Министерство образования и науки Российской Федерации

Санкт-Петербургский Политехнический Университет Петра Великого

—

Институт компьютерных наук и кибербезопасности

ЛАБОРАТОРНАЯ РАБОТА № 2

«Изучение методов исследования исполняемых файлов с элементами самозащиты»

1. по дисциплине «Технологии реверс-инжиниринга программного обеспечения»
2. Выполнил
3. студент гр. 5151004/10101 Веремейчук Я.Ю.

<подпись>

1. Преподаватель
2. асс. преподавателя Овасапян Т.Д.

<подпись>

1. Санкт-Петербург
2. 2023
3. Цель работы

Изучение методов защиты, применяемых во вредоносном программном обеспечении, и способов их обхода. Изучение методов анализа бинарного кода.

1. Ход работы

Существует множество подходов и методов, которые могут быть применены для защиты программного обеспечения. Например, использование алгоритмов шифрования для защиты данных в программном обеспечении, чтобы предотвратить несанкционированный доступ и перехват информации; внедрение механизмов аутентификации для проверки подлинности пользователей и установления их прав доступа к системе; написание скриптов и плагинов к средствам дизассемблерования и отладки, использование средств инструментации исполняемого кода, использование технологии символьного выполнения.

Перед началом работы был получен исполняемый файл crackme3.exe.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1 – Попытка запустить crackme3.exe

Далее необходимо перейти в IDA Pro и открыть там экзешник. Затем нажать Ctrl+E и выбрать TLS Callback, поставить точку останова, начать отладку и с помощью F8 двигаться по коду.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 2 – Выбор функции для начала отладки

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 3 – Запуск отладки

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Рисунок 4 – Пример ошибки при выполнении кода без патчинга

Необходимо обходить все исключения: нопить деления на 0, нопить tls callback, изменять функции прыжков. И нужно дойти до места вывода секретного кода. В коде были обнаружены следующие методы защиты:

* Функция RaiseException;
* Деление на 0;
* IsDebuggerPresent, CheckRemoteDebuggerPresent, NtQueryInformationProcess (антиотладчик)

Для удобства нужно включить отображение байтов.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 5 – Отображение байтов

При отладке был замечен первый call с подозрительным названием, при выполнении которого добиться успеха не удавалось.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 6 – RaiseExeption

Эту команду необходимо занопить.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 7 – Подготовка к замене байтов

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 8 – Замена байтов

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 9 – Результат замены байтов

Далее была замечена команда jmp, которая также приводила к тупиковому результату. Необходимо заменить адрес функции.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 10 – Изменение инструкции

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 11 – Изменение инструкции

Далее было замечено деление на ноль. Функцию div также необходимо занопить.

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 12 – Команда div, приводящая к ошибке

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 13 – Результат патчинга

Далее отладка была запущена с помощью зеленой стрелки запуска и было найдено место ввода пароля. Но команду mov необходимо занопить.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 14 – Ввод пароля

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 15 – Результат патчинга команды mov

При предупреждениях от IDA необходимо нажимать No.

Изображение выглядит как текст, снимок экрана, Шрифт, веб-страница

Автоматически созданное описание

Рисунок 16 – Предупреждение

Далее встретились еще деления на ноль, а также прыжки, приводящие к зацикливанию. У прыжков следует заменить условный на безусловный переход.

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 17 – Деление на ноль

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 18 – Деление на ноль

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 19 – Прыжок

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 20 – Измененный прыжок

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 21 – Еще один прыжок

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 22 – Еще один прыжок

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 23 – Еще один прыжок

Затем было найдено место, где выводится код: ": %d%d\n". Команда test сравнивает регистры с паролями. На нее необходимо поставить точку останова, чтобы быстро проскочить выполнение call’ов, и запустить отладку зеленой стрелочкой. Также стоит обратить внимание на команду jz.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 24 – Место вывода кода

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 25 – Запуск отладки

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 26 – Прыжок

Затем снова можно продолжить отладку зеленой стрелочкой, предварительно поставив еще одну точку останова. Выводится код.

Изображение выглядит как текст, программное обеспечение, Шрифт, веб-страница

Автоматически созданное описание

Рисунок 27 – Вывод секретного кода

Затем был разработан скрипт на Python, находящий механизмы самозащиты в программе и обходящий их. Обход осуществляется путем замены байтов инструкций на nop.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 28 – Результат работы скрипта

Для получения входного значения программы была использована среда символьного выполнения IDA Pro – ponce. Для использования плагина его необходимо скачать: добавить в папку plugins соответствующие dll (https://github.com/illera88/Ponce/releases).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 29 – Установка плагина

После перезагрузки IDA Pro появляется окно конфигурации символьного выполнения.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 30 – Окно конфигурации символьного выполнения

Сначала необходимо найти регистр, в котором хранится введенный пароль, затем отметить необходимое количество символов. В исследуемом примере необходимое количество символов – 37, данное число получено путем исследования функции проверки пароля (число итераций цикла в псевдокоде). Если нажать F5 на функции проверки пароля, которую вызывает call, можно увидеть псевдокод алгоритма проверки.

