servis yang diinginkan, *Nonce2* yaitu suatu nilai yang dikirimkan pemakai untuk dikembalikan lagi oleh TGS sebagai identifikasi bahwa pesan tersebut benar-benar baru , *Tickettgs*, tanda ijin bahwa pemakai telah menerima ijin telah mendapatkan tiket servis dari TGS. Dan terakhir adalah *Authenticatorc*, yang merupakan suatu blok informasi yang diacak dengan kunci sesi yang telah diterima sebelumnya dari pesan (2). *Authenticatorc* ini digunakan untuk mengidentifikasi peminta servis yang terdiri dari data tentang identitas peminta servis, asal dari peminta servis dan TS1 yang merupakan tanda waktu, kapan tiket tersebut diperoleh oleh peminta.

Kelemahan Protokol Kerberos terhadap serangan secara tebakan



Gedung 6. Kelemahan Protokol Kerberos terhadap serangan secara tebakan

Kerberos tidak ditujukan untuk menghadapi serangan secara tebakan, sehingga dapat ditemui kelemahan-kelemahan berikut :[3]

Terdapat data yang dapat dikenali untuk menebak kunci yang dimiliki oleh pemakai, hal ini terlihat pada, pesan 2 misalnya. Pada pesan 2 terdapat informasi mengenai *Nonce1* dan *RealmTGS* yang diacak dengan kunci milik pemakai (KVC), padahal *Nonce1* dapat dilihat pada pesan 1 dalam keadaan tidak teracak, demikian pula dengan yang bukan merupakan rahasia. Dengan demikian pertukaran pesan pada Kerberos perlu diubah sedemikian rupa sehingga informasi tersebut sulit untuk dipakai sebagai acuan tebakan.[2]

Hasil dan Pembahasan

Spesifikasi Proses

Modifikasi yang dilakukan terjadi pada proses otentikasi, sehingga dalam Penelitian ini spesifikasi proses hanya dibuat untuk proses yang di-modifikasi. Pada pihak *Client* proses otentikasi dilakukan oleh proses 1.1.3. otentikasi server, sedangkan pada bagian server pada proses 2.3.3. otetitikasi Client.

Sedangkan proses yang lain tidak dituliskan karena merupakan fasilitas yang disediakan *compiler*.

```
Proses Otentikasi Server
{Mengirimkan pesan no 3}
{Menerima pesan no 2}
Kamus:
        pesan: string
        no_pesan : integer
        kuncisesi : sting
        Bongkar_pesan: Function(pesan:string) ??string
        {menghilangkan Confounder, menata data}
        uji : Function (pesan:string) ??boolean
        {membaca pesan dan mengambil nilai nonce1)
        Confound:Procedure (pesan: string)
        {Menyamarkan data dengan menambahkan confounder pada pesan}
        Krip : Function(pesan, kunci : string) ??string
        {mengacak pesan dengan memakai kunci}
        Kirim: Function (pesan, server :string) ??boolean
        {Mengirim pesan kepada server}
Algoritma:
        If (no_pesan = 2)
        Bongkar_pesan(pesan);
        If (uji(pesan) == notOK)
        Exit; {server tidak terpercaya}
        Else
        Pesan ??gettime() + getcmd() + rand(seed,no) + Authenticator;
        Confound(pesan);
        Krip(pesan,kuncisesi)
        Kirim(pesan, Server);
        Endif
        Endif
```

```
Proses Otentikasi Server
        {Mengirimkan pesan no 2}
        {Menerima pesan no 1,3}
Kamus
        pesan: string
        no_pesan: integer
        kuncisesi : string
        Bongkar_pesan : Function(pesan:string) ??string
        {menghilangkan Confounder, menata data }
        uji : Function (pesan:string) ??boolean
        {membaca. pesan dan mengambil nilai nonce}
        Confound: Procedure (pesan: string)
        {Menyamarkan data dengan menambahkan confounder pada pesan}
        Krip : Function(pesan,kunci : string) ??string
        {mengacak pesan dengan memakai kunci}
        Kirim: Function (pesan, server :string) ??boolean
        {Mengirim pesan kepada server}
Algoritma:
        If (no_pesan = 1)
        Pesan ??gettime() + getcmd() + rand(seed,no) + Authenticator;
        Confound(pesan);
        Krip(pesan,kuncisesi)
        Kirim(pesan, Server);
        Endif
        If (no_pesan = 3)
        Bongkar_pesan(pesan);
        If (uji(Authenticator) == notOK)
        Exit; {Client tidak, terpercaya}
        Else
        Exec(getcmd());
        Endif
        endif
        Endif
        Endif
```

```
Proses Otentikasi Server
        {Mengirimkan pesan no 2}
        {Menerima pesan no 1,3}
Kamus:
        pesan: string
        no_pesan: integer
        kuncisesi : string
        Bongkar_pesan : Function(pesan:string) ??string
        {menghilangkan Confounder, menata data }
        uji : Function (pesan:string) ??boolean
        {membaca. pesan dan mengambil nilai nonce}
        Confound: Procedure (pesan: string)
        {Menyamarkan data dengan menambahkan confounder pada pesan}
        Krip : Function(pesan,kunci : string) ??string
        {mengacak pesan dengan memakai kunci}
        Kirim: Function (pesan, server :string) ??boolean
        {Mengirim pesan kepada server}
Algoritma:
        If (no_pesan = 1)
        Pesan ??gettime() + getcmd() + rand(seed,no) + Authenticator;
        Confound(pesan);
        Krip(pesan,kuncisesi)
        Kirim(pesan, Server);
        Endif
        If (no_pesan = 3)
        Bongkar_pesan(pesan);
        If (uji(Authenticator) == notOK)
        Exit; {Client tidak, terpercaya}
        Else
        Exec(getcmd());
        Endif
        endif
```

