Aplicando Ataques de Dicionário no protocolo Kerberos do Windows 2000

Marcus Cunha Granado
Célio Cardoso Guimarães
Instituto de Computação - UNICAMP
{mgranado,celio}@ic.unicamp.br

Resumo

Este trabalho apresenta uma técnica para eficientemente explorar a implementação do protocolo Kerberos do Microsoft Windows 2000 de modo a obter senhas dos usuários do sistema. Esta técnica abre os sistemas Windows 2000 ao mesmo tipo de ataques de dicionários utilizados no protocolo NTLM do Windows NT 4.0. Um conjunto grande de senhas pode ser descoberto em um intervalo de tempo pequeno.

1 Introdução

O Kerberos, definido no RFC1510 ([1] e [12]), é um protocolo de autenticação originado no MIT no fim dos anos 80. A versão 5 revisão 6 do Kerberos [2] é utilizada amplamente no Microsoft Windows 2000 como um substituto ao protocolo de autenticação NT Lan Manager (NTLM) das versões anteriores deste sistema [2]. As falhas de segurança do protocolo NTLM eram notórias, como os famosos ataques de dicionário aplicados em cima dos *hashes* NTLM trafegando pela rede. Estas falhas têm sido apontadas desde 1997 ([7], [11] e [14]).

O protocolo NTLM é implementado no Windows 2000 apenas por motivos de compatibilidade com sistemas legados que não entendem o protocolo Kerberos (Windows 3.11, Windows 95, Windows 98, e Windows NT 4.0). O NTLM também é utilizado com sistemas Windows 2000 no modo *standalone*, e quando o usuário entra (*logs in*) localmente na máquina sem utilizar a rede.

O Windows 2000 também acrescenta algumas extensões da Microsoft ao Kerberos:

- Utilização de novos tipos de criptografia em pacotes Kerberos ([2] e [5]) (*etypes,* de *Kerberos encryption type*). Verificou-se nesses pacotes dois tipos principais baseados no algoritmo criptográfico RC4 e usando uma verificação (*checksum*) do tipo HMAC-MD5 ([3], [4] e [6]):

  1. KERB_ETYPE_RC4_HMAC (tipo 23)
  2. KERB_ETYPE_RC4_HMAC_EXP (tipo 24, semelhante ao anterior, para exportação para fora dos EUA)

- Utilização do campo AUTHORIZATION-DATA do ticket Kerberos para transportar SIDs (*Windows unique Security Identifier*) do usuário e dos grupos a que ele pertence [2].

A Microsoft vem apresentando o Kerberos como a resposta para todo tipo de problema relacionado à autenticação segura no sistema. Em sua implementação, um mecanismo do Kerberos para dificultar ataques, chamado *pré-autenticação,* é usualmente utilizado. No entanto, como será visto, o tipo de pré-autenticação escolhido pela Microsoft para ser usado, baseado em um *timestamp* (horário) criptografado com uma chave derivada diretamente da senha do usuário, é suscetível a um ataque de dicionário.

Não há uma pesquisa extensiva nesta área, aplicada especificamente aos sistemas operacionais da família Windows. Hobbit, em 1997, liberou um famoso artigo onde o compartilhamento de arquivos e protocolos de autenticação são explorados [7]. Em 1999, Thomas Wu delineou um interessante ataque a um grande ambiente UNIX na Universidade de Stanford usando Kerberos 4 e *etypes* baseados em DES [11]. Kenneth e Leighton [14] também descrevem um trabalho muito interessante usando engenharia reversa no protocolo *Server Message Block* (SMB) do Windows NT.

2 Logon através do Kerberos

O usuário começa realizando o *logon* na rede. Ele digita seu nome (*username*) e sua senha. O cliente Kerberos na máquina do usuário converte sua senha para uma *chave criptográfica de longo prazo* (*long-term key*) e salva o resultado no seu *cache* de credenciais. Esta chave é usada como uma *chave criptográfica simétrica* [6] para criptografar e descriptografar partes importantes da informação que irá atravessar a rede.

