ЖУРНАЛ ОТ КОМПЬЮТЕРНЫХ ХУЛИГАНОВ

ХАКЕР

WWW.XAKEP.RU

ИЮНЬ 06(90) 2006

Жестокий баг в IE

Вскрываем платные спутниковые каналы

Троян для кражи webmoney

Учимся вводить ядовитый PHP-код

Восстановление сдохших хардов

Настраиваем сервер Novell NetWare

(game)land
PUBLISHING FOR ENTHUSIASTS
RUSSIA \ РОССИЯ
WE ARE HACKERS.
WE ARE TOGETHER

ISSN 1609-1019

МИХАИЛ ЕМЕЛЬЧЕНКОВ
/ MICHAEL@EMELTCHENKOV.NET /

Шифрование дисков с помощью Dm-crypt

ОСТАВЛЯТЬ ИНФОРМАЦИЮ НЕЗАШИФРОВАННОЙ — ЗНАЧИТ ПОДВЕРГАТЬ СВОИ ДАННЫЕ ОПАСНОСТИ. СЕГОДНЯ Я РАССКАЖУ ТЕБЕ, С ПОМОЩЬЮ ЧЕГО, КАК И ЗАЧЕМ МОЖНО ШИФРОВАТЬ ДАННЫЕ В ОС LINUX, НАЧИНАЯ ОТ СОЗДАНИЯ ЗАШИФРОВАННОГО ДИСКА И ЗАКАНЧИВАЯ ШИФРОВАНИЕМ ВРЕМЕННЫХ ДАННЫХ. ПОСЛЕ ПРОЧТЕНИЯ ЭТОЙ СТАТЬИ И ПРИМЕНЕНИЯ ПОЛУЧЕННЫХ ЗНАНИЙ НА ПРАКТИКЕ ТЫ СМОЖЕШЬ СПАТЬ СПОКОЙНО.

> После осознания могут быть самые разные. Например, если похищен текст книги, над которой работали многие месяцы, то потери авторов могут составить несколько тысяч долларов, а если книга уже издана, то похищение ее текста может создать книге дополнительную рекламу. Другое дело — искаженная информация. Она представляет гораздо большую опасность. Например, если данные организации об инвентарных описях или списках заказов будут стерты, то работа парализуется надолго. Существует несколько способов шифрования в Linux:
> •шифрование отдельных файлов (выполняется с помощью GnuPG);
> •шифрование дисков (можно выполнить с помощью Dm-crypt, а посредством Dm-crypt — создать виртуальный шифрованный диск, который будет располагаться в физическом файле на диске).
> Device-mapper — новая инфраструктура ядра Linux 2.6, которая позволяет создавать виртуальные устройства, работающие поверх физических. Dm-crypt отличается от Cryptoloop более чистым кодом и удобством в настройке. Я рассмотрю применение Dm-tools для Debian и Gentoo. Для остальных дистрибутивов процедуры настройки будут аналогичны.

Установка Dm-crypt

Сначала необходимо настроить конфигурацию ядра, запустив графическую оболочку для его настройки:

```
# cd /usr/src/linux
# make menuconfig
```

Необходимые опции ядра:

1. Подключаем опции, имеющие статус разрабатываемых или экспериментальных: Code maturity level options -> Prompt for development and/or incomplete code/drivers.

2. Данная опция необходима для корректной работы с udev: General setup -> Support for hot-pluggable devices.

3. Поддержка непосредственно самого Device-mapper: Device Drivers -> Multi-device support (RAID and LVM) -> Device mapper support.

4. Поддержка шифрования через Device-mapper: Device Drivers -> Multi-device support (RAID and LVM) -> Crypt target support.

5. Поддержка алгоритма шифрования AES: Cryptographic options -> AES cipher algorithms.

6. Поддержка алгоритма хэширования SHA1: Cryptographic options -> SHA1 digest algorithm.

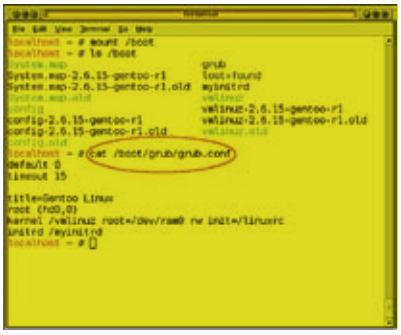
7. Поддержка виртуальных RAM-дисков: Device Drivers -> Block devices -> RAM disk support.

