TUGAS AKHIR
EC5010 KEAMANAN SISTEM INFORMASI
EKSPLOITASI RPC PADA
SISTEM OPERASI WINDOWS

Oleh
Tommy Marki
13202057
(tom2marki[at]gmail[dot]com)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2006

Abstraksi

Pada jaringan (*network*) terdapat berbagai macam protokol yang masing-masing memiliki fungsi yang unik. Salah satunya adalah protokol Remote Procedure Call (RPC). Protokol ini menyediakan suatu mekanisme komunikasi antar proses yang mengijinkan suatu program untuk berjalan pada suatu komputer tanpa terasa adanya eksekusi kode pada sistem yang jauh (remote system). Didalam protokol ini juga terdapat fungsi lain misalnya protokol message, fitur RPC, dsb. Tiap-tiap fungsi ini terkandung pada tiga lapisan RPC yaitu lapisan tertinggi, menengah, dan terendah. Tiap lapisan bersentuhan dengan bagian yang berbeda pada sistem operasi. Implementasi protokol RPC meliputu sektor yang lebih kompleks, mulai dari pemetaan port, bahasa yang digunakan pada pemprograman RPC dan cara kerjanya.

Selain membahas mengenai protokol RPC, pada tugas kali ini akan dibahas mengenai eksploitasi protokol RPC. Eksploitasi yang dimaksud adalah penyalahgunaan fungsi asli protokol ini untuk kegiatan yang lain yang sifatnya merugikan pihak yang di remote.

Daftar Isi

Pendahuluan

Windows XP merupakan salah satu sistem operasi yang banyak digunakan dengan salah satu keunggulannya yaitu user-friendly. Dibalik keunggulan tersebut, integritas sistem operasi untuk berjalan pada aplikasi jaringan jauh tertinggal dibandingkan sistem operasi lainnya seperti Linux atau Unix. Salah satu kelemahan sistem operasi ini yang banyak digunakan para cracker dan juga hacker adalah protokol RPC (Remote Procedure Call).

RPC adalah suatu protokol yang menyediakan suatu mekanisme komunikasi antar proses yang mengijinkan suatu program untuk berjalan pada suatu komputer tanpa terasa adanya eksekusi kode pada sistem yang jauh (remote system).Protokol RPC digunakan untuk membangun aplikasi klien-server yang terdistribusi. Protokol ini didasarkan pada memperluas konsep konvensional dari suatu prosedur dimana nantinya prosedur ini dapat dipanggil dimana pemanggil tidak harus mempunyai alamat yang sama dengan yang lokasi dimana prosedur ini dipanggil. Dimana proses ini dapat dilakukan pada sistem yang sama atau sistem yang berbeda namun terhubung pada jaringan. Namun terdapat kelemahan didalam bagian dari RPC yang berhubungan dengan pertukaran message melalui TCP/IP. Kegagalan terjadi dikarenakan karena penanganan kesalahan pada message yang berisi informasi yang salah. Hasil dari kelemahan ini berakibat pada bagian antar-muka RPC, yaitu bagian yang mendengarkan port RPC yang di-enable. Bagian antar-muka ini menangani objek aktivasi dari DCOM (Distributed Component Object Model) yang dikirimkan oleh mesin klien ke server. Kelemahan ini umumnya dimanfaatkan oleh seorang penyerang untuk dapat menjalankan suatu kode dengan kewenangan Administrator sistem lokal pada sistem yang

terinfeksi. Dengan demikian, maka sistem yang diserang ini dapat diubah-ubah termasuk pengkopian dan penghilangan data sampai pembuatan user baru dengan hak tidak terbatas.

Chapter 1

Remote Procedure Call

Pada chapter ini penulis ingin memberikan deskripsi mengenai protokol RPC. Pembahasan pada chapter ini terbatas pada pengenalan dasar protokol RPC berikut atributnya meliputi protokol message dan fitur-fitur dari RPC. Protokol Message merupakan bagian penting dari RPC karena menyangkut mengenai interaksi yang dilakukan pada prosedur RPC. Fitur-fitur RPC merupakan kemampuan atau keunggulan protokol RPC dalam memberikan layanan dalam proses remote. Pada chapter ini juga akan dibahas mengenai prosedur otentifikasi pada protokol RPC. Proses otentifikasi digunakan untuk mengidentifikasi baik klien, server atau keduanya.

1.1 Definisi

1.1.1 Remote Procedure Calls

RPC adalah suatu protokol yang menyediakan suatu mekanisme komunikasi antar proses yang mengijinkan suatu program untuk berjalan pada suatu komputer tanpa terasa adanya eksekusi kode pada sistem yang jauh (remote system). RPC mengasumsi keberadaan dari low-level protokol transportasi seperti TCP atau UDP untuk membawa pesan data dalam komunikasi suatu program. Protokol RPC dibangun diatas protokol eXternal Data

Representation (XDR), yang merupakan standar dari representasi data dalam komunikasi remote. Protokol XDR mengubah parameter dan hasil dari tiap servis RPC yang disediakan.

Protokol RPC mengijinkan pengguna (users) untuk bekerja dengan prosedur remote sebagaimana bekerja dengan prosedur lokal. Prosedur panggilan remote (remote procedure calls) didefinisikan melalui rutin yang terkandung didalam protokol RPC. Tiap message dari panggilan akan disesuaikan dengan message balikan. Protokol RPC sendiri sebenarnya adalah suatu protokol untuk "meneruskan pesan" yang mengimplemntasikan protokol non-RPC lain seperti panggilan remote batching dan broadcasting. Protokol ini juga mendukung adanya prosedur callback dan select subroutine pada sisi server.

1.1.2 Klien dan Server

Klien adalah komputer atau proses yang mengakses suatu servis/layanan atau resources dari proses atau komputer pada suatu jaringan. Server adalah komputer yang menyediakan servis/layanan dan resources, dan yang mengimplementasikan servis jaringan. Tiap servis pada network adalah susunan dari program remote, dan tiap program remote mengimplementasi prosedur remote. Semua prosedur berikut parameternya dan hasilnya didokumentasi secara spesifik pada protokol suatu program.

