IT-Колледж “Сириус”

ДОКЛАД

по дисциплине “Введение в специальность”
на тему “Взлом ПО”

Выполнил:
Студент группы
1.9.7.2
Бахтинов Владислав Владимирович

Принял:
Старший преподаватель
Тенигин Альберт Андреевич

_____

IT-Колледж “Сириус”
2022

Оглавления

История киберпреступников **2**

Виды взломщиков **5**

Методы взлома ПО **6**

Дзэн-крэкинг **7**

Вспомогательный софт **12**

Заключение **15**

Список литературы **16**

История киберпреступников

Рассвет хакеров начался с **1960-х** годов. Впервые движение хакеры создалось в Массачусетском технологическом институте. Эти люди решили упокоить свое любопытство и применить недавно приобретенные технические умения при изучении новых мэйнфреймов, создаваемые в стенах института [2].

В **1970-х** годах впервые были упомянуты телефонные *фрикеры*. Этот тип взломщиков занимались взламыванием региональных и международных сетей, в результате таких манипуляций они открыли для себя множество возможностей. Они могли звонить бесплатно и бесконечное кол-во раз. Первым из фрикеров являлся Джон Дрейпер. Этот человек неожиданно открыл явление, что простой свисток издает сигнал с частотой 2600 Гц. Точь в точь те же характеристики обеспечивал удаленный доступ к коммутирующим системам AT&T (AT&T Inc. — американский транснациональный телекоммуникационный конгломерат со штаб-квартирой в Далласе, штат Техас. Крупнейшая в мире телекоммуникационная компания и один из крупнейших медиа конгломератов) [2].

Он стал инженером и создателем аппарата, который совместно со свистком и телефоном дало ему возможность совершать бесплатные звонки. Вскоре в журнале Esquire была опубликована статья под наименованием «Секреты маленькой синей коробочки». В данной статье описывается рецепт изготовления этого аппарата. После выкладки число зафиксированных телефонных мошенничеств в США безумно выросло. На эту статью наткнулись даже Стив Возняк и Стив Джобс, они являются будущими основателями компании Apple. Двое этих парней создали свое производство и занимались продажей этих устройств [2].

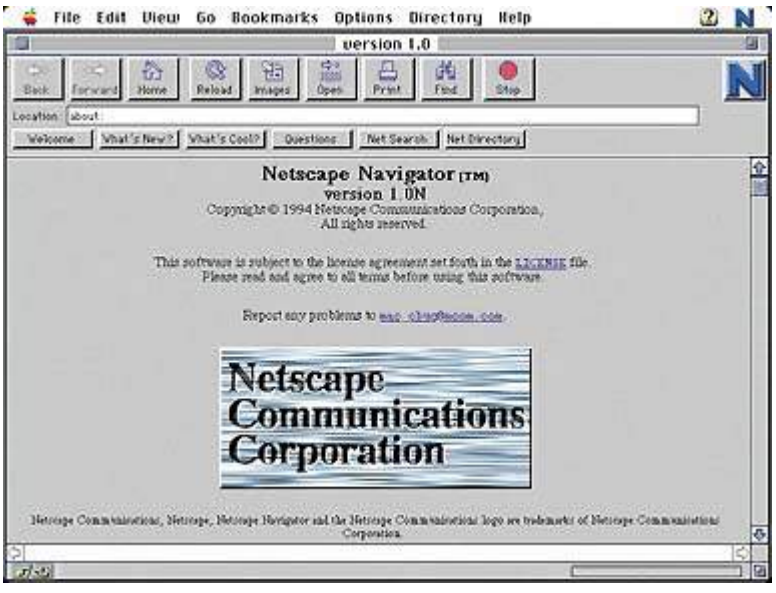
С **1980-го** года стали зарождаться первые хакерские объединения, а также доски объявления. Так как деятельность фрикеров стала уходить в

забвенье, всё стало переходить в сторону компьютерной техники. Стали создаваться первые электронные доски объявлений, такие как: Sherwood Forest и Catch-22. Они превратились в места встреч фрикеров и хакеров, которые обменивались там информацией, продавали друг другу ценные советы и также торговали украденными паролями и номерами кредитных карт [2].