O cliente então envia uma mensagem de requisição de serviço para o *Serviço de Autenticação do Kerberos* (AS, de *Kerberos Authentication Service*), através da emissão do pacote KRB_AS_REQ. Este serviço é oferecido pelo *Centro de Distribuição de Chaves Kerberos* (KDC, de *Key Distribution Center*), existente em cada Controlador de Domínio do NT. A primeira parte desta mensagem identifica o usuário e o nome do serviço para o qual ele está requisitando credenciais (neste caso do *logon* inicial, o TGS (*Ticket-Granting Service*), que está no KDC). A outra parte da mensagem contém *dados de pré-autenticação* que provam que o usuário conhece a senha. Isto é usualmente um *timestamp* criptografado com a chave de longo prazo do usuário, embora o protocolo permita outras formas para pré-autenticação dos dados.

Após o recebimento do pacote KRB_AS_REQ, o KDC procura o nome do usuário em sua base de dados, pega a sua chave de longo prazo, descriptografa os dados de pré-autenticação, e avalia o *timestamp* lá contido. Se este *timestamp* for igual ao *timestamp* local da máquina (dentro de uma faixa de erro - 5 minutos, normalmente), o KDC pode estar seguro que os dados de pré-autenticação foram criptografados utilizando a chave de longo prazo do usuário.

Depois que confirmou a identidade do usuário, o KDC cria credenciais para o cliente apresentar ao TGS, incluindo uma chave de sessão e o TGT (*Ticket Granting Ticket*). O TGT é usado para quaisquer acessos posteriores a recursos do sistema. O KDC criptografa o TGT com sua própria chave de longo prazo[1], e a chave de sessão com a chave de longo prazo do usuário, e envia isto tudo de volta ao usuário em um pacote KRB_AS_REP de resposta do Serviço de Autenticação Kerberos.

De acordo com [5], a chave de cada principal para o etype RC4_HMAC guardada no KDC do Windows 2000, é, por default, gerada utilizando-se a seguinte função MD4 [3]:

```
String2Key(password) = MD4(UNICODE(password))
```[2]

Esta é exatamente a mesma função utilizada para criar os *hashes* NTLM do Windows NT, e foi mantida por motivos de compatibilidade, e para reaproveitamento do material criptográfico das senhas já existentes quando um *upgrade* do Windows NT para o Windows 2000 fosse feito. No entanto, outros tipos de chave, por exemplo, chaves DES [6], podem ser criadas, para utilização de *etypes* que usam o algoritmo criptográfico DES.

[1] Como o TGT somente pode ser enviado ao TGS - e este é uma parte do KDC - então neste caso o KDC é servidor e seu próprio futuro cliente.

[2] Cada caracter UNICODE do Windows é codificado em formato *little-endian* de 2 octetos cada. Então uma operação de *hash* criptográfico MD4 é realizada em apenas os caracteres UNICODE da senha (não incluindo os octetos zeros finais).

3 Atacando o campo de pré-autenticação

O campo de pré-autenticação abaixo *(PADATA)* foi extraído de um pacote KRB_AS_REQ após a execução do comando *'runas administrator'* (equivalente ao *'su'* do UNIX) no Windows 2000. Ele é do tipo PA_ENC_TIMESTAMP, ou seja, contém o *timestamp* local da máquina cliente no formato ASN.1 (*Abstract Syntaxe Notation One*) criptografado com a chave de longo prazo derivada da senha do usuário (no caso, a senha do administrador informada ao programa *'runas'*).

O campo *padata-value* (uma estrutura do tipo *EncryptedData*, conforme a seção 6.1 do RFC1510) possui dois subcampos. O primeiro deles é um campo *etype*, contendo o tipo de algoritmo criptográfico utilizado (no caso, 0x17 = 23 = KERB_ETYPE_RC4_HMAC). O segundo é um campo *cipher* que contém o *timestamp* criptografado.

```
KDC-REQ.PA_DATA (PRE-AUTHENTICATION DATA)
.  .  a3 5f [CONT 3] constr <95>
.  .  .  30 5d [UNIV 16] constr <93>
.  .  .  .  30 48 [UNIV 16] constr <72>
               PA-DATA.padata-type
.  .  .  .  .  a1 03 [CONT 1] constr <3>
.  .  .  .  .  .  02 01 [UNIV 2] 2              2 = PA_ENC_TIMESTAP
               PA-DATA.padata-value
.  .  .  .  .  a2 41 [CONT 2] constr <65>
.  .  .  .  .     04 3f [UNIV 4] <63>
.  .  .  .  .        30 3d
.  .  .  .  .           a0 03 02 01 17  etype  = KERB_ETYPE_RC4_HMAC
.  .  .  .  .           a2 36
.  .  .  .  .              04 34 97     cipher = <texto cifrado segue abaixo>
.  .  .  .  .                 97 dc 07 f7 a6 ee c6 17
.  .  .  .  .                 0e 18 c5 87 4c 2c 60 9b
.  .  .  .  .                 6d 2d 93 ff 8b f6 fc 72
.  .  .  .  .                 e1 f5 77 45 6e 32 91 a7
.  .  .  .  .                 d6 2b 15 a5 fd 3e d2 50
.  .  .  .  .                 fd 28 4a 9b 07 29 96 12
.  .  .  .  .                 4f cc bb b3
```

