

ВОЕННО-КОСМИЧЕСКАЯ АКАДЕМИЯ имени А.Ф.МОЖАЙСКОГО

Кафедра № 27 Математического и программного обеспечения

УТВЕРЖДАЮ

Начальник 27 кафедры

полковник С. Войцеховский

(воинское звание, подпись, инициал имени, фамилия)

« » 2022 г.

Автор: доцент 24 кафедры к.т.н. подполковник С. Краснов

(должность, ученая степень, ученое и воинское звание,
инициал имени, фамилия)

Задание на практическое занятие № 11

Тема: «Настройка механизмов организации замкнутой программной среды.
контроль целостности КСЗ»

(наименование темы семинара, лабораторной работы, практического занятия
и других видов учебных занятий по тематическому плану изучения
дисциплины)

Обсуждено и одобрено на заседании кафедры
(предметно-методической комиссии)

« » 20 г.

протокол №

Санкт-Петербург
2022

Перечень заданий:

Обучающийся должен на практическом занятии выполнить индивидуальное задание определенное преподавателем из списка индивидуальных заданий.

1. Цель работы: изучить принципы и технологии контроля целостности данных (в том числе, комплекса средств защиты – КСЗ), реализованных в ОССН. Освоить умения, необходимые для решения задач подсчёта контрольных сумм файлов и оптических носителей, контроля соответствия дистрибутиву, регламентного контроля целостности и создания замкнутой программной среды.

Материально-техническое обеспечение: ОС Astra Linux Special Edition 1.6 (Версия Смоленск, пользователь leti пароль 11111111) с установленным оперативным обновлением 20211126SE16.iso (оперативное обновление №10), установочный диск ОС AstraLinux_Smolensk-1.6-20.06.2018.iso.

Замкнутая программная среда

Инструменты замкнутой программной среды (ЗПС) предоставляют возможность внедрения цифровой подписи в исполняемые файлы формата ELF, входящие в состав устанавливаемого специального программного обеспечения (СПО), и в расширенные атрибуты файловой системы.

ЗПС обеспечивает противостояние угрозам безопасности информационной системы путем запрещения запуска несертифицированного ПО в ОС. Также, в литературе можно встретить обозначение ЗПС как динамический контроль целостности. ЗПС обеспечивает автоматический контроль доступа субъектов к объектам на предмет запуска программ на выполнение. Контроль осуществляется по наличию или отсутствию у объектов электронной цифровой подписи (ЭЦП), проверяются права доступа и т.п.

Большая часть уязвимостей в информационных системах связана именно с ошибками в программное обеспечение (ПО), которые допускают возможность оперативного вредоносного внедрения в код исполняемых программ напрямую в оперативной памяти. С целью недопущения подобных фактов в ядро ОССН встраивается специальный модуль, который предотвращает исследование выполняемого кода программ в оперативной памяти, определение регистров памяти, запрещение запуска кода программ из сегмента данных, изменение кода в сегментах оперативной памяти.

Режимы работы механизма ЗПС при запуске программы:

штатный режим функционирования;

режим для проверки ЭЦП в СПО;

отладочный режим для тестирования СПО.

В ОС Astra Linux механизм контроля целостности исполняемых файлов и разделяемых библиотек формата ELF при запуске программы на выполнение реализован в модуле ядра `digsig_verif`, который является не

выгружаемым модулем ядра Linux и может функционировать в одном из следующих режимов:

1) исполняемым файлам и разделяемым библиотекам с неверной ЭЦП, а также без ЭЦП загрузка на исполнение запрещается (штатный режим функционирования);

2) исполняемым файлам и разделяемым библиотекам с неверной ЭЦП, а также без ЭЦП загрузка на исполнение разрешается, при этом выдается сообщение об ошибке проверки ЭЦП (режим для проверки ЭЦП в СПО);

3) ЭЦП при загрузке исполняемых файлов и разделяемых библиотек не проверяется (отладочный режим для тестирования СПО).

Механизм контроля целостности файлов при их открытии на основе ЭЦП в расширенных атрибутах файловой системы также реализован в модуле ядра ОС `digests_verify` и может функционировать в одном из следующих режимов:

1) запрещается открытие файлов, поставленных на контроль, с неверной ЭЦП или без ЭЦП;

2) открытие файлов, поставленных на контроль, с неверной ЭЦП или без ЭЦП разрешается, при этом выдается сообщение об ошибке проверки ЭЦП (режим для проверки ЭЦП в расширенных атрибутах файловой системы);

3) ЭЦП при открытии файлов не проверяется.

Также в ОС СН существует механизм контроля целостности файлов при их открытии на основе ЭЦП. Например, мы можем при наличии сгенерированной ЭЦП самостоятельно подписывать файлы. Этот механизм может запрещать открытие файлов, поставленных на контроль, с неверной ЭЦП или же в случае ее отсутствия. Возможен режим функционирования, когда открытие файлов, поставленных на контроль, с неверной ЭЦП или же в случае ее отсутствия разрешается, но при этом выдается сообщение об ошибке проверки ЭЦП (режим для проверки ЭЦП, внедренной в расширенные атрибуты файловой системы).

В аттестованных информационных системах под управлением ОС Astra Linux Special Edition запуск стороннего ПО и СПО без ЭЦП АО «НПО РусБИТех» запрещен (невозможен после включения ЗПС с возникновением ошибки сегментации). Для осуществления возможности запуска стороннего СПО, например, собственной разработки, необходимо получение открытого ключа для подписи у открытые ключи АО «НПО РусБИТех» (рис. 1). Это является подтверждением того, что именно вы, а не злоумышленник, осознанно устанавливаете СПО в аттестованную информационную систему. Например, ситуация, когда организация столкнулась с необходимостью установки в аттестованную информационную систему драйвера, которого нет в составе ОС. Как правило, на организацию заказывается по одному открытому ключу.

В обычным ОС на ядре Linux у всех пакетов есть сертификаты разработчиков и при их установке происходит сверка со списков доверенных

разработчиков. В ОС Astra Linux Special Edition такой режим тоже возможен до включения ЗПС (с потерей аттестации).

НА ФИРМЕННОМ БЛАНКЕ ОРГАНИЗАЦИИ		Генеральному директору АО «НПО РусБИТех» В.И. Пустовому	
		117105, г. Москва, Варшавское ш., д. 26 Тел.: +7 495 648-05-40 Факс: +7 495 648-06-39	
Уважаемый Виктор Иванович!			
Просим Вас сгенерировать комплект ключей для подписания программного обеспечения _____ (название Вашего ПО) на официально приобретенной операционной системе специального назначения «Astra Linux Special Edition» заводской номер № _____.			
Способ получения ключей — передача представителю организации-заявителя.			
Исполнитель:			
ФИО			
Должность			
Контактный телефон			
Мобильный телефон			
E-mail			
Должность подписи	_____ /		Расшифровка
Печать			(подпись)

Рис. 1. Образец заявки на получение ЭЦП

После получения открытого ключа он копируется в каталог /etc/digisig/keys.

Настройка модуля ядра digsig_verif

Настройка режима функционирования модуля digsig_verif осуществляется посредством графической утилиты fly-admin-smc («Панель управления — Безопасность — Политика безопасности — Замкнутая программная среда») или путем редактирования конфигурационного файла /etc/digisig/digisig_initramfs.conf.

Редактирование конфигурационного файла /etc/digisig/digisig_initramfs.conf осуществляется следующим образом:

1) для использования отладочного режима для тестирования СПО необходимо установить для параметра DIGSIG_ELF_MODE значение 0:

DIGSIG_ELF_MODE=0

2) для использования режима для проверки ЭЦП в СПО необходимо установить для параметра DIGSIG_ELF_MODE значение 2:

DIGSIG_ELF_MODE=2

3) для использования штатного режима функционирования необходимо установить для параметра DIGSIG_ELF_MODE значение 1:

DIGSIG_ELF_MODE=1

Рассмотрим механизм ЗПС на конкретных примерах. Войдите в систему под учетной записью с администраторскими полномочиями и высоким уровнем КЦ.

Информация о включенной ЗПС всегда должна быть в формуляре на АРМ или в инструкции администратора.

