Защита программного обеспечения во время выполнения

Отчет по практике
студента 4 курса 451 группы Д. А. Калентьева

Саратовский государственный университет
Им. Н.Г. Чернышевского

Кафедра математической кибернетики
и компьютерных наук

Научный руководитель: к. т. н., доцент
Петров Д. Ю.

2020 г.

Актуальность исследования

Риск копирования, реверсинжиниринга и манипуляции программными решениями ставит под угрозу дальнейшее существование производителей программного обеспечения (ПО). Защита ПО с помощью распознания его модификации является одним из наиболее эффективных методов криптографической защиты программного обеспечения во время выполнения.

Цель и задачи

Так, цель практики заключается в создании электронного ключа и защиты программного обеспечения во время выполнения.

Для достижения цели исследования необходимо решение ряда задач, а именно:

1. Рассмотреть теоретико-методологические аспекты разработки криптографического метода защиты программного обеспечения;
2. Продемонстрировать ход создания электронного ключа с целью защиты программного обеспечения во время выполнения.

Теоретико-методологические основы разработки криптографического метода защиты программного обеспечения

Изучение теоретико-методологических основ разработки крипто графического метода защиты программного обеспечения показало, что одним из наиболее эффективных методов является создание электронного ключа, позволяющего установить программное обеспечение, а также обнаружение модификации ПО. В случае неудачной проверки программное обеспечение не устанавливается или работает в демонстрационном режиме с ограниченными функциями.

Реализуем методы защиты

Процедура закрытия IDA Pro:

```
bool ida_closing_procedure()
{
```

Проверяем, созданы ли мьютексы отладчика, если созданы, то возвращаем соответствующий результат:

```
auto trusted_mtx = OpenMutexA(MUTEX_ALL_ACCESS, FALSE, xor_string("$ IDA trusted_idbs"));

if (!trusted_mtx)
    trusted_mtx = CreateMutexA(nullptr, FALSE, xor_string("$ IDA trusted_idbs"));
else
{
CloseHandle(trusted_mtx);
return false;
}

auto registry_mtx = OpenMutexA(MUTEX_ALL_ACCESS, FALSE, xor_string("$ IDA registry mutex $"));

if (!registry_mtx)
    registry_mtx = CreateMutexA(nullptr, FALSE, xor_string("$ IDA registry mutex $"));
else
{
    CloseHandle(registry_mtx);
    return false;
}

if (!trusted_mtx || !registry_mtx)
return false;
```

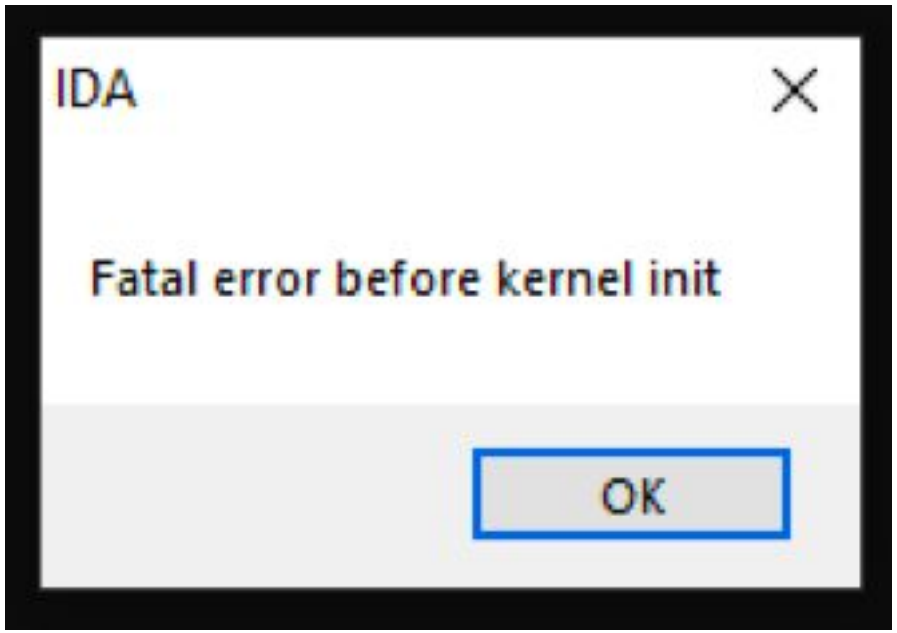
Реализуем методы защиты

Если не созданы, то мы их создаем и отправляем сигнал с ожиданием на наш процесс:

```
WaitForSingleObject(trusted_mtx, INFINITE);

WaitForSingleObject(registry_mtx, INFINITE);
```

В результате, когда отладчик пытается дождаться завершения этих мьютексов, функция WaitForSingleObject возвращается с ошибкой и IDA Pro выдает ошибку и закрывается:



Реализуем методы защиты

```
// Создаем дубликат памяти и накрываем его ключем xor
bool create_duplicate_memory(std::vector<execute_section>& execute_sections, PUCHAR key_ptr, UCHAR key_size)
{
    // Получаем адрес текущего модуля
    auto cur_module = GetModuleHandle(nullptr);

    if (!cur_module)
        return false;

    // Получаем указатель на nt headers нашего модуля
    auto nt_headers = get_nt_headers(cur_module);

    if (!nt_headers)
        return false;

    // Получаем указатель на первую секцию
    auto first_section = IMAGE_FIRST_SECTION(nt_headers);

    if (!first_section)
        return false;
```