1.2 Protokol Message RPC

Protokol Message RPC didefinisikan dengan menggunakan deskripsi data eXternal Data Representation (XDR) yang meliputi struktur, enumerasi dan union. Pembahasan lebih lanjut akan diterangkan pada bab berikutnya mengenai implementasi RPC.

Protokol Message ini membutuhkan faktor-faktor pendukung sebagai berikut :

1. Spesifikasi yang unik untuk tiap prosedur call

2. Respon message yang sesuai untuk tiap message yang diminta

3. Otentifikasi klien untuk tiap layanan dan sebaliknya

Protokol Message RPC memiliki dua (2) struktur yang berbeda, yaitu **call message** dan **reply message**. Tiap klien yang akan melakukan RPC pada suatu server di jaringan akan menerima balasan (*reply*) berupa hasil dari eksekusi prosedur tersebut. Dengan menggunakan spesifikasi yang unik untuk tiap prosedure remote, maka RPC dapat mencocokkan message balasan untuk tiap call message yang diminta klien.

1.2.1 Call Message

Tiap call message pada RPC mengandung nilai-nilai unsigned integer yang digunakan untuk mengidentifikasi prosedur remote yang diminta. Nilai-nilai ini adalah :

1. Nomor Program

2. Nomor Versi dari Program

3. Nomor Prosedur

1.2.2 Reply Message

Reply message yang dikirimkan oleh server jaringan bervariasi tergantung apakah call messages yang diminta klien diterima atau ditolak. Reply message mengandung informasi yang digunakan untuk membedakan kondisi-kondisi yang diminta sesuai dengan call messages. Informasi ini antara lain :

1. RPM mengeksekusi call message dengan sukses

2. Implementasi remote tidak sesuai dengan protokol yang digunakan. Versi yang lebih rendah atau tinggi akan ditolak.

3. Program remote tidak tersedia pada sistem remote

4. Program remote tidak mendukung versi yang diminta klien

5. Nomor prosedur yang diminta tidak ada.

1.3 Fitur dalam RPC

RPC memiliki fitur - fitur sebagai berikut : *batching calls, broadcasting calls, callback procedures* dan *using the* ***select*** *subroutine.*

1.3.1 Batching Calls

Fitur Batching calls mengijinkan klien untuk mengirim message calls ke server dalam jumlah besar secara sequence (berurutan). Batching menggunakan protokol streaming byte seperti TCP / IP sebagai mediumnya. Pada saat melakukan batching, klien tidak menunggu server untuk memberikan reply terhadap tiap messages yang dikirim, begitu pula dengan server yang tidak pernah mengirimkan messages reply. Fitur inilah yang banyak digunakan klien, karena arsitektur RPC didesain agar pada tiap call message yang dikirimkan oleh klien harus ada proses menunggu balasan dari server. Oleh karena itu maka pihak klien harus dapat mengatasi error yang kemungkinan terjadi karena pihak klien tidak akan menerima peringatan apabila terjadi error pada message yang dikirim.

1.3.2 Broadcasting Calls

Fitur Broadcasting mengijinkan klien untuk mengirimkan paket data ke jaringan dan menunggu balasan dari network. FItur ini menggunakan protokol yang berbasiskan paket data seperti UDP/IP sebagai mediumnya. Broadcast RPC membutuhkan layanan port mapper RPC untuk mengimplementasikan fungsinya.

1.3.3 Callback Procedures

Fitur Callback Procedures mengijinkan server untuk bertindak sebagai klien dan melakukan RPC callback ke proses yang dijalankan oleh klien.

1.3.4 Menggunakan select Subrutin

Fitur ini akan memeriksa deskripsi dari suatu file dan messages dalam antrian untuk melihat apakah mereka siap untuk dibaca (diterima) atau ditulis (dikirim), atau mereka dalam kondisi ditahan sementara. Prosedur ini mengijinkan server untuk menginterupsi suatu aktivitas, memeriksa datanya, dan kemudian melanjutkan proses aktivitas tersebut.

1.4 Otentifikasi RPC

Proses otentifikasi adalah proses yang digunakan untuk mengidentifikasi server dan klien pada RPC. Untuk setiap prosedur remote yang dilakukan protokol RPC menyediakan slot yang dipakai sebagai parameter otentifikasi yang berfungsi agar pemanggil (caller) dapat memberikan identitasnya kepada server. Parameter otentifikasi ini dibuat di paket klien.

Otentifikasi RPC terdiri atas beberapa bagian. Berikut ini adalah bagian-bagian pada otentifikasi RPC :

1. Protokol Otentifikasi RPC

   Protokol Otentifikasi RPC disediakan sebagai bagian dari protokol RPC. Untuk setiap prosedur remote, semuanya diotentifikasi oleh paket RPC pada server. Parameter yang digunakan adalah *respon verifier*. Sedangkan pada pihak klien, setiap paket RPC diberikan parameter otentifikasi dan parameter yang digunakan adalah *credential* dan *verifier*.

2. Otentifikasi NULL

   Otentifikasi NULL digunakan pada sistem dimana pemanggil (*caller*) RPC tidak

mengetahui identitasnya sendiri dan server tidak membutuhkan identitas pemanggil.

3. Otentifikasi UNIX

   Otentifikasi Unix digunakan pada prosedur remote di sistem UNIX. Jenis otentifikasi ini dibagi dua (2) yaitu otentifikasi pada sisi klien dan otentifiksi pada sisi server. Pada sisi klien, otentifikasi ini akan membuat otentifikasi handle dengan AIX *permissions* agar dapat berasosiasi dengan parameter credentials pada sistem UNIX. Sedangkan pada sisi server, server harus dapat menentukan tipe otentifikasi yang diberikan oleh pemanggil RPC. Penentuan dukungan terhadap tipe otentifikasi akan memberikan *reply* yang berbeda.

4. Otentifikasi Data Encryption Standard (DES)

   Otentifikasi DES membutuhkan **keyserv** daemon yang harus berjalan baik di sisi server maupun klien. Tiap pengguna pada sistem ini harus memiliki kunci publik (*public key* yang disahkan pada database kunci publik oleh Administrator jaringan tersebut.

