**Защита программного обеспечения** — комплекс мер, направленных на защиту [программного обеспечения](https://ru.wikipedia.org/wiki/Программное_обеспечение) от несанкционированного приобретения, использования, распространения, модифицирования, изучения и воссоздания аналогов.

**Защита от несанкционированного использования программ** — система мер, направленных на противодействие нелегальному использованию [программного обеспечения](https://ru.wikipedia.org/wiki/Программное_обеспечение). При защите могут применяться организационные, юридические, программные и программно-аппаратные средства.

**Защита от копирования** к программному обеспечению применяется редко, в связи с необходимостью его распространения и установки на компьютеры пользователей. Однако, от копирования может защищаться лицензия на приложение (при распространении на физическом носителе) или его отдельные алгоритмы.

## Технические средства защиты

См. также: [Технические средства защиты авторских прав](https://ru.wikipedia.org/wiki/Технические_средства_защиты_авторских_прав)

Методы можно классифицировать ***по способу распространения*** защищаемого программного обеспечения и ***типу носителя лицензии***.

### Локальная программная защита

Требование ввода серийного номера (ключа) при установке/запуске. История этого метода началась тогда, когда приложения распространялись только на физических носителях (к примеру, [компакт-дисках](https://ru.wikipedia.org/wiki/Компакт-диск)). На коробке с диском был напечатан серийный номер, подходящий только к данной копии программы.

С распространением сетей очевидным недостатком стала проблема распространения образов дисков и серийных номеров по сети. Поэтому в настоящий момент метод используется только в совокупности одним или более других методов (к примеру, организационных).

### Сетевая программная защита

* локальная

Сканирование [сети](https://ru.wikipedia.org/wiki/Локальная_вычислительная_сеть) исключает одновременный запуск двух программ с одним регистрационным ключом на двух компьютерах в пределах одной локальной сети.

Недостаток в том, что [брандмауэр](https://ru.wikipedia.org/wiki/Межсетевой_экран) можно настроить так, чтобы он не пропускал пакеты, принадлежащие защищённой программе. Правда, настройка брандмауэра требует некоторых пользовательских навыков. Кроме того, приложения могут взаимодействовать по сети (к примеру, при организации [сетевой игры](https://ru.wikipedia.org/wiki/Сетевая_игра)). В этом случае брандмауэр должен пропускать такой [трафик](https://ru.wikipedia.org/wiki/Сетевой_трафик).

* глобальная

Если программа работает с каким-то централизованным [сервером](https://ru.wikipedia.org/wiki/Сервер_(аппаратное_обеспечение)) и без него бесполезна (например, сервера [онлайн-игр](https://ru.wikipedia.org/wiki/MMOG), серверы обновлений [антивирусов](https://ru.wikipedia.org/wiki/Антивирусная_программа)). Она может передавать серверу свой серийный номер; если номер неправильный, сервер отказывает в услуге. Недостаток в том, что, существует возможность создать сервер, который не делает такой проверки. Например, существовал сервер *battle.da*, который по функциям был аналогичен [Battle.net](https://ru.wikipedia.org/wiki/Battle.net) (от компании Blizzard Entertainment), но пускал пользователей неавторизованных копий игр. Сейчас этот сервер закрыт, но существует немалое количество [PvPGN](https://ru.wikipedia.org/wiki/PvPGN)-серверов, которые также не проверяют регистрационные номера.

### Защита при помощи компакт-дисков

Программа может требовать оригинальный компакт-диск. В частности, такой способ применяется в [играх](https://ru.wikipedia.org/wiki/Компьютерная_игра). Стойкость таких защит невелика, ввиду широкого набора инструментов снятия образов компакт-дисков.[[1]](https://ru.wikipedia.org/wiki/Защита_программного_обеспечения" \l "cite_note-1)

Как правило, этот способ защиты применяется для защиты программ, записанных на этом же компакт-диске, являющимся одновременно ключевым.

Для защиты от копирования используется:

* запись информации в неиспользуемых секторах;
* проверка расположения и содержимого «сбойных» секторов;
* проверка скорости чтения отдельных секторов.

