Säkrare system

roland ronquist

Inledning

När du behöver säkra upp ditt system är det många saker att tänka på, här kommer vi att i första hand ta itu med användarhanteringen och relaterade områden. Förutom det rent tekniska kraven att släppa in de betrodda och hålla slöddret borta så behöver vi rimlig användarvänlighet för systemadministratören. Därför är gemensam administration en viktig sak; gemensam administration av inställningar för många olika program och gemensam administration för användare på olika maskiner.

Gemensam administration av användarkonton över nätet

Sun konstruerade NIS för snart 30 år sedan även om de kallade systemet YP först, systemet har utvecklats och används på många ställen fortfarande. Dock väljer de flesta som bygger nya system att ha användarinformationen i LDAP i stället. Med ett nis-schema så kan en LDAP-katalog i princip användas som en direkt ersättare för NIS. Så långt är allt väl, men den som verkligen är ute efter säkerhet så kan ändå NIS-konceptet kännas en smula förlegat; säkerhet var inte riktigt sysftet från början och även om det hela utvecklats och förbättrats en hel del, så torde den säkerhetsinriktade administratören finna bättre stöd hos Kerberos, som i och för sej också härstammar från 1980-talet, men här har vi säkerhetsfokus redan från början. Inte nog med det, Kerberos ger även så kallad secure single signon (SSO), användaren loggar på och har sedan tillgång till alla tjänster där denne är betrodd. Säkerhet och smidighet i skön förening; det finns naturligtvis situationer då detta inte är något önskvärt beteende, en användare som under hot avslöjar sin inloggningshemlighet (lösenord, eller kod till inloggningsdosa eller liknande) ger ju angriparen en väg in, till alla delar, genom att framtvinga en enda hemlighet. Med separata lösenord (och liknande) så måste rånaren ägna mer energi åt att hämta nödvändig information och löper därför större risk att drabbas av upptäckt. Å andra sidan så kan ju en lat användare välja att fylla sitt kontor med små gula lappar för att hålla rätt på den uppsjö av lösenord som kan behövas för att ta sej runt i systemen utan single signon. Vad som är rätt är i högsta grad en avvägningsfråga för den enskillde administratören och organisationen att ta ställning till.

Central kontohantering verkar bra, vad är problemet?!

På den gamla onda tiden kunde en användare vara lycklig ifall han kunde logga på en dator via en terminal på morgonen, för att sedan ostört producera välstånd i en texteditor eller i något annat program en hel arbetsdag. Nuförtiden har de flesta

betydligt större krav än att låta sej roas med en terminal, eller ens ett terminalfönster...

Idag har folk en rad applikationer att ta hand om, många saker låter göra sej i webbläsaren, andra kräver mer specifika klientprogram (som med epost för den som inte låter sej tillfredställas med webbpost). Vidare skall de lagra filer på diverse filservrar och liknande, data i databaser och -naturligtvis- problem i supportdatabaser. Det behöver inte vara några avancerade användare för att listan på applikationer som kräver inloggning skall bli lång. Du som systemadministratör kanske inte tycker att detta verkar som något problem, men i en större organisation kan det bli väldigt många program som skall underhållas när allt skall integreras i ett centralt system för användarkonton och autenticering. Därför vill vi ha en gemensam infrastruktur för att inloggningshanteringen inte skall bli överväldigande. Eftersom problemet är både generellt och vanligt, och då standardiseringsarbetet inte nått längre här än på andra områden, så har vi tre vanliga system för hantering (så kallade API:er, application programming interface) av inloggning och autenticering:

PAM, SASL och GSSAPI. Enkelt och smidigt, som gjort för att skapa anställningstrygghet för en skicklig systemadministratör.