Um ataque de dicionário *offline* com texto claro conhecido (*known-plaintext attack*) [6] pode então ser aplicado. Caso a senha correta seja utilizada para descriptografar o texto cifrado do campo *padata-value*, um texto claro ASN.1 com a estrutura *timestamp* será encontrado.

Como exemplo, o campo de texto cifrado observado acima resulta no texto claro abaixo caso o algoritmo KERB_ETYPE_RC4_HMAC seja usado com a senha *'vmonlo;'* (sem as aspas simples). Os números entre parênteses correspondem ao tamanho do campo em bytes.

texto claro:

```
97 dc 07 f7 a6 ee c6 17 . checksum (16)
0e 18 c5 87 4c 2c 60 9b .
7d 24 5e 3e 7c 35 57 5f . confounder criptográfico (8)
30 1a                   . estrutura ASN.1 (28)
   a0 11 18 0f          . timestamp '20000529225244Z'
      32 30 30 30 30 35 32 39 32 32 35 32 34 34 5a
   a1 05 02 03
      0c f6 8a          . microssegundos (campo pausec)
```

O *timestamp* obtido acima significa "29/05/2000, 22:52:44", mais os microssegundos. O caracter *Z* do timestamp indica zona de tempo UTC (*Universal Time*).

3.1 Generalizando o ataque

A chave derivada da senha do usuário é utilizada apenas em 4 locais durante o subprotocolo AS de autenticação Kerberos:

1. Quando o cliente criptografa os dados de pré-autenticação em *AS_REQ.PA-DATA.padata-value.cipher;*

2. Quando o KDC descriptografa os dados de pré-autenticação *AS_REQ.PA-DATA.padata-value.cipher;*

3. Quando o KDC criptografa campos importantes como chave de sessão de logon dentro de *AS_REP.enc-part;*

4. Quando o cliente descriptografa o campo *AS_REP.enc-part.*

Um texto cifrado, criptografado com a chave de longo prazo derivada da senha do usuário, atravessa a rede entre as etapas 1-2 e as etapas 3-4, podendo ser capturado. Um ataque de dicionário *off-line* de texto claro conhecido poderia ser aplicado para a descoberta da senha em um tempo prático, da ordem de horas ou de poucos dias.

O ataque anteriormente descrito ao campo de pré-autenticação é aplicado entre as etapas 1-2.

Entre as etapas 3-4, e também na ausência de pré-autenticação (quando as etapas 1 e 2 não existem), é possível realizar um outro ataque, porque o campo *enc-part* devolvido em KRB_AS_REP também é criptografado com a mesma chave de longo prazo derivada da senha do usuário e possui campos com texto claro conhecido [1, sec.5.4.2], principalmente a string '*krbtgt*' e *timestamps* no formato ASN.1. Especialmente importante, dentro de *enc-part* encontra-se a chave de sessão. Com a chave de longo prazo correta, *a chave de sessão pode ser obtida para forjar autenticadores válidos* para o *ticket* TGT de acesso que também foi devolvido junto no KRB_AS_REP. O TGT, junto com autenticadores válidos, permite personificar usuários e ter acesso aos recursos dos servidores, como arquivos, durante o tempo de vida do TGT, mesmo depois que o usuário já saiu (*logged out*) do sistema.

Um interessante estudo de um ataque em larga escala em um ambiente UNIX com Kerberos 4, sem pré-autenticação, e utilizando *etypes* baseados em DES pode ser encontrado em [11].