Первые два метода практически бесполезны из-за возможности снятия полного образа с диска с использованием соответствующего [прикладного ПО](https://ru.wikipedia.org/wiki/Прикладное_программное_обеспечение). Третий метод считается более надежным (используется, в частности, в защите [StarForce](https://ru.wikipedia.org/wiki/StarForce)). Но существуют программы, которые могут эмулировать диски с учётом геометрии расположения данных, тем самым обходя и эту защиту. В StarForce, в числе прочих проверок, также выполняется проверка возможности записи на вставленный диск. Если она возможна, то диск считается не лицензионным. Однако, если образ будет записан на диск [CD-R](https://ru.wikipedia.org/wiki/CD-R), то указанная проверка пройдет. Существует возможность скрыть тип диска, чтобы [CD-R](https://ru.wikipedia.org/wiki/CD-R) или [CD-RW](https://ru.wikipedia.org/wiki/CD-RW) был виден как обычный [CD-ROM](https://ru.wikipedia.org/wiki/CD-ROM). Однако, в драйвер защиты может быть встроена проверка на наличие эмуляции.

В настоящее время наибольшую известность в мире имеют системы защиты от копирования [SecuROM](https://ru.wikipedia.org/wiki/SecuROM), [StarForce](https://ru.wikipedia.org/wiki/StarForce), [SafeDisc](https://ru.wikipedia.org/wiki/SafeDisc), [CD-RX](https://ru.wikipedia.org/w/index.php?title=CD-RX&action=edit&redlink=1) и [Tages](https://ru.wikipedia.org/w/index.php?title=Tages&action=edit&redlink=1).[[2]](https://ru.wikipedia.org/wiki/Защита_программного_обеспечения" \l "cite_note-2)

Для многих программ указанный метод защиты недоступен ввиду отличного способа распространения (например, [shareware](https://ru.wikipedia.org/wiki/Shareware)-программы).

### Защита при помощи электронных ключей



Современные электронные ключи

[Электронный ключ](https://ru.wikipedia.org/wiki/Электронный_ключ) ([донгл](https://ru.wikipedia.org/wiki/Электронный_ключ)), вставленный в один из [портов](https://ru.wikipedia.org/wiki/Порт_(компьютер)) компьютера (с интерфейсом USB, LPT или COM) содержит ключевые данные, называемые также [лицензией](https://ru.wikipedia.org/wiki/Лицензия), записанные в него разработчиком

* информация для чтения/записи (в настоящий момент практически не применяется, так как после считывания ключ может быть сэмулирован)
* ключи аппаратных криптографических алгоритмов (используется наиболее часто)
* алгоритмы, созданные разработчиком программы (ставший доступным сравнительно недавно метод, в связи с появлением электронных ключей с микропроцессором, способным исполнять произвольный код; в настоящее время используется все чаще)

Достоинства защиты с использованием электронных ключей:

* Ключ можно вставлять в любой компьютер, на котором необходимо запустить программу
* Ключ не занимает/не требует наличия дисковода
* Электронный ключ умеет выполнять криптографические преобразования
* Современные ключи могут исполнять произвольный код, помещаемый в них разработчиком защиты (пример — [Guardant Code](https://ru.wikipedia.org/wiki/Guardant), [Senselock](https://ru.wikipedia.org/wiki/Senselock))

Стойкость защиты основывается на том, что ключевая информация защиты (криптографические ключи, загружаемый код) не покидает ключа в процессе работы с ним.

Основные недостатки:

* Цена (15—30 долларов за штуку)
* Необходимость доставки ключа конечному пользователю

Ранее к недостаткам можно было также отнести невысокое быстродействие ключа (в сравнении с CPU компьютера). Однако современные ключи достигают производительности в 1.25 DMIPS (пример — [HASP](https://ru.wikipedia.org/wiki/HASP),[Guardant](https://ru.wikipedia.org/wiki/Guardant)), а техника защиты с их помощью не предполагает постоянного обмена с ключом.

Существовавшие также ранее проблемы с установкой ключа на определенные аппаратные платформы в настоящий момент решены при помощи сетевых ключей (которые способны работать с одной или более копиями защищенного приложения, просто находясь с ним в одной локальной сети) и с помощью программных или аппаратных средств «проброса» USB-устройств по сети.

### Привязка к параметрам компьютера и активация

Привязка к информации о пользователе / серийным номерам компонентов его компьютера и последующая *активация программного обеспечения* в настоящий момент используется достаточно широко (пример: ОС Windows).

