МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(МОСКОВСКИЙ ПОЛИТЕХ)**

Факультет информационных технологий. Кафедра информационной безопасности

Инженерный проект

По дисциплине «Администрирование операционных систем Linux» на тему:

«Вирусы для Linux»

Работу выполнили:

Студенты гр.191-331 очного отделения

Малышева Анастасия Романовна

Юрин Даниил Романович

Научный руководитель:

Гневшев Александр Юрьевич.

г. Москва

2020 г.

**Содержание**

Вступление………………………………………………………………………...3

Глава 1. Безопасность ОС Linux ...........................................................................5

* 1. Уязвимости ОС Linux.....................................................................6
  2. Штатные средства защиты от заражений……………………...10

Глава 2. Вирусы для ОС Linux ............................................................................16

* 1. История развития и принцип работы вирусов…………………16
  2. Виды вирусов и практическая реализация вируса…………….20

Вывод……………………………………………………………………………..30

**Вступление**

В наши дни ОС Linux стремительно набирает популярность. Если на персональных компьютерах она не широко распространена, то активно используется в качестве системного ПО. Кроме того, нельзя не отметить повсеместную популяризацию IoT - устройств, операционной системой которых преимущественно является Linux.

Из-за быстрого роста популярности этой операционной системы, уже длительное время стали ходить споры, есть ли вирусы для Linux и если есть, то стоит ли использовать антивирусы на своем компьютере.

Согласно статистике сервиса StatCounter, 77% ПК работают под управлением Windows и менее 2% на Linux. Поэтому ошибочно будет предположить, что Windows является относительно безопасной системой. Это все связано с тем, что пользователи Windows, как правило, очень часто, создавая свою учетную запись, дают ей права администратора. Так как в таком случае намного проще работать в системе: при установке программ не нужно каждый раз вводить пароль, по этой же причине быстрее и легче запускать от имени администратора некоторые программы и т.п.

Но обратной стороной медали является то, что, попав на такую систему, вирус также легко, как и пользователь, получает доступ ко всем важным частям ОС. По сути, он может делать все, на что запрограммирован.

Совсем иначе обстоят дела у пользователей ОС Linux, так как большую часть времени они работают под обычной учетной записью, под которой нельзя как-либо изменять системные файлы, устанавливать или удалять ПО. Поэтому, даже если на компьютер и попадет какой-либо вирус, доступ у него будет разве что только к файлам в каталоге пользователя, а значит большого вреда системе в целом он не нанесет.

Исходя из выше сказанного, мы делаем небольшой вывод, что ОС Linux действительно безопаснее Windows от Microsoft и менее подвержена атакам. Но разработчики тоже люди и иногда могут ошибаться. Из-за недосмотра или еще по каким-либо причинам в системе появляются уязвимости, которые могут быть использованы вирусами.

**Глава 1.** Безопасность ОС Linux

**Предисловие:**

Относительная безопасность ОС Linux в первую очередь связана с особенностями ее архитектуры. Операционная система UNIX, на базе которой создан Linux специально разрабатывалась для многопользовательской работы. Здесь есть суперпользователь (root), имеющий доступ ко всем файлам и каталогам, а также отдельные пользователи, с доступом только к своим каталогам и разрешенным для них файлам.

Аккаунты пользователей в OC Linux используются, как права доступа, для разрешения запуска программ, файлов и документов только отдельным пользователям, что позволяет ограничить их полномочия, а значит уменьшить шансы заражения машины вирусом, так как вероятность того, что человек зайдя под своим аккаунтом в связи своей неопытности или неосторожности откроет скаченный файл из недостоверных источников, распакует его и тем самым заразит машину вирусом будет равна нулю, так как у него просто не будет прав на подобные действия. Но все-таки шанс заразить Linux вирусом всегда есть.

Очень часто ОС Linux упрекают в том, что в ней нет единства. Существует множество дистрибутивов и окружений рабочего стола, которые могут сильно различаться как по своей работе, так и во архитектурном строении.

Но в плане безопасности данный факт играет положительную роль. Разнообразие дистрибутивов, оболочек, систем управления пакетами, почтовых клиентов, препятствуют широкому распространению вирусов.

Помимо этого, безопасность Linux обусловлена наличием в ней репозиториев. Это тоже очень важный момент, потому что большинство программ в Linux устанавливаются не из сторонних сайтов интернета, а из репозиториев. Там они проверены и подписаны ключами для проверки подлинности. Никто не может их подменить или подсунуть вместо программ вирус, потому что они тщательно проверяются сопровождающими пакетов. В Windows же большинство программ устанавливаются из интернета, и никто не может гарантировать их качество, особенно если пользователи загружают программы не из официальных источников.

Несмотря на все выше перечисленные причины, при большом желании можно взломать все. Так как по своей сути вирус представляет собой обычную компьютерную программу, способную самостоятельно размножаться, используя уязвимости системы и, нередко, наивность пользователя. Linux не лишен уязвимостей, а с появлением новых интуитивно-понятных дистрибутивов, компьютерная грамотность среднестатического пользователя также несколько понизилась.

**Параграф 1.** Уязвимости ОС Linux

Как уже говорилось ранее, при особом желании взломать можно абсолютно любую систему и ОС Linux не исключение. Поэтому перед тем, как перейти непосредственно к теме проекта и рассказать про историю возникновения вирусов, принцип работы Linux вирусов и написанию непосредственно своей вредоносной программы, а также о том, как правильно защитить свою машину, хотелось бы рассказать, почему же Linux не является абсолютно безопасной системой и почему на нее можно установить вирус.

Первая причина – это уязвимость системы. Как мы знаем, что вирусам root права могут быть и не нужны. А если они все-таки они понадобятся, то при определённых обстоятельствах он может их получить. В программном обеспечении, а особенно таком сложном как ядро Linux время от времени обнаруживаются уязвимости. Какие-то из них более серьезные, какие-то менее, но за последнее десятилетие практически каждые несколько лет где-нибудь обнаруживалась громкая уязвимость.

Что такое уязвимость? Это определенная ошибка в программе, которая допущена ее разработчиком. Эту ошибку может использовать любой пользователь для того, чтобы навредить функционированию ПО, в том числе для злоумышленных целей. Наиболее опасные уязвимости это те, которые дают возможность исполнять произвольный код. Все уязвимости можно условно поделить на локальные и удаленные. Локальные могут быть, если у мошенника есть доступ к локальному компьютеру, а удаленные, когда нужен только доступ через Интернет.

