Внимание! Данное пособие не является самодостаточным для прохождения курса и является лишь вспомогательным материалом. Важные элементы заданий, теоретического материала и техническая информация могут изменяться из года в год и не отражаться в данном пособии. Наиболее полная и актуальная информация для выполнения лабораторных работ выдаётся на очных занятиях и консультациях.

Методы и средства криптографической защиты информации

Методические указания для выполнения лабораторных работ

Автор: Недогарок А.А.,
старший преподаватель, кафедра “Информационная безопасность”, факультет информационных технологий, Московский Политехнический университет

Успеваемость

Таблица успеваемости (весна 2024) https://disk.yandex.ru/i/L3XnbKUGPtthgA.

Ссылка для видеоконференции (задержка меньше, чем в Teams) https://meet.jit.si/t80bv.

Шаблон титульного листа лабораторной работы https://disk.yandex.ru/i/ynIlW2RirNLTcg.

Шаблон названия:

- файла отчёта “LR4_181_331_Lastname.docx” (подставить свой номер группы, фамилию или фамилию с инициалами)

- репозитория “181_331_Lastname” (подставить свой номер группы, фамилию или фамилию с инициалами)

Лабораторные работы

Общие требования к оформлению и защите лабораторных работ

1. Если в задании лабораторной работы предусмотрена разработка приложения
    a. Учащийся должен самостоятельно разработать и защитить приложение, требуемое по условиям лабораторной работы.
    b. Учащийся должен обязательно продемонстрировать на своей или лабораторной машине, что оно собирается, запускается и работает без ошибок.
    c. Исходный код должен быть оформлен в едином стиле, способствующем лёгкой читаемости:
        i. переменные, функции, классы и прочие элементы исходного кода имеют осмысленное название, раскрывающее их назначение и смысл.
        ii. каждый завершённый по смыслу блок кода имеет свой отступ, комментарий (как минимум функции и их параметры).
    d. (с 2024) К защите практической части работы (вопросы к исходному коду) принимается только исходный код, в котором нет комментариев кроме комментариев при обхъявлнеии функций/классов. Код, в котором прокомментирована каждая или почти каждая строка считается шпаргалкой и не принимается на защиту.
    e. Весь каталог и репозиторий должен иметь название по шаблону “201-331_Ivanov”, где 201-331 - номер группы, Ivanov - фамилия учащегося латиницей.
    f. Исходный код каждой отдельной лабораторной работы должен располагаться в отдельном каталоге файловой системы и отдельном проекте IDE с соответствующим

однообразным названием латиницей, например, “Lab1”...“Lab6” или “LR1”...“LR6”, или подобным на усмотрение студента.

g. Файлы исходного кода, соответствующие модулям или отдельным классам, должны иметь название по смыслу модуля или название, повторяющее название единственного класса.
h. Выполнение каждой лабораторной работы сопровождается созданием версий в локальном (рекомендуется git, но допускается использовать любую популярную в разработке ПО систему контроля версий: SVN, Mercurial) и удалённом репозитории (рекомендуется закрытый репозиторий https://github.com/ или https://gitlab.com/).
    i. Репозитории, ведение которых начато студентом (создана первая версия файлов) позже, чем за 1,5 месяца до окончания семестра, не принимаются.
    ii. Сохранение версий каждой лабораторной работы ведётся в отдельной ветке, в основную ветку main ветки законченных лабораторных работ сливаются (merge) после защиты.
    iii. Репозитории должны содержать все исходные файлы, необходимые для сборки лабораторных приложений из репозитория на сторонней машине: файлы исходного кода, иконки, изображения, прочие файлы ресурсов и т.п.
    iv. Репозитории не должны содержать автоматически генерируемые при каждой сборке файлы *.ilk, *.pdb, moc_*.cpp, ui_*.h, Makefile, *.stash, *.obj; исполняемые модули и библиотеки, собираемые непосредственно в проекте (*.apk, *.exe и т.п.).
    v. Приложение должно собираться и запускаться из клона репозитория без ошибок на сторонней машине. Для этого, в частности, не следует использовать абсолютные пути к файлам в коде приложения.