5. Protokol Otentifikasi DES

   Protokol Otentifikasi DES meliputi protokol penanganan DES pada proses otentifikasi RPC. Protokol ini mencakup 64-bit blok data DES yang terenkripsi dan menentukan panjang maksimum untuk user name pada jaringan yang digunakan.

6. Enkripsi Diffie-Hellman

   Enkripsi Diffie-Hellman digunakan pada pembuatan kunci public pada otentifikasi DES dengan menggunakan 192-bit kunci. Enkripsi ini memiliki dua buah variabel konstan, yaitu BASE dan MODULUS yang digunakan pada protokol otentifikasi DES.

RPC berhubungan hanya dengan proses otentifikasi, tidak dengan kontrol akses terhadap services/layanan individual yang diberikan. Tiap layanan mengimplementasikan peraturan mengenai kontrol akses masing-masing.

Subsistem otentifikasi pada paket RPC bersifat open-ended, artinya beberapa otentifikasi dapat diasosiasikan pada RPC klien.

Chapter 2

Implementasi RPC

Pada chapter ini akan dibahas mengenai implementasi protokol RPC lebih lanjut. Pembahasan meliputi beberapa bagian, pertama mengenai bahasa RPC yaitu deskripsi mengenai bahasa yang digunakan pada RPC. Kedua meliputi program Port Mapper (pemeta port) yaitu program yang dibutuhkan klien untuk mencari port pada server yang dapat digunakan untuk prosedur RPC. Ketiga mengenai pemodelan RPC dalam penjelasan lapisan-lapisan (*layer*) RPC. Dan terakhir akan dibahas mengenai termasuk cara kerja dari Protokol RPC.

2.1 Bahasa RPC

Bahasa RPC (RPC Language - RPCL) merupakan bahasa yang dikembangkan dari bahasa XDR. Bahasa RPC memiliki kemiripan dengan bahasa XDR namun dengan beberapa penambahan yaitu program definisi. Implementasi layanan protokol dan rutin menggunakan command rpcgen yang berkorespondensi dengan bahasa pemprograman C.

Deskripsi dari bahasa RPC meliputi :

1. Definition

   File dengan bahasa RPC memiliki beberapa definisi, diantaranya adalah : enum, struct, union, typedef, const, dan program.

2. Structure

Struktur pada bahasa RPC dideklarasikan seperti pada pendeklarasian struktur dalam bahasa C

3. Union

   Union pada bahasa RPC berbeda dengan bahasa C. Kemiripan lebih ditunjukkan dengan variasi pada bahasa Pascal

4. Enumeration

   Enumerasi pada bahasa ini memiliki syntax yang sama dengan bahasa C.

5. TypeDef

   Tipe Definisi (Typedef) pada bahasa ini memiliki syntax yang sama dengan typedef pada bahasa C.

6. Constant

   Constant pada bahasa ini dapat digunakan jika variabel integer konstant dibutuhkan.

7. Programs

   Program RPC dideklarasikan dengan syntax berikut secara berurutan : program-definiton, version-list, version, procedure-list, procedure.

8. Declarations

   Dalam bahasa ini, terdapat empat jenis tipe deklarasi yaitu : simple declarations, fixed-length array declarations, variable-length declaration, dan pointer declaration.

2.2 Port Mapper

Port adalah kanal komunikasi diantara klien dan server. Port-port komunikasi ini dibedakan berdasarkan nomor yang dimilikinya dengan fungsi masing-masing. Namun nomor-nomor

port ini, terutama yang memberikan layanan RPC, tidak diberikan pada jaringan transport. Jaringan transport hanya menyediakan layanan pemprosesan message di dalam jaringan.

Untuk mengatasi hal ini, maka program pada komputer klien harus mampu untuk mencari nomor port untuk tiap program di server yang hendak digunakan. Protokol port mapper adalah suatu layanan pada jaringan yang dapat mengatasi masalah ini. Protokol ini memberikan hak pada klien untuk mencari nomor port untuk semua program remote yang disediakan oleh server. Dengan demikian maka implementasi protokol ini pada suatu program port mapper akan memetakan tiap-tiap program RPC dan nomor versinya dengan nomor-nomor port yang spesifik.

2.2.1 Meregister Port

Port Mapper terletak pada nomor port 111 pada setiap mesin (host maupun server) dan merupakan satu-satunya layanan jaringan yang mempunyai port yang khusus dan tetap. Sedangkan untuk jenis layanan jaringan lainnya, nomor portnya dapat statis atau berubah-ubah asalkan kesemuanya terdaftar pada port mapper.

Penempatan nomor port untuk tiap program remote ke dalam port mapper akan mengotomatisasi administrasi nomor-nomor port. Hasilnya akan disimpan dalam satu file dimana file ini akan diduplikat ke setiap klien. Sehingga akan terjadi proses pembaruan data (*update*) setiap kali ada program remote baru yang disediakan oleh jaringan. Salah satu cara alternatif agar sistem tidak harus selalu meng-update file mappernya adalah dengan menempatkan hasil pemetaan port program remote pada suatu file Network File System (NFS) yang di-*sharing*. Namun hal ini membawa kendala apabila server tidak dapat berfungsi, maka seluruh jaringan juga tidak dapat menggunakan fungsi ini.

Pemetaan Port program yang disimpan pada suatu port mapper di server disebut dengan **portmap**. Port mapper ini akan dijalankan secara otomatis tiap kali mesin server

dijalankan. Lalu baik program server maupun klien akan memanggil prosedur port mapper. Kemudian sebagai bagian dari proses inisialisasi, program server akan memanggil port mapper pada host untuk membuat entri pada portmap. Setelah itu, program server akan meng-update entri pada portmap, sedangkan program klien akan memanggil query dari entri portmap ini. Untuk mencari nomor port yang diinginkan, program klien kemudian menggirimkan RPC call message ke port mapper pada server. Apabila proses ini berhasil (server mendukung remote program yang diminta), port mapper server akan mengirimkan nomor port yang sesuai pada RPC reply message. Kemudian proses remote dapat dilakukan dengan menggunakan nomor port tersebut. Proses ini akan selalu dijalankan setiap kali ada permintaan remote program dari klien ke server. Namun untuk meminimalisasi pemanggilan port mapper, pada sisi klien disediakan cache untuk menyimpan nomor-nomor port yang sering digunakan.