Например, в 2014 году обнаружили уязвимость в OpenSSL под названием Heartbleed, которая позволяла получить доступ к любому серверу, в 2018 году обнаружили уязвимости Spectre и Meltdown в процессорах, а в 2019 была проблема с Exim. В ядре Linux время от времени уязвимости тоже появлялись. Все они открывают дорогу вирусам в вашу систему и позволяют им делать с ней всё, что они захотят. Единственный способ защититься - вовремя обновлять систему и программное обеспечение чтобы получать все исправления.

**Dirty COW**- уязвимость, которая была обнаружена в 2018 году. Эта локальная уязвимость появляется во время копирования при записи. Благодаря ей каждый непривилегированный пользователь может получить доступ к ОС. Почти 10 лет такая уязвимость существовала в системе, а после выявления была быстро устранена. Но те владельцы устройств  Andoid, которые не обновляют ядро, до сих пор подвержены риску. Код уязвимости CVE-2016-5195.

**Уязвимость Glibc** (под кодом CVE-2015-7547) была также выявлена в 2016 году. Как оказалось, библиотека Glibc имеет ошибку, которая позволяет мошеннику выполнить свой код по удаленному доступу. Glibc - это реализация стандартной библиотеки Си и С++, она применяется в большинстве программ Linux, и в PHP, Python, Perl. Ошибку допустили в коде разбора ответа DNS сервера. Ее могли использовать злоумышленники, к DNS которых обращались уязвимые машины, а также выполняющие MITM атаку. Эта проблема была с 2008 года. После ее обнаружения, были выпущены специальные патчи.

**Уязвимость нулевого дня ядра** - локальная уязвимость. Она давала возможность повысить права текущего пользователя до root. Это происходила из-за допущенной ошибки в системе работы с криптографическими данными ядра, которые хранятся в памяти. Уязвимость нашли в феврале 2016 года. Она была во всех ядрах начиная от версии 3.8, тоесть ошибка “жила” 4 года и никто о ней не догадывался.

**Уязвимость Shellshock**(под кодом  CVE-2014-6271) существовала 22 года и была обнаружена только в 2014 году. Shellshock появился из-за ошибки в интерпретаторе команд Bash. Он используется по умолчанию в большинстве дистрибутивов Linux. Bash дает возможность объявлять переменные окружения без аутентификации пользователя, а вместе c ними можно выполнить любую команду. Это особо опасно CGI скриптах. Уязвимы серверы и ПК пользователей, маршрутизаторы и другие устройства. Хакер может выполнять удаленно любую команду.

**Уязвимость протокола HTTP/2** - это целая определенная серия уязвимостей. Они позволяют мошеннику очень сильно замедлить работу веб-сервера и выполнить атаку отказ от обслуживания.

**Уязвимость в OpenJDK**существовала с 2013 года. На данный момент исправлена. Это была серьезная уязвимость linux 2016 в Java машине OpenJDK с кодом CVE-2016-0636. Она влияла на пользователей, которые работали  с  Oracle Java SE 7 Update 97 и 8 Update 73 и 74 для Windows, Solaris, Linux и Mac OS X. Позволяла мошеннику выполнить произвольный код за пределами Java машины, если вы откроете специальную страницу в браузере с уязвимой версией Java. Так можно было получить доступ к паролям, личным данным пользователя, запускать программы на ПК.

Следующая причина, почему ОС Linux может с легкостью заразиться вирусом – это как не странно его популярность, а именно популярность на серверах. В отличие от домашних компьютеров, на серверах и различных встраиваемых устройствах, вроде камер и роутеров ОС Linux очень популярна. И для таких платформ вирусы пишут, причём очень активно. В пример можно привести ботнет Mirai, заразивший огромное количество устройств использующих незащищённую службу telnet. Благодаря уязвимостям, описанным выше, вирусы могут проникать на не обновленные серверы и другие устройства. Учитывая, что многие серверы, устройства IoT и Android смартфоны не обновляются и не получают исправления уязвимостей, то уязвимых устройств довольно много.

И наконец, заключительная причина – это универсализация системы. В последнее время мир ОС Linux тоже движется к универсализации. Появились универсальные пакеты snap и flatpack, а самым популярным окружением рабочего стола считается Gnome. Ведь это окружение используется во многих популярных дистрибутивах по умолчанию: в Ubuntu, Fedora, Pop!\_OS и в других. Это может свести на нет все преимущества децентрализации ОС Linux. Например, уже существует вирус EvilGnome, о котором сообщили [Intezer Labs](https://www.intezer.com/blog/linux/evilgnome-rare-malware-spying-on-linux-desktop-users/) в 2019 году. Вирус может скачивать ваши личные файлы, записывать звук с микрофона и делать снимки экрана. Вирусов для ОС Linux всё ещё не особо много, но они начинают активно развиваться.

**Параграф 2.** Штатные средства защиты от заражений.

Безусловно, самым простым и распространенным, но не самым бесполезным штатным средством защиты от заражений является банальное регулярное обновление ПО. Процедура регулярного обновления всего используемого ПО - одна из важнейших в механизме обеспечения безопасности любой операционной системы. Это связано с тем, что злоумышленники постоянно ищут уязвимости в коде программ для того, чтобы взломать систему. Поэтому очень важно регулярно устанавливать новые обновления ПО на машину, для того, чтобы закрыть все ее уязвимости. ОС Linux - яркий пример применения концепции ПО с открытым исходным кодом, поэтому устранение программных недоработок там выглядит совершенно иначе нежели в других операционных системах. Система обновлений одновременно исполняет роль и средства отладки, и технической поддержки. Почти каждый дистрибутив Linux имеет свою, специфичную процедуру обновления программного обеспечения. Для пакетно-ориентированных дистрибутивов, таких как RedHat, используется средство обновления apt-get, которое умеет получать из главного хранилища готовых пакетов дистрибутива (иначе называемом репозиторием) пакеты новых версий используемого в системе ПО. Аналогичные утилиты обновления применяются и в других дистрибутивах - репозиторий Sisyphus для отечественного дистрибутива ALT Linux, служба YaST Online Update дистрибутива SuSE. Все эти механизмы в какой-то мере похожи на метод обновления Windows, но гораздо более масштабны, так как затрагивают все системное ПО в целом. Для Linux возможен и другой путь поддержания актуальности используемого ПО - сборка программ из исходных текстов, причем хранилище исходных текстов регулярно обновляется до актуальной версии, а программы систематически подвергаются перекомпиляции. Активнее всего этот подход используется в дистрибутиве Gentoo, при установке целиком компилирующемся из исходников.