vi. В Readme.md одним абзацем должно быть описано назначение репозитория (или данного каталога), снимок работы кода (окно приложения, либо распечатка вывода приложения в консоли, либо другие отдельно оговариваемые иллюстрации).

i. Для защиты лабораторной работы и выставления положенных за её выполнение баллов учащийся должен
   i. Выполнить все требования по оформлению, описанные в данном разделе и особые требования по оформлению, если они указаны в задании к конкретной лабораторной работе.
   ii. Ответить на вопросы по практической части работы (назначение любой указанной преподавателем строки кода, переменной, функции и прочих сущностей, либо найти те или иные блоки кода, выполняющие указанные преподавателем функции),
   iii. А также ответить на контрольные вопросы к теоретической составляющей ЛР (список вопросов для подготовки приведён в описании каждой ЛР, а при их отсутствии уточняйте у преподавателя).

Преподаватель вправе изменять, корректировать и добавлять данные требования и критерии в случае рациональной необходимости.

График защит лабораторных работ

Количество баллов за лабораторные работы:

- сданные с отставанием до 1 недели включительно, снижается на 3 рейтинговых балла.
- с отставанием - до 2 недель включительно, снижается на 5 рейтинговых баллов.
- с большим отставанием - снижается на 7 рейтинговых баллов.

№ л.р.	**1**	**2**	**3**	**4**	**5**	**6**
Пройдена и	5.03.	16.04.				

№ л.р.	1	2	3	4	5	6
выдана	24	2024				
Срок защиты практической части	19.03.24 (включительно)	17.05.2024				

ЛР1. Защита автоматизированной системы на пользовательском уровне привилегий. Защита приложения Windows от утечки данных, отладки и модификации

Цель

Получение навыков реализации технических методов защиты в ПО пользовательского уровня

Задачи

1. Разработать приложение для персонального компьютера или мобильного устройства для безопасного хранения массива записей учетных данных.
2. Реализовать в приложении защиту
    a. от кражи учётных данных в файловой системе
    b. от кражи учётных в виртуальной памяти,
    c. атак с использованием отладки на пользовательском уровне
    d. атак с использованием модификации исполняемого модуля приложения.

Необходимый теоретический материал

1. Основы устройства исполняемых файлов Windows формата PE https://learn.microsoft.com/en-us/windows/win32/debug/pe-format
    a. Заголовки, сегменты

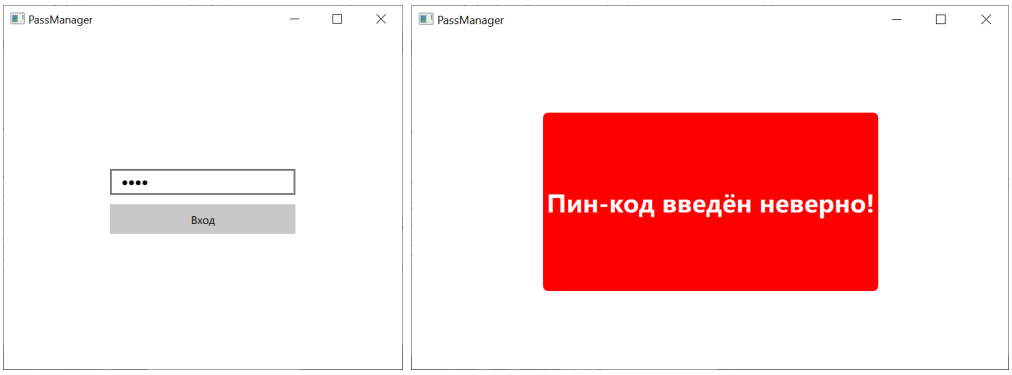
b. Сегмент .text
c. Image base
d. Особенности организации адресного пространства при переносе PE из файла в виртуальную память

2. Классификация возможностей нарушителя по “Методике оценки угроз безопасности информации” ФСТЭК (приложение 8).
3. Обзор нескольких технических реализаций атак.

