

TUGAS AKHIR
EC 7010 KEAMANAN SISTEM LANJUT

APLIKASI KEAMANAN PADA TRANSLASI BINER
DINAMIK



Dikerjakan Oleh :

Nama : Dwi Fendi Dadang Adrianto

NIM : 23203106

BIDANG KHUSUS TEKNOLOGI INFORMASI
PROGRAM STUDI MAGISTER TEKNIK ELEKTRO
PROGRAM PASCASARJANA
INSTITUT TEKNOLOGI BANDUNG
2004

Daftar Isi

Daftar Isi	2
Abstrak	3
Bab I Pendahuluan	4
I.1 Latar Belakang	4
I.1 Tujuan	5
I.1 Ruang Lingkup Masalah	6
I.1 Batasan Masalah	6
I.1 Sistematika Penulisan	6
Bab II Konsep Dasar Keamanan	8
II.1 Aspek-Aspek Sistem Keamanan.....	8
II.2 Motivasi dari Penyerangan Sistem Sekuriti	11
II.3 Tahap-Tahap Untuk Mengcrack	12
II.1 Batasan Masalah	6
I.1 Sistematika Penulisan	6
Bab III Translasi Biner Dinamis	25
III.1. SIND	25
III.2. DynamoRIO	27
III.2.1 Interface DynamoRIO	28
Bab IV Pendekatan Keamanan Translasi Biner Dinamis	30
IV.1. Verifikasi Alamat Pengirim Out-Of-Band.....	30
IV.2. Sistem Panggilan Sandboxing	35
Bab V Penutup	39
Daftar Pustaka	41

ABSTRAK

Komputer pada saat ini bukan lagi merupakan barang mahal yang hanya dimiliki oleh sekelompok orang saja, tetapi sudah merupakan perangkat yang dapat membantu dalam menyelesaikan pekerjaan seseorang. Perlu kiranya seseorang mengamankan dan melindungi data-datanya dari gangguan dan serangan virus yang datang dari dalam maupun luar sistem. Pada akhirnya security komputer dan jaringan komputer akan memegang peranan yang penting dalam pengamanan data.

Suatu translasi biner dinamik adalah suatu system yang dinamik merusak didalam mesin code fragmen yang mungkin langsung dieksekusi atau didalam interpreter software. Ketika mengesekusi interpreter, statistik-statistik dikumpulkan untuk mengetahui seberapa sering fragmen tersebut dieksekusi dan seberapa sering cabang-cabang diambil. Fragmen yang paling sering digunakan ditransformasi dan di cache untuk eksekusi berikutnya. Sekali fragmen berada didalam cache, mereka mungkin akan dimerger dengan fragmen yang lain ketika mereka ditemukan terpisah oleh arah percabangan tanpa syarat. Arsitektur ini digunakan untuk optimalisasi program dinamik, introspeksi instrument dan translasi arsitektur

Kata kunci: sekuriti, translasi, biner, dinamis, memory, rentan

BAB I

Pendahuluan

I.1 Latar Belakang

Sifat mudah terinfeksi virus adalah suatu kelemahan dari perangkat lunak yang memungkinkan program masuk ke suatu bagian tanpa diduga-duga. Suatu kelas yang besar dari sifat mudah terinfeksi virus merupakan kelemahan memori. Penempatan memori boleh berisi data yang disimpan dalam variabel lain atau bahkan struktur yang digunakan dalam environment bahasa pemrograman. Pengguna yang suka mengganggu boleh menulis isi dalam struktur runtime ini dengan nilai-nilai yang secara khusus dipilih, penyerang boleh mengalihkan program mengendalikan input kode mesin dinamis dan mengambil alih proses kerja.

Serangan “Stack-Smashing” yang terus-menerus adalah sebuah contoh dari suatu eksploitasi injeksi kode, suatu kelas dari penyerang yang menggunakan kelemahan memori yaitu sifat rentan atau mudah terinfeksi virus yang menulis terus-menerus dan merusak data dalam aktivasi penyimpanan dalam menjalankan program untuk mengerjakan serangan dengan injeksi kode mesin dinamis. Ada beberapa pendekatan untuk menghentikan atau mengurangi injeksi kode. Pendekatan yang paling alami adalah memperhatikan kelemahan memori yang mempunyai sifat mudah terkena serangan. Dengan menemukan dan menutup sifat mudah terinfeksi virus melalui berbagai analisa source program yang statis atau dengan penggunaan bahasa pemrograman yang lebih kuat atau tahan terhadap serangan virus, yang tidak mengizinkan mengakses memori yang tak aman. Perangkat otomatis dapat membantu dalam proses auditing kode, tetapi kurang akurat atau teliti. Pendekatan dinamis lain untuk mendeteksi ketika suatu memori mudah terinfeksi virus yang mencoba-coba untuk mengubah alur eksekusi dari proses pengerjaan yang berjalan. Pendekatan ini secara khas diterapkan ketika perluasan compiler yang menuntut source program aplikasi mengakses untuk membangun instrumentasi keamanan yang executable. Akhirnya pendekatan lain

yang mencoba untuk mendeteksi ketika kendali telah dialihkan dengan monitoring eksekusi dari proses yang menggunakan perluasan inti sistem operasi.

Banyak pendekatan ketika source program aplikasi atau sistem operasi tidak tersedia. suatu metoda implementasi alternatif untuk berbagai mekanisme keamanan dinamis yang beroperasi tanpa menuntut aplikasi atau source program sistem operasi disebut dengan translasi biner dinamis. Teknik ini dibangun pada awal pekerjaan optimisasi dinamis dalam menjalankan biner. Sistem yang serupa eksekusi run-time translasi untuk eksekusi foreign-architecture biner. Sistem translasi biner dinamis ini berbagi suatu pendekatan yang umum di mana fragmen kode mesin akan ditafsirkan untuk eksekusi langsung dan di mana fragmen yang frequently-used diubah dan diselamatkan untuk penggunaan kemudian membagi-bagi tempat menyembunyikan di mana mereka mungkin dikombinasikan dengan fragmen yang lain. Sistem translasi dinamis biner yang tersedia meliputi SIND dan DynamoRIO. Pendekatan yang diambil oleh sistem translasi biner dinamis dapat digunakan untuk eksekusi perubahan bentuk terkait dengan keamanan untuk mengurangi atau menghentikan eksploitasi injeksi kode. Seperti dengan penggunaan suatu sistem translasi biner yang dinamis, tambahan keamanan ini dapat diterapkan tanpa akses source program atau modifikasi aplikasi atau sistem operasi. Keuntungan ini memungkinkan sistem keamanan dinamis dengan cepat menyebar untuk melindungi suatu lingkup aplikasi yang lebih luas. Dalam beberapa hal, hal tersebut akan menyertakan sistem translasi SIND biner untuk menjalankan proses.

I.2. Tujuan

Berdasarkan latar belakang di atas *paper* tugas akhir EC 7010 ini bertujuan untuk mempelajari keamanan aplikasi translasi biner dinamis yaitu :

- a. mengetahui sifat rentan terhadap serangan virus pada memory
- b. mengetahui konsep dasar translasi biner dinamis, dan
- c. pendekatan keamanan menggunakan translasi biner dinamis.

I.3. Ruag Lingkup Masalah

Permasalahan yang ada di keamanan sistem komputer dapat dipandang relatif luas dan rumit. Permasalahan dalam paper tugas akhir EC 7010 ini adalah :

- a. sifat mudah terinfeksi virus suatu perangkat lunak yang mengijinkan program masuk ke suatu bagian tanpa diduga-duga dan merupakan kelemahan memory. Hal tersebut dikarenakan penempatan data pada memori dengan bentuk variabel lain atau bahkan struktur yang digunakan dalam enviroment bahasa pemrograman
- b. Serangan “Stack-Smashing” yang terus-menerus adalah sebuah contoh dari suatu eksploitasi injeksi kode, suatu kelas dari penyerang yang menggunakan kelemahan memori yaitu sifat rentan atau mudah terinfeksi virus yang menulis terus-menerus dan merusak data dalam aktivasi penyimpanan dalam menjalankan program untuk mengerjakan serangan dengan injeksi kode mesin dinamis.

I.4 Batasan Masalah

Untuk memberikan penekanan khusus sesuai dengan judul tugas ini maka dilakukan pembatasan pada penulisan paper ini:

- a. Pembahasan tentang translasi biner dinamis
- b. Pembahasan tentang pendekatan keamanan translasi biner dinamis

I.5 Sistematika Penulisan

Penulisan dokumen ini disusun dalam beberapa bab, dimana tiap bab dapat dibaca sebagai satu kesatuan informasi tanpa perlu tergantung pada bab sebelumnya. Sehingga pembaca dapat membaca informasi dari bab tertentu saja. Adapun sistematika penulisannya disusun sebagai berikut:

Bab I Pendahuluan berisi latar belakang, maksud dan tujuan, rumusan masalah, pembatasan masalah, dan sistematika penulisan ini sendiri. Bab ini memberikan gambaran umum mengenai isi dari dokumen.