Второе штатное средство для защиты от заражений – это защита безопасности сети. Безопасность OC Linux на сетевом уровне обеспечивает настройка **IPtables**. Это интерфейс, который позволяет управлять работой брандмауэра **netfilter** в ОС Linux. Также, часто под IPtables подразумевают и сам брандмауэр. При помощи данной утилиты можно полностью контролировать доступ вашей системы в интернет. Создавая определенные правила, вы разрешаете или запрещаете входящий и исходящий трафик на любые порты и протоколы с любых **IP** и **Mac** адресов и подсетей. Естественно для этого потребуется время на знакомство с утилитой, но, если вам важна безопасность или контроль над сетью, это того стоит.

Примеры настройки **IPtables.**

Подсистема **IPtables** и **netfilter** встроены в ядро, но вот набор утилит для управления всем этим не всегда поставляется вместе с системой. Для установки утилиты в Ubuntu наберите: sudo apt install iptables.

А в дистрибутивах, основанных на **Fedora**, установка **iptables** выполняется немного по-другому: yum install iptables.

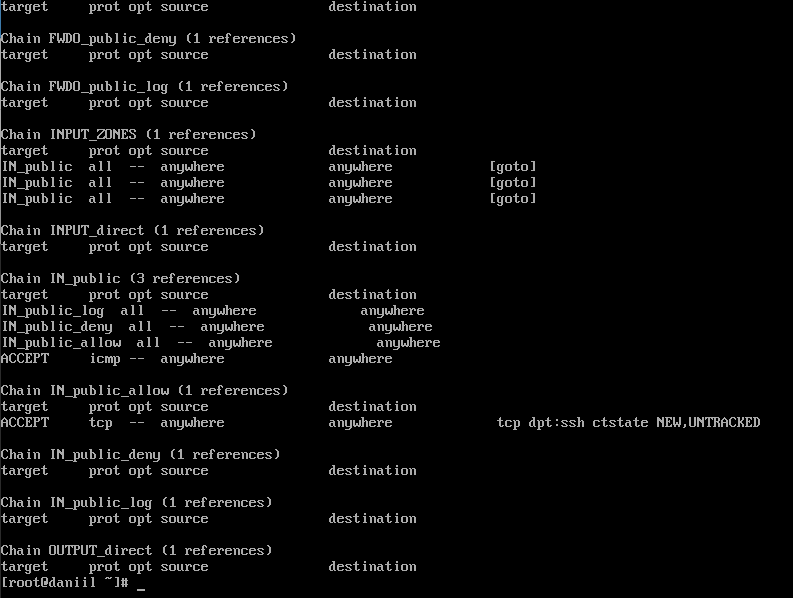
Основные действия, которые позволяет выполнить iptables:

* **-A** - добавить правило в цепочку;
* **-С** - проверить все правила;
* **-D** - удалить правило;
* **-I** - вставить правило с нужным номером;
* **-L** - вывести все правила в текущей цепочке;
* **-S** - вывести все правила;
* **-F** - очистить все правила;
* **-N** - создать цепочку;
* **-X** - удалить цепочку;
* **-P** - установить действие по умолчанию.

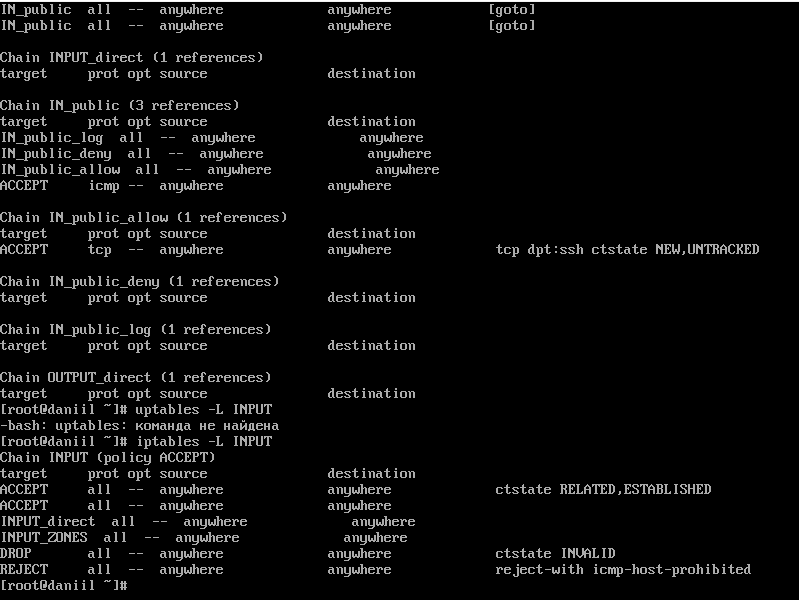
Дополнительные опции для правил:

* **-p** - указать протокол, один из tcp, udp, udplite, icmp, icmpv6,esp, ah, sctp,  
  mh;
* **-s** - указать ip адрес устройства-отправителя пакета;
* **-d** - указать ip адрес получателя;
* **-i** - входной сетевой интерфейс;
* **-o** - исходящий сетевой интерфейс;
* **-j** - выбрать действие, если правило подошло.

1. Сначала рассмотрим, как выполняется просмотр правил iptables, для этого достаточно опции -L: **iptables –L**



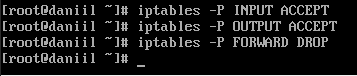
1. Также вы можете указать нужную цепочку, чтобы вывести правила только для нее: **iptables -L INPUT**