В **1986-ом** году из-за учащения зарегистрированных обнаружений взломов компьютерных систем в госучреждениях и различных компаний правительство США вынесло решение ввести наказание, а именно приняли «Акт о компьютерном мошенничестве и злоупотреблениях». В этом акте говорится о том, что проникновение в компьютерную систему стало уголовно наказуемым [2].

В **1988-ом** году был создан первый в мире компьютерный вирус человеком по имени Роберт Моррис-младший. Он запустил в ARPAnet компьютерный вирус, который автоматизированно рассылал своих клонов по электронным каналам. Создатель хотел понять, как его вирус повлияет на компьютерные системы, которые работают на ОС Unix. Его вирус заразил более 6 тысяч ПК и вывел из строя важные федеральные системы [2].

1994-й Год стал одним из ключевых для киберпреступности. Хакерский софт был доступен каждому пользователю благодаря появлению интернета. Использование интернета началось с браузера Netscape Navigator. Его появление сделало сильно проще поиск и получение информации. Хакеры мгновенно перешли в новую среду и перенесли весь свой софт из электронных досок BBS на Web-сайты. После этого хакерское сообщество стало нести действительно высокую угрозу [2].



1999-й Год стал годом создания первых систем компьютерной безопасности. Компания Microsoft выпустила свою ОС Windows 98. Люди начали выкладывать различные публикации списка ошибок, обнаруженных в Windows и иных программах. Разработчики выдвигали множество способов устранения уязвимостей и обновленные варианты программного обеспечения. Очень много самых разных производителей выпустил инструменты защиты от хакеров специально для установки их на домашних компьютерах [2].

2001-й год ознаменовался самой крупной атакой хакеров за все года до 2001-го года. Компания Microsoft была атакована хакерами. Они нанесли многочисленные удары по серверам доменных имен (DNS). Итогом атак стало то, что люди не могли попасть на страницы Microsoft целых 2 дня. Взлом заметили через несколько часов, но уже было поздно [2].

Виды взломщиков

Люди занимающиеся взломом деляться на самые различные виды, группы и подгруппы, но я выделил 3 основных вида взломщиков:

Хакер - изначально этот термин означал того, кто исправлял ошибки в различном программном обеспечении стремительным и элегантным способом и не нес цели причинить вред кому-либо. Но сегодня этот термин разделился на два под термина, такие как «black hat», «white hat» [1].

Black hat - это хакер, злоумышленник, который взламывает системы с целью кражи информации, активирует DDoS-атаки и крадет номера кредитных карт [1].

White hat - это близкий к оригинальному значению термина «хакер» человек, который много знает в программирование и эксперт по безопасности, использующий свои таланты чтобы помогать повышать безопасность компьютерных систем и ловить преступников [1].

Крэкер - это человек, взламывающий системы защит программ для того, чтобы получить к ним доступ незаконным путем. Так же он создает так называемые “Кряки” для программ [1].

Фрикер - это человек, который производит взлом телефонных сетей. Они умеют взламывать электронные системы и получить доступ к деньгам и бесплатным звонкам с мобильных телефонов. Ради этого ими взламываются таксофоны, мобильные сети, телефоны [1].

Методы взлома ПО

Как известно существует множество методов взлома программного обеспечения, но я хотел бы выделить самые основные:

Первым я бы хотел выделить *обнуление триала* - по факту, данный метод не является взломома и довольно прост, так как не требует от пользователя особых махинаций. Этот способ позволяет полулегально продлить срок использования пробного периода программы. Как он работает ? Нужно найти место, где хранится дата первого запуска программы, далее мы должны либо изменить ее, либо же удалить. Таким образом мы получаем обнуление пробного периода программы и можем безвозмездно ей пользоваться [3].

Вторым методом я бы выделил написание *keygen'a* - данный метод является для разработчика самым вредоносным, но для пользователя он имеет только плюсы. Программа начинает считать себя полностью лицензионной и работает исправно, дальнейшего активирования, как у 1-го метода, не требуется.