8. Поддержка так называемого начального RAM-диска: Device Drivers -> Block devices -> RAM disk support -> Initial RAM disk (initrd) support.

Теперь компилируем и устанавливаем ядро в /boot:

```
# mount /boot
# make
# make modules_install install
```

Для простоты настройки сконфигурируем все пункты не модулями,



Консоль: cat /boot/grub/grub.conf

а монолитом. Если ты решишь сделать модульную сборку, то не забудь предварительно подгрузить соответствующие модули с помощью modprobe. После перезагрузки необходимо установить user-space утилиты. Для этого воспользуйся командой:

```
debian# apt-get install cryptsetup
gentoo# emerge device-mapper cryptsetup
```

После чего удостоверься, что device mapper запущен:

```
/dev/mapper/control
```

Проверь также, появился ли crypt target:

```
# /sbin/dmsetup targets
crypt           v1.1.0
striped         v1.0.2
linear          v1.0.1
error           v1.0.1
```

Шифрование root-раздела

Я рассмотрю шифрование корневого раздела только для дистрибутива Gentoo, в Debian все происходит схожим образом. Для шифрования корневого раздела необходимо, чтобы /boot-директория располагалась на отдельном разделе. Linux не поддерживает загрузку с шифрованных разделов напрямую. Вместо этого необходимо использовать initrd (RAM-диск, который грузится до монтирования корневого раздела).

Для корректной работы с udev потребуется собрать multipath-tools:

```
# emerge multipath-tools
```

Необходимо создать и примонтировать initrd. Для этого монтируем /boot-раздел и создаем пустой файл initrd:

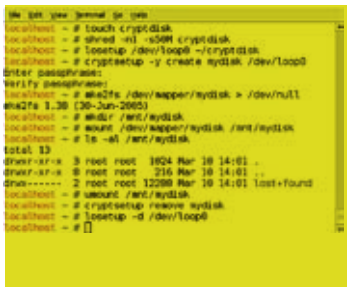
```
# mount /boot
# touch /boot/initrd
```

Заполняю нулями файл /boot/initrd и придаем ему размер 4 Мб:

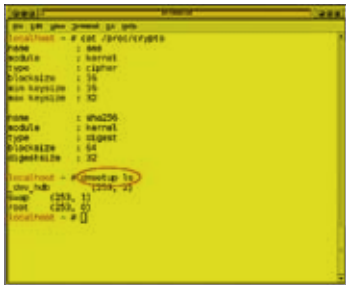
```
# dd if=/dev/zero of=/boot/initrd bs=1M count=4
```

Создаем loopback-устройство для работы с файлом:

```
# /sbin/losetup /dev/loop0 /boot/initrd
```



Консоль: создание зашифрованного раздела в виде файла на диске



Консоль: dmsetup ls

На сайте www.citforum.ru/security/cryptography/ можно найти статьи о криптографии, электронной подписи и программе GnuPG.

```
# числовые операции
test1 = 9
test2 = test1
stop
```

Создаем на нем файловую систему Ext2:

```
# /sbin/mke2fs /dev/initrd
# mkdir /mnt/initrd
```

Монтируем loopback-устройство на каталог диска:

```
# mount /dev/loop0 /mnt/initrd
```

Создаем дерево директорий и файлы скриптов:

```
# cd /mnt/initrd
# mkdir etc dev lib bin proc new
# touch linuxrc devmap_mknod.sh
# chmod +x linuxrc devmap_mknod.sh
```

Где linuxrc — скрипт, отвечающий за непосредственную загрузку, а devmap_mknod.sh — скрипт, который создает необходимые устройства. Оба файла находятся на диске, прилагаемом к журналу. Копируем необходимые файлы:

```
# cp /bin/{sh,cat,mount,umount,mkdir,chroot,cryptsetup,sleep, mknod,
        sed,rm} ./bin
```