Изображение выглядит как текст, снимок экрана, Шрифт, дисплей

Автоматически созданное описание

Рисунок 31 – Псевдокод алгоритма проверки

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 32 – Регистр ecx хранит введенный пользователем пароль

Дважды нажав на выделенный ecx, необходимо нажать на db и выбрать посимвольное выполнение для регистра. Этим действием ставится “бряк” на область памяти, в которой будут происходить изменения. На этом участке памяти программа меняет каждый символ на нужный. Формально программа запускается на всех допустимых тестовых значениях (https://migalin.ru/blog/ponce).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 33 – Место начала хранения введенного значения

Изображение выглядит как текст, Шрифт, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 34 – Выбор для ponce

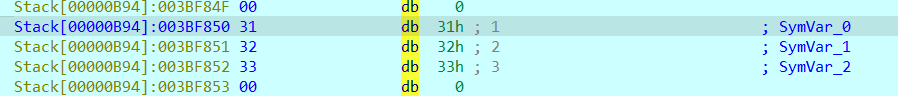


Рисунок 35 – Результат (+37 ниже)

Далее необходимо вернуться, нажав синюю стрелочку, сделать один шаг F8, и дважды нажать на функцию, которая вызывает call. Далее нужно поставить точку останова и продолжить отладку зеленой стрелочкой.

Изображение выглядит как текст, Шрифт, число, программное обеспечение

Автоматически созданное описание

Рисунок 36 – Этап точки останова

Далее необходимо дойти до цикла предсказания пароля. Чтобы получить символ, необходимо на команде в этом цикле нажать на «Negative…» на этапе джампа и пройти по циклу, пока не получится пароль. Negative значит, что мы отрицаем ветку, по которой программа намеренна пойти. Inject – подбор ponce’ом символа, который должен быть для прохода по другой ветке.