\

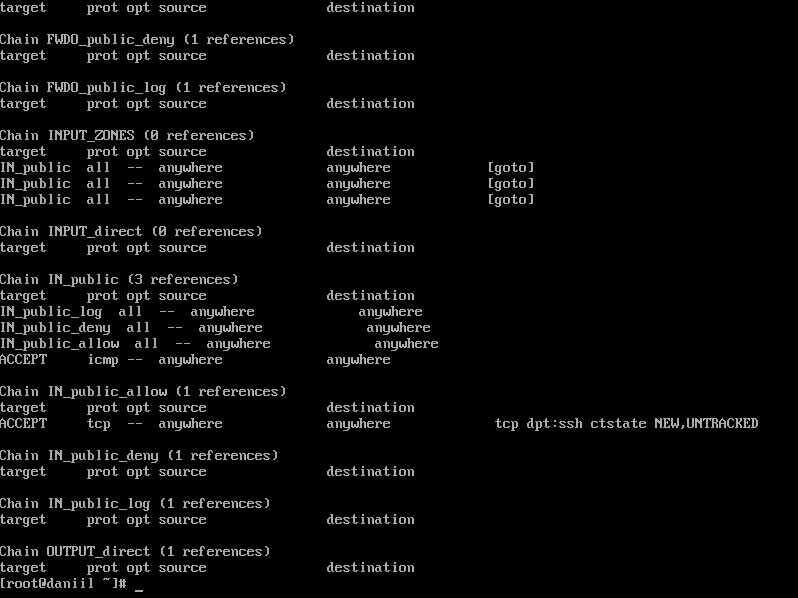
1. Очистка правил. Нельзя просто так отключить **iptables** остановив сервис обновления правил iptables через systemd или даже удалив набор утилит для настройки. Подсистема работает на уровне ядра и не зависит от того, что там у вас установлено. Поэтому если сделаете что-то не так, то нужно будет очистить правила. Для этого надо выполнить: **iptables –F**



1. Если для пакета не подходит ни одно правило, то для него применяется действие по умолчанию. Его можно задать с помощью опции **-p**: **iptables -p INPUT ACCEPT, iptables -p OUTPUT ACCEPT, iptables -p FORWARD DROP**. В этом примере мы разрешаем цепочки **INPUT** и **OUTPUT**, но запрещаем **FORWARD**.



1. Просмотрим **iptables**.



1. Для блокировки пакетов мы можем использовать действие DROP, фильтровать пакеты, которые нужно заблокировать мы можем по множеству критериев, например, протоколу, ip адресу, маске сети, порту и многому другому. Вот так будет выглядеть команда, которая позволяет добавить правило iptables для блокировки всех входящих пакетов от 10.10.10.10: **iptables -A INPUT -s 10.10.10.10 -j DROP**. А теперь исходящие пакеты на этот же адрес: **iptables -A OUTPUT -s 10.10.10.10 -j DROP**

****

1. Блокировка диапазона ip выполняется подобным образом. Для этого нужно использовать маску сети 10.10.10.0/24. Это будут все адреса начиная от 10.10.10.0 до 10.10.10.255:



1. Также можно заблокировать все входящие соединения ssh:



1. Также вы можете полностью очистить iptables выполнив команду с опцией -F: **iptables –F**



На уровне приложений же безопасность обеспечивается с помощью утилиты **WAF**. WAF – это совокупность мониторов и фильтров, предназначенных для обнаружения и блокирования сетевых атак на веб приложение. **WAF** относятся к прикладному уровню модели OSI.

В ОС Linux все события можно отслеживать при помощи лог-файлов. Если кто-нибудь будет пытаться попасть в систему и как-либо ее скомпрометировать, системный администратор (если это сервер) по логам легко сможет отследить, каким образом это осуществляется и что уже успел сделать вирус или взломщик.

**Глава 2**. Вирусы для OC Linux.

**Параграф 1.** История развития и принцип работы вирусов.

Первый в истории вирус для ОС Linux был написан в 1996 году. Хакерская группа VLAD, а точнее ее участник Quantum представил первый рабочий вирус для Linux, написанный на ассемблере с синтаксисом AT&T.

Вирус, получивший имя [Staog](http://www.wiw.org/~meta/vlad.php?read=ARTICLE.2_4&issue=7&desc=STAOG%20Linux%20Virus), при первом запуске получал права root, используя одну из трех уязвимостей: переполнение буфера в командах mount и tip и баг в команде suidperl. Затем прописывался в память ядра с помощью файла /dev/kmem и изменял таблицу системных вызовов так, чтобы функция execve(), запускающая исполняемые файлы, ссылалась на него. В результате при каждом запуске исполняемого файла вирус получал управление, заражал файл, а затем отдавал управление оригинальной функции execve(). Фактически это все, что он делал, никаких зловредных функций, чистый proof of concept. Вскоре уязвимости были пропатчены, и вирус был забыт, так никогда и не обнаружив себя in the wild. Однако метод изменения таблицы системных вызовов еще долго продолжали использовать авторы бэкдоров, пока разработчики Linux не сделали таблицу неэкспортируемой.

Позже, 5 февраля 1997-го, меньше чем через полгода после публикации исходников Staog, компания McAfee опубликовала статью, что вирус для ОС Linux обнаружен «в дикой природе». На самом деле вирус, названный автором Bliss, был выявлен на несколько месяцев раньше, и к моменту публикации пресс-релиза уже существовала его новая, усовершенствованная версия.

Первая версия Bliss была опубликована 29 сентября 1996 года в новостных группах comp.security.unix, alt.comp.virus и comp.os.linux.misc. Однако первые сообщения о вирусе «в диких условиях» появились только 31 января 1997-го в списке рассылки linux-security.

Из-за поднятой шумихи спустя пять дней автор [опубликовал](http://math-www.uni-paderborn.de/~axel/bliss/bliss.txt) обновленную версию вируса (0.4.0), пояснив, что вирус, ходящий по Сети, — это всего лишь альфа-версия и судить его по ней не стоит, он может лучше.

Спустя три дня Рэй Лехтиниеми (Ray Lehtiniemi) провел [анализ вируса](http://math-www.uni-paderborn.de/~axel/bliss/ray_analysis.txt) и выяснил, что даже его новая версия довольно примитивна. Вирус всего лишь находил все доступные для записи исполняемые файлы и перезаписывал их, сохраняя оригинальное приложение в специальном участке файла. Если после этого запускалось инфицированное приложение, вирус извлекал код оригинального приложения в файл /tmp/.bliss-tmp.<pid>, затем форкался и запускал его. Также внутри вируса был найден рудиментарный код, модифицирующий исходники ядра, незаконченная функциональность червя (вирус пытался подключиться к другим хостам по rsh, но, в отличие от червя Морриса, не умел подбирать пароли) и простейший метод защиты от отладки путем завершения самого себя при обнаружении запуска через strace.