Последовательность выполнения

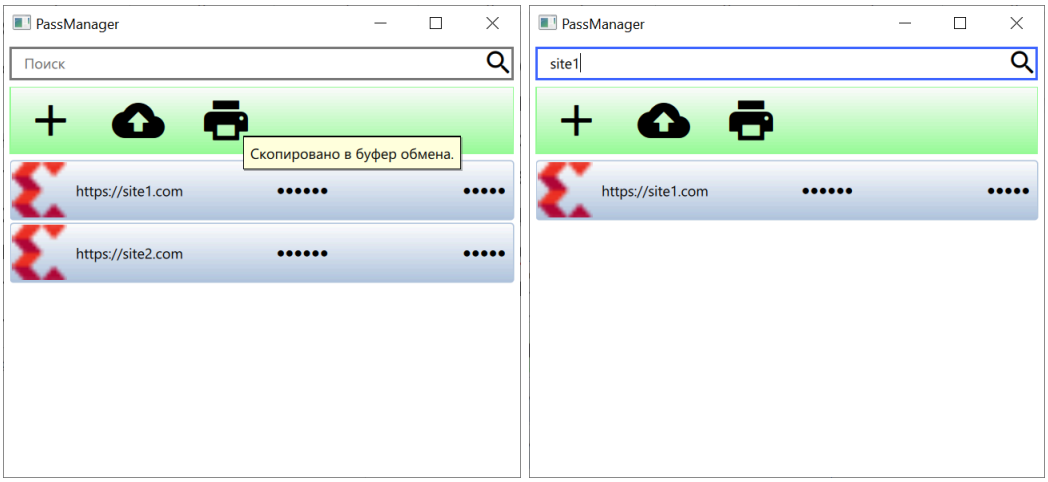
Этап 1. Разработка базового незащищённого приложения. Для выполнения данного задания подходят языки программирования C++/Qt и C#.

1. Реализовать окно (либо панель) аутентификации. Оно должно содержать поле ввода пин-кода для разблокировки хранилища учётных данных. При вводе верного пароля происходит
   a. переключение на следующее окно, отображающее список учётных данных
   b. чтение файла учётных данных, файл должен содержать не менее 5 учётных данных и иметь объем не менее 2 кб,
   c. заполнение списка (или таблицы) учётных данных маскированными символами. Допускается верный пин-код на данном этапе задавать в исходном коде константой в незащищённом виде.



2. Реализовать окно (панель) отображения учётных данных. Содержит

a. поле фильтрации по названию сайта
b. таблицу или список учётных данных; первая колонка содержит url сайта, для которого сохранены данные, вторая и третья - логин и пароль, отображаемый по умолчанию маскирующими символами.
c. ~~кнопки для добавления и удаления учётной записи (не нужно в 2024).~~



При вводе в поле поиска строки в списке учётных данных должны оставаться только те учётные записи, название сайта которых (или иных тегов, предложенных учащимся) включает введённый поисковый запрос.

3. ~~При выходе из приложения учётные данные должны сохраняться в незашифрованном виде в формате JSON.~~ При запуске приложения и успешном вводе пин-кода файл считывается, происходит парсинг и формирование структуры данных, хранящей учётные записи. Вид структуры для хранения учётных данных и прочие нюансы выполнения задания вырабатываются на групповом обсуждении архитектуры приложения на очных занятиях.

Этап 2. Реализация защиты приложения

1. Реализовать защиту от кражи учётных данных из файла. Файл должен быть зашифрован по алгоритму AES-256. Ключ шифрования должен быть получен на основе пин кода с помощью одной из современных криптостойких функций

хеширования (например, SHA-256). Приложение при вводе верного пин-кода должно расшифровывать файл в структуру данных в виртуальной памяти. Расшифрованные данные не должны ни при каких условиях оказываться в файловой системе. Пример использования библиотеки OpenSSL для расшифровки файла, см. https://www.openssl.org/docs/manmaster/man3/EVP_EncryptUpdate.html, первый (из двух) пример функции do_crypt() в конце страницы. Для получения зашифрованного JSON-файла вне приложения можно использовать любое онлайн или оффлайн приложение, поддерживающее 256-битное шифрование AES в режиме CBC, к примеру https://cryptii.com/pipes/aes-encryption.