В процессе установки программа подсчитывает [код активации](https://ru.wikipedia.org/wiki/Код_активации) — контрольное значение, однозначно соответствующее установленным [комплектующим](https://ru.wikipedia.org/wiki/Аппаратное_обеспечение) компьютера и параметрам установленной ОС. Это значение передается разработчику программы. На его основе разработчик генерирует ключ активации, подходящий для активации приложения только на указанной машине (копирование установленных исполняемых файлов на другой компьютер приведет к неработоспособности программы).

Достоинство в том, что не требуется никакого специфического [аппаратного обеспечения](https://ru.wikipedia.org/wiki/Аппаратное_обеспечение), и программу можно распространять посредством [цифровой дистрибуции](https://ru.wikipedia.org/wiki/Цифровая_дистрибуция) (по Интернет).

Основной недостаток: если пользователь производит модернизацию компьютера (в случае привязки к [железу](https://ru.wikipedia.org/wiki/Аппаратное_обеспечение)), защита отказывает. Авторы многих программ в подобных случаях готовы дать новый регистрационный код. Например, [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) в [Windows XP](https://ru.wikipedia.org/wiki/Windows_XP) разрешает раз в 120 дней генерировать новый регистрационный код (но в исключительных случаях, позвонив в службу активации, можно получить новый код и после окончания этого срока).

В качестве привязки используются, в основном, серийный номер [BIOS](https://ru.wikipedia.org/wiki/BIOS) материнской платы, серийный номер винчестера. В целях сокрытия от пользователя данные о защите могут располагаться в неразмеченной области жесткого диска.

До недавнего времени такие защиты разрабатывались и внедрялись разработчиками самого программного продукта. Однако сейчас существуют SDK для работы с программными ключами, например [HASP](https://ru.wikipedia.org/wiki/HASP) SL от компании Aладдин Р. Д. Также все большее распространение получают сервисы, предлагающие одновременно функционал «навесной» защиты и сервера активации/лицензирования (пример — [Guardant Online](https://ru.wikipedia.org/w/index.php?title=Guardant_Online&action=edit&redlink=1), [Protect online](https://ru.wikipedia.org/w/index.php?title=Protect_online&action=edit&redlink=1)).

### Защита программ от копирования путём переноса их в [онлайн](https://ru.wikipedia.org/wiki/Онлайн)

Другим направлением защиты программ является использование подхода [SaaS](https://ru.wikipedia.org/wiki/SaaS), то есть предоставление функционала этих программ (всего или части), как сервиса. При этом код программы расположен и исполняется на сервере, доступном в глобальной сети. Доступ к нему осуществляется по принципу [тонкого клиента](https://ru.wikipedia.org/wiki/Тонкий_клиент). Это один из немногих случаев, когда реализуется защита от копирования.

Код исполняется на «доверенной» стороне, откуда не может быть скопирован.

Однако, и здесь возникает ряд проблем, связанных с безопасностью:

* стойкость такой защиты зависит, прежде всего, от защищенности серверов, на которых он исполняется (речь идет о Интернет-безопасности)
* важно обеспечение конфиденциальности запросов, аутентификации пользователей, целостности ресурса (возможности «горячего» резервирования), и доступности решения в целом

Возникают также вопросы доверия сервису (в том числе правовые), так как ему фактически «в открытом виде» передаются как само ПО, так и данные, которые оно обрабатывает (к примеру, [персональные данные](https://ru.wikipedia.org/wiki/Персональные_данные) пользователей).

### Защита кода от анализа

Можно выделить здесь отдельно средства защиты непосредственно кода приложения от анализа и использования в других программах. В частности, применяются [обфускаторы](https://ru.wikipedia.org/wiki/Обфускация) — программы нужны для запутывания кода с целью защиты от его анализа, модификации и несанкционированного использования.

### Защита программного обеспечения на мобильных платформах

Основная статья: **[Мобильная игра#Защита от копирования](https://ru.wikipedia.org/wiki/Мобильная_игра" \l ".D0.97.D0.B0.D1.89.D0.B8.D1.82.D0.B0_.D0.BE.D1.82_.D0.BA.D0.BE.D0.BF.D0.B8.D1.80.D0.BE.D0.B2.D0.B0.D0.BD.D0.B8.D1.8F)**

Способы защиты программного обеспечения для мобильных платформ от копирования обычно основываются на невозможности рядового пользователя считывать/изменять хранящиеся в [ППЗУ](https://ru.wikipedia.org/wiki/ППЗУ) аппарата данные. Может также использоваться активация программного обеспечения.