Использование враппера является третьим методом - обычно разработчики программ проверяют лицензию единожды, и дальше используют полученные данные — валидна/невалидна (как вариант насколько валидна, если допускается несколько типов лицензии, отличающихся возможностями). Это можно обыграть в нашу сторону, использовав следующий алгоритм:

1. Сказать программе, что лицензия уже проверена
2. Сказать программе то, что лицензия корректна [3].

Но для того чтобы точно описать, как работает взлом именно программ, я возьму за основу 4-й способ, а именно *Дзен-крэкинг* [3].

Дзэн-крэкинг

Так как программа содержит в себе набор алгоритмов, существует 2 метода взлома программ, каждый из них отличается по своему основному принципу, крэкер пользуется одним из них в зависимости от ситуации. Большая часть крэкеров прибегают к дзен-крэкингу, он содержит в себе идею: «я не знаю, как это работает, но я могу это сломать». Иной метод предполагает углубленное изучение и анализ того, как программа функционирует [4].

Но, в обоих случаях обязательно нужно знать команды ассемблера, способы передачи параметров в функции, системные вызовы ОС, различных компиляторов. Без этих познаний изучение крэкинга является бесполезным занятием [4].

Так как для удачного кряка программы не во всех случаях необходимо понимание всех её внутренних функций, иногда можно обойтись очень поверхностным пониманием и анализом встроенной защиты. Первый способ позволит быстрее и эффективнее получить нужный нам результат. Второй потребует колоссальных затрат сил и времени. Конечно, когда происходит взлом вторым способом, количество ошибок и сбоев в программе в дальнейшем будет намного меньше, но разбирать каждую программу по мельчайшим деталям по силам лишь профессионалам высокого класса, к тому же для подобного анализа необходимо иметь опыт и знания, которых у начинающих нет. Тут возникает парадокс: для получения опыта нужно успешно взламывать программы, однако для удачного взлома необходим большой опыт. Именно по данной причине первый метод - исследовать программы, исходя из догадок которые строятся на наблюдении за внешними эффектами, которые производятся программой, очень распространен среди крэкеров. В этом случае крэкер не узнает, что и как происходит глубоко в коде программы, а сначала строит догадки, на что это может быть похоже и каким образом это может быть сделано, как бы он попытался добиться такого же эффекта на месте создателя. При использовании такого метода успех зависит не только от профессионализма, но и от богатого воображения крэкера. Поэтому в данном случае эффективность дзен-крэкинга очень зависит от внимательности и смелости догадок

крэкера потому, что надеяться на то, что разработчики средств защиты используют банальные и давно известные приемы, довольно глупо, нужно творчески подходить к процессу взлома [4].

Чтобы выдвигать гипотезы о работе программы, обязательным этапом будет тщательный сбор информации о ней. Необходимо узнать, упакована она или нет, каковы ограничения в незарегистрированной программе, как происходит процесс регистрации; также желательно узнать, что произойдет, если попытаться использовать программу дольше, чем это задумано ограничениями. Ещё следует посмотреть, что за текстовые строки и ресурсы содержатся внутри программы, проанализировать к каким файлам и ключам реестра программа обращается во время запуска. Будет очень полезно присмотреться к справочной системе программы - там можно обнаружить описание различий между зарегистрированной и незарегистрированной версиями. Имеется вероятность того, что вся эта информация не понадобится в процессе взлома, однако быть подготовленным к неожиданностям всегда лучше [5].

Представим, у нас есть программа и она выполняет определенное действие. Допустим, она перестает запускаться после 15 дней с момента её первого запуска и появляется окошко с предупреждением - это и есть первое наблюдение. Следует провести эксперимент, перевести системное время назад, если программа все равно не запускается, из этого становится ясно следующее - программа как-то проверяет текущую дату и при этом не зависит от показаний часов операционной системы. Можно предположить, что программа сделала пометку что запуск больше невозможен или же определяет время иным методом. Здесь путь делится на два: представить себя на месте создателя и поразмышлять, какими вариантами определения времени он мог воспользоваться. Стоит начать искать в системном реестре или в исполняемых файлах ссылку на запрет запуска, но сначала необходимо проследить за действиями программы, которые являются признаками той или иной реализации защиты. Если программа при запуске шуршит винчестером (если это дисковая версия), то вероятность того, что она совершает проверку файлов на дату и время создания намного больше. В худшем исходе нужно уже разбирать, какие условные переходы, связанные с этим окном, производятся и к каким функциям происходит обращение в данный момент времени [4].