Следующие три года о вирусах для ОС Linux не было слышно ничего, однако с 2000-го их количество начало резко увеличиваться. Скорее всего, потому, что среди рядовых пользователей популярность операционной системы стремительно возрастала и многие компании переходили на новую активно развивающуюся операционную систему с BSD и Solaris.

В 2000 году был обнаружен вирус Virus.Linux.Winter.341, заражавший бинарные файлы формата ELF и отличавшийся крайне скромными размерами — 341 байт. Годом позже прошла информация о первом ZIP-черве, который при запуске помещал свою копию во все найденные ZIP-архивы.

Тогда же появились два довольно интересных вируса Ramen и Cheese, причем второй был червем, закрывавшим бэкдор, открытый Ramen.

Позже добавилось несколько червей, эксплуатирующих уязвимости в веб-сервере Apache. Червь Mighty, обнаруженный в 2002-м, проникал на машину через уязвимость в SSL, затем заходил на специально созданный IRC-канал и ожидал удаленных команд. Червь Lupper также перемещался по веб-серверам, но использовал стандартные уязвимости в CGI-скриптах. Как и Mighty, он имел функцию бэкдора.

В 2007 году появился первый вирус для OpenOffice. Написанный на StarBasic и распространяемый в составе файлов OpenOffice Draw (файлы .odg) вирус BadBunny обходил встроенные средства защиты OpenOffice и получал полный доступ к файлам пользователя. В 2009-м вирус WaterFall был обнаружен в одном из скринсейверов на сайте для пользователей GNOME. После установки он открывал бэкдор и по команде устраивал DDoS-атаку на сайт MMOwned.com.

В 2010 году была обнаружена кроссплатформенная модификация червя Koobface, способная поражать не только Windows, но и ОС Linux и OS X. Червь распространялся через социальные сети с помощью социальной инженерии. После нажатия на ссылку у жертвы открывалась копия сайта youtube.com, а при попытке просмотреть ролик запускался Java-апплет. Далее, используя уязвимости Java, червь получал доступ к системе, устанавливал троян и начинал рассылку вредоносных публикаций в социальных сетях.

Позже были найдены и многие другие виды троянских коней, червей и бэкдоров. Основной целью хакеров были, конечно же, серверы (организовывать ботнеты для выполнения DDoS-атак), но были и вирусы, ориентированные на домашние машины, а также, например, роутеры.

По своей сути, принцип работы вирусов для Windows никак не отличается от вирусов, разработанных специально для ОС Linux. Отличаются лишь структуры файлов в самих ОС.

Исполняемые файлы в Linux имеют специальный формат - ELF (Executable and Linkable Format — формат исполняемых и компонуемых файлов). У каждого ELF-файла есть заголовок, в котором указан адрес входа, т.е. место, где начинается код самой программы. Таким образом загрузчик ELF interpretor узнает, что и в каком количестве загружать. "Классический" вирус записывает себя в место точки входа, чтобы, выполнившись, передать управление оригинальной программе, которую он поместил в другое место.

Загрузчик запускает процесс, расположенный по адресу из заголовка, считая его нормальной программой, а вирус, перехвативший таким образом управление, сам является подобием ELF-файла, но сначала в нем записан исполняемый вредоносный код, а потом - подобие ELF-заголовка, чтобы передать управление оригиналу. Антивирусы, сканируя систему и зная "в лицо" многие вирусы, проверяют содержимое файла на наличие знакомого участка кода (тела вируса) и при необходимости удаляют его.

Именно так работает вирус [Linux.Vit.4096](http://www.viruslist.com/eng/viruslist.asp?id=3135&key=00001000050000200003&f_page=0). Из всего этого следует, что вирус должен иметь доступ на запись к инфицируемому файлу. Поэтому НИКОГДА не используйте root просто так. Вирус попавший в вашу систему может обладать лишь теми же правами, что и вы. Если вы работаете под пользователем, то вирус никогда не сможет нанести вред системе, но нанести вред вашим личным данным может.

**Параграф 2.** Виды вирусов и практическая реализация вируса.

Давайте теперь рассмотрим виды вирусов ОС Linux, которые существуют и могут заразить ваш сервер или систему.

1. **Руткиты**. Руткит от rootkit (набор инструментов root) - это вирус, который встраивается в ядро системы и благодаря этому может скрывать своё присутствие. Обычно руткиты используются не сами по себе, а прикрывают какие-либо другие вирусы, например бэкдоры. Такие вирусы очень опасны, потому что обнаружить их очень сложно, а удалить вообще может быть невозможно. Чтобы проверить систему на наличие руткитов можно использовать утилиту **chkrootkit** или **rkhunter**.
2. **Шифровальщики**. Шифровальщики или Ransomware часто встречались в ОС Windows. Они шифруют ваши файлы и требуют перевести определённую суму денег за расшифровку. В 2015 году такие вирусы появились и для ОС Linux. Первым был вирус, получивший название Encoder. Используя разные уязвимости, он проникал на сервер и шифровал находящиеся там данные с помощью алгоритма AES и RSA. Вирус проникал в систему с помощью уязвимостей в CMS для создания интернет магазинов Magento.
3. **Ботнеты**. Ботнеты менее опасны для компьютеров и серверов, которые они заражают,потому что стараются не вредить и вообще не показывать своего присутствия. Обычно они используются для выполнения DDoS атак на различные сайты и узлы сети. К нашумевшим ботнетам можно отнести Mirai, заражавший роутеры, который уже упоминался выше.
4. **Бэкдоры.** Бэкдоры (от Back door) - это программы, которые позволяют хакеру скрытно проникать в вашу систему. Ещё их называют троянами (Trojan) потому что они помогают злоумышленнику проникать в систему. Сами по себе они не наносят никакого вреда, но позволяют человеку, или вирусу, установившему их проникать в вашу систему и делать с ней всё, что ему надо. В качестве примера распространения бэкдоров можно привести [историю](http://www.webupd8.org/2010/06/linux-trojan-goes-unnoticed-for-year.html) с IRC сервером Unreal, который с 2009 по 2010 год был заражен бэкдором в репозитории разработчиков программы. Все пользователи, устанавливавшие программу в тот период, получали в комплекте с ней бэкдор.
5. **Черви.** Черви называются так, потому что они сами распространяются, используя различные уязвимости в системе или слабые пароли. Например, SSH пароли к серверам очень часто пытаются перебрать различные черви чтобы проникнуть в систему, сделать то, что им надо и атаковать следующие системы. Самый первый вирус, который был созданный для Linux был червем. Он назывался Bliss и был обнаружен в 1997 году. Этот вирус перезаписывал собой все доступные ему исполняемые файлы и пытался распространится по доступным хостам.