Запустите графическую утилиту fly-admin-smc («Панель управления – Безопасность – Политика безопасности – Замкнутая программная среда») (рис. 2).

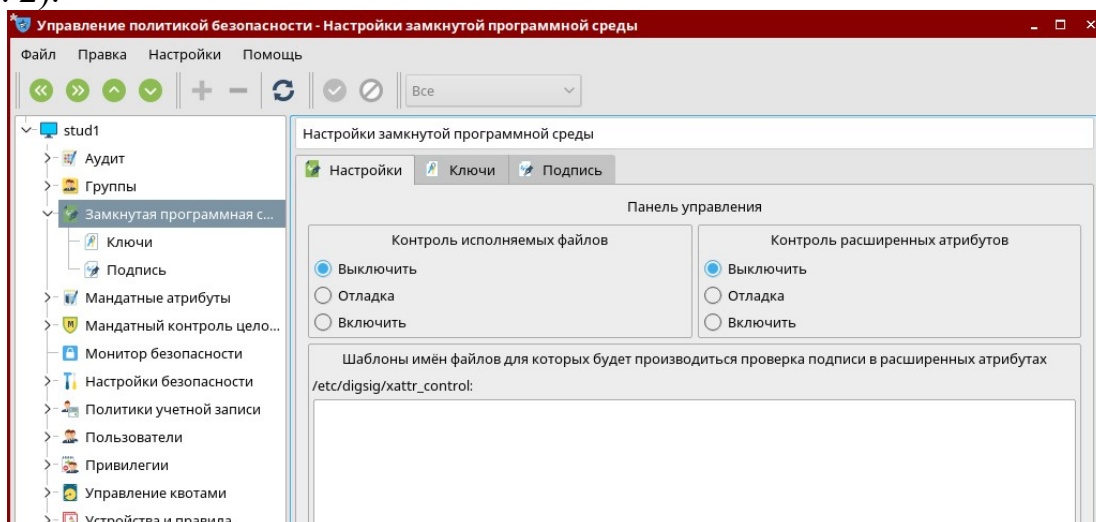


Рис. 2. Меню настройки ЗПС

На рис. 2. видно, что в данный момент времени контроль исполняемых файлов и контроль расширенных атрибутов выключен.

Рассмотрим пример с установкой программ без ЭЦП при отключенной и включенной ЗПС. Запустим браузер. В адресной строке браузера введем адрес <https://busybox.net/downloads/binaries/>. (рис. 3)

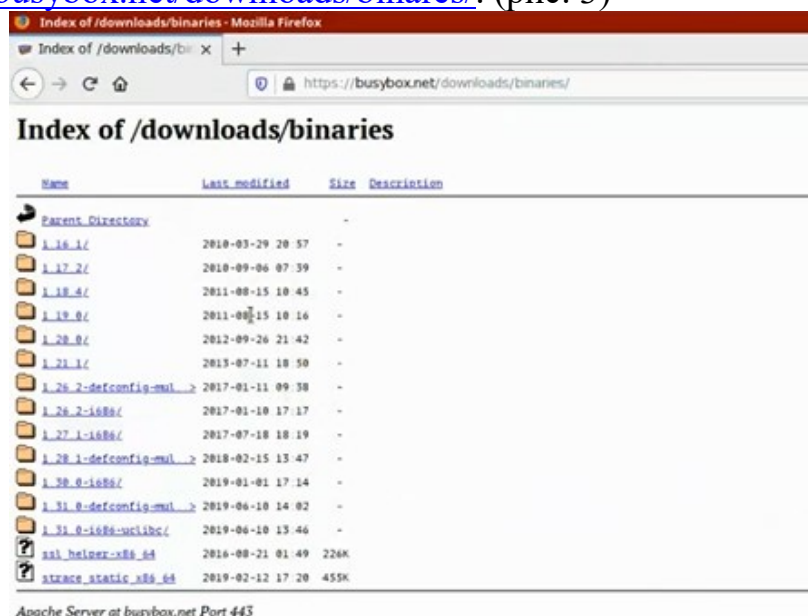


Рис. 3. Содержимое удаленного каталога <https://busybox.net/downloads/binaries/>

Скачиваем файл, например, из каталога 1.16.1 с именем `busybox-x86_64`. Сохраним его в корень домашнего каталога. Просмотрим содержимое домашнего каталога, например, в командном интерпретаторе. Обратите внимание, что скаченный файл не исполняемый и необходимо поменять его атрибуты введя команду **`chmod +x busybox-x86_64`**.

Запускаем программу на исполнение командой **`./busybox-x86_64`**. Убеждаемся, что программа запустилась успешно, потому что у нас отключен механизм ЗПС (рис. 4).

```
administrator@astra:~$ chmod +x busybox-x86_64
administrator@astra:~$
administrator@astra:~$
administrator@astra:~$ ls -l busybox-x86_64
-rwxr-xr-x 1 administrator administrator 973160 дек  8 04:26 busybox-x86_64
administrator@astra:~$
administrator@astra:~$ ./busybox-x86_64
BusyBox v1.16.1 (2010-03-29 11:54:55 CDT) multi-call binary.
Copyright (C) 1998-2009 Erik Andersen, Rob Landley, Denys Vlasenko
and others. Licensed under GPLv2.
See source distribution for full notice.

Usage: busybox [function] [arguments]...
or: function [arguments]...

BusyBox is a multi-call binary that combines many common Unix
utilities into a single executable. Most people will create a
link to busybox for each function they wish to use and BusyBox
will act like whatever it was invoked as.

Currently defined functions:
[, [[, acpid, addgroup, adduser, adjtimex, arp, arping, ash, awk,
```

Рис. 4. Успешный запуск программы `busybox-x86_64`

Теперь активируем контроль исполняемых файлов (рис. 5).

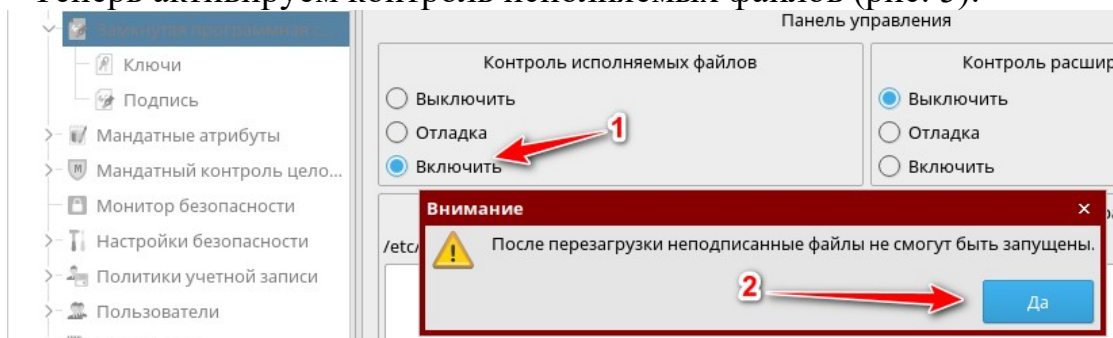


Рис. 5. Активация контроля исполняемых файлов

Подтверждаем применение выбранных параметров (рис. 6).

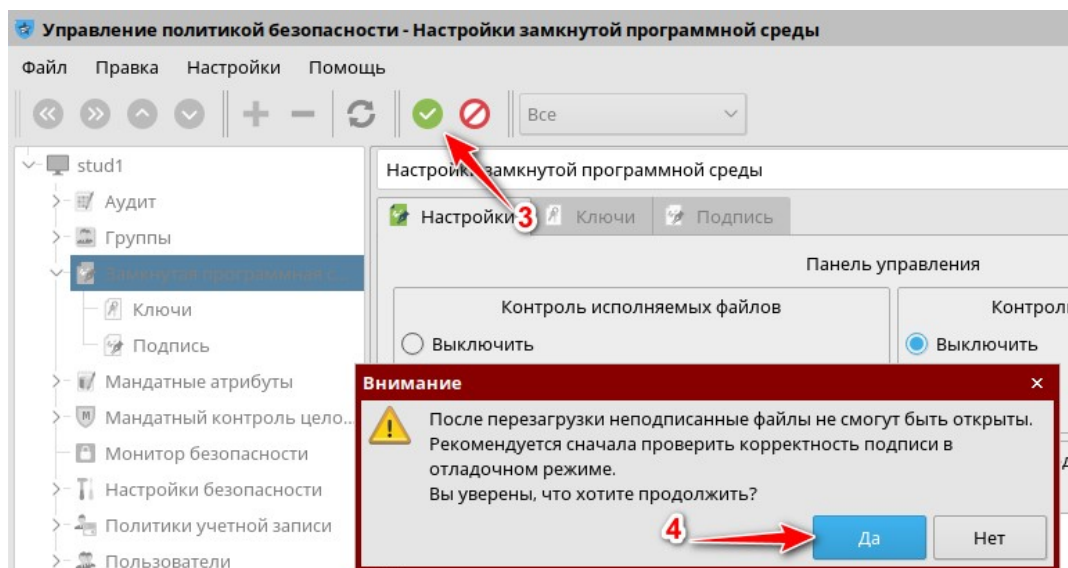


Рис. 6. Завершение активации контроля исполняемых файлов

Далее происходит обновление настроек ЗПС (рис. 7). Процедура обновления настроек занимает длительное время и может занимать даже до 5 минут. Перезагружаем систему.

