Bab 22

Aplikasi - Authentication

Authentication adalah komponen yang sangat penting dalam pengamanan sistem informasi. Boleh dikatakan bahwa semua standard pengamanan sistem informasi, termasuk SSL/TLS, SSH, IPsec, S/MIME dan OpenPGP, mempunyai komponen *authentication*. Pengamanan non-standard biasanya juga mempunyai komponen *authentication*, contohnya menggunakan *password*. Penggunaan suatu komputer dapat mewajibkan pengguna untuk *login* menggunakan *password*. Namun di jaman yang serba terhubung seperti sekarang, fasilitas sistem informasi suatu organisasi yang cukup besar biasanya tidak terdapat pada hanya satu komputer tetapi terdistribusi pada beberapa komputer yang terhubung melalui suatu *local area network* (LAN). Sangat tidak efisien jika pengguna harus *login* ke setiap aplikasi yang tersedia dalam jaringan. Kerberos mencoba membantu dalam masalah ini dengan menggunakan *centralized authentication*, dimana *authentication* dipusatkan di *authentication server*.

22.1 Kerberos

Kerberos dikembangkan di Massachusetts Institute of Technology (MIT) sebagai bagian dari proyek Athena bersama dengan Digital Equipment Corporation (DEC) dan IBM. Proyek Athena berlangsung dari tahun 1983 sampai dengan tahun 1991. Saat Kerberos dikembangkan, satu masalah yang dihadapi dalam penggunaan LAN di MIT adalah tidak adanya *authentication* yang efektif ketika seorang hendak menggunakan aplikasi di komputer lain di jaringan. Contohnya, koneksi ke komputer menggunakan rlogin tidak melalui *authentication* yang aman, komputer yang melayani (*server*) hanya mengecek nama dari pengguna dalam daftar, jadi sangat mudah untuk mengelabui *server*. Salah satu tujuan Kerberos adalah untuk mengamankan berbagai aplikasi yang disediakan oleh proyek Athena, dengan melakukan *authentication* terhadap calon

pengguna.

Karena *authentication* dipusatkan di *authentication server*, protokol Kerberos tidak cukup hanya melakukan *client authentication*, tetapi juga harus memperkenalkan *client* ke *server* yang diminta layanannya oleh *client*. Ada empat aktor dalam protokol Kerberos:

- *client*,
- *authentication server* (AS),
- *ticket-granting server* (TGS), dan
- *server*.

Tugas AS adalah melakukan *authentication* terhadap *client* sedangkan tugas TGS adalah memperkenalkan *client* ke *server*. Meskipun AS dan TGS adalah dua entitas yang berbeda, keduanya bisa ditempatkan di satu komputer. Kombinasi AS dan TGS biasanya disebut sebagai *key distribution center* (KDC). Kunci simetris untuk setiap pengguna (*client*) harus diregistrasi di AS dan kunci simetris untuk setiap *server* harus diregistrasi di TGS. Biasanya kunci pengguna didapat dari *password* melalui *hashing* (lihat bab 9). Protokol Kerberos didasarkan pada protokol Needham-Schroeder dan secara garis besar berjalan sebagai berikut:

1. *Client* melaporkan ke AS bahwa ia ingin menggunakan layanan *server*.
2. AS mengecek daftar *client*. Jika *client* (berikut kuncinya) ada dalam daftar, AS membuat kunci sesi K_1 untuk komunikasi antara *client* dengan TGS. AS kemudian membuat *ticket-granting ticket* (TGT) yang dienkripsi menggunakan kunci TGS dan isinya termasuk identitas *client*, *client network address*, masa berlaku TGT dan K_1. K_1 yang dienkripsi menggunakan kunci *client* dan TGT keduanya dikirimkan ke *client*.
3. *Client* mendekripsi menggunakan kunci *client* untuk mendapatkan K_1 dan kemudian membuat permohonan yang dikirimkan ke TGS dan terdiri dari TGT, identitas *server* dan suatu *authenticator* A_1. A_1 itu sendiri dienkripsi menggunakan K_1 dan isinya termasuk identitas *client*, *client network address* dan *timestamp* (waktu saat A_1 dibuat).
4. TGS mendekripsi TGT menggunakan kunci TGS untuk mendapatkan K_1, identitas *client*, *client network address* dan masa berlaku TGT. Jika TGT masih berlaku maka TGS kemudian mendekripsi A_1 menggunakan K_1 untuk melakukan validasi. Jika validasi sukses maka TGS membuat kunci sesi K_2 untuk komunikasi antara *client* dan *server*. TGS kemudian membuat *client-to-server ticket* (CST) yang dienkripsi menggunakan kunci *server* dan isinya termasuk identitas *client*, *client network address*,

masa berlaku CST dan K_2. K_2 yang dienkripsi menggunakan K_1 dan CST dikirimkan ke *client*.

5. *Client* mendekripsi menggunakan K_1 untuk mendapatkan K_2 lalu membuat *authenticator* A_2 yang dienkripsi menggunakan K_2 dan isinya termasuk identitas *client*, *client network address* dan *timestamp* (waktu saat A_2 dibuat). CST dan A_2 dikirimkan ke *server*.