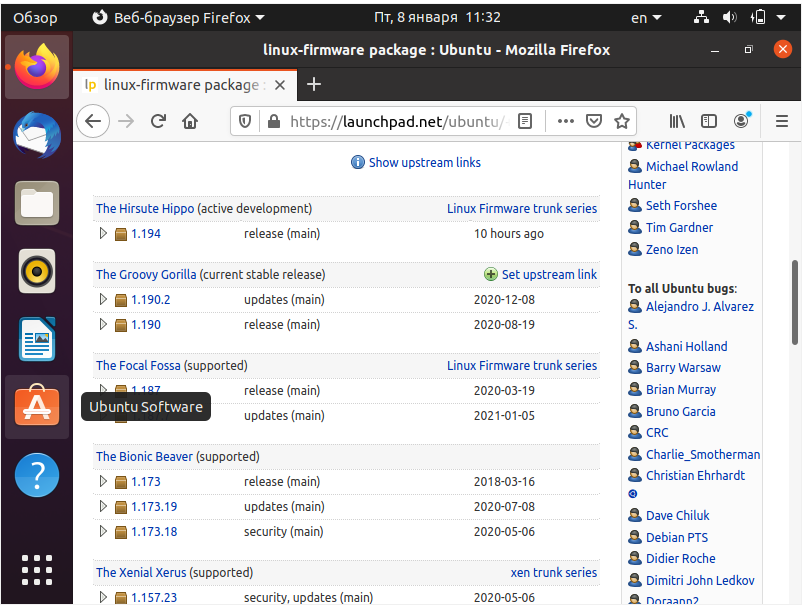
После того, как разобрались с видами вирусов, напишем свой собственный зловредный код. Вирус будем писать на **Ubuntu.**

Для написания вируса “Троянский пингвин” нам понадобятся модифицированные **deb/rpm** пакеты. Пакеты формата **deb** и **rpm** сейчас являются наиболее популярным средством распространения по для Linux.

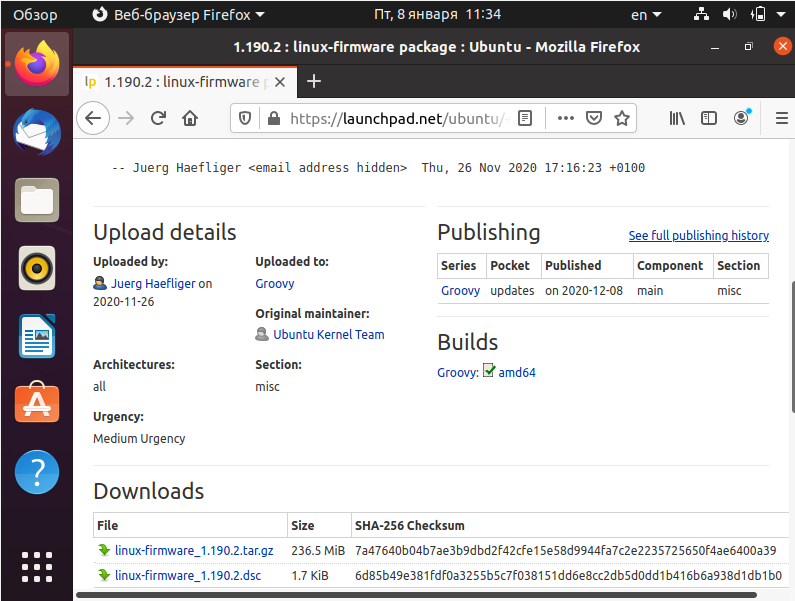
Deb-пакет представляет из себя архив формата [.**ar**](https://ru.wikipedia.org/wiki/Ar_(Unix)), который не использует сжатие. Внутри архива еще три файла: **debian-bynary, control.tar** и **data.tar**  
  
**debian.binary** — текстовый файл, содержащий версию формата deb-пакета, на данный момент там всегда пишут «2.0».  
  
**control.tar** — архив с файлами, содержащими информацию о пакете (например — обязательный файл **control**) и пакеты, необходимые для установки пакета (например — скрипты **preinst**, **postinst**, **prerm** и **postrm**, запускаемые до/после установки/удаления пакета). Может быть сжат с помощью **gzip** или **xz**, в таком случае к имени архива добавляется расширение .**gz** или .**xz** соответственно.  
  
**data**.**tar** — архив с директориями, содержащими устанавливаемые файлы. Директории представлены деревом, в виде которого они должны извлечься в корень файловой системы. Может быть сжат с помощью **gzip**, **bzip2**, **lzma**, **xz**.  
  
Нам необходимо обратить внимание на файл **control** из архива **control**.**tar**. Этот файл содержит информацию о пакете, такую как автор, описание, версию, приоритет пакета в системе и т. д. Нас интересует поле **depends**, в котором указаны зависимости (пакеты, без которых по из данного пакета не может работать). В это поле наш вирус будет дописывать **fakeroot** и **dpkg** — утилиты, которые понадобятся при модификации других пакетов на зараженном компьютере.  
  
Для сборки deb-пакета создается корневая директория пакета. В нее кладутся директории с устанавливаемыми файлами и директория **DEBIAN**, содержащую служебные файлы, среди которых **control** и скрипты для установки/удаления. Затем выполняется команда **fakeroot** **dpkg-deb --build ./path.**

**Шаг 1.** Установка необходимого ПО.

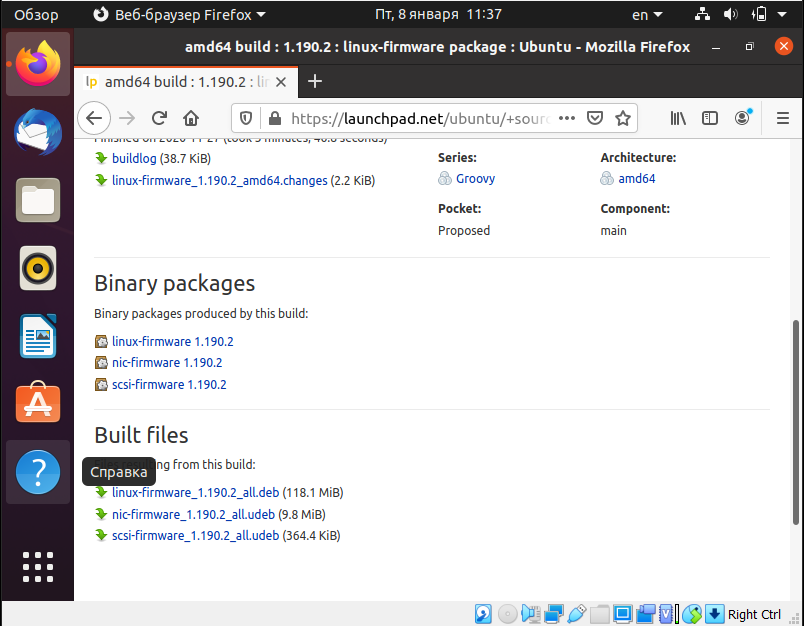
* 1. Для начала установим **deb-пакеты**. Переходим на сайт Launchpad.net и выбираем нужную версию программы.