Для каждого файла узнаем требуемые библиотеки и копируем их в директорию lib:

```
# ldd /bin/sh
        linux-gate.so.1 => (0xffffe000)
        libdl.so.2 => /lib/libdl.so.2 (0xb7f65000)
        libc.so.6 => /lib/libc.so.6 (0xb7e51000)
        /lib/ld-linux.so.2 (0xb7f73000)
# cp /lib/{libc.so.6,ld-linux.so.2} ./lib
```

Создаем необходимые устройства, без которых не удастся загрузить систему (если ты используешь другой раздел с корневой файловой системой, то необходимые параметры mknod можно посмотреть с помощью «ls -l /dev/drive». Например, для /dev/hda3 команда будет выглядеть так: «mknod /mnt/initrd/dev/hda3 b 3 3»):

```
# mknod /mnt/initrd/dev/console c 5 1
# mknod /mnt/initrd/dev/null c 1 3
# mknod /mnt/initrd/dev/sda2 b 8 2
# mknod /mnt/initrd/dev/tty c 5 0
```

Проверим работоспособность всех программ в initrd:

```
# chroot /mnt/initrd /bin/sh
# /bin/chroot --help
# /bin/mkdir --help
```

Размонтируем initrd:

```
# cd ~
# umount /mnt/initrd /boot
```

Теперь редактируем fstab, чтобы он соответствовал новому корневому разделу. В данном примере опция noatime необязательна, но она позволяет ускорить дисковые операции, так как отключает запись времени последнего обращения к файлу.

```
# vi /etc/fstab
/dev/mapper/root / reiserfs noatime 0 1
```

Следующий шаг — настройка grub:

```
# vi /boot/grub/grub.conf
title=Gentoo Linux with initrd
root (hd0,0)
kernel /vmlinuz root=/dev/ram0 rw init=/linuxrc
initrd /initrd
```

Подготовительный процесс завершен. Остается зашифровать корневую файловую систему. Для этого нам понадобится чистый раздел достаточной емкости (обозначим его как /dev/sdb1). Корневой раздел обозначим как /dev/sda2.

```
# mkdir /mnt/{sda2,sdb1}
# mount /dev/sda2 /mnt/sda2
# mount /dev/sdb1 /mnt/sdb1
```

Копируем все содержимое корневого раздела в чистый раздел:

```
# cp -r -p -v /mnt/sda2/* /mnt/sdb1/
# umount /dev/sda2
```

Выполняем подготовительные процедуры для загрузки с копии системы (с */dev/sdb*1):
Редактируем файл */mnt/sdb1/etc/fstab*:

```
# vi /mnt/sdb1/etc/fstab
/dev/sdb1     / reiserfs noatime 0 1
```

Редактируем /boot/grub/grub.conf:

```
# vi /boot/grub/grub.conf
title=Gentoo Linux - Boot from copy
root (hd0,0)
kernel /vmlinuz root=/dev/sdb1
```

Размонтируем устройство:

```
# umount /dev/sdb1
```

После загрузки с копии системы создаем зашифрованный корневой раздел:

```
# /bin/cryptsetup -y create root /dev/sda2
```

Система потребует ввести пароль шифрования с подтверждением. Этот пароль будет спрашиваться при загрузке ОС. По умолчанию используется алгоритм шифрования AES с длиной ключа 256 бит. Это очень надежный алгоритм, применяемый американскими спецслужбами для шифрования секретных документов. Доступные алгоритмы шифрования можно посмотреть командой «cat /proc/crypto». С помощью ключа '-c' можно указать, какой алгоритм использовать, с помощью '-h' — выбрать алгоритм хэширования.
Теперь создаем файловую систему ReiserFS (или любую другую):

```
# mkreiserfs /dev/mapper/root
```

Копируем корневую систему на созданный зашифрованный раздел:

```
# mount /dev/mapper/root /mnt/sda2
# mount /dev/sdb1 /mnt/sdb1
# cp -r -p -v /mnt/sdb1/* /mnt/sda2/
# umount /dev/sdb1 /dev/sda2
```

Все. Система готова к загрузке с зашифрованного раздела.

