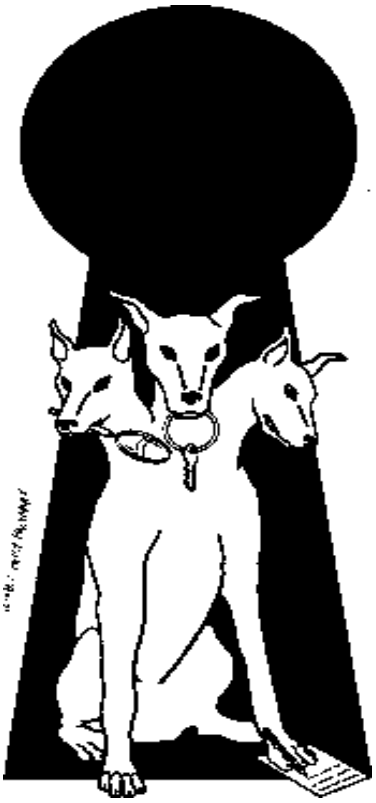
Trabalho de Redes de Computadores I

Professor Otto

Aluno: Marcos Muniz Calôr Filho

Kerberos



Introdução:

Kerberos é um protocolo desenvolvido para fornecer poderosa autenticação em aplicações usuário/servidor, onde ele funciona como a terceira parte neste processo, oferendo autenticação ao usuário.

Para garantir a segurança, ele usa criptografia de chave simétrica, com o DES.

O Kerberos foi desenvolvido como parte do Project Athena, do Massachussets Institute of Technology (MIT). Seu nome vem da mitologia, onde Cerberus (Kerberus para os gregos) é um cão com três cabeças que tem por missão proteger a entrada do inferno de Hades.

O que faz o Kerberos:

Para explicar o que faz o Kerberos, vamos usar algumas analogias baseadas The Moron's Guide to Kerberos, Version 1.2.2.

Na vida real, usamos rotineiramente uma autenticação, na forma de (por exemplo) uma carteira de motorista. A carteira de motorista é fornecida por uma agência, na qual podemos supor confiável, e contém dados pessoais da pessoa como nome, endereço e data de nascimento. Além disto pode conter algumas restrições, como necessidade de óculos para dirigir, e mesmo algumas restrições implícitas (a data de nascimento pode ser usada para comprovar maioridade). Finalmente, a identificação tem um prazo de validade, a partir do qual é considerada inválida.

Para ser aceita como autenticação, algumas regras devem ser observadas: ao apresentar a carteira, não se pode ocultar parte dela (como foto ou data de nascimento). Além disto, quem verifica a autenticação deve reconhecer a agência que forneceu a autenticação como válida (uma carteira de motorista emitida no Brasil pode não ser aceita fora do território nacional).

Kerberos trabalha basicamente da mesma maneira. Ele é tipicamente usado quando um usuário em uma rede tenta fazer uso de um determinado serviço da rede e o serviço quer se assegurar de que o usuário é realmente quem ele diz que é. Para isto, o usuário apresenta um *ticket*, fornecido pelo *Kerberos authenticator server* (AS), assim como a carteira de motorista é fornecida pelo DETRAN. O serviço então examina o *ticket* para verificar a identidade do usuário. Se tudo estiver ok o usuário é aceito.

O *ticket* deve conter informações que garantam a identidade do usuário. Como usuário e serviço não ficam face a face uma foto não se aplica. O *ticket* deve demonstrar que o portador sabe alguma coisa que somente o verdadeiro usuário saberia, como uma senha. Além disto, devem existir mecanismo de segurança contra um atacante que "pegue" um *ticket* e use mais tarde.

Explicando como funciona:

Vejamos como o Kerberos trabalha. Usuários e serviços que utilizem o Kerberos possuem chaves armazenadas no AS. As chaves dos usuários são derivadas de senhas escolhidas por estes, e as chaves dos serviços são geradas aleatoriamente.

Para esta explicação, imaginemos que as mensagens são escritas em papel, e são criptografadas colocando-as em uma caixa-forte, associada a uma chave.