Реализуем методы защиты

```
// Добавляем в вектор только секции в которых исполняется код
for (auto i = 0; i < nt_headers->FileHeader.NumberOfSections; i++, first_section++)
{
    if (first_section->Characteristics & IMAGE_SCN_MEM_EXECUTE)
    {
        // Выделяем память под дубликат исполняемой памяти
        auto alloc_memory = (PUCHAR)VirtualAlloc(nullptr, first_section->SizeOfRawData, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);

        // Проверяем удалось ли выделить память
        if (!alloc_memory)
        {
            execute_sections.clear();
            return false;
        }

        // Указатель на оригинальную секцию
        auto cur_memory = (PUCHAR)(PUCHAR(cur_module) + first_section->VirtualAddress);

        // Записываем дубликат и накрываем его ключем xor
        for (DWORD j = 0; j < first_section->SizeOfRawData; j++)
            alloc_memory[j] = (cur_memory[j] ^ key_ptr[j % key_size]);

        // Устанавливаем дубликат памяти только на чтение
        DWORD old_protect;
        VirtualProtect(alloc_memory, first_section->SizeOfRawData, PAGE_READONLY, &old_protect);

        // Добавляем данные в вектор для дальнейшей проверки
        execute_sections.push_back(
            {
                (PUCHAR)(PUCHAR(cur_module) + first_section->VirtualAddress),
                alloc_memory,
                first_section->SizeOfRawData
            }
        );
    }
}
return true;
}
```

Реализуем методы защиты

```
// Проверяем установлен бряклоинт или нет
bool is_hardware_breakpoint()
{
    CONTEXT ctx;
    memset(&ctx, 0, sizeof(ctx));

    ctx.ContextFlags = CONTEXT_ALL;

    // Проверяем удалось ли получить данные из контекста потока
    if (!GetThreadContext(GetCurrentThread(), &ctx))
        return true;

    // Проверяем занят ли какой либо из отладочных регистров регистров
    if (ctx.Dr0 || ctx.Dr1 || ctx.Dr2 || ctx.Dr3)
        return true;

    return false;
}
```

Заключение

Иные методы

Обычный электронный ключ, позволяющий запустить ПО можно обнаружить с помощью реверс-инжиниринга

Такой механизм неудобен для пользователей, поскольку изменения в аппаратном обеспечении или в операционной системе могут привести к необходимости получения новой лицензии и переустановки программного обеспечения

Создание электронного ключа и проверка модификации ПО

Список использованных источников

1. Барабанов А. В. и др. Статистика выявления уязвимостей программного обеспечения при проведении сертификационных испытаний // Вопросы кибербезопасности. – 2017. – №. 2 (20). – С. 64-77.
2. Белова Е. В., Белов В. Н. USB ключ как один из способ защиты информации // Актуальные вопросы в науке и практике. – 2018. – С. 45-53.
3. Бутин А. А. Методические аспекты разработки систем защиты программного обеспечения // Вестник науки и образования. – 2018. – №. 16-1 (52). – С. 117-124.
4. Бутин А. А. Технологии защиты программного обеспечения // Информационные технологии и математическое моделирование в управлении сложными системами. – 2019. – №. 2. – С. 53.
5. Давыдов В. В., Гребенюк Д. С. Комплекс процедур генерации лицензионного ключа для защиты авторских прав интеллектуальной собственности на программное обеспечение // Системи управління, навігації та зв'язку. – 2017. – №. 1. – С. 11-15.
6. Емельянова Н. Ю. Анализ электронных ключей для защиты программного обеспечения от несанкционированного копирования // Наука в современном обществе: закономерности и тенденции развития. – 2016. – С. 49-50.
7. Прокопенко В. В. Защита программного обеспечения от нелегального использования // Экономические и правовые аспекты развития международной интеграции в современных условиях. – 2017. – С. 235.
8. Селедец И. Е. Защита от взлома программных средств // Синергия Наук. – 2017. – №. 13. – С. 615-625.
9. Татаринов А. А., Болдырихин Н. В. Анализ методов обнаружения вредоносного программного обеспечения на основе поведенческих признаков // Национальная безопасность России: актуальные аспекты. – 2020. – С. 18-22.
10. Хлестова Д. Р., Редников Д. В. Электронные ключи-средство для защиты информации // Инновационное развитие. – 2017. – №. 6. – С. 22-23.
11. Цыцура К. С., Потоловский А. И. Технология защиты от несанкционированного использования программного обеспечения «Электронный ключ» // Актуальные проблемы авиации и космонавтики. – 2016. – Т. 1. – №. 12. – С. 89-94.
12. Rasch A., Wenzel T. The impact of piracy on prominent and non-prominent software developers // Telecommunications Policy. – 2015. – Т. 39. – №. 8. – P. 735-744.
13. Sander, T.; Tschudin C.F. On Software Protection via Function Hiding. Proceedings of Information Hiding ’98. Springer-Verlag. LNCS 1525. – P.111-123.
14. Schaumüller-Bichl1, I.; Piller, E. A Method of Software Protection Based on the Use of Smart Cards and Cryptographic Techniques. Proceedings of Eurocrypt’84. Springer-Verlag. LNCS 0209. – P. 446-454.
15. Shaikh S. A., Londhe B. R. Intricacies of software protection: a techno-legal review. – 2016. – P. 157-165.

Спасибо за внимание!