## Устаревшие технические средства защиты

В прошлом применялись и другие методы защиты ПО от ***нелегального использования***.

### Ключевая дискета

Метод был распространен во времена [MS-DOS](https://ru.wikipedia.org/wiki/MS-DOS), сейчас, в силу устаревания технологии FDD, практически не применяется. Есть четыре основных способа создания некопируемых меток на дискетах:

* Считывание конкретного сектора дискеты (возможно, пустого или сбойного). Это самый простой способ защиты, и при копировании «дорожка в дорожку» дискета копируется.
* Запоминание сбойных секторов дискеты. Перед тем, как записать на дискету информацию, её царапают (или прожигают [лазером](https://ru.wikipedia.org/wiki/Лазер)), после этого записывают номера сбойных секторов. Для проверки подлинности дискеты программа пытается записать в эти сектора информацию, затем считать её.
* Нестандартное [форматирование](https://ru.wikipedia.org/wiki/Форматирование) дискеты. Известна программа FDA (Floppy Disk Analyzer), которая могла проводить исследование и копирование таких дискет.
* «Плавающий бит». Один [бит](https://ru.wikipedia.org/wiki/Бит) записывается так, что в некоторых случаях он читается как «0», в некоторых как «1». Проводится многократное считывание дискеты; среди результатов считывания должны быть и нули, и единицы.

### Запись некопируемых меток на жесткий диск

Некоторые старые программы для [DOS](https://ru.wikipedia.org/wiki/DOS) создавали некопируемые метки на [жёстком диске](https://ru.wikipedia.org/wiki/Жёсткий_диск). Например, файл длиной 1 [байт](https://ru.wikipedia.org/wiki/Байт) занимает на диске один [кластер](https://ru.wikipedia.org/wiki/Кластер_(единица_хранения_данных)) (не менее 512 байт), и в оставшиеся 511 байт можно записать некоторую информацию. Эта практика практически не используется, так как велик риск потери данных.

### Привязка к некоторому физическому объекту

Лицензия программы может привязываться к некоторому физическому объекту, к примеру

* к руководству пользователя. Например, программа выводит: «*Введите 5-е слово на 12-й сверху строке 26-й страницы*». Более изощрённый способ защиты — в руководстве находится важная информация, без которой невозможно пройти игру, этим известна серия [Space Quest](https://ru.wikipedia.org/wiki/Space_Quest). Распространение [сканеров](https://ru.wikipedia.org/wiki/Сканер) и [многозадачных](https://ru.wikipedia.org/wiki/Многозадачность) [операционных систем](https://ru.wikipedia.org/wiki/Операционная_система) положило конец этой практике.
* к некоторому механическому устройству. Игра [Another World](https://ru.wikipedia.org/wiki/Another_World) поставлялась с «кодовым колесом». В системе защиты от копирования [Lenslok](https://ru.wikipedia.org/wiki/Lenslok), применявшейся в играх для [ZX Spectrum](https://ru.wikipedia.org/wiki/ZX_Spectrum), надо было, посмотрев на картинку через систему [призм](https://ru.wikipedia.org/wiki/Призма_(оптика)), увидеть двухбуквенный код.

## Недостатки технических методов защиты ПО

### Уязвимости современных методов защиты ПО

Основная статья: [**Взлом программного обеспечения**](https://ru.wikipedia.org/wiki/Взлом_программного_обеспечения)

Уязвимости современных методов защиты можно достаточно строго классифицировать в зависимости от использованного метода защиты.