Bab II Konsep dasar keamanan

Bab III Analisis masalah yang translasi biner dinamis

Bab IV Pendekatan keamanan translasi biner dinamis

Bab V Penutup berisi kesimpulan

BAB II

KONSEP DASAR KEAMANAN

II.1. Aspek-Aspek Sistem Keamanan

Masalah keamanan dan kerahasiaan data merupakan salah satu aspek yang penting dalam system informasi. Seseorang akan merasa kesal apabila data penting yang dimiliki dalam komputernya tiba-tiba hilang. Maka akan ada rasa curiga terhadap orang-orang yang berada disekitarnya, kadang tidak tahu siapa pelakunya. Data hilang dapat disebabkan karena disengaja maupun tidak sengaja atau penyebab dapat datang dari dalam maupun dari luar. Berkaitan dengan hal tersebut maka perlu adanya keamanan data pada komputer yang kita miliki. Lingkup keamanan data dari suatu system komputer menampilkan bermacam-macam komponen dalam network security yang selalu penuh syarat dengan pendekatan teknologi sekarang. Komponen-komponen tersebut terbentuk dalam program aplikasi seperti : Indentifikasi, Ontentikasi, Otorisasi, Acces Control, Confidentiality (Kerahasiaan), Data Integrity, Nonrepudiation, Denial Service.

a. Indentifikasi (Identification)

User mengindentifikasi ke program aplikasi melalui user identifier atau user ID. User id biasanya diisi dengan data-data singkat dan mudah dihafal. Untuk itu kita menampilkan gambar dibawah ini. Gambar ini menampilkan lebih dari satu aplikasi pada tiap server. Untuk tiap aplikasi, user biasa memiliki ID yang berbeda. Agar lebih ideal, seharusnya ada satu ID pada user untuk mengakses beerbagai aplikasi dalam network. Bagaimanapun juga ID membutuhkan koordinasi dari user, system administrator, dan security administrator.

b. Ontentifikasi (Aunthentication)

Mengontentifikasi adalah proses untuk mmeriksa indentifikasi apa yang dipakai oleh para user. Pemeriksaan ini membutuhkan pertukaran informasi dari aplikasi ke para user. Pada umumnya aplikasi menanyakan password yang banyaknya lebih dari 6 - 8 karakter. Users seringkali menyimpan password ditempat yang

aman dan tidak dapat diketahui orang. Untuk aplikasi ini password disimpan dalam database yang aman dan selalu tersedia dan siap digunakan untuk para users. Cara kerja database ini sangatlah sederhana yaitu seperti ini : user mengirim password ke aplikasi. Setelah password itu diterima aplikasi tersebut mengecek kembali identitas user dengan membandingkan dengan password yang diterima dari user dengan password yang tersimpan pada database user ID. Cara tersebut digunakan ketika user melakukan login dengan ID dan password pada workstation yang menghendaki untuk mengakses database. Database server menerima user ID dan password lalu database server meminta security server untuk menverifikasi ulang user ID dan password. Setelah itu security server membandingkan password dari database server dengan password yang disimpan dalam security sever. Jika 2 password tersebut cocok, security server memberitahu kepada sever database, bahwa user tersebut telah sah. Dan jika password tersebut tidak cocok maka database server tidak memperbolehkan user untuk masuk.

Proses diatas adalah untuk memeriksa indentitas user yang cocok dan proses tersebut dapat dinamakan one way authentication. Pengecekan ini dilakukan ketika memberikan bukti otentik kepada aplikasi, tetapi user tidak menanyakan keotentikan aplikasi tersebut. Two way authentication dapat dikatakan ketika kedua-duanya, yaitu user dan aplikasi memberikan bukti otentiknya masing-masing.

c. Otorisasi (Authorization)

Otorisasi adalah proses pemberian akses kepada user ID. Hak untuk mengakses meliputi spesifikasi seperti berikut ; seperti user meminta untuk membaca, menulis, atau memperbarui suatu file.

d. Akses Kontrol (Access Control)

Akses Kontrol menyinggung ke proses penyelenggaraan penggunaan akses untuk network resources. Acces Kontrol mempunyai hak untuk pemberian izin diterima atau ditolaknya izin user untuk mengakses resource dan melindungi resource dengan memberi batas akses untuk otentifikasi saja dan otorisasi users.

e. Kerahasiaan (Confidentiality)

Kerahasiaan adalah proses yang digunakan untuk melindungi informasi yang bersifat rahasia dari pihak yang tidak bertanggung jawab. Rahasia data membutuhkan perlindungan ketika disimpan maupun di transmisikan melalui network. Ketika disimpan secara lokal, data dapat dilindungi dengan mengenkripsikan atau dengan cara menyelenggarakan yang dilakukan Acces Kontrol. Ketika data rahasia tersebut ingin di transmisikan melalui network, data tersebut harus sudah terlebih dahulu dienkripsikan. Dalam mengimplementasi enkripsi, juga membutuhkan keamanan distribusi dalam mengenkripsi kata untuk si pengirim dan si penerima dalam mengenkrip data.

f. Data Integritas (Data Integrity)

Data integritas juga memberikan deteksi dari pihak yang tidak bertanggungjawab dalam memodifikasi data. Pada umumnya data integritas mendetect data apa yang telah dimodifikasi dan sedang berlangsung dalam transmisi. Seperti modifikasi bisa menghasilkan sebuah serangan atau error transmission. Files yang bersifat rahasia dalam server database bisa disimpan sebagai encrypted text. Sebagai detailnya, password pada security server harus disimpan dalam bentuk encrypted text agar dapat mencegah dari pencuri password dalam file. Untuk lebihnya, password bisa ditransmisikan hanya dalam bentuk encrypted text atau lebih baik tidak ditransmisikan sama sekali.

g. Tidak bertolakan (Nonrepudation)

Tidak bertolakan adalah kemampuan untuk memberikan bukti yang asli pada data atau bukti dari pengiriman data. Ini dapat dilindungi dari usaha untuk pengirim yang salah dalam pengiriman data, atau penerima yang menerima data yang salah. Tidak bertolakan atau Nonrepudiation seringkali dibutuhkan untuk mengatur transaksi yang bersifat komersial melalui internet.

h. Penolakan dalam pelayanan (Denial of Service)

Serangan penolakan dalam pelayanan adalah salah satu dari serangan crackers mengambil alih atau menggunakan resource, sehingga tidak dapat dipakai oleh

siapapun. Contohnya serangan seperti ini meliputi virus yang menggunakan memory dari system atau ketika crackers yang menyerang host dan mengambil alih host yang legitimate.

i. Keamanan networks (Secure Networks)

Pada umumnya client/server network bisa meliputi beberapa dari user dengan client workstation, server printer, server database, server directory, server security, dan server communication menuju internet atau network lainnya. Proses mendesign keamanan yang umum untuk sebuah client/server network harus menjalani berbagai tahapan. Ini meliputi bukti autentik dari users, mengamankan workstation users, menyelenggarakan akses kontrol, mengimplementasi dari distribusi pelayanan keamanan, melindungi network pribadi dari pihak yang tak bertanggung jawab dalam internet, dan mememanajemenkan keamanan data sebaik mememanajemenkan data.

II.2 Motivasi dari Penyerangan Sistem Sekuriti

Sejauh ini kita merasa beruntung, banyak dari yang kita tahu bahwa penyerangan system security bersifat terpisah dan sporadis . Motif dari penyerangan ini bervariasi. Pada umumnya ada empat type dari motif di belakang penyerangan system security diantaranya : memata-matai perindustrian tertentu, mengambil keuntungan dari perbankan , balas dendam, dan peringatan. Tipe pertama dari motif dibelakang penyerangan sistem security adalah memata-matai dunia perindustrian. Disini cracker masuk lewat file pribadi perusahaan , mencari celah-celah rahasia, dan mengirimkan file yang dibutuhkan kepada perusahaan saingan. Memata - matai perusahaan merupakan ancaman utama bagi perusahaan yang membuat produk untuk jangka panjang dan perusahaan yang membuat sumber daya dan pembangunan. Salah satu tujuan dari penyerangan adalah mencegah serangan cracker dari situs perusahaan lain yang mencoba mencari tahu tentang penyerangan cracker tersebut atau menghilangkan jejak. Pergerakan untuk suatu bank memperoleh keuntungan dapat juga dijadikan sebagai motif cracker untuk menyerang sistem security. Dalam kasus ini cracker berusaha untuk mencuri uang atau mencuri sumber-sumber penting untuk membuat uang. Para crackers

jumlahnya terus bertambah dan mereka terus berusaha untuk menambah uang mereka. Perusahaan yang curang dapat mentransfer uang mereka untuk para cracker yang membantu perusahaan tersebut. Penggelapan credit card dan penggelapan lewat telephone dapat juga dianggap sebagai penyerangan para crackers. Balas dendam merupakan salah satu alasan sebagai motivasi dibelakang penyerangan sistem security oleh para crackers. Rasa tidak puas para karyawan terhadap perusahaan dapat menjadi suatu boomerang bagi perusahaan itu sendiri contohnya bila karyawan tidak puas terhadap perusahaan, para karyawan dapat menyerang perusahaan dengan menyerang security sistem perusahaan itu sendiri. Dengan menginstall software time bomb atau virus sebelum mereka meninggalkan perusahaan tersebut . Ketika software time bom itu bekerja dengan meledakkannya sendiri atau virus bekerja /beroperasi, maka software atau virus itu dapat merusak kunci-kunci dan sumber-sumber yang penting dalam perusahaan tersebut. Tipe ke empat motif dibelakang penyerangan security sistem oleh para crackers adalah peringatan/pengumuman . Mereka telah menyerang via internet dengan sendiri ataupun oleh suatu grup. Dalam usaha ini mereka (crackers) menunjukkan kebolehan atau keunggulan mereka dalam meng-hack kepada seseorang ataupun para pengguna di seluruh dunia .