1. Primeiro o usuário envia uma mensagem ao AS: "Eu, J. Random User, gostaria de falar com o servidor Foo".
2. Quando o AS recebe a mensagem, ele faz duas cópias de uma nova chave registrada. Estas chaves são chamadas de chaves de sessão. Elas serão usadas nas trocas diretas entre serviço e usuário.

3. Ele coloca uma das chaves de sessão na Caixa 1, junto com um pedaço de papel com o nome "Servidor Foo" escrito. A caixa é trancada com a chave do usuário (o AS conhece todas as chaves de usuário e todas as chaves de serviço).
4. *Por que este papel aqui? Devemos nos lembra que a caixa é na realidade apenas uma mensagem criptografada, e que a chave de sessão é uma seqüência randômica de bytes. Se a Caixa 1 somente contivesse a chave de sessão, o usuário não teria como reconhecer se a resposta veio do AS, ou não saberia se a decriptação teve sucesso. Pela adição da mensagem "Servidor Foo", o usuário (mais precisamente a aplicação do usuário) poderia saber se a caixa veio do AS, e se a decriptação obteve sucesso.*
5. Ele coloca a outra chave de sessão em uma Caixa 2, junto com um pedaço de papel com o nome "J. Random User". Esta caixa é fechada com a chave do serviço.
6. Ele envia ambas as caixas ao usuário.
7. *Na versão 4 do Kerberos, a Caixa 2 era colocada (sem necessidade) dentro da caixa 1, mas na versão 5 isto foi corrigido.*
8. O usuário destranca a Caixa 1 com sua chave, extraindo assim a chave de sessão e o papel com o nome "Servidor Foo" escrito nele.
9. O usuário não consegue abrir a Caixa 2 (ela foi trancada com a chave do serviço, que só o AS e o serviço conhecem). Então ele coloca um pedaço de papel com a hora corrente numa Caixa 3, e tranca esta com chave de sessão, e envia ambas ao serviço.
10. O serviço abre a Caixa 2 com sua própria chave, extraindo a chave de sessão e o papel com "J. Random User" escrito nele. Ele abre a Caixa 3 com a chave de sessão para extrair o pedaço de papel com a hora corrente nele. Estes itens demonstram a identidade do usuário.

> *O timestamp é colocado na Caixa 3 para prevenir que alguém faça uma cópia da Caixa 2 (devemos nos lembrar que as caixas são na verdade mensagens eletrônicas) e a utilize para se passar pelo usuário mais tarde. Como os relógios da rede nunca são perfeitamente sincronizados, uma pequena margem (em geral 5 minutos) é permitida entre o timestamp e a hora atual. Além disto, o serviço mantém um lista das autenticações recebidas recentemente para garantir que elas não foram reenviadas.*

A chave de serviço é gerada aleatoriamente e armazenada em um arquivo especial chamado de SECRETE KEY FILE. O Kerberos assume que este é seguro, que ninguém pode copiá-lo e se passar pelo serviço.

No Kerberos, a Caixa 2 é chamada de *ticket*, e a Caixa 3 de *authenticator*. O *authenticator* tipicamente contém mais informações do que as listadas acima. Algumas destas informações são adicionadas em decorrência do fato de que as mensagens são eletrônicas (por exemplo, existem checksum). Pode existir também uma chave secreta usada para criptografar as mensagens que serão trocadas entre usuário e serviço após a autenticação, garantindo assim a privacidade.

Autenticação de serviço:

Algumas vezes o usuário pode querer o serviço seja autenticado no retorno. Para isto, o serviço pega o timestamp do *authenticator* (Caixa 3), acrescenta um ao seu valor, coloca em uma Caixa 4, junto com um pedaço de papel com o nome "Servidor Foo" escrito nele, tranca esta caixa com a chave de sessão e envia de volta ao usuário. Ao receber a Caixa 4, o usuário abre com a chave de sessão, e verifica o timestamp. Se for maior que o enviado, significa que o serviço conseguiu abrir a Caixa 2 (decriptografar o *ticket*).

Isto acontece na versão 4 do Kerberos. Na versão 5 o serviço

faz mais que somente adicionar uma unidade ao timestamp.