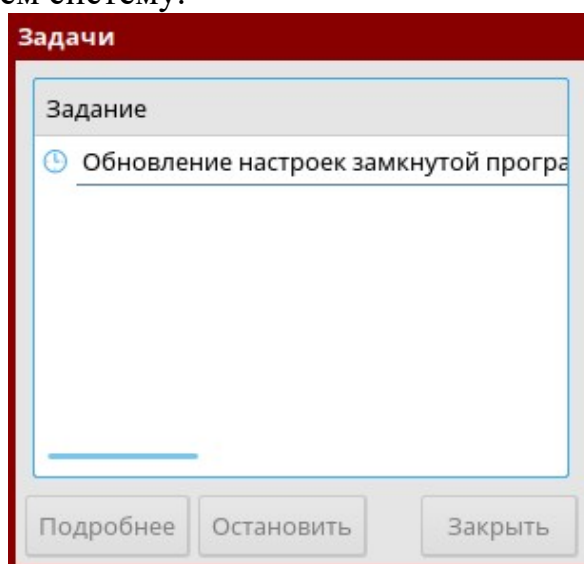


Рис. 7. Обновление настроек ЗПС

Попробуем запустить программу `busybox-x86_64`, предварительно перейдя в каталог с программой. Как видим, запуск программы завершился ошибкой сегментирования.

Теперь установим сертифицированную и подписанную ЭЦП программу `busybox`. Сертифицированная программа `busybox` входит в состав стандартной поставки ОС Astra Linux Special Edition и находится на установочном диске ОС `AstraLinux_Smolensk-1.6-20.06.2018.iso`.

Подключите оптический диск с ОС через интерфейс виртуальной машины, не забыв подмонтировать диск в системе (например, командой **`mount /dev/cdrom /media/cdrom`** или во всплывающем окне возле трея ОС) (рис. 7 и 8).

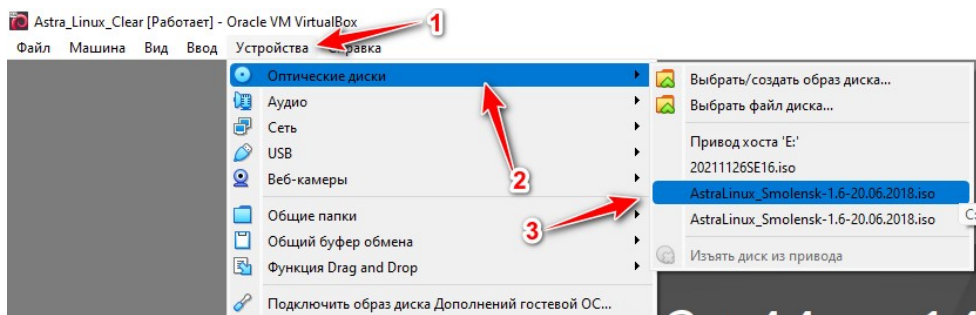


Рис. 8. Подключение оптического диска с ОС

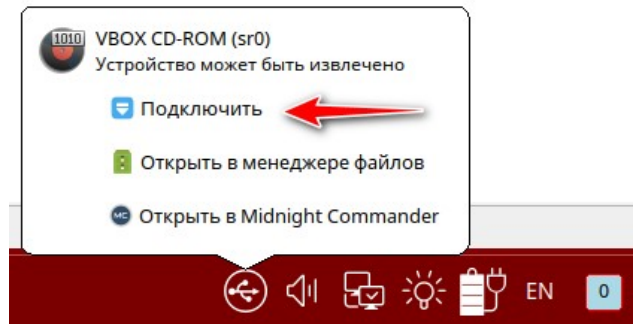


Рис. 9. Монтирование оптического диска в ОС

Установите программу введя команду: **sudo aptitude install busybox** (рис. 10).

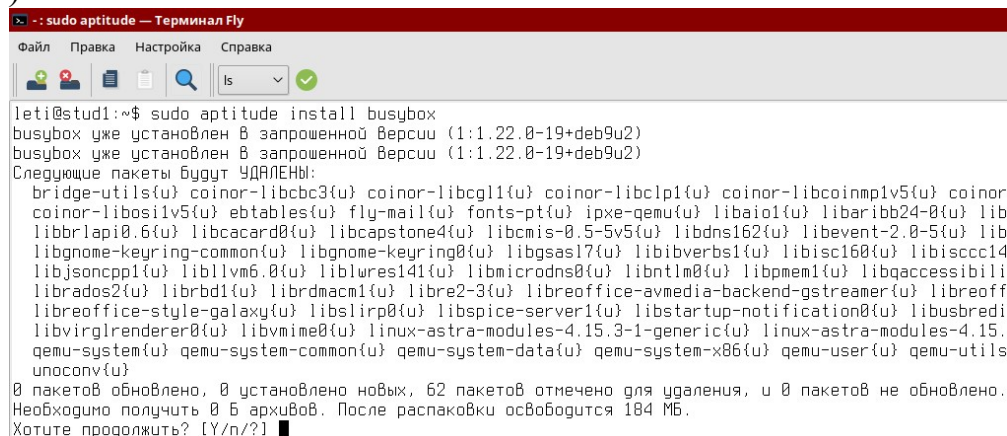
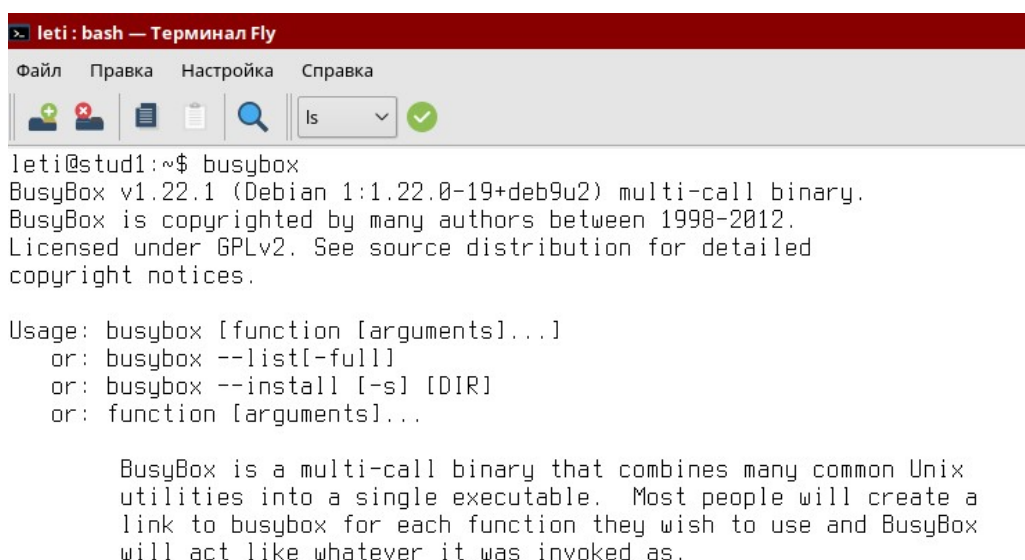