2. Реализовать защиту от компрометации учётных данных в дампе оперативной памяти, добавив второй слой шифрования. После расшифровки файла доступен массив структур, в которых хранится url сайта в открытом виде, а логин и пароль зашифрованы вторым слоем шифрования. Таким образом после расшифровки файла приложение может выстроить список и производить поиск по адресу сайта, но логины и пароли продолжают храниться в зашифрованном виде. Это позволит защитить учётные данные даже если злоумышленник получит доступ к виртуальной памяти или её дампу. Отдельные логины и пароли расшифровываются для использования только после их ручного выделения пользователем в списке и повторного ввода пин-кода.
3. ~~TODO Проверка и хранение пин-кода~~

4. ~~Реализовать защиту от отладки на основе использования функции API Windows IsDebuggerPresent(): при запуске приложение вызывает функцию IsDebuggerPresent() и, при обнаружении отладчика отображает пользователю предупреждение и прекращает работу (до ввода пин-кода).~~
5. ~~Реализовать защиту от отладки на основе метода т.н. “Самоотладки”:~~

a. Создать приложение-спутник, которое будет подключаться к менеджеру паролей как отладчик с помощью функции DebugActiveProcess().
b. Реализовать старт менеджера паролей из приложения-спутника в отдельном процессе, и подключение к нему спутника как отладчика, с обработкой сообщений отладки (WinAPI функции WaitForDebugEvent() и ContinueDebugEvent(), пример в документации https://learn.microsoft.com/en-us/windows/win32/debug/writing-the-debugger-s-main-loop).

Данный элемент задания можно технически реализовать только на C/C++. Также следует учитывать, что данный метод защиты надёжнее работает в компиляторах MSVC: при изменении константы хеша в исходном коде хеш-сумма сегмента ".text" чаще остаётся неизменной.

Для защиты данного элемента задания учащимся необходимо попытаться подключиться к защищаемому менеджеру паролей с помощью стороннего отладчика (x64dbg) и пронаблюдать эффект.

6. Реализовать защиту от модификации (патча) приложения с помощью самопроверки контрольной суммы.
   a. Определить виртуальный адрес начала сегмента .text.
   b. Определить размер сегмента .text.
   c. Вычислить контрольную сумму блока данных, расположенного в сегменте .text.
   d. Реализовать сравнение эталонной контрольной суммы и суммы, подсчитываемой каждый раз при запуске приложения, а также отображение предупреждения, если контрольные суммы не совпадают.

Дополнительные задания (выполняются только по согласованию с преподавателем индивидуально)

1. [количество баллов зависит от сложности реализации, за комментариями обращаться к преподавателю] Реализовать в приложении модуль для распознавания владельца по лицу и

разблокировки учётных данных по биометрическим параметрам (а именно, по изображению лица с камеры, см. по ключевому запросу “person recognition”). Для выполнения данного задания не допускается использовать готовые встроенные в ОС реализации для аутентификации по лицу (например, Windows Hello или аналогичные из мобильных ОС) и сторонние приложения/сервисы. Допускается использовать сторонние библиотеки. Базу для авторизации по лицу необходимо подготовить/обучить самостоятельно.

2. [количество баллов зависит от сложности реализации, за комментариями обращаться к преподавателю] Реализовать в приложении защиту от атак с перехватом буфера обмена.
3. [количество баллов зависит от сложности реализации, за комментариями обращаться к преподавателю] Реализовать защиту процесса менеджера паролей с помощью специально разработанного драйвера уровня ядра от
    a. отладки,
    b. снятия дампа
    c. других атак.
4. (лимит на данную задачу в 201-351 превышен) [количество баллов зависит от сложности реализации, за комментариями обращаться к преподавателю] Реализовать в менеджере паролей
    a. Удаление учётных записей из файла.
    b. Добавление новых учётных записей в файл.
    c. Ведение автоматического архивирования версий.
    d. Отправку всех или выбранных учётных записей на печать.