O Ticket Granting Server:

Existe um problema no esquema apresentado acima. Ele será usado toda vez que o usuário requisitar um serviço, e a cada vez que isto acontecer ele terá que entrar com sua senha (destrancar a Caixa 1 com a chave do usuário). Uma primeira solução seria armazenar a chave do usuário gerada a partir da sua senha. Mas armazenar a chave do usuário é muito perigoso, pois com uma cópia desta um invasor se passaria pelo usuário até que sua senha fosse modificada.

O Kerberos resolve este problema introduzindo um novo agente chamado de *ticket* granting server (TGS). O TGS é logicamente distinto do AS, mas eles podem residir na mesma maquina. A função do TGS é a seguinte: antes de acessar qualquer serviço, o usuário requisita um *ticket* para contatar o TGS, como se ele fosse um serviço qualquer. Este *ticket* é chamado de *ticket granting ticket* (TGT).

Depois de receber o TGT, a qualquer momento que o usuário desejar requisitar um serviço, ele irá requerer um *ticket* não mais do AS, mais sim do TGS. Além disto, a resposta não será criptografada com a chave secreta do usuário, mas sim com a chave de sessão providenciada pelo AS para ser usada entre usuário e TGS. O conteúdo desta resposta é uma chave de sessão que será usada com o serviço regular. O resto da comunicação continua como descrito acima.

Este mecanismo funciona de maneira semelhante a um visitante num local de trabalho. Você mostra sua identificação(pode ser a carteira de motorista) e recebe uma identificação de visitante. Agora quando quiser entrar numa das várias salas, ao invés de apresentar sua identificação oficial a cada sala (corre-se o risco de ter a identidade perdida ou roubada), basta mostrar a identidade de visitante. Se sua identidade de visitante for perdida ou roubada, você poderá torna-la invalida e substituí-la rapidamente e com facilidade, algo que não acontece com sua carteira de motorista.

A vantagem é que enquanto senhas usualmente são válidas por meses, O TGT é válido somente por um curto período de tempo, tipicamente 8 horas. Depois deste tempo, o TGT não pode ser usado por ninguém, nem usuário nem invasor. Este TGT, como qualquer *ticket* que o usuário obtém é armazenado no *credentials cache*. Existe um número de comandos que permitem ao usuário manipular seu próprio *credentials cache*.

Cross Realm Authentication:

Um único AS e um único TGS funcionam bem se o número de requisições for pequeno, mas com o crescimento da rede, cresce o número de requisições de serviços, e o AS/TGS passa a ser um gargalo no processo de autenticação. Em outras palavras pode-se dizer que o sistema não esta escalado, o que é muito ruim para um sistema distribuído como o Kerberos.

Entretanto o Kerberos oferece a vantagem de se dividir a rede em *realms*. Esta divisão muitas vezes é feita sobre limites organizacionais. Cada *realm* tem seu próprio AS e seu TGS.

Para realizar uma *cross-realm authenticathion* (o usuário em um *realm* acessa um serviço noutro *realm*) é necessário primeiro que o *realm* do usuário registre um remote TGS (RTGS) no *realm* do serviço.

Note que quando o TGS foi adicionado, uma comunicação a mais foi somada ao protocolo. Aqui uma outra comunicação foi adicionada: primeiro o usuário contata o AS para acessar o TGS, então ele contata o TGS para acessar o RTGS. Finalmente ele contata o RTGS para acessar o serviço.

Em muitos casos, onde coexistem muitos *realms*, é ineficiente registrar cada *realm* em cada outro *realm*. Para evitar isto, a versão 5 do Kerberos permite uma hierarquia de *realms*, de maneira que para contatar um serviço num outro *realm*, pode ser necessário contatar RTGS em um ou mais *realms* imediatos. O nome de cada um destes *realms* é gravado no *ticket*.

Como utilizar o Kerberos:

Para o usuário:

Para o usuário utilizar o Kerberos, primeiro ele deve estabelecer um *Kerberos principal*. Um *Kerberos principal* é algo parecido com uma conta em uma máquina. O nome do *principal* é do tipo your_name@YOUR.REALM. A parte antes de @ é uma string você escolhe (normalmente é a mesma coisa que seu login name). A parte posterior é o nome do *realm*.