Шифрование SWAP

Для Debian:

1. Удостоверься, что в файле /etc/defaults/cryptdisks присутствует следующая строка:

```
# vi /etc/defaults/cryptdisks
CRYPTDISKS_ENABLE=Yes
```

2. Отредактируй файл /etc/crypttab для настройки шифрования свопа (где /dev/sda5 – имя раздела со свопом).

```
# vi /etc/crypttab
cryptswap /dev/sda5 /dev/urandom swap,cipher=aes,size=256,swap
```

3. Отредактируй /etc/fstab, чтобы вместо обычного своп-раздела использовался раздел Device-mapper.

```
# vi /etc/fstab
/dev/mapper/cryptswap   none swap   sw 0 0
```

Для Gentoo порядок действий будет выглядеть следующим образом:

1. Добавь в файл /etc/conf.d/cryptfs строчку:

```
# vi /etc/conf.d/cryptfs
swap=cryptswap source='/dev/sda5'
```

2. Отредактируй файл /etc/fstab:

```
# vi /etc/fstab
/dev/mapper/cryptswap   none swap   sw 0
```

В результате мы получим своп-раздел cryptswap, шифруемый случайным ключом. Но сначала неплохо бы затереть старый своп случайными данными, так как в свопе могут находиться куски приватных данных, оставленные там после работы различных программ:

```
# swapoff
# dd if=/dev/urandom of=/dev/sda5 bs=1M
```

После этого можно перезагрузиться и начать использовать новый зашифрованный своп. Пароль при загрузке запрашиваться не будет.

Шифрование home-раздела

Далее создадим шифрованный home-раздел:

1. Затираем раздел случайными данными:

```
# dd if=/dev/urandom of=/dev/sdb1 bs=1M
```

Где /dev/sdb1 — раздел, на котором будет располагаться шифрованный диск, а michael — имя логического диска (/dev/mapper/michael). Внимание: пароль должен совпадать с паролем логина.

```
# cryptsetup –y create michael /dev/sdb1
```

2. Проверим, что это работает:

```
# dmsetup ls
michael (254, 1)
cryptswap (254, 0)
```

3. Создаем файловую систему Ext3 (естественно, вместо Ext3 может выступать любая ФС):

```
# mke2fs -j /dev/mapper/michael
```

4. Отмонтируем раздел:

```
# dmsetup remove michael
```

В случае использования целого диска, а не раздела, можно везде указывать /dev/sdb вместо /dev/sdb1.

Для автомонтирования раздела скачиваем (ftp.debian.org/debian/pool/main/libp/libpam-mount/) и устанавливаем libpam-mount:

```
# dpkg –i libpam-mount_0.9.22-6_i386.deb
```

В случае с Gentoo скачиваем ebuild с bugs.gentoo.org/attachment.cgi?id=64090 и распаковываем его в /usr/local/portage/sys-libs. Далее добавляем строчку в /etc/make.conf:

```
# vi /etc/make.conf
PORTDIR_OVERLAY=/usr/local/portage
```

Собираем pam_mount (сначала добавим строчку в /etc/portage/package.keywords, так как он помечен нестабильным):

```
# echo “sys-libs/pam_mount ~x86” >> /etc/portage/package.keywords
# emerge pam_mount
```

Для Debian процесс настройки заключается в редактировании конфигурационных файлов /etc/login.defs и /etc/pam.d/{common-auth,common-session,pam_mount.conf}:

```
# vi /etc/pam.d/common-auth
auth        optional        pam_mount.so use_first_pass
# vi /etc/pam.d/common-session
session optional            pam_mount.so

# vi /etc/security/pam_mount.conf
volume michael crypt - /dev/sdb1 /home/michael cipher=aes - -

# vi /etc/login.defs
CLOSE_SESSIONS yes
```

Для Gentoo ситуация схожа:

1 Добавляем в файл */etc/pam.d/login* строки:

```
# vi /etc/pam.d/login
auth        optional        /lib/security/pam_mount.so use_first_pass
session     optional        /lib/security/pam_mount.so
```

2 Добавляем в файл /etc/security/pam_mount.conf строку:

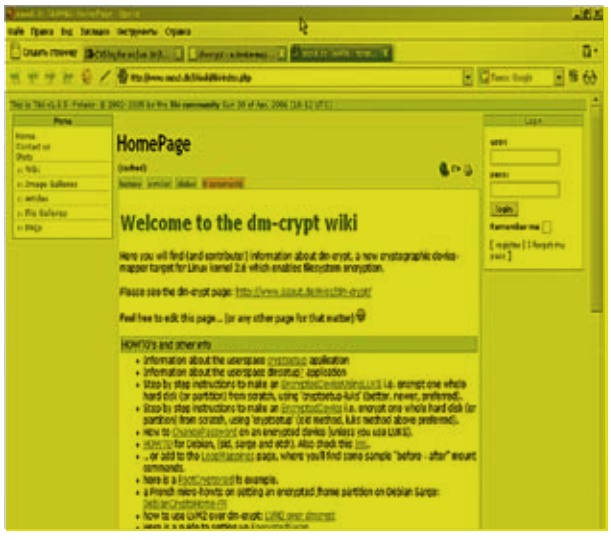
```
# vi /etc/security/pam_mount.conf
volume michael crypt - /dev/sdb1 /home/michael cipher=aes - -
```

После этого входим под именем michael. Диск должен автоматически примонтироваться на /home/michael. Беда в том, что права доступа у каталога будут root:root. Необходимо их сменить на пользовательские:

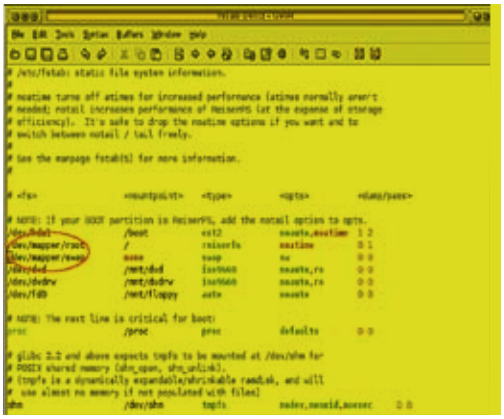
```
# chown michael:users /home/michael
```

Кроме того, существует возможность шифрования раздела не паролем логина, а отдельно сгенерированным ключом. В этом случае можно легко сменить пароль логина без необходимости решифрации диска. Правда, зная пароль логина, можно легко получить этот ключ. Поэтому использование данного способа шифрования я считаю неоправданным. Другое дело, если ключ находится на внешнем носителе, например на USB-флешке. Тогда, зная пароль, но не имея флеш-носителя, расшифровать раздел будет невозможно.
Устанавливаем openssl:

```
debian# apt-get install openssl
```



Консоль: wiki, посвященная Dm-crypt



Консоль: gvim /etc/fstab

```
gentoo# emerge openssl
```

Создаем ключ длиной 256 бит:

```
# cat /dev/urandom l head –c 32 > /home/michael.key
```

Создаем шифрованный home-раздел:

```
# dd if=/dev/urandom of=/dev/sdb1 bs=1M
# cat /home/michael.key l cryptsetup create michael /dev/sdb1
# mke2fs -j /dev/mapper/michael
# dmsetup remove michael
```

Шифруем ключ, которым зашифрован диск:

```
# cat /home/michael.key l openssl aes-256-ecb > /home/michael.key
```

В запросе на ввод пароля пишем пароль логина.
Далее происходит процесс настройки — он такой же, как и с шифрованием без ключа, за исключением одной строчки:

```
# vi /etc/security/pam_mount.conf
volume michael crypt - /dev/sdb1 /home/michael cipher=aes aes-256-ecb /home/michael.key
```

Раздел в виде файла на диске

Для начала необходимо создать файл заранее определенного размера, предположим, 50 Мб:

```
# touch cryptdisk
# shred -n1 -s50M cryptdisk
```

Желательно затереть cryptdisk именно таким способом, так как в результате получится набор случайных данных, и нельзя будет точно узнать, сколько реальной информации хранится в этом файле. Настало время создать шифрованный раздел посредством loopback-устройства:

```
# losetup /dev/loop0 ~/cryptdisk
# cryptsetup -y create mydisk /dev/loop0
# mkreiserfs /dev/mapper/mydisk
# mkdir /mnt/mydisk
# mount /dev/mapper/mydisk /mnt/mydisk
```

Зашифрованный раздел теперь доступен через /mnt/mydisk. После завершения работы с приватными данными его следует размонтировать, а затем удалить loopback-устройство, как показано ниже:

```
# umount /mnt/mydisk
# cryptsetup remove mydisk
# losetup -d /dev/loop0
```

При желании можно написать небольшой скрипт, который будет выполнять все эти команды автоматически.