Pluggable Authentication Modules, PAM

PAM är gjort för den lokala hanteringen på ett UNIX-system. Detta på ett dubbelriktat sätt, dels kan den som skriver ett program använda PAM för att sköta användare och inloggning på ett sätt som är helt kompatibelt med hanteringen för inloggningsuppgifter via ssh och gdm. Smidigt för programmeraren, praktiskt för administratören, som kan använda samma konton för både traditionell åtkomst till maskinen och för att släppa in användare på den nya applikationen som programmeraren försett med pam-stöd. För att även ge administratören lite längtan efter PAM har anrättningen kryddats med möjligheten att länka in så väl nya användardatabaser som autenticeringsredskap. Det är exempelvis vanligt att leverantörer av fingeravtrycksläsare, irisskanners och lösenordsgeneratorer, alla levererar PAM-moduler så att administratören enkelt kan göra den nya hårdvaran till standard autenticering oavsett vilket program användaren vill använda. Som om inte detta vore nog för att locka säkerhetsorienterade administratörer, så finns även möjlighet att modifiera uppdateringen av lösenord så att användarna inte kommer undan med varken qwerty, hitler eller kuken. Med PAM går enkelt att kolla lösenord mot ordlistor och mot regler som ställer krav på exempelvis längd och blandning av olika typer av tecken. Likaså kan defaultvärden enkelt ställas in för bästföredatum och liknande så att användarna tvingas uppdatera sina lösenord regelbundet.

Det finns naturligtvis en helt omvänd orsak att använda den här tekniken också, den som exempelvis driver en ISP (Internet Service Provider, internetleverantör på svenska) kan vilja ha kundkonton separerade från unixanvändarna. Därför är det vanligt med exempelvis epostservrar (imap, webbmejl och liknande) där epostanvändarna inte är kopplade till de användare som finns på systemet , ftpservrar är ett annat vanligt exempel på detta.

För att konfigurera PAM behöver du dels gå in i diverse konfigurationsfiler under

`/etc/pam.d`

beroende på linuxsmak så kan dessa filer vara allt från milt kommenterade till ganska välkommenterade. Konfigurationsfilerna pekar på diverse shared objects (så kallade libbar), de som följer med är normalt dokumenterade av man-sidor som, ovanligt nog, är

mer inriktade på att tala om vad respektive lib skall åstadkomma snarare än att fokusera på ”hur” och ”var” som ju annars brukar vara helt dominerande på en man-sida. Både gentoo och andra har gjort försök att dokumentera inställningar och funktion, men bästa förklaringen brukar många tycka är:

`http://tuxradar.com/content/how-pam-works`

GSSAPI, Generic Security Services API

GSSAPI kan dels ses som ett generellt verktyg att integrera olika system för autenticering och kryptering i applikationer, dels är det också den officiella vägen att integrera Kerberos-stöd. Problemet där är att det saknas ett generellt kerberos-protokoll, olika kerberossystem skulle kunna betraktas som inkompatibla om det inte vore för att de levereras med GSSAPI som stöd för de program som vill ansluta mot kerberos. För att du skall kunna använda GSSAPI krävs två saker, dels att applikationen som skall säkras upp har stöd för GSS och dels att du har en gssmodul som passar till det säkerhetssytem du använder (ofta kerberos och andra system för användarhantering, men det kan även röra sej om exempelvis krypteringsapparater).

SASL, Simple Authentication and Security Layer

Skapades i första hand för att ta hand om epost och ehandel, men det finns inga specifikt tekniska parametrar som begränsar användningen av SASL till just dessa områden. Däremot kommer du att se detta tydligt när du tittar på utbudet av program som stöder detta protokoll. För den här kursen räcker det med att känna till SASL som ett alternativ och ni skall veta att en lång rad tjänster för autenticering och användarhantering stöds; inklusive GSSAPI så att SASL kan integreras så att ändringar i kerberos och liknande slår igenom utan att du behöver pilla på specifika inställningar för SASL igen.

Sammanfattningsvis om API:erna för inloggning

PAM gör tre saker: skapar möjlighet för alla program på systemet att använda samma infrastruktur för användare och autenticering, vidare skapar det möjlighet för administratören att administrera och välja infrastrukturen, till sist tillför PAM kraftfulla redskap att tvinga användarna till en vettig hantering av lösenord och liknande.