Associado a cada *principal* existe um nome, uma senha e algumas outras informações. Estes dados são armazenados na base de dados do Kerberos. Esta base de dados é criptografada com uma chave mestra do Kerberos, pode ser replicada para servidores escravos, e ela não pode ser examinada por qualquer um.

Para o usuário, o Kerberos é quase transparente. Existe um número de serviços setados para requerer autenticação via Kerberos. Para usar um destes serviços, o usuário precisa obter um TGT primeiro. O comando para isto é kinit:

```
% kinit

Password for your_name@YOUR.RELAM
```

Quando o usuário entrar sua senha, o programa kinit fará uma requisição ao AS para contatar o TGS. A senha será usada para computar a chave secreta do usuário (observe que a senha será digitada apenas uma vez e não trafegará pela rede), que será usada para decriptar parte da resposta do AS (a parte que contém a confirmação da requisição e a chave de sessão). Se a senha estiver correta, o usuário terá um TGT. Pode-se verificar isto usando o comando klist:

```
% klist
Ticket cache: /var/tmp/krb5cc_1234
Default principal: your_name@YOUR.REALM

Valid starting          Expires              Service principal
24-Jul-95 12:58:02    24-Jul-95 20:58:02    krbtgt/YOUR.REALM @YOUR.REAL
```

O campo Ticekt cache mostra qual arquivo contém as credenciais do usuário. O *principal* default é aquele fornecido pelo TGT. A saída é uma lista dos *ticket* existentes, Assim se o usuário apenas requisitou o TGT, este será o único *ticket* e o único *principal* será do serviço o do TGS (krbtgt). Pode-se observar que os *ticket* são válidos por um determinado período de tempo, neste caso 8 horas.

Se o usuário estiver usando uma versão kerberizada do rlogin, este programa irá usar o TGT do *credentials* cache para requisitar um *ticket* para o rlogin daemon na máquina em que está logado. Isto acontece automaticamente como pode-se notar abaixo:

```
% rlogin newhost.domain
last login: Fri 21 12:04:40 from ...
```

A única maneira de se notar a diferença é listando o credentials cache:

```
% klist
Ticket cache: /var/tmp/krb5cc_1234
Default principal: your_name@YOUR.REALM

Valid starting          Expires             Service principal
24-Jul-95 12:58:02    24-Jul-95 20:58:02   krbtgt/YOUR.REALM @YOUR.REALM
24-Jul-95 13:03:33    24-Jul-95 21:03:33   host/newhost.domain @YOUR.REALM
```

O significado do Service principal é o seguinte: A primeira parte, antes da barra, é o nome do *principal*. A Segunda parte, entre a barra e o @ é chamado de *instance*. Para serviços é normalmente o hostname da máquina onde o serviço está sendo executado, no caso de serviços do Kerberos (como kinit) ele é o nome do *realm*. Para usuários, normalmente é nulo (neste caso não existe nem a barra), ou quando o usuário acessa algum serviço privilegiado, existe uma indicação disto (como your_name/admin ou your_name/secure). O último componente, após o @ é o nome do *realm*.

Por default, o rlogin deixa qualquer *ticket* que ele tenha obtido no cache. Isto não é um problema de segurança, a menos que alguém tenha acesso ao terminal em que o usuário esteja logado,

Caso o usuário não deseje deixar os *credentials* no cache, ele pode destruí-los usando o comando kdestroy:

```
% kdestroy
% klist
klist: No credentials cache file found while setting cache flags
(ticket cache /var/temp/krbrcc_1234)
```

O comando kdestroy remove todos os *ticket* (incluindo o TGT) do cache.

Para o administrador:

Para o administrador a coisa é um pouco mais complexa. O AS e o TGS (normalmente o mesmo executável) devem ser configurados e iniciados. *Principals* devem ser registrados, e o mais importante, os serviços devem ser disponibilizados para uso do Kerberos.