Рис. 10. Установка программы busybox

Программа успешно установилась. Запустите программу на выполнение, введя команду **busybox**. Программа успешно запустилась (рис. 11). Ответьте на вопросы, почему программа запустилась просто по команде, без указания пути к исполняемому файлу? Какой командой можно найти, в каком каталоге расположен исполняемый файл **busybox**? Почему запустилась программа **busybox** (в чем отличие от скаченной **busybox-x86_64**)?



```
leti@stud1:~$ busybox
BusyBox v1.22.1 (Debian 1:1.22.0-19+deb9u2) multi-call binary.
BusyBox is copyrighted by many authors between 1998-2012.
Licensed under GPLv2. See source distribution for detailed
copyright notices.

Usage: busybox [function [arguments]...]
or: busybox --list[-full]
or: busybox --install [-s] [DIR]
or: function [arguments]...

BusyBox is a multi-call binary that combines many common Unix
utilities into a single executable. Most people will create a
link to busybox for each function they wish to use and BusyBox
will act like whatever it was invoked as.
```

Рис. 11. Успешный запуск программы busybox с активированной ЗПС

Можно просмотреть файл журнала ОС на предмет работы ЗПС. Введите команду **cat /var/log/messages|grep DIGSIG**. Видим регистрацию события, связанного с попыткой запуска несертифицированного ПО (флаг NOT SIGNED) с указанием пути, по которому была попытка запуска программы.



```
Dec 8 15:34:09 astra org.ally.atspi.Registry[1295]: SpiRegistry daemon is running with well
-known name - org.ally.atspi.Registry
Dec 8 15:34:38 astra kernel: [ 117.022539] DIGSIG:[ERROR] NOT SIGNED: path=/home/administ
rator/busybox-x86_64 uid=1000 gid=1000
Dec 8 15:37:44 astra kernel: [ 221.898448] Parsec integrity level: 63
Dec 8 15:37:44 astra kernel: [ 303.439108] DIGSIG:[ERROR] NOT SIGNED: path=/home/administ
rator/busybox-x86_64 uid=1000 gid=1000
administratorgastra:~$
administratorgastra:~$
administratorgastra:~$ cat /var/log/messages
```

Рис. 12. Фрагмент файла журнала ОС /var/log/messages

Запрет исполнения неподписанных исполняемых файлов распространяется только на исполняемые бинарные файлы и никак не влияет на исполняемые скрипты.

В графической утилите fly-admin-smc («Панель управления – Безопасность – Политика безопасности – Замкнутая программная среда») есть вкладка для управления открытыми ключами (рис. 13). Ключи можно создавать самостоятельно (рис. 13–4) или копировать ключи для подписывания ПО полученные, например от АО «НПО РусБИТех» (рис. 13–2).

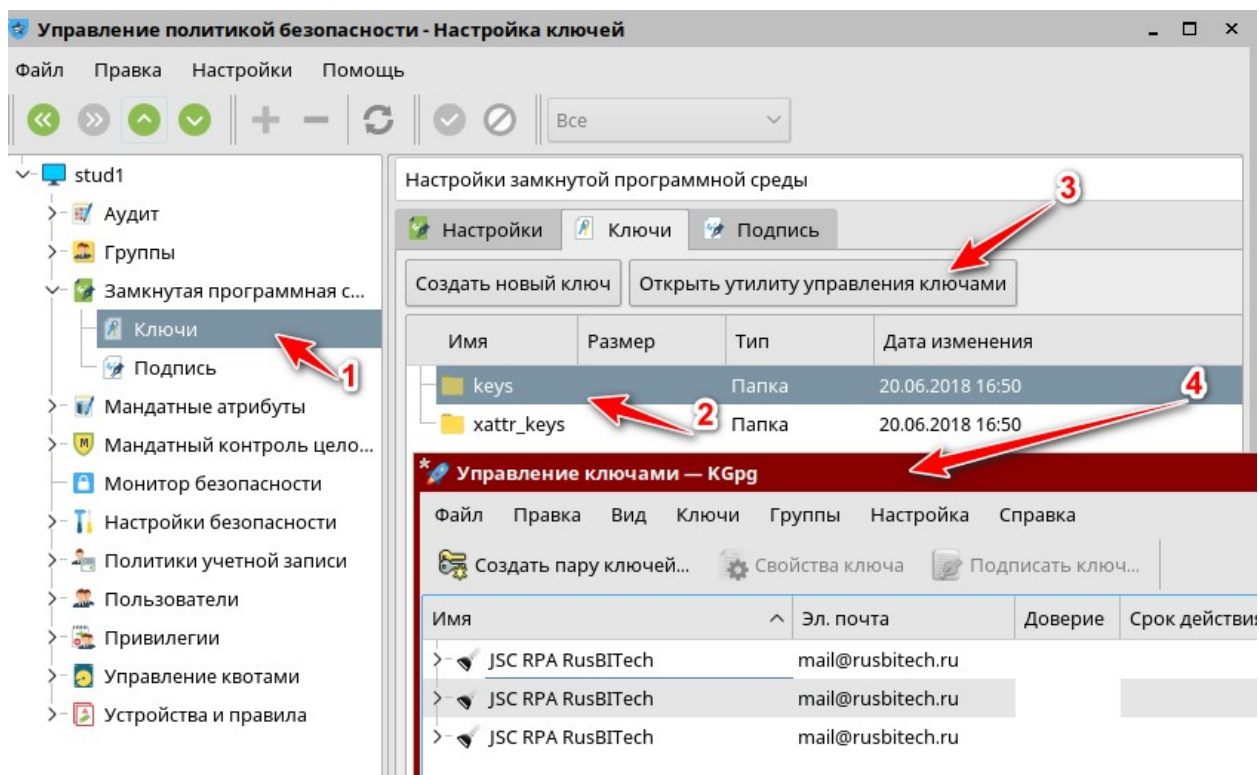


Рис. 13. Вкладка для управления открытыми ключами

Управление открытыми ключами возможно и через командную строку.

Введите команду **su -**, затем пароль от учетной записи root (если пароль не установлен, установите его командой **sudo passwd root**, указав в качестве пароля 11111111), тем самым повысив свои привилегии в системе.

Перейдите в каталог `/etc/digsg/` и изучите его содержимое.

В модуль `digsg_verify` встроены открытые ключи АО «НПО РусБИТех», которые используются:

- для проверки подписи исполняемых файлов;
- для проверки подписи открываемых файлов в расширенных атрибутах;
- для проверки подписи на дополнительных ключах, загружаемых из `/etc/digsg/keys`.

В каждый момент времени модуль использует два набора ключей:

основной набор (изначально три встроенных ключа АО «НПО РусБИТех») – для проверки любых подписей;

дополнительный набор (изначально пустой) – только для проверки подписи открываемых файлов в расширенных атрибутах.

Если дополнительный набор не содержит ключа, который использовался для подписи файла в расширенных атрибутах, модуль ищет подходящий ключ в основном наборе.

Каталог `/etc/digsg/keys` содержит открытые ключи в формате `gnupg --export`, которыми расширяется основной набор ключей при загрузке системы. Каждый ключ, использованный для подписывания СПО, необходимо скопировать в каталог `/etc/digsg/keys/`, например, с использованием команды:

cp /<каталог>/<файл ключа> /etc/digsg/keys/

В каталоге `/etc/digsig/keys/` может располагаться иерархическая структура ключей. Каждый ключ должен быть подписан уже загруженным ключом основного набора в момент его добавления в основной набор. Иерархия подкаталогов `/etc/digsig/keys` обрабатывается сверху вниз таким образом, что:

файлы ключей в `/etc/digsig/keys` должны быть подписаны встроенными в модуль `digsig_verify` ключами АО «НПО РусБИТех»;

файлы ключей в `/etc/digsig/keys/subdirectory` могут быть подписаны и встроенными ключами АО «НПО РусБИТех», и ключами из `/etc/digsig/keys`;

в целом, каждый ключ из подкаталога внутри `/etc/digsig/keys` должен быть подписан либо ключом из вышележащего каталога, либо встроенным ключом АО «НПО РусБИТех».

Подписывание файлов формата ELF в ОС

В модуле ядра `digsig_verif` реализован механизм, позволяющий использовать несколько ключей при подписывании файлов формата ELF.