II.3 Tahap-Tahap Untuk Mengcrack

Mungkin kita akan melihat bagaimana seorang remaja mengacau balaukan berbagai sistem komputer perusahaan raksasa yang jahat, mengadakan duel hidup-mati di cyberspace, dan akhirnya menyelamatkan dunia. Di dunia nyata ceritanya tidak seheboh itu. Tapi memang benar, tiap cracker, yang baik maupun yang buruk, pasti melakukan penyusupan. Aktivitas orang yang akan meng-hack suatu sistem komputer (sistem sekuriti) sebenarnya dapat dibagi menjadi 4 tahapan :

- a. Mencari sistem komputer untuk dimasuki (mengumpulkan informasi).
- b. Menyusup masuk
- c. Menjelajahi system tersebut (termasuk mencari akses ke seluruh bagiannya).
- d. Membuat backdoor dan menghilangkan jejak.

a. Mencari Sistem Komputer

Pada masa-masa awal cracker, komputer masih jarang, dan tidak semuanya terhubung ke dalam jaringan. Akibatnya, cracker-cracker di masa itu mencari system dengan cara menghubungi nomor-nomor telepon yang dicurigai terhubung ke jaringan dengan modem mereka. Cara praktisnya adalah dengan mencari nomor telepon perusahaan besar (yang kemungkinan besar memiliki jaringan), lalu menghubungi nomor-nomor yang berdekatan hingga menemukan nomor yang memberikan sinyal carrier (berarti terhubung ke suatu modem computer juga). Ini dapat dilakukan secara manual, tapi agar lebih mudah sering dilakukan dengan program khusus yang disebut prefix scanner, Demon Dialer, atau War Dialer. Contohnya program ToneLoc.

Di zaman internet ini, sebagian besar computer sudah terhubung ke dalam Internet, maka sepertinya cracker-cracker sekarang lebih umum untuk mencari sasaran lebih umum untuk mencari sasaran mereka di antara computer-komputer host yang ada (lebih gampang dibandingkan metode pertama). Pada masa sekarang ini, yang perlu dicari bukanlah komputernya, tapi pintu masuk yang bisa dimanfaatkan dalam sistem komputer itu atau system security computer itu. "Pintu masuk " ini berupa port, yaitu jalur-jalur keluar-masuknya data dari dan ke suatu komputer. Pengaksesan komputer melalui port ini disebut "port surfing". Pencarian port yang dapat digunakan biasa dilakukan dengan program khusus yang disebut port scanner. Contoh dari port scanner ini adalah Rebellion, PortPro dan PortScanner (mampu memeriksa sekelompok alamat IP untuk mencari port yang terbuka). Port yang sering terbuka misalnya port 23 (telnet), 43 (whois), 79 (finger), 25 (Simple Mail Transfer Protocol). Servis-servis yang sering terdapat ini kemudian dapat digunakan untuk mengumpulkan informasi lebih jauh tentang host yang akan dijadikan sasaran. Servis yang paling umum dipakai adalah finger. Finger, servis yang umum bagi system operasi UNIX (meskipun sudah ada versi Windowsnya, misalnya Finger32) adalah perintah yang dapat menampilkan informasi mengenai seorang pemakai jaringan. Cara kerjanya adalah sebagai berikut. Bila seseorang menjalankan program utilities finger client (misalnya Finger32 tadi) untuk mencari keterangan tentang A di sistem B, misalnya,

program finger client tadi akan mengirimkan permintaan ke finger daemon di sistem B.

Daemon, dalam konteks ini, bukanlah makhluk merah yang berekor dan bertanduk (dan dihidungnya ada angka 666) itu. Daemon adalah program yang ditempatkan sebagai "penunggu" port-port pada host Internet, yang bertugas menjalankan perintah-perintah dari luar secara otomatis. Misalnya saja, mailer daemon, finger daemon. Finger daemon itulah yang kemudian akan mengirimkan informasi yang diminta ke orang tersebut. Kelengkapan informasi yang diberikan bisa berbeda-beda, tergantung konfigurasi sistem tersebut. Informasi yang umum ditampilkan misalnya login name, waktu login terakhir, dan nama pemakai. Finger juga bisa digunakan untuk melihat daftar pemakai dalam suatu sistem. Dalam hal ini, yang difinger adalah hostnya, bukan nama pemakai secara spesifik. Contoh perintahnya: **finger @host_sasaran.com**

Untuk alasan keamanan, banyak perusahaan/badan-badan yang menonaktifkan finger daemon pada sistem mereka (dengan mengedit file /etc/inetd.conf). Bila finger dilakukan pada system-sistem semacam ini, akan muncul pesan Connection Refused. Selain melakukan scanning terhadap port yang terbuka, selanjutnya juga biasa dilakukan scanning terhadap sistem secara umum. Biasanya untuk mengetahui jenis sistem operasi dalam komputer tersebut, tipe daemon, file share (NETBIOS pada sistem berbasis Windows). Contoh program scanner komersial yang menscan kelemahan sistem secara umum adalah Internet Security Scanner. Contoh lain yang populer, dan gratis, adalah COPS dan SATAN. Banyak program semacam ini, sebenarnya dibuat dengan tujuan membantu administrator sistem untuk menemukan celah-celah pada sistemnya (untuk kemudian diperbaiki). Bahkan, SATAN sendiri merupakan singkatan dari Security Administrator Tool for Analyzing Networks. Namun, dalam perkembangannya, program-program ini bisa diibaratkan sebagai pedang bermata dua, karena dapat juga dimanfaatkan orang yang hendak menerobos ke dalam system. Seorang cracker juga bisa melihat jumlah server yang ada di balik suatu domain name dengan perintah whois. Bila di balik suatu domain name terdapat beberapa mesin/suatu jaringan,

maka cracker juga akan berusaha melakukan pengumpulan informasi dan pemetaan terhadap jaringan tersebut. Misalnya saja dengan menggunakan perintah `showmount`, `rpcinfo`, melihat NFS export dari komputer-komputer yang menjalankan `nfsd` atau `mountd` untuk menemukan komputer-komputer yang memiliki hubungan "terpercaya" (trusted). Pencarian komputer-komputer yang saling berhubungan juga bisa dilakukan dengan melihat website dan dokumen-dokumen yang disediakan host yang bersangkutan. Website dan dokumen-dokumen semacam itu kadangkala memuat informasi tentang host-host lain yang "sekelompok" dengan host yang akan dimasuki. Beberapa dari perintah-perintah yang digunakan untuk eksplorasi tersebut memang biasanya hanya ada di UNIX. Tapi, Windows juga menyediakan beberapa perintah TCP/IP yang cukup bermanfaat, misalnya:

Arp

Perintah ini berguna untuk menampilkan address translation table dari Internet ke Ethernet. `tracert` (windows) / `tracert` (unix). `Tracert` (tracert) berguna untuk menampilkan rute yang ditempuh suatu paket ke tujuannya. Bisa digunakan untuk mengetahui adanya gateway-gateway antara beberapa daerah/jaringan.

ping

Berguna untuk mengetahui aktif/tidaknya suatu host. Cara kerjanya adalah dengan mengirimkan paket ICMP (Internet Control Message Protocol) ECHO-REQUEST. Bila host yang diping aktif, ia akan membalas dengan paket ICMP ECHO-REPLY. Ping juga akan memberitahukan waktu yang diperlukannya untuk mencapai host sasaran dan kembali lagi ke komputer kita.

netstat

Menampilkan informasi jaringan (jenis informasi tergantung parameter perintah).

nbtstat

Menampilkan informasi NETBIOS jaringan. Selain itu, masih banyak perintah lain yang sering digunakan, misalnya perintah `nslookup`. Dengan informasi yang diperoleh, seorang cracker bisa menyusun hubungan host yang dituju dengan

komputer-komputer lain, data-data umum, dan mungkin kelemahan-kelemahannya. Ini bisa mempermudah dalam memilih cara yang akan digunakan untuk menyusup masuk.

b. Menyusup Masuk

Inti dari kegiatan menyusup masuk ini adalah dengan menipu sistem pengamanan, biasanya berupa proteksi password. Seorang cracker dapat menembus sistem pengamanan dengan berbagai cara, antara lain :

- 1) Social engineering
- 2) Menebak password
- 3) Menyadap password
- 4) Mengeksploitasi kelemahan pada sistem sasaran
- 5) Trashing

1) Social Engineering

Social engineering, atau rekayasa sosial, atau "con", menurut istilah umum adalah cara yang cukup antik dan tidak banyak berhubungan dengan komputer. Sebenarnya, cara ini lebih umum digunakan oleh para penjahat komputer. Amat jarang cracker yang menyinggung hal ini. Praktek social engineering adalah dengan berpura-pura menjadi orang yang akan "dimanfaatkan" untuk menembus sistem (misalnya, seorang pekerja) atau seseorang dari dinas pemerintah. Modalnya adalah pengetahuan tentang orang yang akan ditiru selain modal nekat. Social engineering yang paling umum adalah melalui telepon, dengan modal kecepatan berpikir, pengetahuan, kemampuan menyamarkan suara (dapat dengan bantuan alat pengubah suara). Salah satu cara bertindak yang umum adalah berpura-pura menjadi seorang karyawan yang kehilangan password / gagal mengakses komputer kerjanya (agar diberi seperangkat password baru). Social engineering juga dapat dilaksanakan melalui pos, misalnya dengan berpura-pura menjadi sebuah perusahaan yang sedang mengadakan survei dengan mengirimkan daftar isian pada orang-orang yang menjadi sasaran. Setelah data-data penting diambil, data-data tersebut dapat digunakan untuk sarana social engineering lewat telepon.

2) Menebak password

Sesuai namanya, cara ini dilakukan dengan cara menebak kombinasi username dan password dalam sistem yang dijadikan sasaran. Dulu sekali, orang masih belum begitu sadar akan pentingnya password yang sulit ditebak, sehingga cara menebak password ini masih mudah untuk dilakukan secara manual. Sekarang, cara menebak password ini dilakukan dengan menggunakan program. Prinsip dasar program pemecah password dengan cara ini cukup sederhana. Biasanya, cracker yang ingin menggunakan program tersebut terlebih dahulu menyusun sebuah daftar kata yang sering digunakan orang. Bisa berupa nama aktor, nama tempat, dan sebagainya. Lalu, program pemecah password tersebut akan mencoba kombinasi dari kata-kata dalam daftar tersebut hingga mendapatkan kombinasi yang sesuai. Ada juga program yang mencoba seluruh kombinasi karakter yang mungkin, baik huruf, angka maupun karakter lain. Metode ini disebut "brute forcing". Beberapa program yang lebih baik, misalnya LophtCrack, memiliki fasilitas untuk melakukan baik brute forcing maupun penebakan berdasarkan daftar kata.