Inicializando o servidor Kerberos: Tipicamente, o administrador deve seguir os seguintes passos para inicializar o servidor Kerberos:

1. Instalar os arquivos binários. Clientes podem ser colocados em um diretório comum. Certos binários (como ksu e todos os root processes) devem instalados pelo root para funcionarem corretamente. Em geral, make install da fonte de instalação fará tudo para o administrador.
2. Editar o arquivo krb5.conf. Este arquivo contém as opções de configuração para os clientes, como onde estão todos os KDC's, qual KDC é local, quais maquinas pertencem a quais *realms*, etc..
3. Editar o arquivo /.k5login. Cada linha consiste de um único nome de *principal*, que indica os direitos para o ksu. Ele é apenas uma lista de controle de acesso.
4. Editar o arquivo /etc/services.
5. Editar o arquivo /etc/inetd.conf. Idealmente todos os serviços devem estar fora do ar. Existem algumas versões Kerberizadas de comandos BSD que podem ser incluídos. Para fazer com que o daemon passe a responder esta nova versão deste arquivo, o administrador pode encontrar o inetd do processo e enviar o comando kill –HUP para o processo.
6. Editar o arquivo /etc/rc.<hostname>, adicionando linhas para fazer com que os daemons krb5kdc e kadmin sejam iniciados quando a máquina é bootada.
7. Criar a base de dados

8. Adicionar entradas para usuários e serviços.
9. Executar os servidores krb5kdc e kadmin em background. Eles farão isto por default a partir disto.

Registrando principals: Uma vez inicializada a base de dados, pode-se executar o kadmin para adicionar *principals*.

Kerberizando aplicações: Esta é a parte mais difícil do uso do Kerberos. Reconstruir uma aplicação para que ela passe a usar o Kerberos é chamado de kerberizar a aplicação. A aplicação kerberizada deve:

1. Encontrar a identidade do usuário.
2. Localizar o cache de *credentials* do usuário.
3. Checar para ver se existe um *ticket* para o serviço requisitado.
4. Se não existir, usar o TGT e a chave de sessão para enviar uma requisição para obter um.

Isto nem sempre é uma programação trivial. Entretanto existem aplicações pré-kerberizadas como POP e R-comandos Berkeley (por exemplo rlogin).

Limitações do Kerberos:

- O Kerberos é bastante adequado para aplicações usuário/servidor e não ponto a ponto.
- Como o cachemento das chaves, *ticket* e *principals* são feitos no diretório /tmp, é necessário que as estações de trabalho sejam seguras.
- Apresenta problemas com multi-homed hosts, que utilizam mais de um endereço IP.
- Os relógios devem estar sincronizados (em geral a diferença não pode ultrapassar cinco minutos) devido ao timestamp.
- É vulnerável contra senhas fracas, possibilitando que um invasor intercepte uma mensagem e utilize um ataque dicionário para descobrir a senha do usuário.
- Existe a possibilidade de modificação, por um atacante, das aplicações kerberizadas.

Bibliografia

- TUNG, BRYAN - "The Moron's Guide to Kerberos, Version 1.2.2".
- NEUMAN, B. CLIFFORD e TS'O, THEODORE - "Keberos: An Authentication Service For Computer Networks" IEEE Communications Magazine, Volume 32, No 9, pag. 33-38, Set 1994.
- NEUMAN, B. CLIFFORD e STUBBLEBINE, G. STUART - "A Note on the use of Timestamps as Nonces".
- NEUMAN, B. CLIFFORD - "Protection and Security Inssues for Future Systems".
- RFC 1510.
- NUNES, ANTÔNIO CARLOS NUNES e CARLE, EMERSON - "RSA: um método seguro de criptografia?".
- RIVEST, R. L. e SHAMIR, A. e ADLEMAN, L. - "A methode of obtaining digital signatures and public-key criptosystems".
- BELLOVIN, M. STEVEN e MERRITT, MICHAEL - "Limitations of the Kerberos Authentication System".
- NAKAMURA, EMILIO T. - "Kerberos Um seviço de Autenticação para redes de computadores". UNICAMP Trabalho para a disciplina Tópicos de Computadores II.