Последовательность проверки практического задания

1. TODO описать, как демонстрируется каждый метод защиты
2. Репозиторий, ветки

Вопросы на защиту теоретической части

1. Криптография, криптоанализ и криптология. Определения и взаимосвязь.

2. Перечислите основные задачи криптографии.
3. Назовите определения и подчеркните отличия расшифрования и дешифрования.
4. Шифр, шифрсистема, шифртехника. Определения и взаимосвязь.
5. Криптографический протокол, криптосистема. Определения и взаимосвязь.
6. Классификация криптосистем. Перечислить или изобразить на схеме выделяемые классы.
7. Примеры бесключевых криптосистем.
8. Определение и примеры симметричных криптосистем.
9. Определение и примеры асимметричных криптосистем.
10. Определение и примеры поточных и блочных криптосистем. Определение и стандарты паддинга.
11. Криптостойкость, определение и числовые метрики.
12. Принцип Керкгоффса.
13. Хеширование, определение, области применения и примеры функций.
14. Требования к криптостойким хеш-функциям, коллизия первого и второго рода.

~~Литература~~

1. ~~https://en.wikipedia.org/wiki/Kernel_Patch_Protection~~
2. ~~https://www.elastic.co/blog/protecting-windows-protected-processes~~
3. ~~https://tipsmake.com/bad-guys-can-steal-data-by-freezing-ram-sticks-with-liquid-nitrogen~~
4. ~~~~

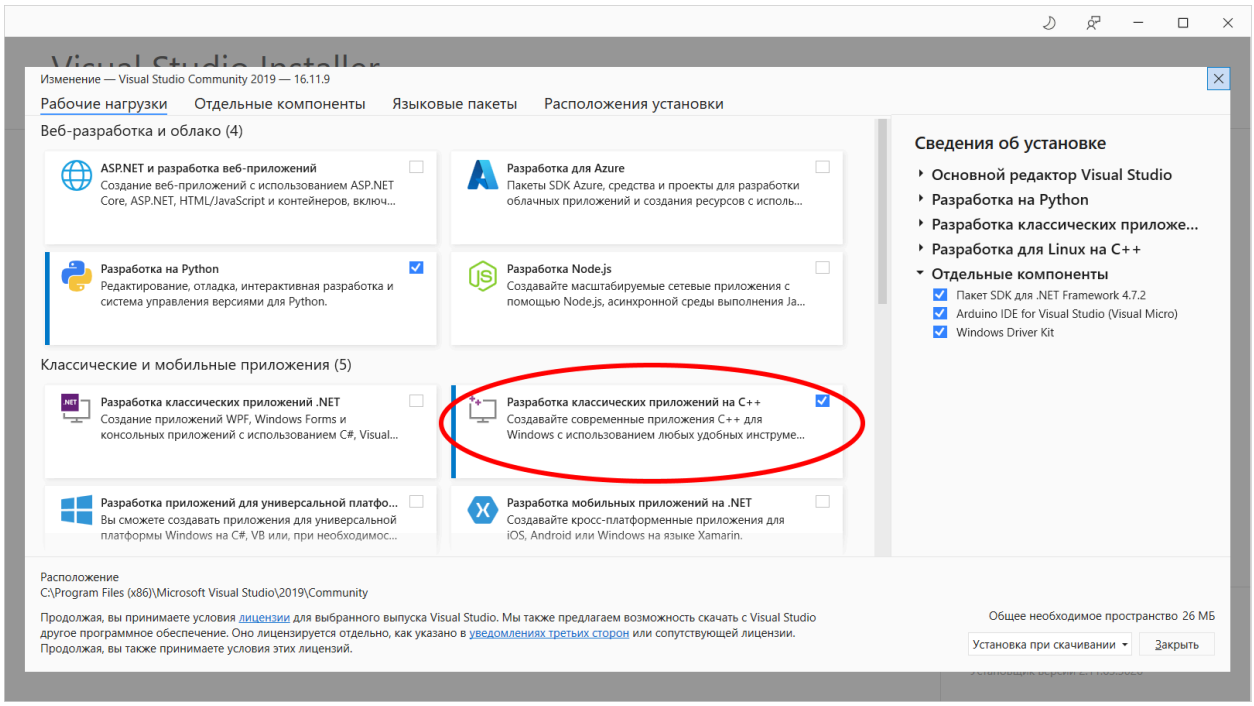
~~Необходимое ПО и инструкции по установке~~