* Проверка оригинального носителя. Можно обойти при помощи копирования / [эмуляции](https://ru.wikipedia.org/wiki/Эмуляция) диска (специальная программа полностью копирует диск, затем создается драйвер виртуального дисковода, в который помещается образ, который программа принимает за лицензионный диск. Во многих играх применяется вариант этого метода под названием «Mini Image», когда подставной диск имеет маленький размер (несколько [мегабайт](https://ru.wikipedia.org/wiki/Мегабайт), содержащие только лицензионную информацию), программа признаёт его лицензионным
* Ввод серийного номера. Основной уязвимостью является возможность беспрепятственного копирования и распространения дистрибутива вместе с серийным номером. Поэтому в настоящее время практически не используется (либо используется в совокупности с другими методами).
* Активация программного обеспечения. В отличие от предыдущего метода, активационный код генерируется с использованием уникальной информации (S/N оборудования, информации о пользователе) и является уникальным. В этом случае, в момент генерации кода активации в процессе установки программы есть риск эмуляции «универсального» аппаратного окружения (как то перехват обращений программы при считывании соответствующей информации, либо запуск программы изначально в [виртуальной среде](https://ru.wikipedia.org/wiki/Виртуальная_машина)). Также, при неиспользовании запутывания кода защищенного приложения (или использовании слабых методов), злоумышленник может найти код генерации кода активации и вынести его в отдельную утилиту (т. н. «генератор ключей aka keygen»), ровно как и вырезать всю процедуру активации (что, однако, сложнее, так как он может вызываться в разных частях приложения)
* Использование [электронных ключей](https://ru.wikipedia.org/wiki/Электронный_ключ). Часто встречается мнение о возможности эмуляции электронного ключа или библиотек интерфейса [API](https://ru.wikipedia.org/wiki/API), используемого при обращении к электронному ключу. Это действительно можно сделать при неграмотной реализации защиты на электронном ключе (к примеру, программа только проверяет наличие ключа и читает/пишет в него что-либо). Однако встроенные в программу защитные механизмы собственной разработки, основанные на вызове симметричных и асимметричных алгоритмов электронного ключа практически исключают возможность его эмуляции, так как обращения к ключу происходят каждый раз разные и накопить достаточное количество статистики для создания полного статистического аналога невозможно. Таким образом, стойкость защиты сильно зависит от реализации (в том числе от наличия уникальных защитных механизмов, реализованных разработчиком защиты). Тем не менее потенциально стойкость такой защиты может быть очень высока.
* «Отключение» защиты путём модификации программного кода (к примеру, удаления проверок лицензии). Может быть реализовано при неиспользовании (или использовании слабых) [инструментов запутывания кода](https://ru.wikipedia.org/wiki/Обфускация). В результате программа дизассемблируется (или даже декомпилируется, в худшем случае), код исследуется на наличие защитных механизмов, найденные проверки удаляются.

Многие защиты предоставляют инструменты противодействия взлому: дестабилизация [отладчика](https://ru.wikipedia.org/wiki/Отладчик); [шифрование](https://ru.wikipedia.org/wiki/Шифрование) кода, исключающее изучение кода в статике при помощи [дизассемблера](https://ru.wikipedia.org/wiki/Дизассемблер); запутывание кода, «ложные ветви», сбивающие [хакера](https://ru.wikipedia.org/wiki/Хакер) с толку; проверка целостности файла, не дающая накладывать [патчи](https://ru.wikipedia.org/wiki/Патч); виртуализация кода с собственной системой команд. Все эти методы препятствуют изучению и анализу логики защиты, повышают её стойкость.

### Использование автоматических средств защиты

Существует проблема, связанная с недостатком ресурсов (в том числе временных) у разработчиков ПО. Им может не хватать времени, финансов или квалификации на реализацию собственной стойкой защиты. Они вынуждены пользоваться сторонними автоматическими средствами защиты ПО. Эти средства пристыковывают к скомпилированной программе защитный модуль. Преимущество такой защиты в том, что её можно установить на любую программу (даже без доступа к исходному коду программы). Недостаток в самом подходе — «шаблонности» метода. Стандартные защиты имеют большую вероятность быть взломанными, так как устанавливаются на несколько программ и тем самым обеспечивают спрос на рынке взлома.

Тем не менее, автоматические средства затрудняют взлом программы. Их иногда целесообразно использовать либо когда защиты нет вообще, либо в совокупности с реализацией собственной уникальной защиты.

# Взлом программного обеспечения

**Взлом программного обеспечения** ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *software cracking*) — действия, направленные на устранение [защиты программного обеспечения](https://ru.wikipedia.org/wiki/Технические_средства_защиты_авторских_прав) (ПО), встроенной разработчиками для ограничения функциональных возможностей. Последнее необходимо для стимуляции покупки такого [проприетарного](https://ru.wikipedia.org/wiki/Собственническое_ПО) ПО, после которой ограничения снимаются.

## Виды взлома

Практически любой взлом сводится к использованию одного из следующих способов:

* Ввод *серийного номера (регистрационного кода)* ([жарг.](https://ru.wikipedia.org/wiki/Жаргонизм) *серийник*) ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *serial number, S/n*) — взлом программы посредством введения правильного [регистрационного ключа](https://ru.wikipedia.org/wiki/Регистрационный_ключ) (или фразы), полученного нелегальным способом. Ключ может генерироваться на основе какой-либо информации (имени владельца [ПО](https://ru.wikipedia.org/wiki/Программное_обеспечение), характеристик аппаратной части компьютера, и т. п.), либо иметь фиксированное значение. Для генерации регистрационного ключа используется тот же алгоритм, что и в программе.