Hingga saat ini, cara menebak password masih banyak dipakai, dan program-programnya banyak tersedia, baik untuk NT, UNIX maupun DOS. Misalnya saja, program CrackerJack dan StarCrack. Program-program ini sebenarnya diciptakan untuk keperluan pengamanan sistem. Seorang administrator sistem biasanya memiliki beberapa program semacam ini, lengkap dengan seperangkat file daftar kata, untuk memastikan bahwa password-password dalam sistemnya sudah benar-benar kuat dan tidak mudah dibobol. Pada awalnya, penggunaan metode penebakan password memiliki eksese yang cukup jelek, yakni jumlah kegagalan login yang tinggi (akibat dari berbagai kombinasi yang salah) sebelum kombinasi yang sesuai ditemukan. Ini terutama terjadi pada penerapan metode "brute forcing". Jumlah login yang gagal selalu dicatat dalam log system, dan log ini hampir pasti diperiksa administrator secara rutin. Sebagai akibatnya, ini akan memberitahu administrator bahwa suatu upaya pembobolan tengah berlangsung. Oleh karena itu, para cracker sekarang akan berusaha menggunakan teknik-teknik tertentu untuk menyalin file daftar password dari sistem sasaran ke komputer

mereka sendiri, lalu mencoba memecahkannya dengan program penebak password saat off-line. Dengan demikian, kemungkinan terdeteksi menjadi lebih kecil.

3) Menyadap password

Menyadap password dapat dilakukan dengan berbagai cara. Cara yang umum sekarang adalah dengan menyadap dan memeriksa paket-paket data yang lalu-lalang dalam suatu jaringan (lazim disebut "sniffing" atau packet monitoring). Sniffing (mengendus/menghidu) biasanya dilakukan terhadap paket-paket data yang dikirim dalam bentuk teks biasa (tidak terenkripsi). Praktek sniffing yang lebih maju dibantu oleh sebuah protocol analyzer, yang untuk membaca paket data yang "dibungkus" protokol tertentu. Pemeriksaan paket-paket data dalam suatu jaringan sebenarnya adalah juga praktek yang bermanfaat di kalangan administrator sistem, untuk mengawasi gejala-gejala awal penyusupan. Namun, cara ini juga umum di kalangan cracker maupun penjahat komputer. Sebuah program pemonitor paket dapat ditugaskan untuk mencari paket-paket yang berisikan kata-kata kunci semacam "password", "login" dan sebagainya, lalu menyalin paket tersebut untuk dianalisa kemudian oleh cracker yang bersangkutan. Contoh program sniffer misainya Netmon, EtherPeek, LanWatch, tcpdump.

4) Mengeksploitasi kelemahan pada sistem sasaran

Di dunia ini, tidak ada yang sempurna. Demikian juga sistem komputer. Memang, kelemahan tiap sistem biasanya berbeda-beda, dan hanya orang yang mempelajari sistem tersebut secara mendalamlah yang mengetahuinya (kecuali bila ada yang mengirimkan informasi tentang kelemahan itu ke suatu forum). Contoh "lubang" pada sistem yang populer (meskipun bentuknya berbeda) adalah adanya default username dan password, yang sudah diprogram ke dalam beberapa sistem sejak di pabrik. Berikut adalah contoh default username yang sudah tersebar luas :

a) Sistem UNIX

Login Password

root root/system

sys system

daemon daemon

uucp uucp

tty tty

Sistem VAX/VMS

Username Password

SYSTEM OPERATOR

SYSTEM MANAGER

SYSTEM SYSTEM

SYSTEM SYSLIB

Selain default, kesalahan pada program juga dapat dieksploitasi oleh para calon penyusup. Program-program versi awal maupun yang cukup rumit seringkali memiliki bug yang dapat membuat program tersebut macet/crash lalu "membuang" pemakainya ke sistem operasi. Program lapis luar (menjalankan servis untuk umum) yang "bermasalah" dan memiliki banyak perintah pengaksesan ke bagian dalam suatu jaringan dapat macet dan "meleparkan" seorang pemakai ke dalam jaringan bila program tersebut kebetulan "kumat". Memori (stack buffer) sering menjadi titik lemah (dan sasaran utama cracker) dalam sistem-sistem demikian. Buffer overflow, misalnya pada server POP (Post Office Protocol) dengan qpopper dari Qualcomm, bisa digunakan untuk langsung mengakses direktori root. Pada beberapa perangkat lunak lain, biasanya para cracker menciptakan kondisi buffer overflow untuk mendapatkan sebuah shell (sebagai hasil "proses anak"/child process) dalam sistem terproteksi, yang bisa digunakan untuk hacking lebih jauh. Kadangkala seorang cracker sengaja mengirimkan input data atau rangkaian perintah yang dirancang sedemikian rupa untuk mengeksploitasi kelemahan sistem ini. Selain kesalahan pada programnya sendiri, seorang cracker bisa juga mencari kelemahan akibat kesalahan konfigurasi program. Misalnya pada anonymous FTP (File Transfer Protocol),

yang bila tidak dikonfigurasi dengan benar, bisa dimanfaatkan untuk mengambil file-file penting (misalnya /etc/passwd) . Hal yang mirip adalah pada tftp (Trivial File Transfer Protocol), daemon yang dapat menerima perintah transfer file tanpa adanya proteksi password. Contoh yang lebih tua lagi adalah eksploit PHF, yang dulu sering digunakan untuk menghack

b) Situs Web

Beberapa program scanner yang telah disebutkan di atas (misalnya SATAN) juga bisa mendeteksi adanya daemon-daemon dalam port, dan apakah daemon-daemon tersebut dari versi yang "bermasalah". Cara lain yang cukup umum adalah dengan memanfaatkan script CGI (Common Gateway Interface). CGI adalah antarmuka yang memungkinkan komunikasi antar program klien dan server. Script CGI sebenarnya sering digunakan untuk pembuatan efek-efek pada webpage, namun dalam kaitannya dengan keamanan, sebuah script CGI juga memungkinkan akses file, penciptaan shell, maupun pengubahan file secara ilegal. Eksploitasi CGI ini sebenarnya bukan merupakan kesalahan pada bahasa penulisan scriptnya, tapi merupakan teknik pemrograman yang cerdas (biasanya dengan manipulasi validasi input).

Selain memori dan penggunaan skrip, ada kalanya cracker menemukan cara untuk mengeksploitasi kelemahan yang muncul dari feature baru suatu program. Misalkan saja pada server berbasis Win32. Win32, seperti kita ketahui, mendukung dua macam penamaan file. Penamaan yang panjang (misalnya namafilepanjang.com), dan penamaan pendek (namafi~1.com), yang berguna untuk menjaga kompatibilitas dengan sistem operasi lama. Nah, dalam kasus ini, kadang-kadang seorang administrator hanya memprotek file berdasarkan nama pendeknya (pada contoh di atas, namafi~1.com). Bila hal itu terjadi, cracker bisa mengakali proteksi tersebut dengan mengakses file itu menggunakan nama panjangnya (namafilepanjang.com). Kelemahan lain yang umum dikenal adalah eksploitasi terhadap fasilitas file sharing (terutama pada Windows NT). Fasilitas file sharing, bila dikonfigurasi dengan kurang baik, memungkinkan sembarang orang untuk mengakses data-data (apapun!) dalam komputer dengan mudah. Di

tingkat perangkat keras, evolusi perangkat keras juga menimbulkan berbagai celah baru. Beberapa perangkat keras jaringan modern, yang secara fisik bukan merupakan komputer, sebenarnya berisi komponen yang fungsinya menyerupai komputer. Misalkan saja printer. Beberapa printer berisi CPU SPARC, dan menjalankan sistem operasi UNIX (Solaris UNIX), sehingga secara teoritis memiliki user account sendiri, dan bisa dihack, sama seperti komputer biasa

5) Trashing

Cara lain, selain social engineering, yang tidak melibatkan komputer. Trashing, sesuai namanya, adalah metode untuk mengumpulkan informasi (password) dengan cara membongkar kertas-kertas/dokumen buangan dari instansi/perusahaan sasaran. Meskipun nampaknya kuno, tapi cara ini kadang-kadang masih berhasil. Salah satu kasus yang tercatat adalah kasus cracker "Control-C", anggota Legion of Doom yang diburu Michigan Bell hingga akhirnya tertangkap pada 1987. Namun nasibnya tidak jelek - ia sendiri mendapatkan pekerjaan di bidang keamanan komputer di Michigan Bell, sementara wajahnya menghiasi poster-poster internal Michigan Bell (ia bahkan juga menandatangani poster-poster tersebut), dengan pesan peringatan pada karyawan untuk selalu menghancurkan sampah-sampah kertas mereka. Menjelajahi sistem komputer (dan mencari akses yang lebih tinggi) Setelah seorang cracker sejati berada dalam sistem, ia kemungkinan besar akan berkeliling, melihat-lihat isi dari sistem yang baru saja dimasukinya, dan mencoba berbagai perintah untuk mengetahui fungsinya. Salah satu contoh perintah yang paling sering digunakan (dalam UNIX) adalah perintah ls. Perintah ini serupa dengan dir dari DOS, yang gunanya untuk melihat isi direktori. Contoh hasil output dari perintah ls ini adalah :

```
drwx--x--x 5 art delapan 433 Oct 18 12:30 daftar
drwx--x--x 3 opique delapan 512 Oct 24 10:00 lotte
-rwx--x--x 1 opique delapan 311 Dec 22 18:22 web
-rwxrwxrwx 6 airboy delapan 240 Dec 23 10:03 dokumen
```