2.2.2 Prosedur Port Mapper

Program port mapper mendukung dua protokol, yaitu UDP dan TCP/IP. Program ini terhubung pada port 111 untuk kedua protokol ini. Berikut ini adalah prosedur-prosedur yang digunakan program port mapper pada kedua protokol ini :

1. NULL

   Prosedur ini tidak berfungsi, prosedur ini tidak memberikan parameter dan juga tidak memberikan hasil.

2. SET

   Prosedur ini akan meregister program pada port mapper dengan memberikan parameter sebagai berikut : program number (prog), version number (vers), transport protocol number (prot), dan nomor port yang diminta untuk layanan ini. Hasil dari prosedur ini berupa Boolean True atau False yang mengindikasikan sukses tidaknya

proses mapping.

3. UNSET

   Prosedur ini digunakan untuk me-unregister program pada port mapping jika program remote tidak lagi digunakan. Parameter yang dibawa sama dengan prosedur SET dikurangi nomor protokol dan nomor port.

4. GETPORT

   Prosedur ini memberikan parameter berupa nomor program (prog), version number (vers), dan transport protocol number (prot) untuk mendapatkan hasil berupa nomor port untuk program yang diminta.

5. DUMP

   Prosedur ini akan mencatat semua entri dalam database port mapper. Prosedur ini tidak membutuhkan parameter dan memberikan hasil berupa (prog), (prot), (vers), dan nomor port.

6. CALLIT

   Prosedur ini digunakan untuk memanggil suatu program remote lain pada mesin yang sama tanpa harus mengetahui nomor port dari program yang diminta.

2.3 Lapisan RPC

Bagian antar-muka (*interface*) dari RPC dibagi menjadi 3 lapisan / bagian (*layer*) yaitu :

1. Lapisan Tertinggi (Highest Layer)

   Lapisan ini merupakan lapisan yang sangat bersentuhan langsung dengan sistem operasi, mesin dan jaringan tempat RPC berjalan. Layer ini umumnya banyak digunakan pada pembuatan dan pemprograman RPC karena penggunaan layer ini sama

saja dengan penggunaan RPC. Banyak servis / layanan pada layer ini yang berhubungan langsung dengan informasi yang banyak dibutuhkan, misalnya fungsi rnuser() yang berguna untuk memberikan informasi jumlah user / pengguna pada sistem remote.

Berikut ini jenis-jenis servis lainnya yang banyak digunakan pada layer ini :

Rutin	Deskripsi
rnusers	mengembalikan jumlah user pada sistem remote
rusers	mengembalikan informasi mengenai user tertentu
havedisk	memeriksa keberadaan disk pada mesin remote
rstats	melihat kinerja dari kernel remote
rwall	menulis untuk menentukan mesin remote tertentu
yppasswd	mengupdate password dari user dalam Yellow Pages

Tabel 1. Servis pada Layer Tertinggi RPC

2. Lapisan Menengah (*Intermediate Layer*)

   Lapisan ini merupakan implementasi dari RPC sesungguhnya. Pada layer ini, seorang user tidak harus berhubungan dengan soket, sistem operasi atau implementasi lo-level lainnya. Pada layer ini, seorang user hanya melakukan proses remote pada suatu mesin. Layer ini merupakan layer yang digunakan untuk semua program RPC. Pada layer ini terdapat rutin-rutin mengenai "registerrpc()", "callrpc", dan scv run. Dua rutin yang disebut pertama adalah rutin-rutin yang fundamental. "registerrpc() digunakan untuk memperoleh nomor unik dari tiap prosedur identifikasi dalam tiap sistem. Sedangkan "callrpc()" digunakan untuk mengeksekusi prosedur remote. Implementasi layer diatasnya dilakukan pada layer ini.

3. Lapisan Terendah (*Lowest Layer*)

   Lapisan ini merupakan lapisan yang mengatur tentang soket dan sistem call. Lapisan

ini tidak memberikan data dan servis secara detail untuk dapat digunakan. Umumnya program yang dibuat untuk lapisan ini merupakan program yang paling efisien. Permasalahan yang timbul pada sistem ini berkaitan dengan penyesuaian implementasi RPC untuk sistem operasi yang berbeda.

2.4 Model dan Cara Kerja RPC

Prosedur call umumnya berkaitan dengan penggunaan stack, penyimpanan parameter yang diterima dalam stack tersebut dan pengalokasian ruang untuk lokal variabel. Namun selain itu ada yang disebut dengan Prosedur Call remote, yang berarti pelaksanaan proses diatas namun pada suatu sistem lain yang berhubungan melalui suatu jaringan. Sistem prosedur remote ini memiliki cara kerja yang sedikit banyak mirip, namun berbeda dengan prosedur call biasa.

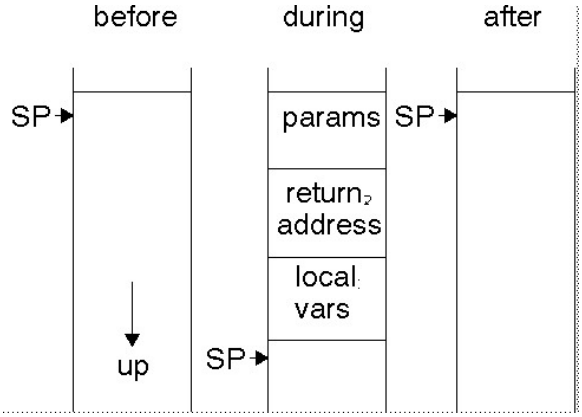

Figure 2.1: Prosedur Call Lokal

Tiap prosedur yang dipanggil dalam RPC, maka proses ini harus berkoneksi dengan server remote dengan mengirimkan semua parameter yang dibutuhkan, menunggu balasan

dari server dan melakukan proses kemudian selesai. Proses di atas disebut juga dengan *stub* pada sisi klien.