1. **Visual Studio Community 2022**

В настоящий момент (март 2023 г) для разработки драйверов поддерживается только версия Visual Studio 2022. Скачать бесплатную некоммерческую версию Community можно по ссылке

https://visualstudio.microsoft.com/ru/free-developer-offers/ (старая версия Visual Studio Community 2019 версия по ссылкам

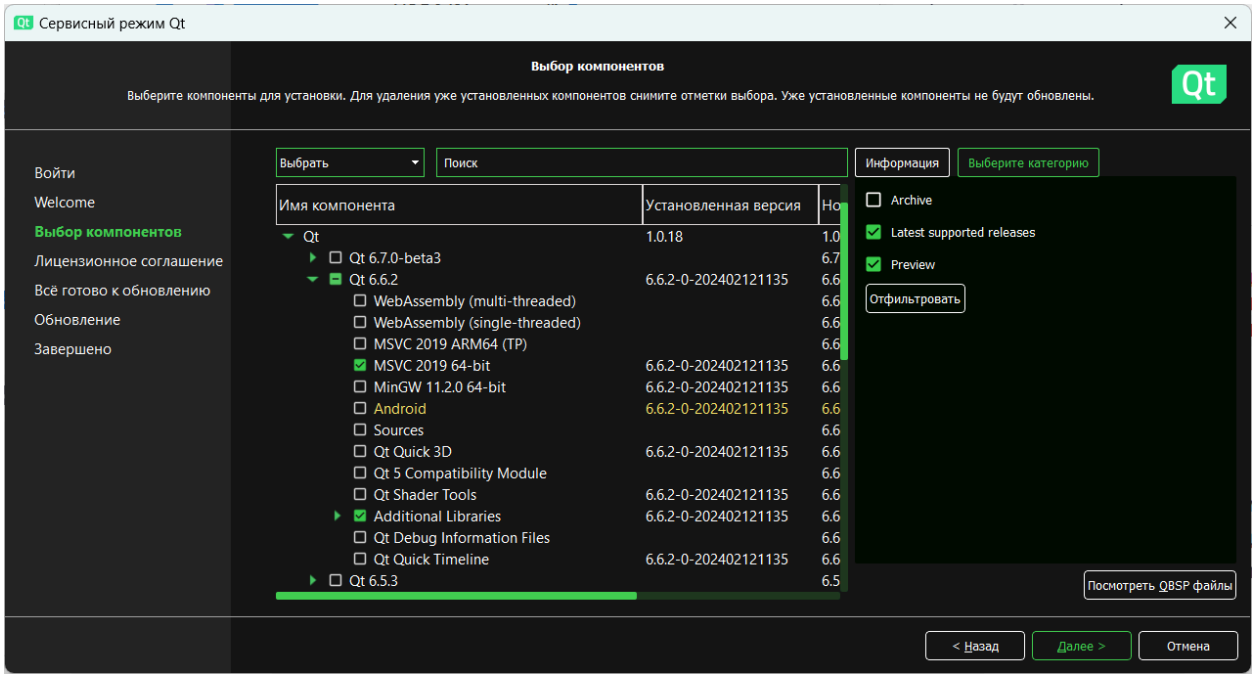
- https://visualstudio.microsoft.com/ru/vs/older-downloads/
- или https://docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes,
- (2024) https://apps.microsoft.com/detail/xp8cdjnzkfm06w?ocid=pdpshare&hl=ru-ru&gl=RU,
- (2024) https://aka.ms/vs/16/release/vs_community.exe.
  При установке выбрать комплект разработки для C++.