Kriteria Pengukuran

Tingkat kerahasiaan pada sistem keamanan ini diukur dari waktu yang diperlukan penyerang untuk membongkar sistem keamanan. Penyerang dalam hal ini memakai metode tebakan terhadap kunci dari pesan dan menguji kebenaran tebakannya dengan membandingkannya dengan pesan-pesan yang telah berhasil dicuri.

Waktu yang diperlukan sebelum sistem keamanan terbongkar harus memenuhi persyaratan yaitu penyerangan terhadap kunci pada pesan harus mampu ditunda hingga minimal 60 menit. Karena setiap satu jam program hasil implementasi akan melakukan perubahan kunci sesi yang digunakan untuk berhubungan dengan server.

Pengukuran Tingkat Kerahasiaan Pengacak Pesan

Algoritma pengacak dalam sistem keotentikan ini memakai algoritma DES. Algoritma ini metransformasikan masukan berupa data 64 bit menjadi data yang berbeda sebesar 64 bit berdasarkan variabel sebesar 56 bit. Jika ke 64 bit masukan dipakai dan ke 56 bit variabel kunci dipilih secara acak, maka tidak ada cara lain untuk mencari kunci yang dipakai selain dengan mencoba setiap kemungkinan kunci. Algoritma ini menjamin bahwa akan ada lebih dari 70 000 000 000 000 000 (256) kemungkinan yang mungkin menjadi kunci.

Kelemahan tersebut diatasi dengan penambahan *confounder* sebagaimana disebutkan dalam teori dimuka. Bilangan acak dibangkitkan dengan memakai nilai awal waktu (dalam detik) saat proses dilakukan. Sedangkan nomor bilangan acak yang dipakai sebagai *confounder* adalah waktu tersebut ditambah dengan nilai 1/100 kali 100 (misalnya jika nilai waktu saat itu 1,23. Maka nomor bilangan acak adalah 24). Dengan demikian setiap detik akan terjadi perubahan urutan bilangan acak yang dibangkitkan dan setiap 1/100 detik terjadi perubahan nomor bilangan acak yang akan dipergunakan sebagai *confounder*.

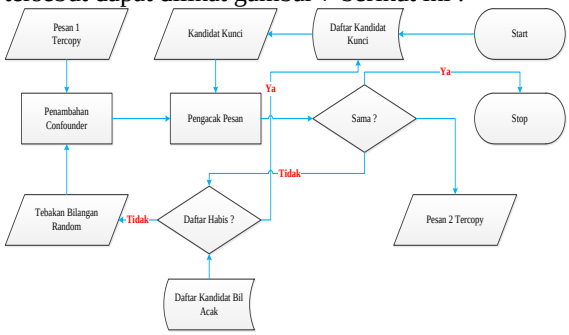
Asumsi Pengukuran Tingkat Kerahasiaan Melawan Serangan Secara Tebakan

Asumsi yang dipakai adalah sebagai berikut :