Sedangkan Stub pada sisi server adalah proses menunggu tiap message yang berisi permintaan mengenai prosedur tertentu. Server harus membaca tiap parameter yang diberikan, kemudian memberikan prosedur lokal yang sesuai dengan permintaan dan parameter. Kemudian setelah eksekusi, server harus mengirimkan hasil kepada pihak pemanggil proses.

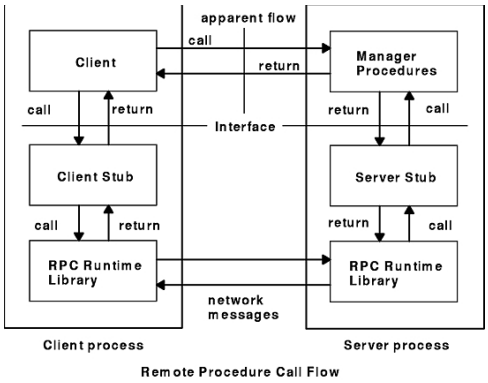

Figure 2.2: Remote Procedure Call Flow

Diagram diatas memberikan gambaran mengenai flow dari eksekusi dalam proses RPC. Berikut ini adalah diagram yang akan menjelaskan secara rince mengenai proses yang terjadi pada klien dan server dalam eksekusi suatu prosedur RPC :

Berikut penjelasan dari diagram diatas :

1. Klien memanggil prosedur stub lokal. Prosedur Stub akan memberikan parameter dalam suatu paket yang akan dikirim ke jaringan. Proses ini disebut sebagai **marshalling**.

2. Fungsi Network pada O/S (Operating system - Sistem Operasi) akan dipanggil oleh stub untuk mengirim suatu message.

3. Kemudian Kernel ini akan mengirim message ke sistem remote. Kondisi ini dapat berupa *connectionless* atau *connection-oriented*.

4. Stub pada sisi server akan melakukan proses **unmarshals** pada paket yang dikirim pada

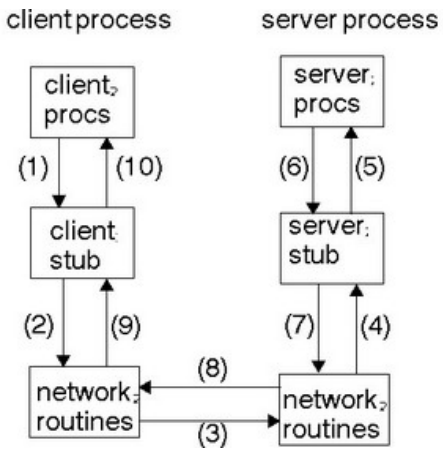

Figure 2.3: Proses RPC

network.

5. Stub pada server kemudian mengeksekusi prosedur panggilan lokal.

6. Jika eksekusi prosedur ini telah selesai, maka eksekusi diberikan kembali ke stub pada server.

7. Stub server akan melakukan proses marshals lagi dan mengirimkan message nilai balikan (hasilnya) kembali ke jaringan.

8. Message ini akan dikirim kembali ke klien.

9. Stub klien akan membaca message ini dengan menggunakan fungsi pada jaringan.

10. Proses **unmarshalled** kemudian dilakukan pada message ini dan nilai balikan akan diambil untuk kemudian diproses pada proses lokal.

Proses diatas akan dilakukan berulang-ulang (rekursif) dalam pengeksekusian RPC dalam suatu remote sistem.

Chapter 3

Kelemahan dan Eksploitasi RPC

Tujuan utama penggunaan protokol RPC adalah untuk mempermudah komunikasi dalam pembangunan aplikasi klien - server yang terdistribusi. Namun dalam perkembangannya, seiring dengan perkembangan sistem operasi, protokol ini banyak disalah-gunakan. Hal ini terkait dengan kelemahan protokol ini yang dimanfaatkan oleh pihak-pihak tertentu untuk mendapatkan keuntungan atau sekedar untuk mengacaukan sistem yang berlaku. Tindakan ini disebut juga dengan eksploitasi RPC.

3.1 Kelemahan RPC pada Sistem Operasi Windows

Sistem operasi Microsoft Windows adalah salah satu sistem operasi yang mendukung protokol RPC. Sebagai sistem operasi yang paling banyak digunakan di seluruh dunia baik untuk PC Desktop maupun server, sistem operasi Windows menjadi salah satu sistem operasi yang banyak dijadikan target mulai dari serangan virus, worm, sampai trojan. Protokol RPC juga tidak luput menjadi sasaran serangan oleh pihak yang tidak bertanggung jawab.

Alasan penyerangan pada protokol ini disebabkan karena kemudahan ketergantungan sistem operasi Windows pada servis/layanan RPC. Banyak layanan pada sistem operasi ini yang didesain untuk bergantung pada layanan RPC, baik digunakan maupun tidak. Dan sayangnya, pengguna sistem operasi ini tidak dapat mengdisfungsikan layanan RPC.

Pengguna bisa saja menggunakan firewall untuk memblok port RPC, namun karena sistem operasi Windows terlalu bergantung pada mekanisme RPC sebagai fungsi dasarnya maka hal ini kadang-kadang tidak berhasil. Bahkan sebagai studi kasus, pada sistem operasi WIndows Server 2003 terdapat kelemahan pada fungsi RPC itu sendiri (Informasi lengkapnya dapat dilihat pada tabel kelemahan sistem operasi ini pada lampiran).