6. *Server* mendekripsi CST menggunakan kunci *server* untuk mendapatkan K_2, identitas *client*, *client network address* dan masa berlaku CST. Jika CST masih berlaku, *server* kemudian mendekripsi A_2 menggunakan K_2 untuk melakukan validasi. Jika validasi sukses maka *server* membuat konfirmasi yang isinya termasuk *timestamp* dan identitas *server*. Konfirmasi dienkripsi menggunakan K_2 dan dikirimkan ke *client*.

7. Menggunakan K_2, *client* mendekripsi dan mengecek konfirmasi. Jika tidak bermasalah maka *client* dapat memulai permintaan pelayanan dari *server*.

8. *Server* dapat melayani permintaan *client*.

Authentication yang dilakukan AS terhadap *client* bersifat tidak langsung, yaitu dengan mengenkripsi kunci sesi antara *client* dengan TGS menggunakan kunci *client*. Jika seorang yang bukan *client* mengaku sebagai *client* dan meminta TGT kepada AS, maka itu akan sia-sia karena orang tersebut tidak bisa mendapatkan kunci sesi (ia tidak memiliki kunci *client* dan tidak memiliki kunci TGS). Tanpa kunci sesi antara *client* dengan TGS, *authenticator* A_1 yang *valid* tidak bisa dibuat.

Protokol Needham-Schroeder versi awal tidak menggunakan *timestamp*. Akan tetapi, tanpa *timestamp*, protokol rentan terhadap *replay attack*. Oleh sebab itu Kerberos menggunakan *timestamp* untuk mencegah *replay attack*. Validasi *authenticator* yang dilakukan TGS dan *server*, selain mengecek identitas *client* dan *client network address*, juga mengecek *timestamp*. Jika umur *timestamp* melebihi batas yang ditentukan (biasanya 5 menit), maka *authenticator* dianggap tidak *valid*. TGS dan *server* juga mengelola daftar terdiri dari permintaan yang telah divalidasi dalam 5 menit terahir. Jika ada permintaan ulang dalam jangka waktu 5 menit (ini dapat dicek menggunakan daftar), maka permintaan ulang tersebut ditolak.

Kerberos dapat di*download* dalam bentuk *source code* dari MIT. *Web site* untuk Kerberos adalah `http://web.mit.edu/Kerberos/`. Saat bab ini ditulis versi terbaru Kerberos adalah versi 5-1.7. Berbagai dokumentasi mengenai Kerberos juga bisa didapat dari *web site* Kerberos, termasuk *Installation Guide*, *User's Guide* dan *Administrator's Guide*.

Untuk dapat menggunakan aplikasi dengan Kerberos, tentunya aplikasi harus mendukung protokol Kerberos (istilahnya aplikasi sudah *Kerberized*),

baik disisi *client* maupun disisi *server*. Ini dapat dilakukan dengan mengintegrasi protokol Kerberos langsung dalam aplikasi atau menggunakan suatu *wrapper* yang mendukung Kerberos. Aplikasi disisi *client* juga harus mengetahui kunci *client* yang diregistrasi dengan AS, sedangkan aplikasi disisi *server* juga harus mengetahui kunci *server* yang diregistrasi dengan TGS. Beberapa program Unix yang *Kerberized* sudah termasuk dalam Kerberos versi 5-1.7, antara lain telnet, rlogin, ftp, rsh, rcp dan ksu.

Setiap *administrative domain* (misalnya `athena.mit.edu`) mempunyai *key distribution center* (KDC) sendiri dan dinamakan *realm*. Jika *client* ingin menggunakan layanan *server* di *realm* yang lain, maka diperlukan *cross-realm authentication*. TGS untuk *server* (sebut saja STGS) harus diregistrasi di TGS untuk *client*, atau setidaknya harus ada rantai

client, AS, TGS, TGS1 , ... , TGSn, STGS, *server*

dimana STGS diregistrasi di TGSn, TGSn diregistrasi di TGSn-1, ... , TGS2 diregistrasi di TGS1 dan TGS1 diregistrasi di TGS. Untuk *authentication*, *client* harus melalui rantai tersebut:

- oleh AS, *client* diberikan TGT untuk TGS;
- jika TGT dapat divalidasi oleh TGS, *client* diberikan TGT1 untuk TGS1;
- jika TGT1 dapat divalidasi oleh TGS1, *client* diberikan TGT2 untuk TGS2;
- dan seterusnya.

Hanya STGS yang dapat memberikan CST kepada *client* untuk kemudian diberikan kepada *server*.

Kerberos banyak digunakan untuk *single-sign-on*. Sebagai contoh, menggunakan sistem Unix, pengguna hanya mengetik *password* satu kali. Setiap kali menggunakan program *client-server* yang sudah *Kerberized* seperti telnet, pengguna tidak perlu mengetik *password* lagi.

22.2 Ringkasan

Di bab ini kita telah bahas *client-server authentication* menggunakan Kerberos. Kerberos banyak digunakan untuk implementasi *single-sign-on* dan dapat di*download* dalam bentuk *source code* dari

`http://web.mit.edu/Kerberos/.`