4 O Protótipo

No programa não-otimizado desenvolvido para testar a viabilidade de encontrar o texto claro original do campo de pré-autenticação, cada estrutura *padata-value* pode ser descriptografada em *0,30 ms* em um Pentium 166MHz (isto equivale a uma taxa de 3333 tentativas por segundo). Ele foi desenvolvido baseando-se no RFC 1510, em documentos da Microsoft [5], e em códigos fontes de distribuições Kerberos encontradas na Internet, como o pacote *Heimdal* [10] e o pacote *Krb5* do Linux. Versões especiais do 'tcpdump 3.5' (para obter os pacotes de *logon* Kerberos da rede) e 'John the Ripper 1.6' (para aplicar o ataque de dicionário com variações de palavras) foram criadas para implementar o ataque.

4.1 O Algoritmo

O algoritmo abaixo é funcionalmente equivalente ao utilizado no Windows 2000, conforme foi comprovado através do correto funcionamento do protótipo.

```
; Algoritmo para obter o texto claro
; usando etype=KERB_ETYPE_RC4_HMAC
;
; Obtém a chave do usuário
K=String2Key(senha)          (etapa 1)
;
;
; Decodifica PADATA:
; . campo cifrado = chksum,edata
```

```
; . edata = dados criptografados
; . data  = dados em texto claro
; . T=1 (KRB_AS_REQ), T=8 (KRB_AS_REP)[5]
; . K1,K3 = hashes temporários
;
K1=HMAC_MD5(K,T)              (etapa 2)
K3=HMAC_MD5(K1,chksum)        (etapa 3)
data=RC4(K3,edata)            (etapa 4)
```

Caso a senha correta seja usada, depois da etapa 4 o campo *data* contém uma estrutura ASN.1 com um *timestamp*.

4.2 Otimizações

Várias otimizações podem ser feitas ao algoritmo inicial, destacando-se:

- As etapas 1 e 2 somente precisam ser feitas uma vez para cada palavra do dicionário. A função *String2Key* para o *etype* analisado do Windows 2000 não utiliza o conceito de '*salt*'[3] das senhas UNIX de forma que, *para cada senha* do dicionário, é possível armazenar o resultado do fim da etapa 2 e aplicar *apenas* as etapas 3 e 4 para todos os textos cifrados dos usuários encontrados (as etapas 3 e 4 são dependentes do texto cifrado).
- Calcular antecipadamente as etapas 1 e 2 para cada uma das palavras do dicionário, e armazenar os resultados em memória/disco para utilizar na hora do ataque.
- Descriptografar apenas os primeiros 9 bytes de *edata*[4]. Os primeiros 8 bytes são o *confounder,* que é composto por uma string aleatória, cujo texto claro não se conhece. O byte seguinte deve ser o valor 0x30 (em hexadecimal), que indica uma estrutura ASN.1. Se não for 0x30, então o resto pode ser ignorado. Se for, então ainda há uma chance em 256 de que o valor foi uma coincidência. Mais alguns bytes devem ser descriptografados e comparados para assegurar que o texto foi descriptografado corretamente.

5 Possíveis defesas ao ataque

Algumas defesas podem ser utilizadas para evitar esta vulnerabilidade. Textos claros não deveriam ser cifrados com a senha do usuário ou com qualquer função que possua apenas a senha como parâmetro, já que na média os usuários escolhem senhas simples.

O acoplamento da autenticação inicial Kerberos (subprotocolo AS) com protocolos modernos de transmissão segura de senhas, como os protocolos PAK (*Provably Secure Password Authentication and Key Exchange using Diffie-Hellman*) de Phil MacKenzie [15], SRP (*Secure Remote Password Protocol*) de Thomas Wu [8], e os protocolos seguros de R.Perlman e C.Kaufmann [9], onde as chaves derivadda senha do usuário nunca são utilizadas para transmitir qualquer informação (*zero-knowledge*), e portanto são imunes a ataques de dicionários.