****

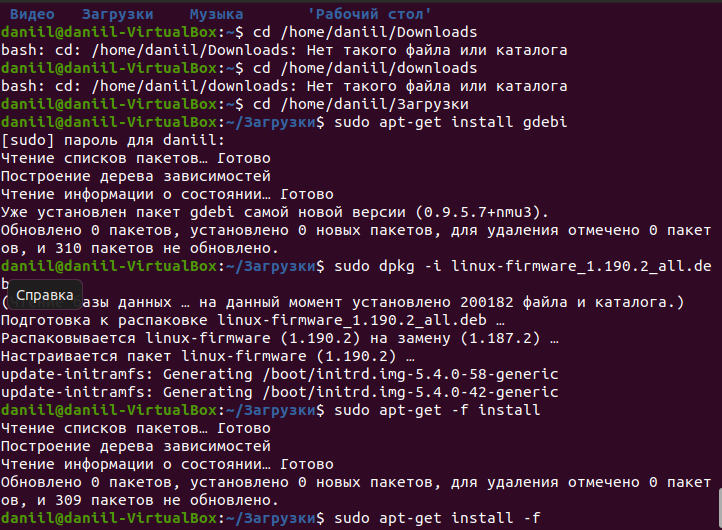
* 1. Дальше выбираем архитектуру:

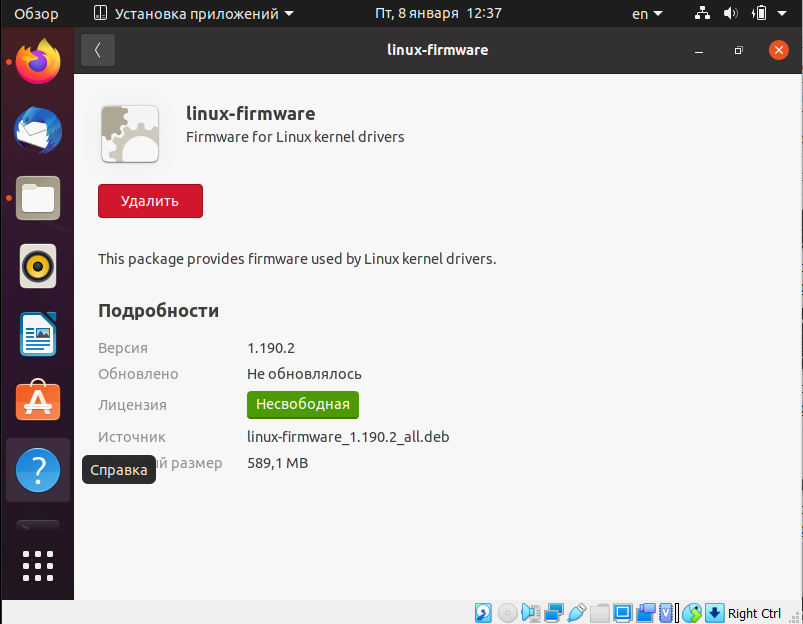
****

* 1. И осталось получить deb файл для нашей системы:



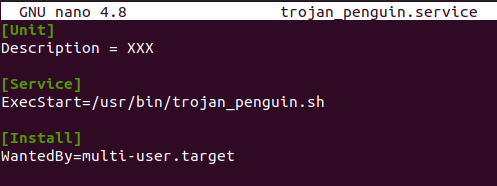
* 1. Все благополучно установилось. Пришлось сделать ввести пару консольных команд. Так как через графический интерфейс выдавало ошибку. Вот эти команды: **sudo apt install gdebi**, **cd home/daniil/Загрузки/, sudo dpkg -i libux-firmware\_1.190.2\_all.deb, sudo apt-get install -f**





**Шаг 2.** Создаем скрипты.

* 1. Cсоздали файл **trojan\_penguin.service**, который будет помещаться в директорию **/lib/systemd/system**, и добавил в нее следующее:



**ExecStart=/usr/bin/trojan\_penguin.sh** — тут я указал путь к файлу (к будущему вирусу), который должен запускаться при старте системы.

* 1. Перейдем в директорию **/usr/bin/trojan\_penguin.sh**  и напишем там следующий код.

****

создаем папку для записи логов, если таковой нет -**if ! [ -d $debug/var/log/trojan\_penguin/ ]; then mkdir $debug/var/log/trojan\_penguin fi**

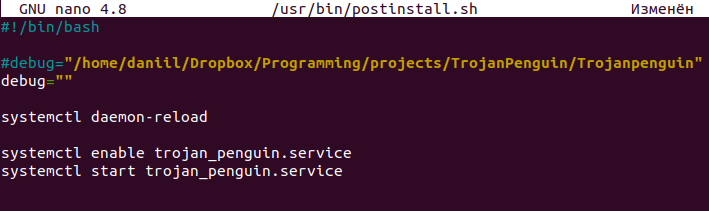
работаем в бесконечном цикле, делая паузы по 10 минут - **while [ 1 ]**

**list=$(find /home -name "\*.deb")** - ищем deb-пакеты для каждого найденного пакета выполняем следующий цикл - **for line in $list**

**debug/usr/bin/tp\_infect.sh $line >> $debug/var/log/trojan\_penguin/log** - запускаем цикл, модифицирующий пакет, а логи записываем в файл log done **date > $debug/var/log/trojan\_penguin/last\_start** - записываем время последней отработки вируса (мне это помогло во время отладки) **sleep 600** - пауза (60 \* 10 сек = 10 мин)