*Примечание1*: Регистрационный код может распространяться в *ключевом файле (файле лицензии)* ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *keyfile*), который обычно помещается в каталог с установленной программой.

*Примечание2*: Для массового взлома, зачастую, создаётся (и в дальнейшем используется) [*генератор ключей*](https://ru.wikipedia.org/wiki/Генератор_ключей) (жарг. *кейген*) ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *keygen* сокр. от key generator) — [программа](https://ru.wikipedia.org/wiki/Компьютерная_программа) для генерации регистрационных ключей (см. выше). Данный вид взлома наиболее востребован (особенно, когда программа часто обновляется или рег. ключ генерируется на основе какой-то информации (см. выше)) и поэтому наиболее ценится. Как правило, требует бо́льшей квалификации взломщика по сравнению с другими видами взлома, но не всегда.

* Использование *загрузчика* (жарг. *лоадер*) ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *loader*) — способ обходить некоторые виды защиты [ПО](https://ru.wikipedia.org/wiki/Программное_обеспечение), заключающиеся в использовании внешних (навесных) систем защиты. Состоит в изменении определённых фрагментов программы в [оперативной памяти](https://ru.wikipedia.org/wiki/Оперативная_память) сразу после её загрузки в эту память, но перед её запуском (то есть перед выполнением кода в [точке входа](https://ru.wikipedia.org/wiki/Точка_входа)).
* Применение [*(бинарного) патча*](https://ru.wikipedia.org/w/index.php?title=Бинарный_патч&action=edit&redlink=1) (часто жарг. *крэк* или *кряк* от [англ.](https://ru.wikipedia.org/wiki/Английский_язык) *crack*) ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *byte patch*) — способ, похожий на «загрузчик», но модификация производится статически в файлах программы. Как правило, это один из самых простых и быстрых способов взлома ПО.
* Использование *взломанной версии файла(ов)* ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *cracked*) — способ заключается в подмене оригинальных файлов программы файлами, которые уже взломаны.
* Использование *эмулятора ключа* ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *key emulator*) — способ используется для обмана защит, построенных на использовании в качестве защиты [электронного ключа](https://ru.wikipedia.org/wiki/Электронный_ключ) (как правило, подключаемого к [LPT](https://ru.wikipedia.org/wiki/LPT) или [USB](https://ru.wikipedia.org/wiki/USB) порту компьютера). Заключается в снятии дампа внутренней памяти ключа. Файл с содержимым этой памяти подаётся на вход специальной программе — эмулятору, которая подключает свой [драйвер](https://ru.wikipedia.org/wiki/Драйвер)-фильтр в стек драйверов и обманывает защищённую программу, эмулируя работу с аппаратным ключом. В случаях наличия в программе обращений к ключу для аппаратного шифрования участка памяти этот метод используется в связке с методом *Бинарный патч*.
* Подмена официального [сайта](https://ru.wikipedia.org/wiki/Сайт) программ и/или соответствующее изменение настроек с целью обойти проверку ключа, если она была вынесена разработчиками на какой-либо интернет-ресурс (В абсолютном большинстве случаев — для предотвращения взлома, реже — для учета и ведения статистики, сбора сведений). Чаще всего осуществляется на примитивном уровне путем модифицирования [файла hosts](https://ru.wikipedia.org/wiki/Hosts) и запуска различных эмуляторов, иногда — использование различных программ ([Денвер](https://ru.wikipedia.org/wiki/Денвер_(программа))) или использование реально существующего веб-ресурса.

При взломе сложных защит, а также при необходимости достичь максимального эффекта, применяется комбинация вышеперечисленных способов. В редких случаях, это происходит при недостаточной квалифицированности взломщика.

Этот список не является исчерпывающим, а лишь обозначает наиболее встречаемые способы взлома.

Вид взлома, в большинстве случаев, обусловлен [видом защиты](https://ru.wikipedia.org/wiki/Защита_программного_обеспечения" \l ".D0.92.D0.B8.D0.B4.D1.8B_.D0.B7.D0.B0.D1.89.D0.B8.D1.82). Для некоторых защит возможно использовать различные виды взлома, для других — способ может быть единственным.