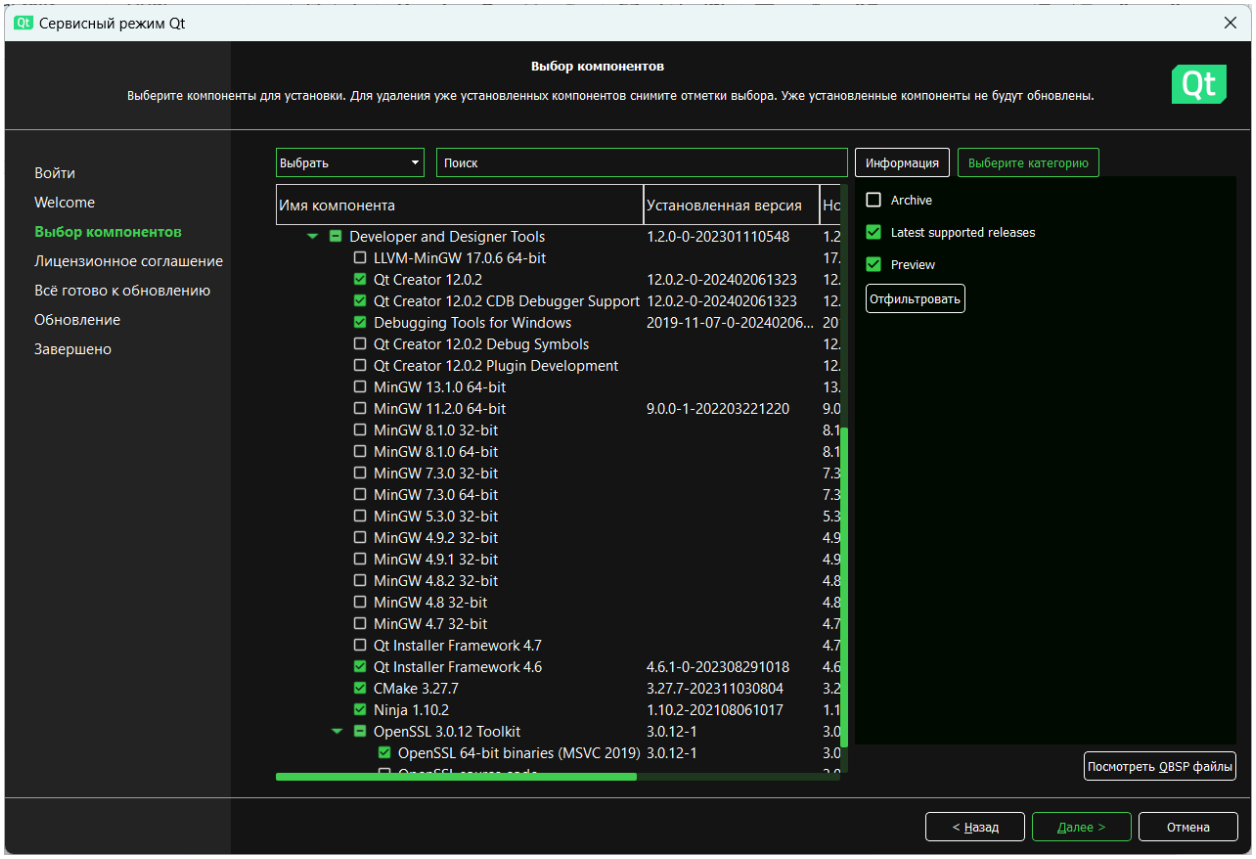


После установки убедиться, что в Visual Studio создаётся и запускается шаблонное консольное приложение C++.