Порядок использования ключей для `digsig_verif`: дополнительные ключи записываются в `/sys/digsig/keys` или `/sys/digsig/xattr_keys` в иерархической последовательности цепочек подписей. Все дополнительные ключи должны быть подписаны главным ключом или другим дополнительным ключом, подпись которого может быть проверена (за исключением первого ключа в каталоге `/sys/digsig/xattr_keys`).

Для создания дополнительных ключей используется утилита GNU Privacy Guard. Данная утилита присутствует, как правило, во всех ОС на ядре Linux. Модифицированный GnuPG (`gpg` в отечественных ОС СН использует, в том числе, алгоритмы ГОСТ) выводит ГОСТ Р 34.11-94 в списке доступных алгоритмов. Для получения списка доступных алгоритмов необходимо выполнить команду: **`sudo gpg --version`**. Введите команду для просмотра поддерживаемых алгоритмов шифрования (рис. 14). Найдите среди

```
leti@stud1:~$ sudo gpg --version
gpg (GnuPG) 2.1.18
libgcrypt 1.7.6-beta
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /root/.gnupg
Поддерживаются следующие алгоритмы:
С открытым ключом: RSA, ELG, DSA, ECDH, ECDSA, EDDSA,
                   GOST_R34.10-2001, GOST_R34.10-2012
Симметричные шифры: IDEA, 3DES, CAST5, BLOWFISH,
                   AES, AES192, AES256, TWOFISH, CAMELLIA128,
                   CAMELLIA192, CAMELLIA256, GRASSHOPPER
Хеш-функции: SHA1, RIPEMD160, SHA256, SHA384, SHA512,
             SHA224, GOST_R34.11-2012, GOST_R34.11-94
Алгоритмы сжатия: Без сжатия, ZIP, ZLIB,
                 BZIP2
```

Рис. 14. Поддерживаемые утилитой GNU Privacy Guard алгоритмы шифрования

По умолчанию в ОССН используется алгоритмы GOST_R34, 10-2001 и GOST_R34, 10-2012. Когда мы хотим создать свой ключ для подписи то используем ключевую пару, обязательно включающую GOST_R34, 10-2001. Созданный ключ будет располагаться в домашнем каталоге пользователя, который создает этот ключ.

Создадим свой ключ (рис. 15). Запустим терминал в непривилегированном режиме (\$) и введем команду: **gpg --gen-key**. Утилита попросит ввести нас полное имя (генерируется ID пользователя для идентификации ключа), адрес электронной почты. Проверим введенные данные и примем их, нажав клавишу О. Утилита во всплывающем окне предложит придумать нам фразу-пароль для защиты создаваемого ключа (если вы работаете без графического режима, то вам будет предложен консольный вариант ввода фразы-пароля). Этот пароль мы будем использовать при подписи программы созданным ключом и этот ключ будет защищен от использования злоумышленниками.

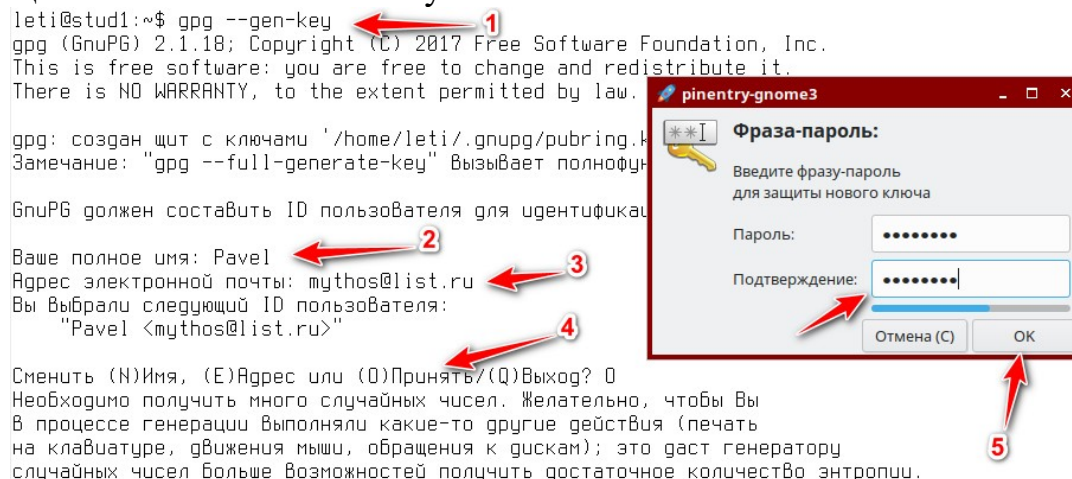


Рис. 15. Создание ключа для подписи утилитой GNU Privacy Guard

После принятия исходных данных происходит генерация ключа. Для ускорения генерации выполняем рекомендации утилиты. Теперь мы можем работать с созданными ключами (рис. 16).

```
gpg: /home/leti/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ B244B003E5B1CC9B помечен как абсолютно доверенный
gpg: создан каталог '/home/leti/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/leti/.gnupg/openpgp-revocs.d/3231ABC4
открытый и секретный ключи созданы и подписаны.

pub  rsa2048 2022-02-17 [SC] [   ] gogen go: 2024-02-17]
      3231ABC430B7E0E3B06C87C8B244B003E5B1CC9B
      3231ABC430B7E0E3B06C87C8B244B003E5B1CC9B
uid                               Pavel <mythos@list.ru>
sub   rsa2048 2022-02-17 [E] [   ] gogen go: 2024-02-17]
```

Рис. 16. Сгенерированные ключи для подписи

Вновь созданные ключи расположены в домашнем каталоге пользователя. Перейдите в данный каталог, например командой **cd ~** (хотя скорее всего вы уже в своем домашнем каталоге). В нем найдите подкаталог **.gnupg** и просмотрите его содержимое командой: **ls -al .gnupg/** (рис. 17).

```

leti@stud1:~$ ls -la .gnupg
утого 28
drwx----- 4 leti leti 4096 фев 17 12:34 .
drwxr-x--- 18 leti leti 4096 фев 17 12:14 ..
drwx----- 2 leti leti 4096 фев 17 12:34 openpgp-revocs.d
drwx----- 2 leti leti 4096 фев 17 12:34 private-keys-v1.d
-rw-r--r-- 1 leti leti 1408 фев 17 12:34 pubring.kbx
-rw----- 1 leti leti 32 фев 17 12:23 pubring.kbx~
-rw----- 1 leti leti 1240 фев 17 12:34 trustdb.gpg

```




Рис. 17. Содержимое каталога .gnupg

В каталоге `private-keys-v1.d` и расположены наши сгенерированные ключи. Просмотрите содержимое указанного каталога командой: **`ls -al .gnupg/private-keys-v1.d/`** (рис. 18). При необходимости использования сгенерированных ключей на других АРМах мы можем их экспортировать (простое копирование не используется из-за ограничений по подписи ключей вышестоящими ключами и особенностями файла `gpg`) введя команду:

```

gpg --export "Pavel <mythos@list.ru>" > /tmp/copiya_klucha.gpg
leti@stud1:~$ ls -la .gnupg/private-keys-v1.d/
утого 16
drwx----- 2 leti leti 4096 фев 17 12:34 .
drwx----- 4 leti leti 4096 фев 17 12:34 ..
-rw----- 1 leti leti 1174 фев 17 12:34 633886D9906E7554D98BC09B5969377E56F10450.key
-rw----- 1 leti leti 1158 фев 17 12:34 E7352F8C82514FADF856EE5C6BD18AA374D0FD4E.key
leti@stud1:~$ gpg --export "Pavel <mythos@list.ru>" > /tmp/copiya_klucha.gpg

```

Рис. 18. Содержимое каталога private-keys-v1.d

Далее копируем созданный файл на необходимый АРМ и импортируем ключ в систему введя команду: **`gpg --import /tmp/copiya_klucha.gpg`**

ВАЖНО! Местоположение ключа может быть другим, а не `/tmp/copiya_klucha.gpg`

Поскольку действия по импортированию выполнялись на одном АРМе, то ключ импортирован не был, так как он уже присутствует в системе (рис. 19). Самостоятельно импортируйте ключ на новый АРМ.

```

leti@stud1:~$ gpg --import /tmp/copiya_klucha.gpg
gpg: ключ B244B003E5B1CC9B: "Pavel <mythos@list.ru>" не изменен
gpg: Всего обработано: 1
gpg:                               неизмененных: 1