Dalam tampilan di atas, huruf paling kanan menunjukkan jenis (d untuk direktori dan - untuk file,), 3 huruf berikutnya menunjukkan izin file bagi pemakai

(r berarti izin menulis, w izin membaca, x izin menjalankan/execute). 3 huruf berikutnya adalah izin file bagi kelompok pemakai, dan 3 huruf terakhir adalah izin file bagi umum (dalam contoh di atas, file dokumen dapat ditulis, dibaca dan dijalankan oleh siapa saja). Untuk hasil yang maksimal, perintah ls ini diikuti dengan option -a (menampilkan semua file, termasuk yang berawalan titik) -d (menampilkan direktori sebagai file biasa—tidak menampilkan isinya) dan -l (memberikan tampilan yang lengkap—ukuran dsb. Perintah lain yang banyak digunakan adalah perintah man, yang digunakan untuk menampilkan manual online dari suatu perintah. Sintaksnya adalah man (nama perintah). Perintah-perintah yang umum lain misalnya:

- " pwd (menampilkan direktori aktif)
- " whereis/find (untuk menentukan letak program tertentu)
- " who (menampilkan nama pemakai yang online)
- " cp (menyalin file)
- " cat (menampilkan isi file)
- " vi (editor)
- " cd(mengubah direktori aktif)
- " chmod (mengubah izin file)

Setelah beberapa lama berputar-putar di dalam sistem, mungkin seorang cracker akan melihat bahwa aksesnya amat terbatas, sebatas akses yang dimiliki orang yang account-nya ia gunakan. Ia pasti akan berusaha mencari akses tertinggi (superuser), yang memungkinkan ia melakukan apa saja di dalam sistem yang ia masuki. Pencapaian akses tertinggi ditandai dengan diizinkan-nya cracker tersebut untuk mengakses direktori akar/root pada sistem tersebut. Pada UNIX ditunjukkan dengan prompt #, dan ini tidak akan didapat dengan mudah, kecuali pada komputer milik sendiri atau sistem yang administratornya kurang sadar keamanan.

d. Membuat Backdoor dan Menghilangkan Jejak

Seorang cracker yang ahli akan selalu berusaha agar aksinya dan keberadaannya tidak diketahui oleh pemilik sistem yang dimasukinya. Kenapa? Sebab urusannya akan panjang bila ia tertangkap. Sebagian orang menganggap cracker sebagai teroris informasi, sebagian lagi menyamaratakan seluruh cracker sebagai penjahat

komputer. Pokoknya, pasti ujungnya tidak enak. Bagaimana cara cracker menghilangkan jejak? Berkaitan dengan waktu, cara untuk memperkecil kemungkinan terdeteksi adalah dengan melakukan aktivitasnya di saat sistem yang akan dimasukinya tidak atau kurang diawasi. Misalnya pada waktu subuh, atau akhir minggu. Berkaitan dengan teknik, salah satu cara yang paling umum adalah mengedit file-file log (catatan aktivitas) pada sistem yang dimasuki, dan menghilangkan semua entry yang berkaitan dengan dirinya. Aktivitas yang berlangsung selama hacking, misalnya saja aktivitas scanning, bisa disamarkan dengan memasang kuda troya/trojan ke program-program yang biasa digunakan untuk mengawasi jaringan (misalnya netstat). Untuk penyembunyian identitas saat melakukan finger, dapat digunakan dengan perintah semacam : **finger @host1 @host2**, Ini adalah cara untuk memperoleh informasi finger tentang host2, tapi perintahnya dialihkan sehingga seolah-olah datang dari host1. Cara lain untuk menyamarkan identitas saat beraksi misalnya dengan melakukan bouncing. Bouncing adalah memanfaatkan suatu sistem sebagai "basis operasi" untuk memasuki sistem lain. Dalam praktek bouncing ini, jejak-jejak akan mengarah ke komputer yang dijadikan "basis operasi" dan bukan ke lokasi cracker sebenarnya. Bouncing dapat dilakukan melalui FTP, mesin proxy server, wingate, atau host lain (dengan memanfaatkan telnet atau rsh). Selain melalui Internet, bouncing juga bisa dilakukan melalui manipulasi hubungan telepon. Bouncing semacam ini bisa dibilang sukar sekali dilacak, tapi tingkat kesulitannya juga tinggi. Manfaat bouncing, selain untuk menyulitkan pelacakan, sebenarnya juga untuk melewati berbagai program proteksi. Katakanlah komputer A (sebuah web server) dan B (FTP server) berada dalam satu jaringan/subnet yang terhubung ke Internet. A diproteksi terhadap akses dari Internet oleh sebuah firewall. Cracker yang ingin masuk ke komputer A dapat memerintahkan komputer B (yang terbuka untuk umum) untuk melakukan koneksi ke port tertentu dari A. Koneksi ini kemudian bisa dimanfaatkan oleh cracker. Karena B dan A ada di dalam subnet yang sama, maka lalu-lintas data yang terjadi tidak akan disaring oleh program firewall (yang hanya akan menyaring paket dari luar subnet).

Tentunya, selain berpikir untuk menghilangkan jejak, cracker juga perlu memikirkan cara untuk kembali ke sistem tersebut. Tapi tidak mungkin bila tiap kali ingin masuk ia harus mengulang semua proses di atas? Nah, untuk tujuan inilah cracker menggunakan apa yang disebut backdoor (pintu belakang). Backdoor, pada prinsipnya, adalah "jalan tembus" yang dibuat cracker setelah masuk, yang berguna untuk kembali tanpa perlu melalui sistem proteksi lagi. Contoh pemasangan backdoor misalnya adalah modifikasi file `hosts.equiv` dan `.rhosts` (pada sistem UNIX). Modifikasi pada file-file tersebut memungkinkan komputer host cracker diberi status `trusted` oleh sistem sasaran, hingga pengaksesan dari komputer cracker tidak akan "disaring". Backdoor juga bisa dibuat dengan menambahkan account-account baru pada file daftar password (`/etc/passwd` pada UNIX standar). Alternatif dari penambahan account ini adalah pembuatan daftar password palsu, yang kemudian akan "ditukar" tempatnya dengan file password asli pada waktu-waktu tertentu (dengan bantuan cron-fasilitas administrator waktu pada UNIX). Selain cara-cara itu, penyalinan SUID shell ke direktori yang mudah diakses (misalnya `/tmp`) juga sering digunakan. Untuk mencegah kecurigaan, nama shell ini akan diubah. Misalkan saja dari `/bin/csh`, disalin menjadi `/tmp/.data01`. Satu lagi cara pembuatan backdoor yang umum, adalah dengan mengubah konfigurasi servis. Konfigurasi tidak begitu sering diperiksa (berbeda dengan daftar password dan log), sehingga kemungkinan terdeteksinya lebih kecil. File-file yang dimodifikasi misalnya `/etc/inetd`. Modifikasi ini biasanya dilakukan dengan cara mengganti daemon dari servis yang jarang digunakan dengan suatu proses yang akan memberikan shell bagi cracker (terutama akses ke root).

BAB III

Translasi Biner Dinamis

Translasi biner dinamis adalah suatu sistem dinamis untuk menguraikan eksekusi kode mesin. Membagi-bagi yang akan dieksekusi secara langsung dalam interpreter perangkat lunak. Saat eksekusi dalam suatu interpreter, angka-angka statistik dikumpulkan terhadap fragmen yang sering dieksekusi dan bagaimana sering cabang spesifik di fragmen diambil. Fragmen yang sering digunakan diubah untuk eksekusi cache berikutnya. Waktu fragmen-fragmen berada dalam cache, mereka akan digabungkan dengan fragmen-fragmen lain ketika mereka ditemukan terpisah oleh suatu percabangan tanpa syarat. Arsitektur umum ini digunakan untuk optimisasi program yang dinamis, introspeksi, instrumentasi, dan translasi arsitektur. Beberapa proyek terkemuka yang menggunakan translasi biner dinamis adalah SIND, DynamoRIO dan lainnya.

III.1. SIND

SIND adalah suatu frame open-source multi-platform untuk translasi biner yang pada saat ini dalam proses pengembangan. SIND secara umum menggunakan sistem arsitektur translasi biner dinamis, tetapi mempergunakan modul dan didesain berorientasi objek, untuk dilakukan riset pengembangan aplikasi dan teknik frame yang mudah dan fleksibel untuk translasi biner dinamis di masa datang. Sekarang target platform Solaris adalah pada SPARC dan Mac OS X pada PowerPC.

Sistem SIND terdiri dari beberapa modul, yang ditunjukkan pada Gambar 3.1.