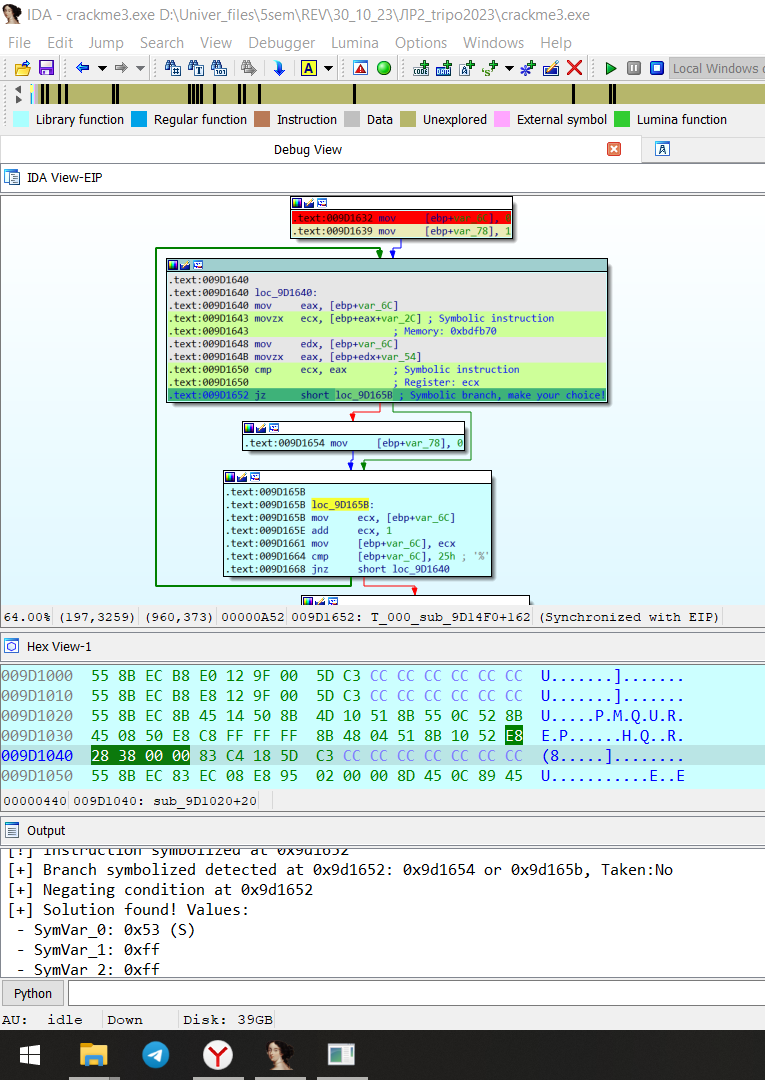


Рисунок 37 – Получение первого символа

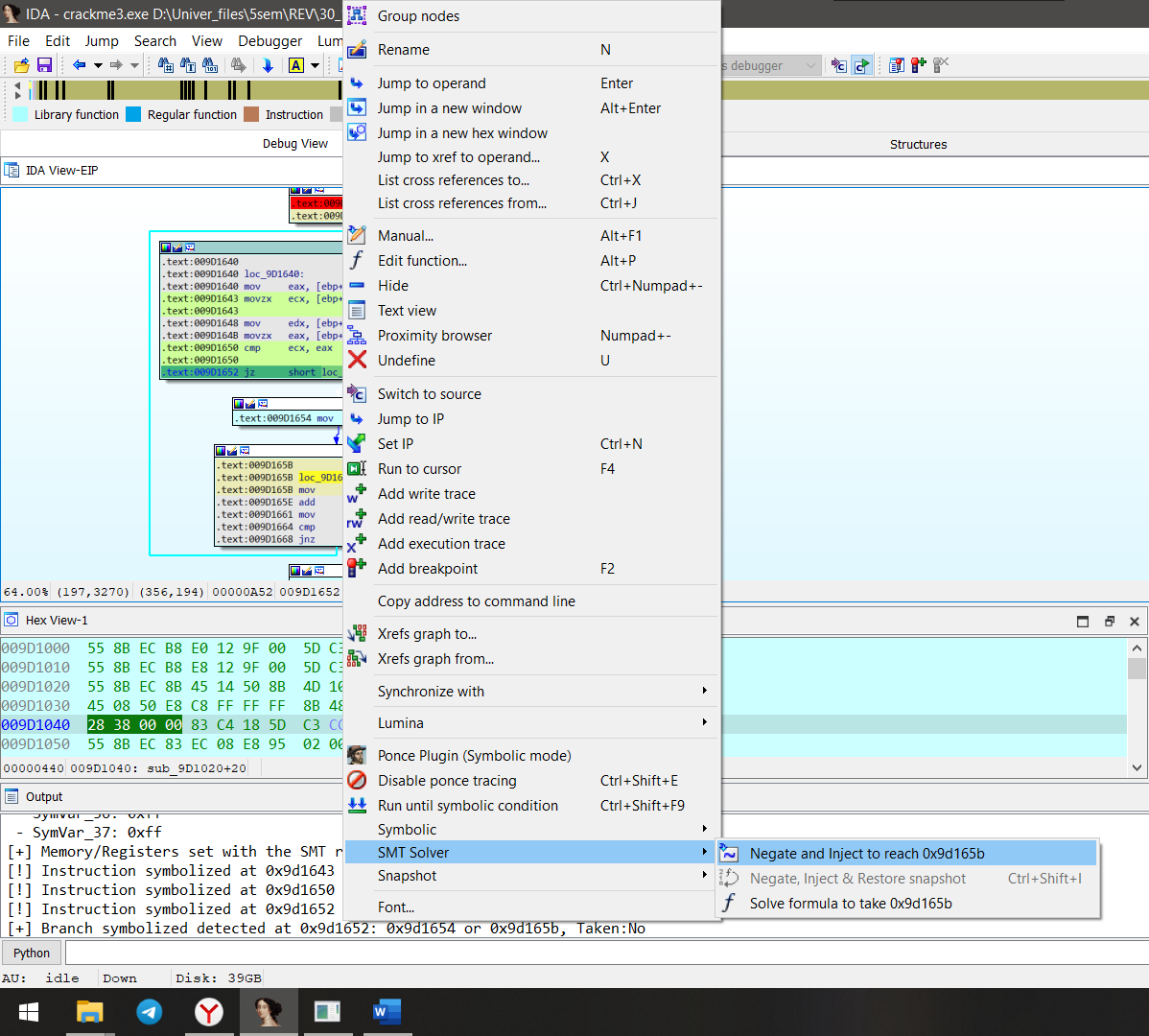


Рисунок 38 – Необходимая команда для предсказания

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 39 – Получение второго символа

Итоговый пароль: SuperSe`retBackdnnrP`rsvnrd0123456789d

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 40 – Успешный ввод пароля

Далее был рассмотрен второй бинарный файл.

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Рисунок 41 – Стартовое окно

Аналогично тому, как происходил патчинг в предыдущем файле, было найдено успешное завершение программы.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 42 – Место вывода удачи или фэйла

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 43 – Успешное завершение программы

1. Выводы

В ходе работы были изучены методы защиты, применяемые во вредоносном программном обеспечении, и способы их обхода, а также методы анализа бинарного кода.