1. Tebakan dilakukan dengan memakai 100 buah kandidat kunci, dengan anggapan bahwa kunci pasti ketemu setelah 100 kali tebakan.
2. Pengacak pesan adalah sebuah fungsi yang diketahui oleh penyerang.
3. Cara-cara penambahan *Confounder* diketahui oleh penyerang.
4. Penyerang memulai serangan dengan pengetahuan akan waktu pengiriman ± 5 detik dari waktu yang sebenarnya.
5. Penyerang mempunyai contoh pesan sebelum dikirimkan dan pesan sesudah dikirimkan.
6. Algoritma penyerangan sebagai berikut :

```
while (Kunci_tidak_ketemu)
while (tebakan_masih_ada or Kunci_tidak_ketemu)
tebakan ??guest[ k ];
Hasil_tebakan_awal = dekrip(tebakan, ciphertek);
Waktu_asli ??Waktu XOR Nonce;
Baca_waktu(detik,milidetik);
Rand_seed(detik);
for i=1 to (detik * milidetik)
Confounder = random(10000);
Hasil_tebakan_akhir = Hasil_tebakan_awal XOR Confounder
If (Hasil_tebakan_akhir = plaintek)
Kunci_ketemu = tebakan,
Else
If (tebakan_masih_ada)
K??k+ 1;
endif
endif
end while.
k ??1
end while.
```

Untuk mempermudah pembacaan algoritma tersebut dapat dilihat gambar 7 berikut ini :



Gambar 7. Diagram alur penyerangan hasil modifikasi

Pembahasan Pengukuran

Dengan memakai algoritma diatas usaha untuk mencari kunci dilakukan. Perulangan yang dilakukan oleh algoritma tersebut untuk mencari kunci sangat tergantung perkiraan waktu terjadinya pertukaran pesan tersebut dituliskan oleh pengirim pesan. Dalam pengujian dilakukan penghitungan waktu yang diperlukan untuk menemukan kunci dengan anggapan tebakan waktu transaksi tidak lebih dari ±5 detik. Sehingga tebakan dilakukan dengan tebakan terhadap waktu pesan selama waktu 10 detik.

Algoritma yang telah disebutkan diatas untuk melakukan tebakan pada satu buah tebakan selama ± 0,1 detik. dengan asumsi bahwa harus melakukan 100 macam tebakan untuk selang waktu 10 detik, maka program harus melakukan tebakan sebanyak 10 (detik) kali 100 (milidetik) kali 1(1 buah kandidat kunc1) buah tebakan hasilnya harus melakukan tebakan sebanyak 1 000 kali. Sehingga waktu yang diperlukan untuk menebak kunci dengan memakai seluruh kandidat kunci adalah adalah 100 000 kali 0,l detik (waktu yang diperlukan untuk satu tebakan kunci) dengan kata lain waktu yang diperlukan adalah 10 000 detik atau 2 jam 46 menit 40 detik. Lebih dari yang telah didefinisikan diatas.

Kesimpulan dan Saran

Dari pembahasan diatas dapat disimpulkan sebgai berikut :

1. Metode penambahan Confounder yang diterapkan dalam sistem keotentikan dapat dimplementasikan dalam sistem operasi Linux sebagai bagian pengujian keaslian pemakai jika pemakai ingin mendapatkan pelayanan dari jarak jauh.
2. Metode L. Gong yang diterapkan akan meningkatkan ketahanan terhadap serene secara tebakan dengan meningkatkan usaha penyerang untuk melakukan tebakan. Usaha penyerangan ditingkatkan dengan memaksa penyerang untuk melakukan tebakan terhadap *confounder* terlebih dahulu sebelum dapat menguji kebe-naran hasil tebakannya.
3. Dengan asumsi bahwa tebakan terhadap waktu pengiriman pesan mempunyai rentang ± 5 detik, ketahanan protokol keotentikan meningkat 1000 kali lipat dibanding jika metode Li Gong tidak diterapkan.

Berikut adalah beberapa saran yang dapat digunakan untuk penelitian selanjutnya :

1. Biasakan untuk membuat kata sandi yang tidak dapat dikaitkan dengan pemakai atau kata-kata yang bersesuaian dengan yang ada pada kamus.
2. Biasakan untuk mengganti kata sandi dalam selang waktu yang sesering mungkin.

Daftar Pustaka

[1] L. Gong, M. A. Lomas, R. M. Needham, and J. H. Saltzer, "Protecting Poorly Chosen Secret from Guessing Attacks", *IEEE Journal On Selected Areas In Communication*, Vol 11, No 5, Juni 1993. Hal 648-656.

[2] S.Bellovin and M. Merrit, " Encripted key exchange : Password based protocols secure againts dictionary attacks", *IEEE Symp. Security Privacy*, Oakland, CA, Mei 1992. hal 72-84.

[3] J. Tackett Jr, D. Gunter, and L. Brown, "*Spesial Edition"*: Using Linux", QUEC®,1995.

[4] Randal L. Schwartz, "Learning Perl", O'Reilly & Associates, Inc. 1994

[5] R. Bird, L. Gopal, A. Herzberg, P. A. Janson, S.Kutten, RMolva, M.Yung, " Systematic Design of a Family of Attack Resistant Authentication Protocols",*IEEE Journal On Selected Areas in Communication*, Vol 11, No 5, Juni 1993. Hal 679 - 693.