Kelemahan protokol RPC pada sistem operasi ini terletak pada fungsi RPC yang berhubungan dengan pertukaran message melalui protokol TCP/IP. Hal ini mempengaruhi bagian antar-muka Distributed Component Object Model (DCOM) yang berhubungan dengan RPC, yang akan mendengarkan (*listen*) port-port RPC yang tersedia. DCOM adalah protokol yang berfungsi untuk mengaktifkan komponen pada perangkat lunak (software) agar dapat berkomunikasi langsung dengan jaringan. Protokol ini didesain untuk penggunaan jaringan multi tranport termasuk protokol Internet misalnya HTTP. RPC melalui HTTP v1 (pada sistem operasi WIndows NT 4.0, Windows 2000) dan v2 (Windows XP, Windows Server 2003) mempunyai suatu fitur baru yaitu mendukung protokol transportasi RPC yang mengijinkan RPC untuk beroperasi melalui port TCP 80 dan 443. Hal ini menyebabkan komunikasi antara klien dan server dapat dilakukan walaupun dalam pengawasan server proxy dan firewall. COM Internet Services (COM) mengijinkan DCOM untuk melakukan RPC melalui HTTP untuk komunikasi DCOM klien dan DCOM server.

Port Portmap 111 adalah nomor port yang paling banyak diakses, namun umumnya nomor port ini telah difilter oleh firewall. Kecenderungan lain berpindah pada nomor port lain yang dapat digunakan untuk mengirim pesan tertentu yang telah dimanipulasi, seperti port 135, 139, 445, 593 pada remote komputer. Melalui port-port ini maka seorang user dapat melakukan permintaan yang dapat mengekploitasi dengan menjalankan kode dengan hak sistem lokal. Berikut ini tabel protokol yang sering dimanfaatkan pada ekploitasi RPC :

Urutan Protokol yang digunakan oleh Endpoint Mapper	Port TCP atau UDP
ncacn ip tcp	TCP/135
ncacn ip udp	UDP/135
ncacn np pipe emapper	TCP/139 dan TCP/445
ncacn http	TCP/593
ncacn http dengan servis COM Internet aktif	TCP/180

Tabel 2. Port dan Protokol yang umum dieksploitasi

Masalah ini semuanya disebabkan oleh kelemahan pada servis RPCSS. Servis ini berhubungan dengan aktivasi DCOM. Kegagalan terjadi pada penanganan messages yang salah sehingga mempengaruhi aktivasi DCOM yang mendengarkan port UDP 135, 137, 138, 445 dan port TCP 135, 139, 445, 593. Ditambah port 80 dan 443 (CIS atau RPC over HTTP) jika diaktikan. Dengan kesalahan ini, maka seorang klien dapa menggunakan kegagalan ini untuk mengeksekusi kode yang dapat dijalankan pada server.

Pada makalah ini akan dibahas secara khusus eksploitasi pada sistem operasi Microsoft Windows dan variannya.

3.2 Implementasi Eksploitasi RPC

Pada implementasi ini digunakan beberapa tools yang digunakan untuk melakukan eksploitasi pada RPC. Pada contoh kasus ini, implementasi dilakukan pada Local Area Network (LAN) dengan host yang menggunakan sistem operasi Microsoft Windows XP. Eksekusi eksploitasi ini dibagi menjadi beberapa tahap. Tahap pertama adalah deteksi sistem yang memiliki kelemahan (*vulnerability*). Tahap kedua adalah penyerangan terhadap sistem tersebut. Biasanya pada tahap ini digunakan program yang didesain khusus untuk mengeksploitasi RPC. Pada contoh kasus ini digunakan program bernama "Kaht". Tahap terakhir adalah tahap eksekusi kode pada sistem yang telah ter-remote. Pada tahap ini,

seorang penyerang dapat melakukan apa saja mulai dari pembuatan account user dengan hak administrator sampai pengambilan data pada sistem.

3.2.1 Deteksi Sistem

Tools yang digunakan pada tahap ini adalah RPCScan v.2.03 buatan Foundstone.inc dan digunakan pada sistem operasi Microsoft Windows. Sebetulnya tujuan awal penggunaan perangkat ini adalah untuk mendeteksi sistem yang memiliki kelemahan keamanan bagi para Administrator, namun dapat dipakai untuk tujuan lain. Software ini dapat mendetaksi sistem operasi yang memiliki kelemahan yaitu pada layanan DCOM (Distributed Component Object Model). Sesuai dengan penjelasan pada bagian sebelumnya, layanan DCOM merupakan komponen yang berhubungan dengan servis RPC.

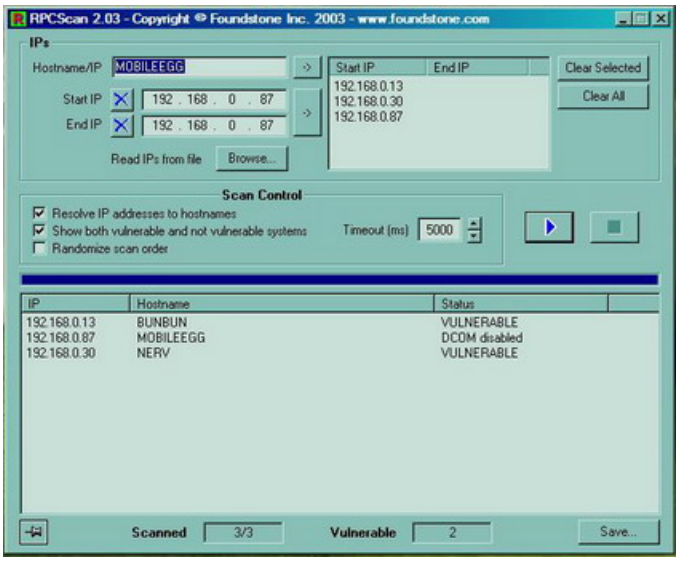

Figure 3.1: Sistem yang memiliki Kelemahan

Sistem yang masih mengaktifkan layanan DCOM akan diberi status "Vulnerable" dan sebaliknya akan diberi status "DCOM Disabled"(Figure 3.1). Batasan pen-scan-an pada perangkat ini terbatas pada jaringan dengan protokol IP v4. Namun mengingat masih

banyaknya sistem yang menganut protokol ini, maka implementasi pengecekan sistem dapat terjadi pada siapa saja. Serangan akan ditujukan pada sistem dengan keterangan Vulnerable.

