**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).

**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 или его отдельные алгоритмы.

## Технические средства защиты

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. 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.

* 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)

### Защита при помощи компакт-дисков

Программа может требовать оригинальный компакт-диск. Стойкость таких защит невелика, ввиду широкого набора инструментов снятия образов компакт-дисков.

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

### Защита при помощи электронных ключей

[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
* ключи аппаратных криптографических алгоритмов (используется наиболее часто)
* алгоритмы, созданные разработчиком программы

### 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).

Основной недостаток: если пользователь производит модернизацию компьютера (в случае привязки к [железу](https://ru.wikipedia.org/wiki/Аппаратное_обеспечение)), защита отказывает.

### 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.

### 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)

# 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)

## 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.
* 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
* 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.
* 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). Заключается в снятии дампа внутренней памяти ключа.
* Подмена официального сервера программ и/или соответствующее изменение настроек с целью обойти проверку ключа, если она была вынесена разработчиками на какой-либо интернет-ресурс

При взломе сложных защит, а также при необходимости достичь максимального эффекта, применяется комбинация вышеперечисленных способов.

## 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).

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.

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.

# 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.