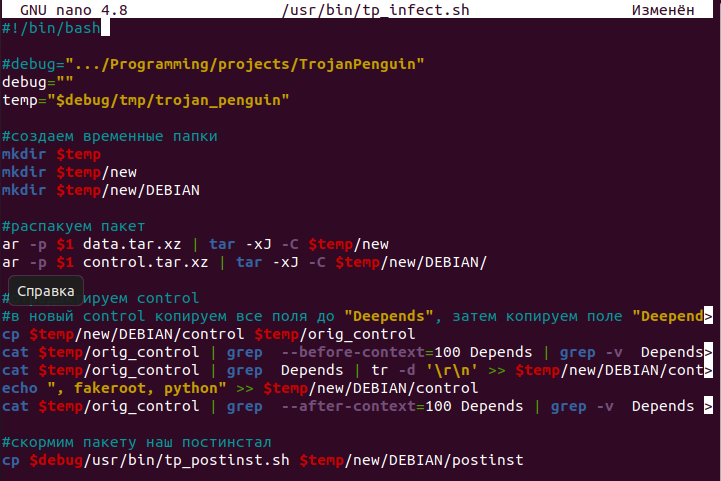
Мы ищем deb-пакеты в разделе **/home**, пути к найденным файлам записываем в переменную **list.** Потом просто перебираем все строки из **line** и для каждого файла запускаем скрипт **tp\_infect.sh**, который заразит этот файл. Когда писали вирус, скрипты находились в отдельной директории, и для удобства создали переменную **debug**, в которой я прописал путь к этой папке.  
  
Демон готов, осталось научиться его запускать при старте системы. Для этого напишем скрипт **postinstall.** Он будет запускаться сразу после установки зараженного пакета и указывать, чтобы наш вирус запускался вместе с системой. Разместим его в директории **"/usr/bin/"**, чтобы оттуда копировать его в заражаемые пакеты.

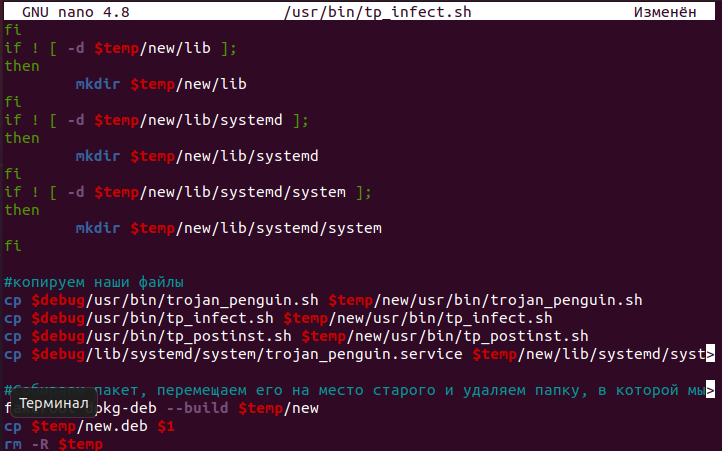
* 1. Создали файл **postinstall** и записали в него следующий код.



#### **Шаг 3.** Модифицируем deb-пакет.

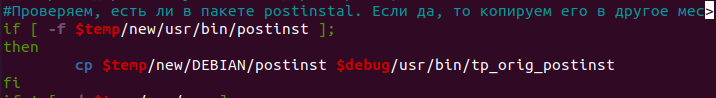
* 1. Архивы, содержащиеся в deb-пакете могут иметь разные разрешения. Мы рассмотрели только тот случай, когда архивы сжаты с помощью**. xz**. Файл **/usr/bin/tp\_infect.sh**, отвечающий за модификацию, получил такое содержимое:



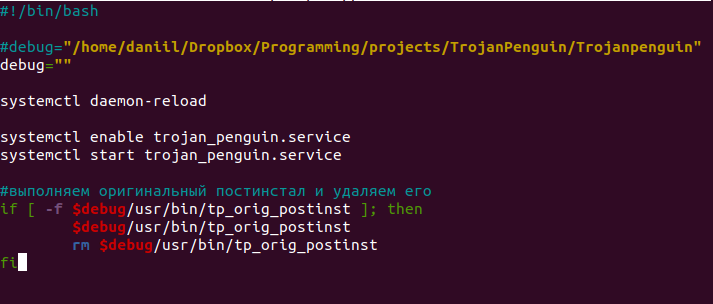


#### **Шаг 4.** Проблемы с postinstall.

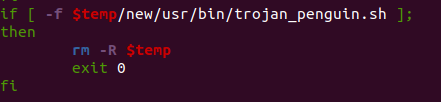
* 1. Все бы хорошо, но теперь у нас проблема. Оригинальный **postinstal** может быть написан на разных языках: **python**, **bash**, что не позволят нам просто взять и дописать свой **postinstall** в него. Решили эту проблему следующим образом: добавили в скрипт **tp\_infect.sh** следующие строки:

****

* 1. В **postinstall** добавим следующе строки:

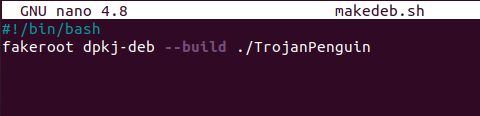
****

* 1. Одна проблема решена, но появилась другая. Наш вирус будет модифицировать пакет, даже если он уже заражен. При модификации вирус увидит, что в пакете есть **postinstal**, переместит его в **/usr/bin/,** тем самым перезаписав оригинал. Чтобы этого избежать, добавили проверку в **tp\_infect.sh**:

****

**Шаг 5.** Собираем воедино.

* 1. Для того, чтобы собрать вирус, необходимо выполнить следующую команду – **sudo nano makedeb.sh** и впишем там следующее**:**

****

* 1. Вирус готов.

**Итог:**

К сожалению, данный вирус не обнаружил антивирус, так как он достаточно простой и не является опасным для системы.

**Вывод:**

Как мы поняли, вирусы и операционные системы, в том числе и Linux, вещи все же совместимые.

За последнее время их появилось достаточно много. Другое дело, что они ориентированы, либо на уязвимые и не обновляемые системы, IoT устройства, или социальную инженерию.

Поэтому, для того, чтобы защититься от них, очень важно своевременно устанавливать обновления и следовать тем правилам, что были описаны в Главе 1, параграф 2. И тогда ваша машина будет под надежной защитой.