Dispatch Interp
Bootstrap
Transformer
Fragment Cache
Transformer
Memory
Manager
Running
Binary

Gambar 3.1: modul SIND

Modul-modul utama SIND adalah bootstrapper, dispatcher, interpreter, memory manager, code transformers, dan cache fragment. Salah satu tujuan dalam pengembangan SIND adalah untuk evaluasi dari teknik translasi biner dinamis. Oleh karena itu, banyak modul diterapkan penggunaannya untuk beberapa pendekatan dari bermacam-macam kompleksitas dan efisiensi. Ini akan memungkinkan sistem yang dibangun menggunakan SIND menggunakan implementasi yang paling cepat kecuali di mana suatu implementasi yang lebih rumit diperlukan oleh sistem translasi biner dinamis. Pendekatan ini juga memungkinkan implementasi modul dioptimalkan menggunakan feature platform-specific supaya lebih mudah dibawa tetapi mempunyai implementasi yang lebih lambat. Bootstrapper adalah kontrol mekanisme aplikasi yang diasumsikan oleh sistem SIND. Sekarang ini, tiga teknik diterapkan penggunaannya fasilitas proses mencari kesalahan dari program, memproses file system, dan fasilitas penghubung yang dinamis dari sistem operasi Solaris. Ketika perangkat breakpoint diaktifkan, keadaan aplikasi disimpan dan diperbaiki dalam interpreter perangkat lunak dimana eksekusi dikerjakan. Teknik yang ketiga menggunakan suatu share library. Dimana berlangsung inisialisasi instrumen yang dapat dieksekusi untuk diterima dan dikontrol ke SIND, ketika eksekusi dilakukan maka memasukkan nilai dimulai pula. Pekerjaan pemetaan halaman yang berisi symbol-simbol suatu sinyal handler yang dapat dieksekusi dan registerisasi untuk pegangan ketika halaman di eksekusi mencoba tombol kontrol penghubung antara eksekusi kode aplikasi secara langsung pada prosesor dan dalam interpreter perangkat lunak. Halaman juga bertanggung jawab atas diterapkan identifikasi jejak untuk menciptakan fragmen, menentukan fragmen untuk ditempatkan di dalam cache, dan ketika fragmen yang sering dieksekusi dihubungkan ke dalam suatu super-fragment. Halaman juga bertanggung jawab untuk memilih modul implementasi yang sesuai, didasarkan dengan sisi-sisi istimewa dalam sekumpulan platform. SIND interpreter adalah suatu set perintah arsitektur interpreter yang berbasis perangkat lunak. Tidak sama dengan arsitektur emulator, interpreter hanya menginterpretasikan aliran instruksi yang sebenarnya dan efisiensi, mungkin tanpa menirukan detail level yang lebih rendah dari arsitektur yang sama. Ini akan disederhanakan dengan hanya menginterpretasikan mode perintah pengguna.

Selama penafsiran, interpreter mengumpulkan nilai statistik dan profil data dengan kode fragmen untuk penggunaan potensial karena perubahan kode. Pengelolaan memori mengontrol pemisahan pembacaan dan penulisan memori dalam alamat ruang dari aplikasi ketika bekerja dalam interpreter. Pemisahan ini memungkinkan SIND untuk beroperasi dalam alamat ruang yang berbeda dari aplikasi jika diinginkan di mana akses yang timbal balik ke data bisa disajikan bersama lewat memori. Yang dilakukan akan sangat melindungi struktur SIND dan cache fragmen dari suatu aplikasi yang berpotensi merusak. Sebagai alternatif, pengelola memori bisa dengan jelas mengizinkan SIND dan aplikasi untuk berbagi alamat ruang bersama. Penggunaan data yang dikumpulkan selama melacak penafsiran kode, pemindahan kode SIND menulis kembali aliran perintah untuk peningkatan kemampuan, monitoring, atau keamanan aplikasi. Pemindahan yang beragam adalah rangkaian yang secara bersama-sama melaksanakan beberapa operasi pada kode fragmen. Suatu alat penghubung instruksi yang diset ringkas digunakan oleh pemindah untuk menerapkan perubahan bentuk set instruksi yang mandiri dipindahkan. Tempat cache fragmen mengatur dan menyimpan kode aplikasi yang paling sering dieksekusi. Sebagai tambahan, cache fragmen bertanggung jawab atas penyisipan kata pendahuluan dan bagian kode terakhir untuk mengatur tombol SIND ketika kontrol masuk dan meninggalkan fragmen. Tempat cache fragmen juga akan mendukung penghubung fragmen-fragmen dengan jalan membagi-bagi cabang untuk satu sama lain berhubungan bersama-sama dalam suatu super-fragment, tak perlu penghapusan tombol pengatur SIND.

III.2. DynamoRIO

DynamoRIO adalah suatu sistem optimisasi biner dinamis untuk HP-PA biner. Sistem operasi HP-UX. DynamoRIO menyertakan suatu versi yang terbaharu tentang Dynamo yang beroperasi pada IA-32 biner di bawah both Windows dan Linux. Walaupun masih dalam pengembangan, introspeksi RIO runtime dan optimisasi arsitektur yang digunakan untuk implementasi suatu metode keamanan eksekusi program pemanggil. RIO memerlukan suatu pendekatan ringan yang sedikit berbeda ke translasi biner dinamis dari SIND dan Dinamo sebelumnya.

Sebagai ganti menginterpretasikan biner yang sebelumnya mengumpulkan fragmen-fragmen, semua kode aplikasi dieksekusi ke luar dari kode cache. Kode aplikasi dipecah dalam blok dasar, urutan instruksi yang berakhir dengan instruksi perpindahan kontrol tunggal, yang dieksekusi secara langsung ke luar dari cache blok dasar. Pada bagian akhir suatu blok dasar, RIO mulai lagi mengontrol dan menguji target dari kontrol perpindahan instruksi itu. Jika target bukan di cache blok dasar, RIO akan menempatkannya dalam tempat cache dan pelaksanaan resume dalam blok dasar yang baru. Jika target telah berada dalam cache, dua blok dasar akan dihubungkan bersama-sama dengan suatu lompatan langsung. Selama eksekusi, DynamoRIO mengidentifikasi *traceheads* yang potensial.

```
void DynamoRIO_init();
void DynamoRIO_exit();
void DynamoRIO_basic_block(void*, app_pc, InstrList*);
void DynamoRIO_trace(void*, app_pc, InstrList*);
void DynamoRIO_fragment_deleted(void*, app_pc);
```

Gambar 3.2: Fungsi-fungsi DynamoRIO

Eksekusi mempertahankan penghitungan untuk masing-masing dari *traceheads*. Ketika suatu eksekusi yang sudah dikenal dicapai, *hot* trace ditempatkan dalam cache trace. Pengumpulan hot trace untuk kode fragmen yang paling sering dieksekusi harus merupakan target optimisasi yang dinamis.

III.2.1 Interface DynamoRIO

Pengiriman DynamoRIO pada suatu Aplikasi Interface Pemrograman atau *Application Programming Interface* (API) memungkinkan untuk konstruksi optimisasi dan keperluan instrumentasi, memanggil klien DynamoRIO. API meliputi abstrak yang tersusun baik untuk membuat dan menggerakkan perintah IA-32 seperti halnya Struktur data DynamoRIO yang spesifik. Ini mengizinkan klien programmer DynamoRIO untuk beroperasi pada arus instruksi tanpa pengetahuan yang terperinci tentang instruksi encoding IA-32, hanya perlu suatu pemahaman yang di-set dari instruksi IA-32. API juga mendefinisikan beberapa fungsi hubungan yang akan dipanggil oleh DynamoRIO jika mereka didefinisikan oleh pemakai share library. Deklarasi dari fungsi ini seperti ditunjukkan pada Gambar 3.2. DynamoRIO masuk dan DynamoRIO keluar mempunyai kaitan

fungsi dipanggil ketika DynamoRIO menginisialisasi dan hampir berhenti, secara berurutan. Ketika membuat suatu blok dasar, fungsi blok dasar DynamoRIO adalah memanggil untuk mengijinkannya mengubah bentuk kode fragmen sebelum ditempatkan di cache blok dasar. Dengan cara yang sama, fungsi trace DynamoRIO adalah memanggil ketika suatu trace diijinkan untuk diubah sebelum ditempatkan di cache trace.

BAB IV

Pendekatan Keamanan Translasi Biner Dinamis

Beberapa pertahanan yang digunakan untuk melawan terjadinya eksploitasi injeksi kode adalah mengimplementasikan ekstensi compiler. Sebagai contoh, StackGuard dan Stack Shield adalah implementasi ekstensi untuk GNU C Compiler. Karena sejumlah pertimbangan, re-compilation dari aplikasi keamanan yang kritis. Ini berkaitan dengan tidak tersedianya suatu source program, waktu, atau keahlian. Beberapa teknik memungkinkan menggunakan mekanisme proteksi translasi biner dinamis. Bagian berikut mendiskusikan penggunaan dari translasi biner dinamis untuk implementasi beberapa teknik melindungi terhadap eksploitasi injeksi kode. Verifikasi alamat pengirim Out-Of-Band adalah suatu teknik menulis ulang kode yang bisa untuk menghentikan semua serangan stack-smashing. Sistem Panggilan Sandboxing adalah suatu teknik untuk membatasi akibat-akibat dari kesalahan dan kemungkinan besar kode malicious

IV.1 Verifikasi Alamat Pengirim Out-Of-Band

StackShield adalah suatu tool untuk melindungi aplikasi terhadap serangan stack smashing, yang tidak memerlukan modifikasi source program. Salah satu dari teknik yang diterapkan oleh StackShield adalah memodifikasi subroutine yang berurutan memanggil untuk menyimpan alamat pengirim dalam bagian stack bandingkan dengan aktivasi perekaman segmen stack program. Implementasi menambahkan suatu ukuran array yang fix untuk segmen data dari program penyimpanan secara menyeluruh untuk stack alamat pengirim. Bagian terakhir Subroutine dimodifikasi untuk menekan alamat pengirimnya ke stack ini. Sebelum kembali, subroutine membandingkan alamat pengirim dari aktivasi perekam dengan alamat pengirim yang meninggalkan stack secara menyeluruh. Jika mereka bertemu, subroutine akan kembali normal. Jika mereka tidak bertemu, program tergantung pada suatu pilihan compile-time berhenti atau terus melanjutkan eksekusi menggunakan alamat pengirim kembali dari stack alamat pengirim menyeluruh.

Diterapkannya suatu penyesuaian dari teknik yang digunakan oleh StackShield, yang memanggil verifikasi alamat pengirim Out-Of-Band sebagai suatu klien DynamoRIO. Instrumenkan masing-masing instruksi panggilan untuk menekan alamat pengirim yang diharapkan ke suatu stack out-of-band alamat pengirim dan masing-masing menginstrumenkan kembali instruksi untuk menyamakan tujuan dengan alamat di atas stack out-of-band. Jika alamat pengirim dari perekam aktivasi stack program tidak memenuhi atas out-of-band stack alamat pengirim, ini menandakan adanya alamat pengirim telah menjadi overwritten dan beberapa tindakan yang dapat diambil adalah: membukukan percobaan yang mengganggu dan untuk menghentikan atau melanjutkan eksekusi dengan menggunakan out-of-band alamat pengirim.