```

Рис. 19. Импорт ключа

Далее после импорта созданного ключа в новую систему его обязательно нужно подписать вышестоящим ключом этой системы. Введите команду:

`gpg --sign-key "Pavel <mythos@list.ru >"`

На рис. 19 процесс подписания не требуется из-за ограничений, оговоренных выше. Если бы это был новый АРМ, то система сообщила бы об успешной подписи ключа вышестоящим ключом (ключом Росбиттех).


```

leti@stud1:~$ gpg --sign-key "Pavel <mythos@list.ru>"

gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: глубина: 0 достоверных: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
gpg: срок следующей проверки таблицы доверия 2024-02-17
sec rsa2048/B244B003E5B1CC9B
    создан: 2022-02-17    гоген до: 2024-02-17    назначение: SC
    доверие: абсолютное достоверность: абсолютное
ssb rsa2048/1B871731CA8AC754
    создан: 2022-02-17    гоген до: 2024-02-17    назначение: E
[ абсолютно ] (1). Pavel <mythos@list.ru>

"Pavel <mythos@list.ru>" уже подписан ключом B244B003E5B1CC9B
Нечего подписывать ключом B244B003E5B1CC9B

Ключ не изменялся - обновление не нужно.

```

Рис. 19. Подписывание ключа вышестоящим ключом

Теперь можно подписывать скрипты и программы. **(сторонние программы, исполняемые в ОС Astra Linux с включенной ЗПС подписываются только используя ключ, полученный от АО «НПО РусБИТех».** Самостоятельно создайте файл, например touch script, сделайте его исполняемым, например командой chmod 755. Просмотрите свойства созданного файла. Как видно на рис. 20 на данный момент файл не подписан никаким ключом.

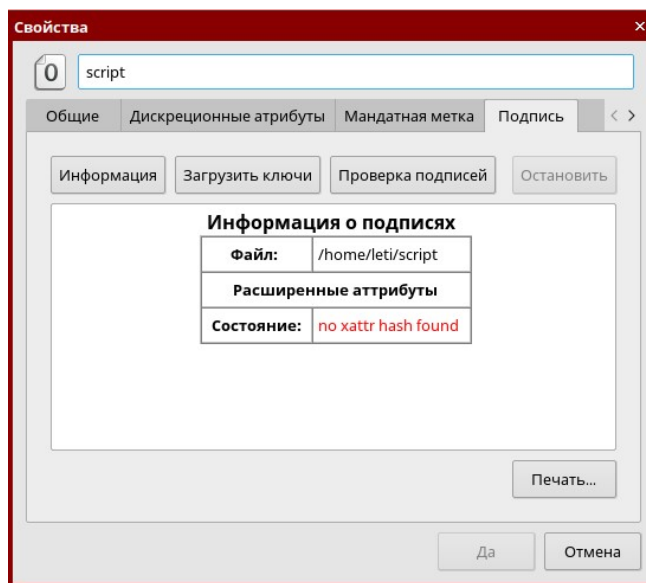


Рис. 20. Свойства файла script

Подпишем файл, используя консоль (можно сделать и при помощи графических утилит). Просмотрим расширенные свойства файла введя команду: **bsign -w script** (рис. 21).

```

leti@stud1:~$ touch script
leti@stud1:~$ chmod 755 script
leti@stud1:~$ bsign -w script
bsign: no xattr hash found in 'script'

```

Рис. 21. Свойства файла script

Подпишем файл закрытым ключом введя команду: **bsign --sign script** (рис. 22). Введем фразу пароль. Все, файл подписан.

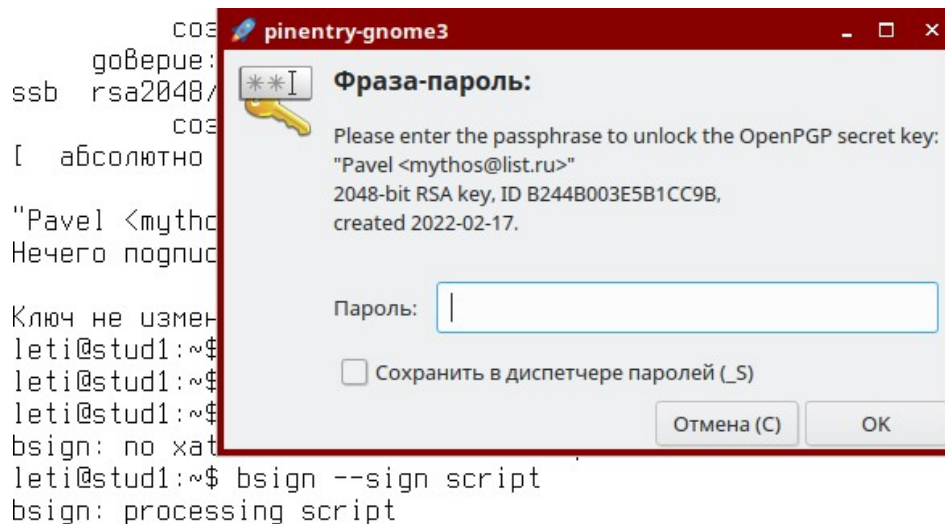


Рис. 22. Процесс подписи файла script

Убедимся, что файл подписан, просмотрев его свойства (рис. 23).

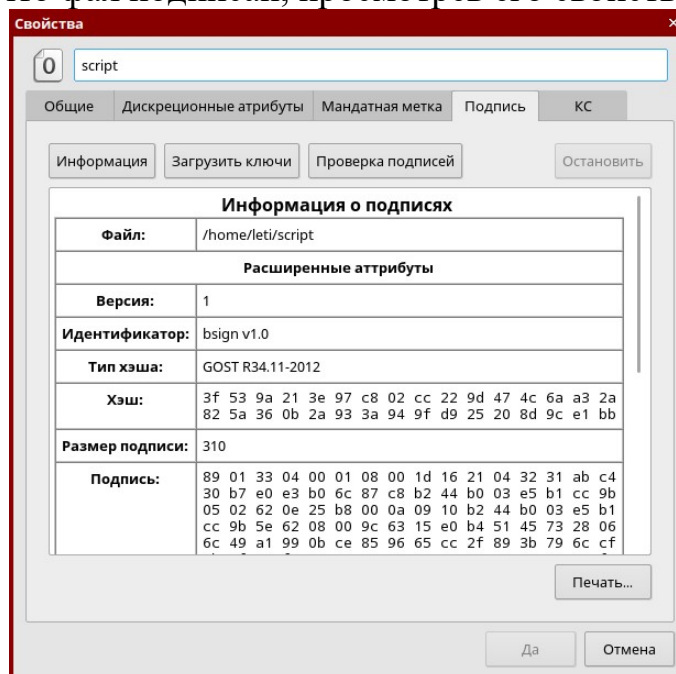


Рис. 23. Свойства файла script

Также, можно проверить подпись файла при помощи командной строки (рис. 24). Введем команду: **bsign -w script**


```

version: 1
id: bsign v1.0
xattr hash: {GOST R34.11-2012} 3f539a213e97c802cc229
xattr signature_size: 310
xattr signature:
  89 01 33 04 00 01 08 00 1d 16 21 04 32 31 ab c4
  30 b7 e0 e3 b0 6c 87 c8 b2 44 b0 03 e5 b1 cc 9b
  05 02 62 0e 25 b8 00 0a 09 10 b2 44 b0 03 e5 b1
  cc 9b 5e 62 08 00 9c 63 15 e0 b4 51 45 73 28 06
  6c 49 a1 99 0b ce 85 96 65 cc 2f 89 3b 79 6c cf
  eb 2f 34 f4 c9 c4 16 22 37 49 0c 5a 1c c2 44 fc
  19 14 2d ee 83 85 ba 96 94 a6 89 cf fa 36 74 a0
  df fd 12 26 f8 15 54 d8 9f 63 d8 e2 fb d4 71 63
  46 3b 39 31 73 55 70 91 7b f0 de c2 8a e3 de b3
  8c c2 19 fa af f8 a5 3d b9 3f cd 22 7c 78 a0 ee
  26 57 42 be b8 bd b9 1f 62 c6 28 82 a1 0c 55 d4
  76 ba 14 97 a1 ab 9d 65 86 bc f2 73 6f 25 37 17
  1b 22 06 85 42 f1 44 af 56 ef d9 61 90 ac e4 07
  c7 fa 9c 00 27 4d 18 79 28 c4 f8 16 27 a1 19 4e
  0f 2c 9e 54 1a 71 5d 74 d7 9e aa 41 44 7b b2 94
  9c a1 56 ae 27 9c 6c 62 4b 39 b0 38 a2 48 26 d0
  10 71 fa f5 77 28 20 16 7d e7 c5 3e f9 91 a4 ae
  a2 12 db ba 8e 03 d2 df 16 a6 90 e1 0e 80 c2 f3
  e3 b0 46 71 6c c2 4c e3 37 85 db 81 f8 a1 17 19
  e8 73 16 63 27 57
signer: B244B003E5B1CC9B
timestamp: 17 Feb 2022 13:38:48 (1645094328)
bsign: good xattr hash found in 'script'