Любые защитные функции имеют свои дыры и уязвимости, какой бы профессионал их не делал. Слабое место может быть спрятано в самых глубоких недрах кода - однако оно есть. Достаточно найти его и совершить точную атаку, тогда защита не выдержит и даст сбой. Из этого следует вывод, залог успешного взлома в отыскивании уязвимостей в защите [4].

Самыми подходящими для крэкеров уязвимости - это глобальные переменные. В этих переменных хранятся сведения о состоянии программы, например «активирована - не активирована». Функции возвращающие критичное для защиты значение (число запусков или дней до истечения испытательного срока, результат проверки серийного номера на правильность), и процедуры, выводящие сообщение об успешной или неуспешной попытке регистрации, а также об истечении триального срока программы. Иногда изменение 0 на 1 дает полный доступ к ПО и делает его зарегистрированным, но для этого нужно найти эту дыру, что и является первостепенной задачей крэкера. По этим причинам поиск констант (количество дней) и их изменение является наиболее успешным решением так как в этом случае происходит минимальное вмешательство в код программы [4][5].

Ещё одна дыра, сильно облегчающая жизнь крэкерам - это проблема условного перехода, заключающаяся в реализации проверки некоторого условия, не использовав напрямую команду условного перехода. В чем же отличия команд условных переходов? А в том, что каждый такой переход легко превратить в этот же, но с противоположным условием. Обычно для этого хватит всего-то исправить один бит. Несмотря на простоту, правка условных переходов наименее предпочтительна, чем модификация функций. Это обоснуется тем, что условных переходов, которые имеют отношение к защите, в программе может быть много (обычно сильно больше, нежели фрагментов кода, отвечающих за возврат результата функции), но их поиск требует особой внимательности [4][5].

Впрочем при правке условных переходов начинаются проблемы так, как не всегда следует верить в происходящее в программе после изменения данных. Одной из ошибок может быть неверная интерпретация собранной информации. Защитные функции могут быть прямо под носом у крэкера, но из-за того, что они встроены в функции, которые относятся к стандартным вызовам, их можно не увидеть. Изменение условного

перехода также не во всех случаях приводит к планируемому результату. Часто, даже если все признаки неактивированной версии отсутствуют, программа продолжает думать, что она не активирована. Это происходит из-за того, что защита имеет несколько скрытых слоев. Но при этом эффект может оказаться полностью противоположный - изменение условного перехода приводит к регистрации программы, но в этом случае все равно выводятся окна с оповещением о регистрации или продления срока действия программы [4].

Приемы взлома, написанные выше, используются почти во всех ситуациях, однако в некоторых случаях защита встроена в установщик, которым производится установка программы, так как большинство программ предоставляется в виде инсталляционного пакета. В нем, помимо файлов, которые нужно установить на пк пользователя, содержится сценарий инсталляции - скрипт, который описывает алгоритм инсталляции. Так как написать свой инсталлятор довольно сложно, для большей части небольших программ используют один из готовых продуктов. Инсталляционный скрипт описывает, каким методом устанавливается тот или иной файл (добавление, замена, замена с предварительной проверкой версии и т.п.), какие данные нужно внести в реестр или в файлы конфигурации, какие программы запускаются до, во время самой установки и после. Также параллельно с этим могут происходить проверка серийных номеров, создание записей в реестре и распаковка и запуск программ. Из-за этого защиту от запуска незарегистрированной или триальной со сроком действия версии следует искать в инсталляционном файле. Но реализация защиты может кардинально различаться, как от простых меток в реестре и до создания точки отсчета времени от системной библиотеки самой операционной системы. В виде метки с помощью которой программа будет проверять ограничения по времени или по количеству запусков, может использоваться какой-то файл, который создается в процессе установки, особенно если такой файл будет спрятан глубоко в системных директориях, но активно использоваться программой. Наилучший и надежный метод обнаружения таких функций защиты - это декомпиляция инсталляционного файла и разбор его по байту. Но стоит учитывать одно, такая операция требует много времени и большого опыта. Однако в случае, когда инсталляционная программа создана другим разработчиком, можно написать универсальный патч, который подходит для обхода защиты всех программ использующих такую же версию