3.2.2 Eksploitasi Protokol RPC menggunakan Program

Pada tahap ini akan dilakukan eksploitasi protokol RPC, agar terjadi kekeliruan penanganan message dari penyerang (host1) ke sistem yang diserang (host2) sehingga host1 dapat melakukan eksekusi kode pada host2. Tools yang digunakan pada tahap ini adalah program bernama "Kaht". Program ini akan mengeksploitasi port 135 dari sistem yang terserang. Sistem yang akan diserang adalah sistem dengan IP "192.168.0.13". Berikut ini tampilan eksekusi dari program ini :

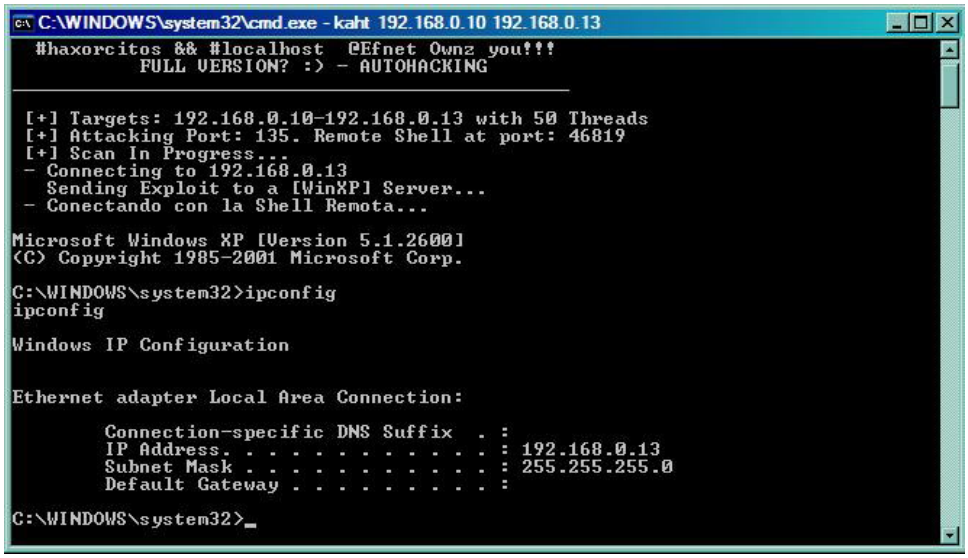

Figure 3.2: Penyerangan Pada Sistem

Gambar diatas menyatakan bahwa host1(IP 192.168.0.87) telah berhasil masuk kedalam sistem remote pada host2 (IP 192.168.0.13). Keberhasilan eksploitasi ini juga dapat dilihat

pada kondisi port-port yang terhubung antara host1 dan host2. Sebelum tereksploitasi, tidak ada port TCP pada host1 yang terhubung dengan siapapun. Setelah ekploitasi pada host2 berhasil dilakukan, maka terlihat bahwa pada port 135 terjadi koneksi / koneksi sedang berjalan dan tersambung (Figure 3.3). Dengan demikian maka pengeksekusian kode pada host2 oleh host1 dapat dilakukan. Sampai tahap ini, host1 hanya merupakan user dengan hak biasa saja. Berikutnya adalah implementasi pengeksekusian kode misalnya membuat account user baru dengan hak administrator.

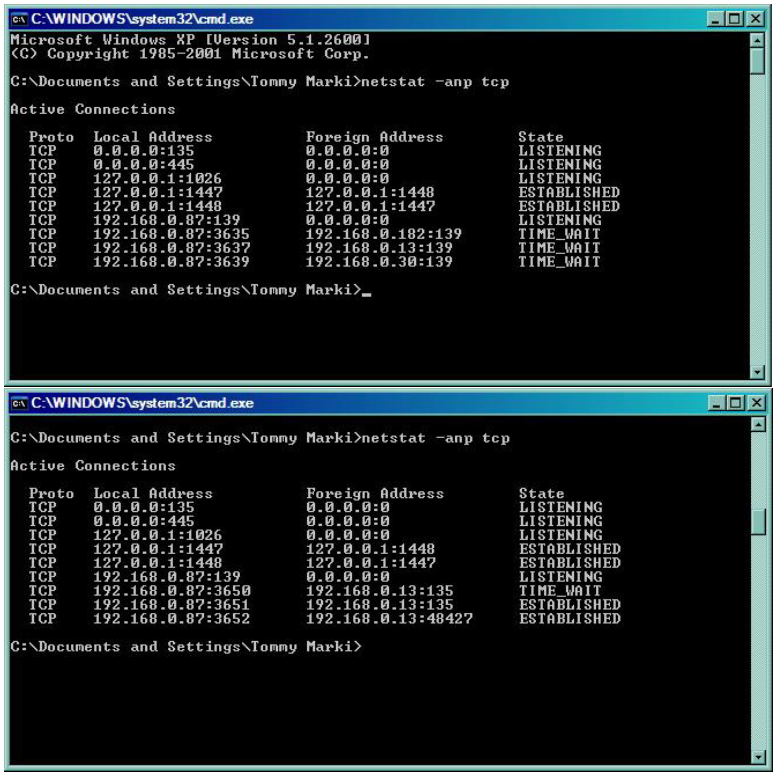

Figure 3.3: Sebelum dan Sesudah Eksploitasi

3.2.3 Eksekusi Kode

Tahap ini merupakan tahap eksekusi kode yang diinginkan. Pada tahap ini host1 dapat melakukan melakukan apa saja. Pada contoh kasus ini akan dicoba untuk membuat sebuah account baru dengan hak seorang administrator. Setelah masuk pada sistem host2, maka gunakan perintah berikut untuk mengeksekusi pembuatan user baru :
net user tom2 /add

Kemudian hak user ini diubah menjadi hak Administrator :
net localgroup Administrators tom2 /add

Dengan demikian pada host2 akan didapatkan satu user baru bernama "tom2" dengan hak seorang administrator.