## Принципы взлома

Как правило, в основе работы крэкера лежит исследование [ассемблерного кода](https://ru.wikipedia.org/wiki/Язык_ассемблера), полученного из [машинных инструкций](https://ru.wikipedia.org/wiki/Машинный_код) с помощью специально предназначенной для этого программы-[дизассемблера](https://ru.wikipedia.org/wiki/Дизассемблер). В зависимости от выбранного способа взлома, результат исследования может использоваться, например, для построения [генератора ключей](https://ru.wikipedia.org/wiki/Генератор_ключей) или для внесения необходимых изменений в [исполняемый файл](https://ru.wikipedia.org/wiki/Исполняемый_файл). Последний способ в большинстве случаев наиболее лёгкий, так как не требует изучения алгоритма проверки правильности ключа: зачастую взлом сводится к поиску проверки нескольких условий (наподобие «ВведённоеЧисло *равно* ЭталонномуЧислу?») и замене такого условия на безусловный переход ([goto](https://ru.wikipedia.org/wiki/Goto), jmp), или, реже, на противоположное (то есть для данного примера на «ВведённоеЧисло *не равно* ЭталонномуЧислу?»).

Кроме того, внесение изменений в исполняемый файл ([патч](https://ru.wikipedia.org/wiki/Патч)) может производиться с целью отключения нежелательных действий со стороны программы (например, напоминание о необходимости регистрации), сокращения функциональности программы. В этих случаях, часто, соответствующие команды процессору заменяются на байты со значением 90h (в [шестнадцатеричной системе счисления](https://ru.wikipedia.org/wiki/Шестнадцатеричная_система_счисления)), что соответствует ассемблерной команде [nop](https://ru.wikipedia.org/wiki/NOP) (*No Operation*), то есть «пустой команде», не выполняющей никаких действий. Если таких команд много, то применяется безусловный переход (перепрыгивание ненужного кода). Возможно также расширение возможностей программы написанием дополнительного кода, но, как правило, это слишком трудоёмкий процесс, не оправдывающий временных затрат.

Между тем, патч возможен, как правило, в том случае, когда исполняемый файл программы не защищён специальными «пакерами» и «протекторами» — программами, скрывающими реальный код исполняемого файла. Для последнего типа программ зачастую используется самая интеллектуальная часть [обратной разработки](https://ru.wikipedia.org/wiki/Обратная_разработка) ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *reverse engineering*) — исследование кода программы при помощи [отладчика](https://ru.wikipedia.org/wiki/Отладчик) и создание генератора ключей, но возможны и другие решения, например, создание загрузчика (см. выше).

# Реверс-инжиниринг виртуализированного и мутированного кода

1. Мутирование — это метод обфускации кода, при котором исходный граф потока управления разбивается дополнительными вершинами, ветвлениями, дополняется мусорными инструкциями, циклами, не нарушая исходного алгоритма программы. Часто исходные инструкции мутируются в некоторое подмножество других инструкций выполняющих одну и ту же работу.  
  
2. Виртуализация — это метод обфускации кода, при котором исходные инструкции алгоритма, транслируются в инструкции виртуальной машины, сгенерированной протектором. На место исходного алгоритма встраивается код, который во время выполнения передаёт промежуточные инструкции на вход в виртуальную машину, интерпретирующую их.  
  
Оба способа усложняют как статический так и динамический анализ исполняемого кода и часто протекторы допускают комбинирование способов.

##### 0x01. Преимущества и недостатки протекторов

Очевидным преимуществом мутации конечно же является невозможность визуального исследования алгоритма. Перед исследователем на первый взгляд лежит просто тарелка с горой спагетти, разобрать которую в ручную запредельно сложно, именно на это и делают ставки современные протекторы.

Недостатков у таких технологий тоже хватает. Первый из которых, это потеря производительности, ведь мутируемый код увеличивается в сотни, а то и в тысячи раз. Не меньше это касается и виртуализации, обычно виртуальная машина намного тяжелее мутируемого кода.

##### 0x02. Проблема неполноты защиты кода

Получая на вход бинарный файл, даже при наличии отладочной информации существует ряд ограничений, нарушение которых приведёт к неуниверсальности протектора, либо повреждению защищаемого приложения.

Исходное местоположение кода и данных не меняется, хотя перемещение некоторых других секторов возможно.