Teknik ini sangat efektif untuk melawan terhadap semua eksploitasi stack smashing yang berlebihan dan metoda eksploitasi kembali alamat tujuan di stack. Dalam serangan sebuah stack smashing, hanya data-data pengikut yang ditulis melebihi batas buffer, sehingga stack out-of-band disimpan di dalam memori stack tidak bisa ditulis kembali dan penulisan kembali alamat yang melebihi akan terdeteksi. Format string atau perubahan stack menyerang kembali tujuan alamat dalam stack program yang mirip akan dideteksi dan dicegah.

IV.1.1 Implementasi

Kita implementasikan Verifikasi Alamat Pengirim Out-Of-Band sebagai klien DynamoRIO yang menggunakan interface DynamoRIO. Implementasi itu bisa untuk eksekusi biner DynamoRIO dan dengan sukses mendeteksi modifikasi dari alamat pengirim tanpa kelihatan terjadi penurunan kemampuan. Untuk memastikan bahwa semua kode tertangkap sebelum eksekusi, diterapkan instrumentasi pada tingkat blok dasar yang menggunakan blok dasar DynamoRIO. Sebelum ditempatkan ke dalam cache kode, masing-masing blok adalah kode instrumen untuk memanggil suatu fungsi pemantau sebelum eksekusi tiap-tiap panggilan langsung, panggilan yang tidak langsung, atau kembalinya perintah atau instruksi. Pemantauan ini berfungsi untuk menyimpan kembali koreksi alamat out-of-band pada fungsi panggilan dan memverifikasi kembali persetujuan

alamat dengan nilai fungsi out-of-band. Sekarang implementasi menggunakan suatu ukuran array yang fix diperlakukan suatu stack untuk penyimpanan alamat pengirim. Pada suatu fungsi pemanggil, alamat dari instruksi yang mengikuti instruksi pemanggil ditekan ke stack. Pada suatu fungsi balik, alamat tujuan dari hasil dibandingkan dengan alamat yang disimpan kembali pada alamat stack out-of-band. Jika alamat tidak ketemu, maka kejadian tersebut dicatat sebagai gangguan. Dalam posisi ini, beberapa tindakan boleh dilakukan (tergantung pada pilihan sistem pengelolanya). Proses harus dihentikan oleh suatu panggilan untuk keluar dan mengakhirinya atau menggugurkan untuk mengakhiri dan mengeluarkan bagian dalam atau inti untuk pemeriksaan berikutnya. Sebagai alternatif, alamat pengirim out-of-band digunakan untuk langkah eksekusi selanjutnya. Walaupun eksekusi program akan terus berlangsung, overwritten mempunyai struktur data lain pada stack yang merupakan percobaan untuk menghancurkan stack, sehingga dapat menyebabkan program hancur.

IV.1.2 Tantangan

Implementasi mengajukan beberapa tantangan rancang-bangun. Khususnya, ada beberapa kesulitan yang memastikan bahwa masing-masing panggilan subroutine telah sesuai dengan hasil subroutine. Beberapa kasus yang khusus memerlukan untuk identifikasi panggilan palsu seperti uraian berikut. Dalam kaitan dengan beberapa aspek dari run-time yang saling berhubungan, tidak setiap panggilan akan menghasilkan suatu rekaman alamat pengirim. Sebagai contoh, posisi kode independent sering terpaksa menentukan penempatan lokasi eksekusi memori. Model kode pemakai tidak bisa secara langsung mengakses daftar petunjuk instruksi EIP (daftar konter program yang ada pada arsitektur IA-32) maka posisi kode independent sering menggunakan suatu trik untuk mendapat kembali nilai dari petunjuk instruksi yang menggunakan salah satu metoda seperti pada Gambar 4.2.

a: nop

b: bn,a a

c: bn,a b

call c

Memindahkan nilai program counter untuk mendaftarkan r31 pada PowerPC:

```
a: xor. r5, r5, r5
```

```
bnel a
```

```
mflr r31
```

Memindahkan nilai instruksi petunjuk untuk mendaftarkan eax pada IA-32:

```
call next
```

```
next: pop %eax
```

Gambar 4.2: Metode Memperoleh Alamat dimana saat Kode mencegah eksekusi

Dalam kode yang menggunakan metoda ini, tidak ada instruksi yang berhubungan, maka “panggilan” seperti itu tidaklah direkam kembali di stack alamat out-of-band. Semua instruksi panggilan dengan source dan alamat tujuan dalam memori menghubungkan runtime dengan mengabaikan kode yang executable sebab hasilnya lebih sering dibanding hasil panggilan yang palsu dan mengabaikan keamanan, seperti diuraikan dibawah. Untuk pertimbangan yang serupa, kembalikan source dan tujuan alamat pengirim dalam menghubungkan runtime kode yang diabaikan. Kembali dari menghubungkan runtime dalam suatu fungsi share library adalah hasil dari resolusi simbol yang didatangkan dari luar. Ketika mendatangkan pertama kali dari luar fungsi share library memanggil suatu fungsi penghubung runtime untuk melihat alamat dari fungsi yang diinginkan. Khusus kasus yang serupa juga diperlukan untuk panggilan dan hasil yang menyertakan share library DynamoRIO. Ketika suatu program dihubungkan dengan library DynamoRIO, DynamoRIO mengambil alih eksekusi dari program selama inisialisasi share library. Blok dasar yang pertama ditempatkan ke dalam cache dan isi dieksekusi kembali dari hasil inisialisasi DynamoRIO. Identifikasi dari panggilan dan untuk kembali dari penghubung runtime yang menghubungkan source dan alamat tujuan untuk menunjukkan daerah memori yang di proses dipetakan. Sekarang implementasi membaca `/proc/self/maps1` pada waktu inisialisasi dan menyimpan jangkauan alamat dan pathname dari obyek yang dipetakan ke dalam jangkauan didaftar sebuah penghubung. Untuk masing-masing kontrol memindahkan source instruksi atau alamat tujuan, alamat dihubungkan ke obyek pemetaan dengan melalui pendaftaran. Ini digunakan

untuk mengidentifikasi apakah panggilan atau hasil harus diabaikan atau tidak sebelum diuraikan. Pengabaian panggilan dan perintah kembali akan aman sebab penulisan alamat pengirim oleh aplikasi maupun kode share library penghubung Runtime tidak berhadapan dengan input pemakai di overflow.

1. `/proc/self/maps` adalah suatu pseudo-file dalam file sistem `procfs` yang berisi suatu daftar naskah dari segmen-segmen memori yang dipetakan dari proses yang sedang berjalan.
2. penghubung interaksi yang utama dengan pemakai selesai environment variables seperti `LD PRELOAD` yang diabaikan ketika mengerjakan setuid yang executable.

IV.1.2 Kesimpulan

Setelah mengidentifikasi kasus khusus dan panggilan palsu yang diabaikan, teknik implementasi membuktikan dengan sangat terbuka. Teknik dan implementasi dengan sukses mendeteksi dan mencegah beberapa teknik eksploitasi tanpa terlihat terjadi penurunan kekuatan. Sekarang ada beberapa pembatasan-pembatasan. implementasi yang menggunakan suatu ukuran array yang fix untuk menyimpan kembali alamat. Ini secara tak wajar membatasi ukuran panggilan yang maksimum. Suatu pengukuran kembali array yang dinamis dapat menyediakan akses secara tetap dan membiarkan ukuran panggilan maksimum yang diijinkan oleh memori. Saat teknik yang diuraikan dengan sukses menghentikan semua “off-the-shelf” serangan-serangan terhadap stack alamat pengirimnya, maka akan rentan sebagai sasaran serangan yang menggunakan sifat mudah kena luka yang diijinkan berbagai alamat untuk overwritten (ini meliputi format string dan tumpukan yang rentan terhadap penggerogotan). Jika stack alamat pengirim out-of-band disimpan dalam sebuah tempat yang diketahui atau lokasi yang dapat direncanakan, penyerang boleh (overwrite) untuk kedua alamat yang kembali dalam segmen stack dan di stack out-of-band.

Dengan menggunakan suatu array yang ukurannya dinamis, stack alamat penerima tidak akan tinggal pada suatu lokasi penempatan ukuran kembali dan dapat mejadi sasaran serangan. Beberapa pendekatan bisa diambil untuk meningkatkan performance. Penggunaan dari suatu daftar yang dihubungkan

untuk menyimpan pemetaan memori bisa ditingkatkan untuk memperkecil waktu lookup oleh berbagai teknik variasi.