GSSAPI och SASL gör i princip båda samma sak, men av historiska skäl är de tekniskt olika och saknar egentlig kompatibilitet. Anledningen till att vi behöver båda är att de flesta program bara stöder den ena. En enkel tumregel är att epost, webb och liknande som kör SSL eller TLS (Transport Layer Security, modern variant av SSL) ofta kan konfigureras att använda SASL, under det att de flesta andra saker i stället föredrar GSSAPI. Den som behöver både GSSAPI och SASL kan välja att sköta båda subsystemen helt separat, men det finns även möjlighet att koppla SASL så att den hämtar användar och autenticering från GSSAPI. Detta är inte en variant på att äta kola med pappret på, det är ett sätt att minska din arbetsbörda som administratör. När du konfigurerat alla saker så att de fungerar så hamnar du -tyvärr- allt för snabbt i situationer då saker behöver ändras. Om SASL hämtar all ”kraft” (användare och annan information) ur GSSAPI, så behöver du bara konfigurera om ditt GSSAPI, för att inställningarna skall slå igenom i SASL. Lika smidigt som att ha släp efter lastbilen...

Både GSSAPI och SASL hjälper applikationer som kommunicerar över nätverket att dels

använda externa (och förhoppningsvis säkra) infrastrukturer för användarhantering och autenticergin, dels krypterar de applikationsdata så att inget viktigt behöver sändas i klartext över nätet. Precis som med PAM får du möjlighet att skaffa intressant hårdvara som enkelt kan integreras i dina GSS- eller SASL-anslutna program. Den som inhandlat en kryptoaccelerator kan enkelt snabba upp krypteringen av datatrafiken om den levererats med stödmodul för SASL eller GSS. Naturligtvis kan du inte använda en SASL modul i ditt GSS-subsystem eller tvärt om; du måste ha moduler som passar. Lika så kan det vara träligt att försöka blanda 32- och 64-bitars operativsyst/moduler med varandra; fast där kan det många gånger gå med envishet, arbete och lite tur.

Ni kommer även att finna få saker som låter sej konfigureras för att använda Kerberos utan att antingen gå via GSSAPI eller SASL, även om den skicklige programmeraren naturligtvis kan skapa stöd utan färdiga paket.

Läs gärna wikipedia.

Kerberos

Med kerberos separerar du användarhantering och autenticering från applikationerna. Användaren loggar på och autenticeras mot kerberos och får som bevis för detta en biljett (så kallad ticket). Biljetten har en giltighetstid och en lista på tjänster där den gäller. Detta betyder att användaren loggar på en gång och får sedan tillgång till alla kerberoskompatibla tjänster utan ytterligare inloggning. Som vi sett ovan så har vi två huvudtyper av kerberoskompatibla tjänster vid sidan av den rena påloggningen på själva operativsystemet.

En kerberos-server (KDC, Key Distribution Center) består av två huvudkomponenter: AS (Authentication Server) och TGS (Ticket Granting Server). När användaren loggar in får denne en speciell biljett, en TGT (Ticket Granting Ticket) som han sedan kan använda för att (utan ytterligare autenticering) hämta ytterligare biljetter för olika tjänster (så kallade ST, Service Tickets). De tre främsta fördelarna med systemet är att användaren loggar på en gång, vilket förutom att det är bekvämt även betyder att själva autenticeringsdialogen sker vid få tillfällen vilket kan försvåra så väl brute-force- som man-in-the-middle-angrepp. Kerberosservrarna belastas mindre.

Användare, maskiner och tjänster som skyddas med kerberos kallas alla var och en för principal. Området, domänen, eller vad du nu är van vid för terminologi från andra system heter här realm. Eftersom biljetterna är tidsstämplade så kan ni räkna med att kerberos fungerar dåligt om inte tiden i nätet är väl synkroniserad via exempelvis ntp.

NFS v4

Med kerberos och ntp faller nfs in i tjänsteutbudet på ett väldigt naturligt sätt. Dels för att få nfs att fungera smidigt utan ntp, men de flesta brukar märka att nfs utan korrekt tid är tämligen jobbigt, eftersom version 4 av nfs autenticerar användningen på användarnivå så är det synnerligen praktiskt med fungerande autenticeringstjänst och där kommer kerberos naturligt in med både autenticering och single sign on.