Uma solução de curto prazo seria utilizar extensões Kerberos que permitam chaves públicas/privadas resistentes a ataques de dicionários como a extensão PKINIT [13]. Existe uma implementação PKINIT no Windows 2000, mas esta necessita que *smartcards* sejam usados no *logon* inicial. Esta não é uma configuração usual, e são raras as instalações que utilizam estes dispositivos, provavelmente pelo custo e por não conhecerem este ataque eficiente ao campo de pré-autenticação do Kerberos no Windows 2000. Outro problema relacionado com este tipo de extensão é que é baseada em 'alguma coisa que o usuário tem' (*stored-key approach*), que apenas

[3] O *salt* é uma string aleatória acrescentada no fim de cada senha do usuário. Quando o *salt* existe, tanto ele quanto a senha são parâmetros da função de cálculo *String2Key*, de modo que dois usuários com a mesma senha dificilmente vão ter a mesma chave. Alguns *etypes* utilizam *salt*.

[4] O algoritmo RC4 [6] permite descriptografar um texto cifrado byte a byte.

verifica a presença do dispositivo, e não 'alguma coisa que o usuário sabe' (*knowledge-based approach*), que verifica a presença de vida humana. Além disto, toda uma complicada infraestrutura de chave pública deve ser implementada e gerenciada.

6 Conclusão

O ataque descrito ao campo de pré-autenticação Kerberos permite a descoberta *off-line* de senhas do sistema em um período de tempo prático, abrindo o Windows 2000 ao mesmo tipo de ataques de dicionário existentes no protocolo NTLM do Windows NT 4.0.

Isto mostra que o método PA_ENC_TIMESTAMP para pré-autenticação Kerberos usado no Windows 2000 foi uma má escolha para um sistema que está se esforçando para ser seguro.

Referências

[1] J.Kohl, C.Neuman, *The Kerberos Network Authentication Service (V5)*, RFC1510, Setembro de 1993, e draft-ietf-cat-kerberos-revisions-06.txt, 14 de Julho de 2000

[2] *Windows 2000 Server Authentication*, Microsoft White Paper, 1999 e Microsoft Platform SDK, *Kerbcon.h*, 2000

[3] R.Rivest, *MD4/MD5 Message-Digest Algorithm,* RFC-1320/RFC-1321, Abril de 1992

[4] H.Krawczyk, M.Bellare, R.Canetti, *HMAC: Keyed-Hashing for Message Authentication*, RFC 2104, Fevereiro de 1997

[5] M.Swift, J.Brezak, *The Windows 2000 RC4-HMAC Kerberos encryption type*, draft-brezak-win2k-krb-rc4-hmac-03.txt, Junho de 2000

[6] B.Schneier, *Applied Cryptography.* New York: Wiley, 1996

[7] Hobbit, *CIFS: Common Insecurities Fail Scrutiny*, Avian Research, Janeiro 1997. Disponível em http://www.avian.org

[8] T.Wu, *The Secure Remote Password Protocol*, Proceedings of the 1998 Network and Distributed System Security Symposium, Disponível em http://www.isoc.org/ndss98/

[9] R.Perlman, C.Kaufman, *Secure Password-Based Protocol for Downloading a Private Key*, Proceedings of the 1999 Network and Distributed System Security Symposium. Disponível em http://www.isoc.org/ndss99/proceedings

[10] Heimdal - A free Implementation of Kerberos 5. Disponível em http://www.pdc.kth.se/heimdal

[11] T.Wu, *A Real-World Analysis of Kerberos Password Security*, Proceedings of the 1999 Network and Distributed System Security Symposium. Disponível em http://www.isoc.org/ndss99/proceedings

[12] J.G.Steiner, B.C.Newman, and J.I.Schiller. *Kerberos: An Authentication service for open network systems.* In USENIX Conference Proceedings, pages 191-202, Fevereiro de 1988.

[13] B.Tung et al, *Public Key Cryptography for Initial Authentication in Kerberos*, draft-ietf-cat-kerberos-pk-init-12.txt, Julho de 2000.

[14] L.Kenneth, C.Leighton, *DCE/RPC over SMB - SAMBA and Windows NT Domain Internals*, Macmillan Technical Publishing, 2000

[15] P.MacKenzie, V.Boyko, S.Patel, *Provably Secure Password Authentication and Key Exchange Using Diffie-Hellman*, EuroCrypt 2000, pp. 156-171 (PAK, PAK-X, and PPK protocols). Disponível em http://www.bell-labs.com/user/philmac/pak.html