IV.2. Sistem Panggilan Sandboxing

Suatu sandbox adalah suatu lingkungan eksekusi yang terbatas untuk menjalankan kode yang berpotensi untuk tidak dipercayai. Lingkungan bahasa pemrograman secara khas menggunakan sandbox untuk memastikan keamanan eksekusi kode yang mobile. Sebagai contoh program bahasa Java menggunakan suatu sandbox ketika eksekusi Java Applets. Sandbox Java membatasi akses dari kode tidak dipercaya untuk metoda kelas yang berpotensi membahayakan. Suatu pendekatan yang serupa harus digunakan untuk kontrol eksekusi tentang segala proses sistem dengan pembatasan penggunaan panggilan sistem yang UNIX

Sistem panggilan suatu permintaan untuk melayani dari pemakai ke sistem operasi. Sistem operasi kernel bertindak sebagai suatu monitor antara pemakai dan sumber daya sistem. Dengan cara ini, kode user-mode tidak secara langsung saling berhubungan dengan sumber daya seperti file atau tool, tetapi sebagai gantinya membuat sistem panggilan meminta sistem operasi kernel untuk melaksanakan operasi yang diinginkan. Secara khas, sistem panggilan menyediakan kemampuan untuk saling berhubungan dengan file, tool, proses yang lain, atau jaringan. Sistem Panggilan menjalankan model keamanan yang UNIX. Tindakan dari suatu proses secara efektif dibatasi dengan pembatasan sistem panggilan yang harus dilakukan, dan diijinkan oleh sistem panggilan yang harus digunakan. Sebagai contoh, dengan pembatasan sistem panggilan tertentu, suatu proses yang tidak dipercayai harus dibatasi untuk menggunakan jaringan atau melakukan perubahan apapun pada sistem file walaupun file mengijinkannya. Sistem panggilan adalah sistem panggilan yang ditempatkan dalam daftar EAX sampai enam argumentasi ditempatkan dalam daftar EBX, ECX, EDX, ESI, EDI, dan EBP, karena dalam pesan. Jika ada lebih dari enam argumentasi dipanggilan sistem, disimpan dalam struktur di memori dan struktur alamat seperti yang pertama dan hanya argumentasi di EBX. Sistem Panggilan akhirnya diaktifkan dengan suatu int \$. Setelah instruksi, hasil nilai dari sistem panggilan

terdapat dalam daftar EAX. Jika ada suatu kesalahan dalam melaksanakan sistem panggilan, nilai hasil adalah - 1.

```
...  
mov %edx, 0x10(%esp)  
mov %ecx, 0xc(%esp)  
mov %ebx, 0x8(%esp)  
mov %eax, $0x00000005  
int $0x80  
...
```

Gambar 4.1: Instrumentasi Sistem Panggilan

4.2.1 Implementasi

Pendekatan yang kita lakukan adalah untuk menggantikan sistem panggilan dengan panggilan yang berfungsi memonitor yang melaksanakan sistem panggilan untuk kepentingan aplikasi. Fungsi monitor sistem panggilan (mirip untuk keadaan *strace* dan *truss* utility UNIX) dan dengan memilih eksekusi sistem panggilan yang didasarkan pada pemeriksaan dari sistem panggilan yang spesifik atau argumentasi panggilan. Mekanisme ini digunakan untuk suatu perencanaan sandbox yang didasarkan pada pembatasan penggunaan sistem panggilan. Suatu klien DynamoRIO, menguji blok dasar sebelum eksekusi dan menggantikan sistem panggilan, IA-32 instruksi `int $ 0x80`, (Gambar 4.2) dengan suatu urutan perintah eksekusi keamanan status aplikasi, pemantauan memanggil fungsi *chk syscall*, memperbaiki kembali status aplikasi, dan melanjutkan eksekusi (Gambar 4.2.). Sistem panggilan memonitoring fungsi *chk syscall* yang diambil sebagai argumentasi sistem panggilan yang meminta nomor dan parameter. Hasil nilai dari *chk syscall* akan ditafsirkan sebagai hasil nilai dari aplikasi sistem panggilan. Sebagai contoh, jika sistem panggilan menolak, *chk syscall* harus menset *errno* ke EPERM (pemberian isyarat bahwa operasi tidak diijinkan atau ijin ditolak) dan kembali.

```
...  
mov 0x4002079c, %esp  
mov %esp, 0x400207d4
```

```

pushf
pusha
push %ebp
push %edi
push %esi
push %edx
push %ecx
push %ebx
push %eax
call $0x40023f98
mov 0x40020780, %eax
add %esp, $0x1c
popa
popf
mov %esp, 0x4002079c
mov %eax, 0x40020780
...

```

Gambar 4.2: Instrumentasi Setelah Sistem Panggilan

Perluasan yang sederhana boleh menghasilkan kebijakan pembatasan file mengakses sistem file (menirukan sistem panggilan *chroot*) atau jaringan komunikasi

IV.2.2 Kesimpulan

Sejumlah kecil sistem panggilan dalam system operasi Unix membuat interface yang ideal membatasi untuk implementasi kemajuan model-model keamanan lebih lanjut. Penggunaan translasi biner dinamis untuk menulis kembali kebijakan sistem panggilan ini untuk keamanan dikuatkan dalam kode pemakai. Sebab kebijakan harus dipilih ketika aplikasi diluncurkan, aplikasi yang berbeda harus dikontrol oleh kebijakan model keamanan. Manfaat lain dari penerapan kebijakan keamanan dalam pengembangan kode user-mode menjadi lebih sederhana. Mekanisme kebijakan keamanan harus ditulis dan merunut program tanpa

menambahkan kompleksitas pengembangan level kernel. Strategi implementasi yang efisien boleh memperkecil ongkos eksploitasi ini.

Dalam kaitan perancangan DynamoRIO sistem translasi biner dinamis, adalah bentuk sandbox yang tidak terhindarkan. Sebagai contoh, seorang pemakai malicious boleh mencoba untuk membypass instrumentasi sistem panggilan dengan melompat ke kode yang melaksanakan instruksi panggilan. Dalam hal ini, tujuan dari lompatan akan diperlakukan sebagai sebagai suatu blok dasar yang baru dan sistem pemanggil kode instrumentasi akan dimasukkan. Kode pemakai malicious boleh mencoba untuk memodifikasi kode di DynamoRIO blok dasar dan cache trace.

BAB V

Penutup

Sekarang dapat diketahui adanya suatu penggolongan baru tentang sifat rentan terhadap serangan virus pada sistem keamanan komputer, yaitu kerentanan pada memori. Sifat memori yang rentan terhadap virus adalah kelemahan perangkat lunak yang memungkinkan memori mengakses di luar ilmu semantik dari bahasa pemrograman, di mana perangkat lunak telah dibuat. Keadaan sifat memori yang rentan terhadap virus meliputi overflow di penyangga (buffer), format string, overflow interger, akses array out-of-bounds, dan double-free yang rentan virus. Sifat memori yang rentan terhadap virus akan digunakan untuk eksploitasi sebuah injeksi kode memanfaatkan suatu serangan yang mengalihkan kontrol program ke dalam mesin.injeksi kode yang dinamis. Keamanan komputer, terutama keamanan aplikasi adalah suatu senjata dalam perlombaan antara penyerang dan pelindung. Karena sejak pertama dulu buffer overflow rentan terhadap virus. Sebagai contoh, perbedaan dalam pengalaman antara metoda eksploitasi yang digunakan oleh worm Morris dan eksploitasi yang kreatif dari Apache yang rentan virus.

Kemampuan untuk menemukan sifat rentan virus dalam aplikasi telah tumbuh dengan mantap melalui penggunaan rancang-bangun dan black-box yang menguji secara otomatis. Sebagai contoh, Apache yang menyandi sifat rentan virus di open source web server yang sangat populer. Dalam waktu yang sama, banyak pekerjaan telah dilakukan dalam menciptakan pertahanan melawan eksploitasi injeksi kode. Translasi biner yang dinamis telah membuktikan dapat menjadi solusi yang sangat sehat untuk menerapkan system pertahanan yang dinamis. Seperti ditunjukkan oleh dua pendekatan implementasi, translasi biner dinamis dapat digunakan untuk implementasi pendekatan yang akan secara khas diimplementasikan sebagai compiler yang statis atau ekstensi sistem operasi. Penggunaan translasi biner dinamis untuk menulis kembali kode aplikasi pada run-time dengan meniadakan kebutuhan akan source program aplikasi atau perubahan permanen aplikasi, memungkinkan aplikasi berubah secara acak pada run-time, membuat tujuan aplikasi bergerak. Implementasi Sistem Panggilan

sandbox menunjukkan bagaimana translasi biner dinamis digunakan untuk implementasi sistem yang secara khas diterapkan dalam sistem operasi. Ini memungkinkan sistem dibangun untuk melindungi aplikasi tanpa menuntut perubahan global pada sistem operasi (yang tidak boleh mungkin tanpa akses source program sistem operasi). Translasi biner dinamis bukan hanya suatu alat untuk re-implementing sistem yang berjalan, juga memungkinkan penggunaannya pada suatu generasi baru dari sistem pertahanan yang dinamis untuk riset pengembangan pada masa datang.

Daftar Pustaka

1. Budi Rahardjo: *Keamanan Sistem Informasi Berbasis Internet*, 2001.
<http://budi.insan.co.id>
2. CERT: *Computer Security*, 2001. http://www.cert.org/tech_tips/
3. David Moore : *The spread of the code-red worm*,
http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml.
15 Desember 2004.
4. Gerardo Richarte : *Four different tricks to bypass stackshield and stackguard protection*,
<http://www.corest.com/files/files/11/StackguardPaper.pdf>, 12 Desember 2004.
5. Last Stage of Delirium Research Group : *Unix assembly codes development for vulnerability illustration purposes*,
<http://www.lsdpl.net/documents/asmcodes-1.0.2.pdf>, 12 Desember 2004.
6. Mark J. Cox. Apache httpd : *vulnerability with chunked encoding*,
<http://online.securityfocus.com/archive/1/277268>, 15 Desember 2004.
7. Robert Morris : *Morris worm source code*,
http://sunsite.bilkent.edu.tr/pub/security/cerias/doc/morris_worm/worm-src.tar.gz, 15 Desember 2004.
8. Tim Newsham : *Format string attacks*, Technical report, Guardent, Inc., September 2000.
9. Tool Interface Standards Committee : *Executable and linking format*,
<http://x86.ddj.com/ftp/manuals/tools/elf.pdf>, 12 Desember 2004.
10. Trek Palmer, Dino Dai Zovi, and Darko Stefanovic : *SIND: A framework for binary translation*, Technical Report TR-CS-2001-38, University of New Mexico, 2001.
11. Vendicator : *Stack shield*,
<http://www.angelfire.com/sk/stackshield/info.html>, 15 Desember 2004.
12. Wahana Komputer Semarang : *Memahami Model Enkripsi dan Security Data, Andi Offset, Yogyakarta, 2003*