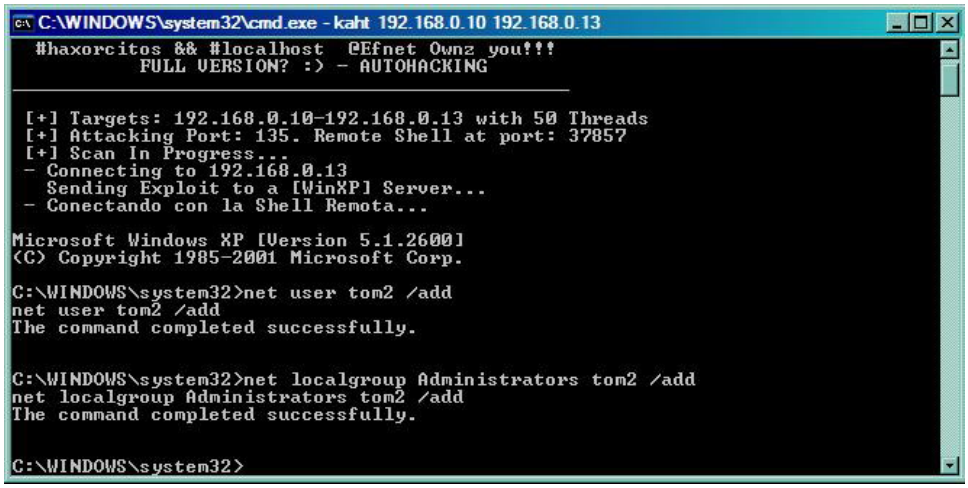

Figure 3.4: Pembuatan User

Implementasi ini dapat dilakukan untuk tujuan baik, misalnya pada kasus hilangnya password seorang user yang memiliki hak sebagai Administrator. Namun pada dasarnya

hal ini sendiri adalah kelemahan utama didalam sistem keamanan pada sistem operasi.

Selain pembuatan user baru, eksekusi kode yang dapat dilakukan lainnya antara lain upload atau download data dari dan ke host2, perusakan sistem, dll.

3.3 Pencegahan Eksploitasi RPC

Pentingnya nilai informasi yang terkandung dalam suatu server atau sebuah komputer pribadi mengharuskan penggunanya baik user atau administrator harus melakukan tindakan preventif guna menghindari adanya kebocoran informasi ini. RPC sendiri adalah suatu layanan yang vital dan ada di dalam setiap sistem operasi, khususnya sistem operasi Microsoft Windows yang bergantung pada jenis layanan ini. Oleh karena itu menghilangkan layanan RPC pada sistem operasi terasa lebih susah ketimbang dilakukan pengawasan ekstra terhadap layanan ini. Berikut ini adalah beberapa cara yang dapat digunakan untuk melakukan hal ini (dikutip dari buletin keamanan Microsoft MS03-026):

1. Memblokir port 135, 137, 138 dan 445 pada UDP dan port 135, 149, 445, dan 593 pada TCP melalui Firewall. Disfungsikan COM Internet Services (CIS) dan RPC melalui HTTP yang menggunakan port 80 dan 443 terutama pada jaringan remote yang menggunakan VPN (Virtual Private Network) atau sejenisnya.

2. Gunakan personal Firewall seperti Internet Connection Firewall.

3. Blokir semua port sering diekploitasi dengan menggunakan filter IPSEC.

4. Disfungsikan fitur DCOM pada setiap komputer atau server. Disable DCOM on all affected machines

5. Khusus Sistem operasi buatan Microsoft, selalu update security Patch untuk meningkatkan keamanan sistem operasi tersebut.

Kesimpulan

Information is Power

Suatu sistem dibuat untuk mempermudah pengaturan di dalam suatu komunitas. Namun pengaturan ini perlu diperbaharui seiring dengan kemajuan teknologi dan jaman.Setiap sistem baru dirancang dengan kompleksitas yang tinggi guna memperbaiki setiap kelemahan yang terdapat di sistem sebelumnya. Namun di dalam kekompleksitasan inilah banyak terkandung kelemahan-kelemahan baru.

Sesuai dengan kutipan diatas, bahwa saat ini begitu tingginya nilai informasi sehingga banyak pihak yang merasa membutuhkan informasi yang belum tentu dan selayaknya dimiliki. Dianalogikan dengan sistem keamanan pada protokol RPC, dimana protokol ini awalnya digunakan untuk mempermudah adanya komunikasi antar klien server untuk aplikasi yang terdistribusi. Namun perkembangan berikutnya menunjukkan bahwa fungsi asli protokol ini digunakan pihak tidak bertanggung jawab untuk memperoleh suatu informasi yang bukan miliknya.

Simpulan terakhir adalah, sesungguhnya tidak ada sistem yang seratus persen aman dari kebocoran dan kelemahan. Yang ada adalah sistem yang belum teruji keamanannya. Oleh karena itu, sebagai seorang pemilik komputer personal atau seorang administrator sudah seyogyanya untuk terus menerus mengambil tindakan preventif agar sistem yang dijaganya tetap stabil dan terhindar dari kelemahan yan bisa dimanfaatkan orang lain.

Bibliography

[1] AIX Version 4.3 Communication Programming Concepts, *Remote Procedure Call Chapter 8*, Sun Microsystem.
`http://www.dlib.indiana.edu/doc_link/en_US/a_doc_lib/aixprggd/progcomc/ch8_rpc.htm.`

[2] Internet Security Systems, *PRC Signature Quality, ISS X-Force White Paper*, Juni 2002.

[3] Newmarch, Jan, *Remote Procedure Call*, 1995.
`http://jan.netcomp.monash.edu.au/OS/l14_1.html`

[4] Microsoft Security Buletin MS03-026, *Buffer Overrun In RPC Interface Could Allow Code Execution (823980)*, 2003.
`http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx,`

[5] Nicholas. Petreley, *Security Report: Windows vs Linux*, Security White Paper - Reg Research, 2004.

[6] R. Spangler, *Analysis of the Microsoft Windows DCOM RPC Exploit*, Packetwatch Research, 2004.

[7] Sun Microsystem, *RPC Guide*, 1993. Sun Microsystems, Inc. 2550 Garcia Avenue Mountain View, California 94043