инсталляционного пакета. Исходя из способа минимального сопротивления используется метод мониторинга системы во время процесса инсталляции. В таком случае, не сильно разбирая код файла, можно обнаружить запросы к файлам и создание ключей в реестре. Такие утилиты мониторинга за ОС дают возможность делать слепки системы на различных этапах ее установки, а к тому же методом сравнения исходных значений с полученными можно определить результаты после запуска инсталляционного пакета. Но нужно отметить, далеко не у всех инсталляционных пакетов имеются многочисленные возможности в защите, из-за этого довольно часто встречается такое решение - запуск самой программы после установки или дополнительных программ на фоне процессов, которая и создает защиту [4].

Обязательно нужно знать, инсталлятор вдобавок может иметь серийные ключи, алгоритм их создания или проверки, но только в том случае, если при установке обязательно нужно ввести ключ. Благодаря определенным манипуляциям с инсталляционным файлом можно раздобыть эту функцию, с помощью которой и защищается серийный ключ. Всего имеется три наиболее популярных способа проверки серийного ключа: сравнение с эталонными значениями, включенными в инсталлятор, проверка серийного ключа по его хэшу и сверка хэшей серийных ключей. Однако, в инсталляторе может быть множество номеров, это поможет нам сделать выполнение задачи проще. Следовательно, можно определить используемый метод в нашем случае и как это может нам помочь [4].

Конечно, я рассмотрел далеко не все способы защиты и пути их обхода.Стоит повториться, что для крэкинга нужно иметь довольно творческий склад ума и навык импровизации, но самое важное - это опыт. К любой программе нужно искать индивидуальный подход для обхода ее защиты. В этом русле дзен-крэкинг будет самым распространенным методом взлома программ.

Вспомогательный софт

Инструменты - один из козырей всех крэкеров, но без умения их использовать это бесполезные программы. Однако от того, насколько хороши и качественны выбранные инструменты, зависит скорость и эффективность действий. Софт должен поддерживаться и получать регулярные обновления, поскольку программы не стоят на месте и постоянно становятся безопаснее и устойчивее к взломам. Инструментов для крэкеров большое разнообразие, главное выбрать самые эффективные и мощные. Существует множество типов программ, но я расскажу об основных [5].

Отладчик - программа, заточенная под поиск ошибок в коде программ, ядрах операционных систем, SQL-запросах. Он предоставляет возможность выполнить пошаговую трассировку, изменить значения переменных во время процесса выполнения кода, устанавливать и удалять контрольные точки или условия остановки и т.д [6].

Дизассемблер - это программное обеспечение, которое преобразует инструкции машинного языка в инструкции языка ассемблера (также известный как обратный инжиниринг). Как следует из этого термина, дизассемблер выполняет операции, обратные операциям, выполняемым ассемблером [7].

Этими инструментами часто пользуются в паре, поскольку дизассемблер выдает в основном «чистый код», но новые дизассемблеры могут выделять локальные переменные в процедурах, распознавать вызовы стандартных функций и многое другое. Используя дизассемблер, можно только предположить, что за данные получает функция в качестве параметров и что они означают. Для того, чтобы это понять, требуется изучение большей части программы [5].

Декомпилятор - это преобразователь исполняемого или готового к запуску программного кода в форму более высокого языка программирования, которую людям легче понимать. Декомпиляция - это тип reverse engineering, который выполняет операции, противоположные

компилятору. Декомпиляция была впервые использована в 1960-х годах для облегчения переноса программы с одной платформы на другую [8].