```

Рис. 24. Подпись файла script

Контроль целостности в ОС

Рассмотрим, как проверяется подписанные файлы в системе, т.е. каким образом осуществляется контроль целостности КСЗ (обратите внимание, это не мандатный контроль целостности, а средство подсчета контрольных сумм файлов, пакетов и оптических дисков).

Для обеспечения контроля целостности (в т. ч. контроля целостности КСЗ) в ОС реализованы:

- средство подсчета контрольных сумм файлов и оптических дисков;
- средство подсчета контрольных сумм файлов в deb-пакетах;
- средство контроля соответствия дистрибутиву;
- средства регламентного контроля целостности;
- средства создания замкнутой программной среды.

Для решения задач контроля целостности предназначена библиотека libgost, в которой для вычисления контрольных сумм реализованы функции хэширования в соответствии с ГОСТ Р 34.11-94, ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит и ГОСТ Р 34.11-2012 с длиной хэш-кода 512 бит. Названная библиотека используется в средствах подсчета контрольных сумм файлов и оптических дисков, контроля соответствия дистрибутиву и регламентного контроля целостности, модулях аутентификации.

В ОС реализован механизм, обеспечивающий проверку неизменности и подлинности загружаемых исполняемых файлов формата ELF. Проверка производится на основе контрольных сумм, вычисляемых в соответствии с ГОСТ Р 34.11-94 и ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит, и ЭЦП, реализованной в соответствии с ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012, которые внедрены в исполняемые файлы формата ELF в процессе сборки ОС. Данный механизм предназначен для выявления фактов

несанкционированного изменения исполняемых файлов формата ELF (в т. ч. относящихся к КСЗ) и предотвращения их загрузки.

В ОС реализован механизм, обеспечивающий проверку неизменности и подлинности файлов. Проверка производится на основе контрольных сумм, вычисляемых в соответствии с ГОСТ Р 34.11-94 и ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит, и ЭЦП, реализованной в соответствии с ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012, которые внедряются в расширенные атрибуты файловой системы. Данный механизм предназначен для выявления фактов несанкционированного изменения исполняемых файлов и предотвращения их открытия.

Также при проверке целостности происходит проверка ЭЦП, которую мы изучили и отработали ранее.

Проверим контрольные суммы оптических дисков. Перейдем в режим суперпользователя введя команду: **sudo su -**. Для подсчета контрольных сумм файлов и оптических дисков в состав ОС включена утилита командной строки **gostsum**. Для просмотра справки по утилите **gostsum** с целью ознакомления с возможными вариантами подсчета контрольных сумм введите команду: **gostsum -h**

Подмонтируйте оптический диск с дистрибутивом ОС AstraLinux_Smolensk-1.6-20.06.2018.iso. Произведите подсчет контрольной суммы оптического диска, введя команду **gostsum -d /dev/cdrom** (рис. 25)

```
root@stud1:/home/leti# gostsum -d /dev/cdrom
7cc13b25295767082190d42c8e9f5191a4f880e68a4532156445aaaaa12c3dbb /dev/cdrom
root@stud1:/home/leti# █
```

Рис. 25. Контрольная сумма оптического диска

Средство подсчета контрольных сумм файлов в deb-пакетах

Для подсчета контрольных сумм файлов в deb-пакетах в состав ОС включена утилита командной строки **gostsum_from_deb** (с ключом **-p**). Самостоятельно произведите подсчет контрольной суммы какого-нибудь пакета из состава дистрибутива ОС. Подсчет контрольной суммы производится для каждого файла из состава пакета. Самостоятельно произведите подсчет контрольной суммы какого-нибудь стороннего скачанного пакета из состава дистрибутива ОС. Что произошло? Объясните почему.

```

ntax/swig.syntax
ab29870e1af2efe584c0ab8323a6d1e76f3a88e8ec91e29fa8530d7c590f39d0 /usr/share/mc/sy
ntax/syntax.syntax
0c558de841c735c19145fc49443cd701804d8ddd306f443e8bbc88588b6cf9e9 /usr/share/mc/sy
ntax/tcl.syntax
55c3fdc013d98a01ee2d200e70bc9b839ed80ca9371d7f1a8b429e2a559a9347 /usr/share/mc/sy
ntax/texinfo.syntax
8f0757f27adec129d44480ab8898d79111f4fd75c984a7c466b96e087ea8c25a /usr/share/mc/sy
ntax/ts.syntax
919cf6b0bbf6e44a7892420fd4eabd5564f3926592550ea1909d177f803bfd4e /usr/share/mc/sy
ntax/tt.syntax
6c459962fd60981a605c4bf77cfe16e261b7eabedbb84beba43833417c4dd306 /usr/share/mc/sy
ntax/unknown.syntax
173a705dd213786c8fdd10f616e41c9d54ab52ad15984a08e140244a1b14975b /usr/share/mc/sy
ntax/verilog.syntax
b608868a9ad951f513e7bd496953436b68fa30a843f3bd60b5c1e298cc7cf3e2 /usr/share/mc/sy
ntax/vhdl.syntax
ad470d38fdef0dd7ae2da864974f66ef19e71391ca12fe76ab9aa9872afaf560 /usr/share/mc/sy
ntax/xml.syntax
f1d7c788d0424a8525d53cfe1c130211e31f2340fd7591a23aec382896058e2f /usr/share/mc/sy
ntax/yaml.syntax
b756ad895cc6cb1e35476a8517c66f6fd1616ecfe4000beb096567688c97d9e9 /usr/share/mc/sy
ntax/yum-repo.syntax
beda8adf808dd9e0b4ef771238a0f75313bb0951b1a25c438e81ab15ef857e88 /usr/share/mc/sy
ntax/yxx.syntax
root@astra:~#
root@astra:~#
root@astra:~# gostsum_from_deb -p /media/cdrom/pool/main/m/mc/mc-data_4.8.21-1astr
al_all.deb █

```

Рис. 26. Контрольная сумма пакета

Средство контроля соответствия дистрибутиву

Средство контроля соответствия дистрибутиву предоставляет возможность для контроля соответствия объектов ФС ОС дистрибутиву. Для обеспечения контроля целостности объектов ФС ОС (в т. ч. СЗИ) в состав дистрибутива входит файл `gostsums.txt` со списком контрольных сумм по ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит для всех файлов, входящих в пакеты программ дистрибутива. Используя графическую утилиту `fly-admin-int-check`, можно провести вычисление контрольных сумм файлов системы и проверку соответствия полученных контрольных сумм файлов системы эталонным контрольным суммам. Более подробное описание утилиты см. в электронной справке. Просмотрите содержимое файла `gostsums.txt` введя команду: `cat /media/cdrom/gostsum.txt` При установке пакетов система сверяет данные контрольных сумм с данными в этом файле. При установке обновлений системы также происходит сопоставление контрольных сумм пакетов с данными файла, который находится в составе диска с обновлениями.

Самостоятельно подсчитайте количество строк в данном файле, используя советуемые команды (например, `wc`). Сколько строк в файле? Для какого количества файлов произведен подсчет контрольных сумм?