API-шпионы и другие утилиты мониторинга - при крэкинге довольно часто нужно знать выполняемые действия той или иной программы, какие стандартные функции и с какими параметрами вызывает, откуда она читает и куда записывает данные. Для добычи этой информации на выручку приходят утилиты мониторинга и делятся они на две группы: те, который отслеживают факт появления определенных событий и позволяющие выявить несколько типов изменений, которые произошли в системе за определенный отрезок времени. К первой группе можно отнести различные API-шпионы, они перехватывают вызовы системных функций, популярными утилитами в этой группе будут: Reg, File, PortMon. Такие утилиты предоставляют множественную информацию об отслеживаемых событиях, но имеются и недостатки, они генерируют довольно большие и не интуитивно понятные для анализа логи, если отслеживаемые события происходят часто. Ко второй группе можно отнести всевозможные программы создающие снимки реестра, жесткого диска, системных файлов и т.д. Такие утилиты дают возможность сравнивать состояние ПК до и после какого-то события, выстроить список различий в этих состояниях [9].

Распаковщики и утилиты для дампинга процессов

Дизассемблировать запакованную или зашифрованную программу невозможно, но если весьма хочется заполучить даже какой-то листинг, то можно попробовать вытянуть из памяти компьютера дамп программы во время ее работы. Данный дамп уже можно довольно удачно дизассемблировать. Более того, на основе такого дампа возможно воссоздать выполняемый файл программы и такой файл будет благополучно загружаться и работать. Собственно этот принцип и лежит в основе работы большинства сегодняшних распаковщиков: испытуемая программа запускается под контролем распаковщика, далее распаковщик дожидается определенного события, знаменующего о том, что программа полностью распаковалась, здесь же производит остановку программы и сбрасывает ее дамп на диск. Защитные системы зачастую стараются мешать получению работоспособного дампа через манипуляций с сегментами и таблицами импорта-экспорта. В таких ситуациях требуется использовать PE-реконструкторы, другими словами программы,

обнаруживающие в дампе некорректные ссылки на функции и предпринимающие попытки их восстановить. Некоторые продвинутые дамперы и распаковщики обладают интегрированными средствами восстановления секций импорта [9].

Заключение

В ходе выполнения этой работы я ознакомился с множеством методов получения доступа к программе, не приобретая лицензии. Также смог распределить киберпреступников на различные типы, каждый из которых преследует свои цели и своими способами достигает их. Если обобщить, то взломщики делятся на три основных типа: для кого взлом является просто увлечением и занимается этим для самообразования, преступников, которые стремятся заполучить какие-либо блага с помощью взлома, а также люди, которые специально взламывают программы и сообщают об ошибках разработчикам за определенную плату, или же напрямую работают в компании. Крэкинг и т.д. - это само по себе противоречивое занятие, с одной стороны, всё это является незаконной деятельностью, которая может нести вред как компаниям, так и ни в чем невинным людям. Однако это способствует развитию систем защиты и обороны от киберпреступности, программы становятся сильнее в этом плане и взломщикам с каждым годом всё труднее подвергать их взлому, из-за чего порог вхождения в эту деятельность выше, следовательно меньше людей стремится этим заниматься. Но к сожалению, создать идеальную систему защиты просто невозможно, абсолютно любая программа рано или поздно найдет того, кто подвергнет её удачному взлому, так что это можно назвать бесконечной войной, которая вряд ли когда либо прекратится.

Список литературы

1.Хакинг. Искусство эксплойта. 2-е издание [2018] Джон Эриксон
2.https://www.osp.ru/cw/2001/28-29/42777
3.Компьютер глазами хакера [2012] Михаил Фленов
4.Введение в крэкинг с помощью OllyDbg
5.Введение в reverse engineering для начинающих [Денис Юричев]
6.ЛитРес: Самиздат Отладчик (Debugging) в SAP ERP (S/4HANA) для блондинок
7.«Ассемблер и дизассемблирование» [Владислав Пирогов]
8.Инструментальная среда восстановления исходного кода программы – декомпилятор TyDec Е. Н. Трошина
9.Искусство дизассемблирования (Крис Касперски)