Средства регламентного контроля целостности

Организация регламентного контроля целостности ОС, прикладного ПО и СЗИ обеспечивается набором программных средств на основе «Another File Integrity Checker». В указанном наборе реализована возможность для проведения периодического (с использованием системного планировщика заданий `cron`) вычисления контрольных сумм файлов и соответствующих им

атрибутов расширенной подсистемы безопасности PARSEC (мандатных атрибутов и атрибутов расширенной подсистемы протоколирования) с последующим сравнением вычисленных значений с эталонными. В указанном наборе программных средств реализовано использование библиотеки libgost, обеспечивающей подсчет контрольных сумм в соответствии с ГОСТ Р 34.11-94.

Эталонные значения контрольных сумм и атрибутов фалов хранятся в БД. База контрольных сумм и атрибутов может быть создана при помощи команды: **afick -i**

Для вычисления контрольных сумм могут использоваться алгоритмы: MD5-Digest, SHA1 и ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит

Есть графическая утилита для регламентного контроля целостности fly-admin-int-check (рис. 27). Запустите утилиту из консоли или при помощи панели управления.

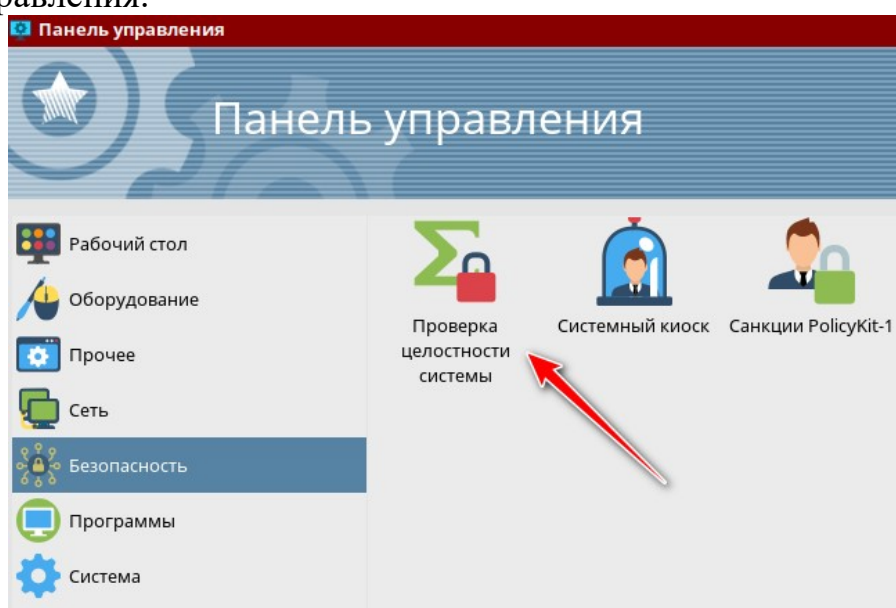


Рис. 27. Запуск утилиты для регламентного контроля целостности fly-admin-int-check

С помощью этой утилиты (рис. 28) можно произвести вычисление контрольных сумм файлов в системе и проверить полученные данные с эталонными значениями.

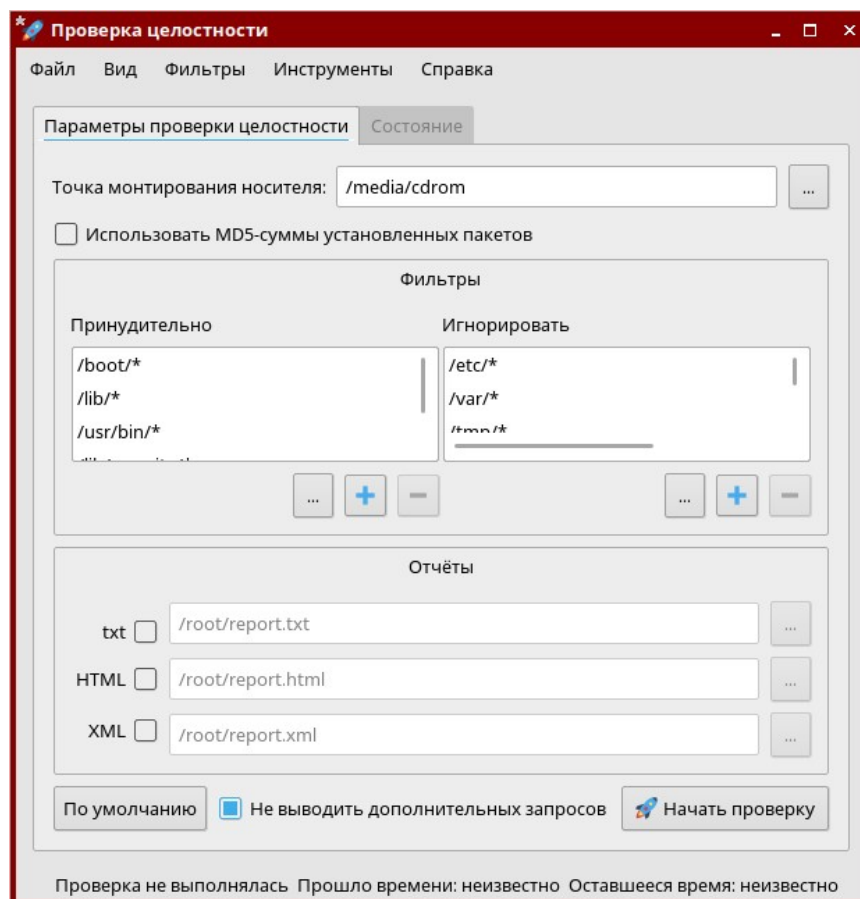


Рис. 28. Утилита для регламентного контроля целостности fly-admin-int-check

Так, например, на рис. 28 сравнение полученных контрольных сумм будет происходить с эталонными значениями источника, в качестве которого выступает cdrom (можно задать другой источник эталонных значений, например, локальный репозиторий).

Часть каталогов можно установить для принудительной проверки. Как видно на рис. 28 это системные каталоги ОС, которые не меняются в системе со временем и их изменение говорит о проникновении в систему. Можно добавить свои каталоги. Справой стороны, указываются каталоги, значения контрольных сумм в которых игнорируются. Перечислите эти каталоги и объясните почему их необходимо игнорировать. Можно задать местоположение отчетов о проверке, а также задать фильтры.

При проверке контрольных сумм необходимо учитывать, что в систему были установлены обновления и в качестве эталонного использовать файл с диска с обновлениями.

Критерии выставления оценок обучающимся:

Знания, умения и навыки обучающихся определяются оценками «отлично», «хорошо», «удовлетворительно», «неудовлетворительно». Общими критериями, определяющими оценку знаний и умений обучающихся на текущем практическом занятии, являются:

«отлично» – наличие глубоких и исчерпывающих знаний в объеме материала практического занятия, правильные, уверенные действия по применению полученных знаний на практике, грамотное, логичное изложение материала при ответе.

«хорошо» – наличие твердых и достаточно полных знаний в объеме материала практического занятия, незначительные ошибки при освещении вопросов, правильные действия по применению знаний на практике, четкое изложение материала при ответе.

«удовлетворительно» – наличие твердых знаний в объеме материала практического занятия, изложение ответов с ошибками, уверенно исправляемыми после дополнительных вопросов, необходимость в наводящих вопросах экзаменуемому, правильные действия по применению знаний на практике.

«неудовлетворительно» – наличие грубых ошибок в ответах, непонимание сущности излагаемых вопросов, неумении применять знания на практике, неуверенности и неточности в ответах на дополнительные и наводящие вопросы.

Порядок оценки выполнения задания

По выполнению работы каждый курсант должен представить отчет. Отчет должен содержать:

- название практического занятия;
- текст индивидуального задания;
- исходный текст программы;
- результаты тестирования решения.

В процессе выполнения индивидуального задания или после завершения его выполнения преподаватель проводит собеседование с каждым курсантом по теме выполненной работы, проверяя также практические навыки, приобретенные в ходе занятия. Отчетный материал предоставляется преподавателю, а результаты защищаются.

ПОДПОЛКОВНИК С. Краснов
(воинское звание, подпись, инициал имени, фамилия автора)
« ____ » _____ 202_ г.