2. **Qt Open Source (https://www.qt.io/download-open-source)**

В инсталляторе отменить комплект библиотек для сборки в MSVC 2019, Qt Creator, Qt Creator CDB Debugger Support, Debugging Tools for Windows, CMake, Ninja и OpenSSL 64-bit binaries.





После установки необходимо убедиться, что в Qt собирается и запускается шаблонное приложение типа Qt Widgets.

ЛР2. Защита автоматизированной системы на уровне ядра ОС. Прозрачное шифрование дискового ввода/вывода драйвером ядра ОС

~~//TODO в PreOperationSassThrough не имеет смысла расшифровывать буфер при MJ_READ, так как буфер недоступен (вписать в методику + снимок)~~

Цель

Ознакомление с приёмами использования модулей ядра ОС для защиты автоматизированных систем.

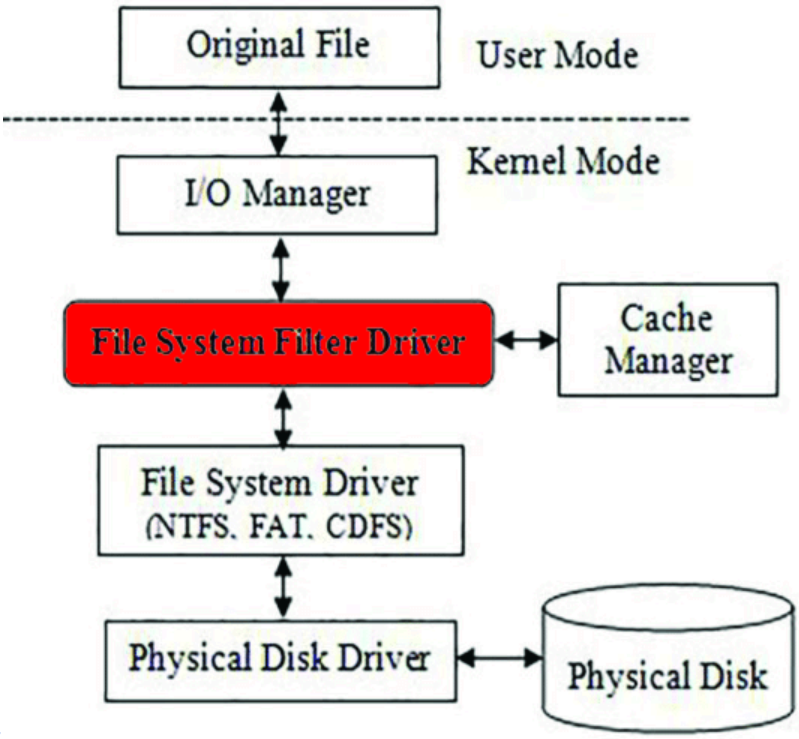
Задачи

1. Ознакомиться с основами устройства ядра ОС и архитектурой драйвера на примере MS Windows.
2. Развернуть комплекс программного обеспечения для разработки драйверов для ОС Windows, включающий среду разработки (IDE), SDK, WDK, WinDbg и виртуальную машину для тестирования.
3. ~~Разработать минифильтр-драйвер дискового ввода/вывода для избирательного “прозрачного” шифрования файлов.~~
4. ~~Разработать клиентское приложение для тестирования и управления функциями разработанного драйвера.~~

~~Необходимый теоретический материал~~

1. ~~Кольца защиты команд ЦП. Привилегии, доступные коду, исполняемому в разных кольцах защиты. ПО, работающее с привилегиями уровня ядра. Взаимодействие ПО, работающего на уровне ядра и на пользовательском уровне.~~
2. ~~Основы архитектуры драйвера уровня ядра ОС Windows (опционально - Linux).~~
3. ~~Основы Си, алгоритмы и библиотеки шифрования данных.~~
4. ~~Основы владения средствами разработки (компилятор, компоновщик, IDE) и отладки.~~

5. Основы криптографических алгоритмов и их программных реализаций (OpenSSL).
6. Информация по коммерческим аналогам - https://www.easefilter.com/.

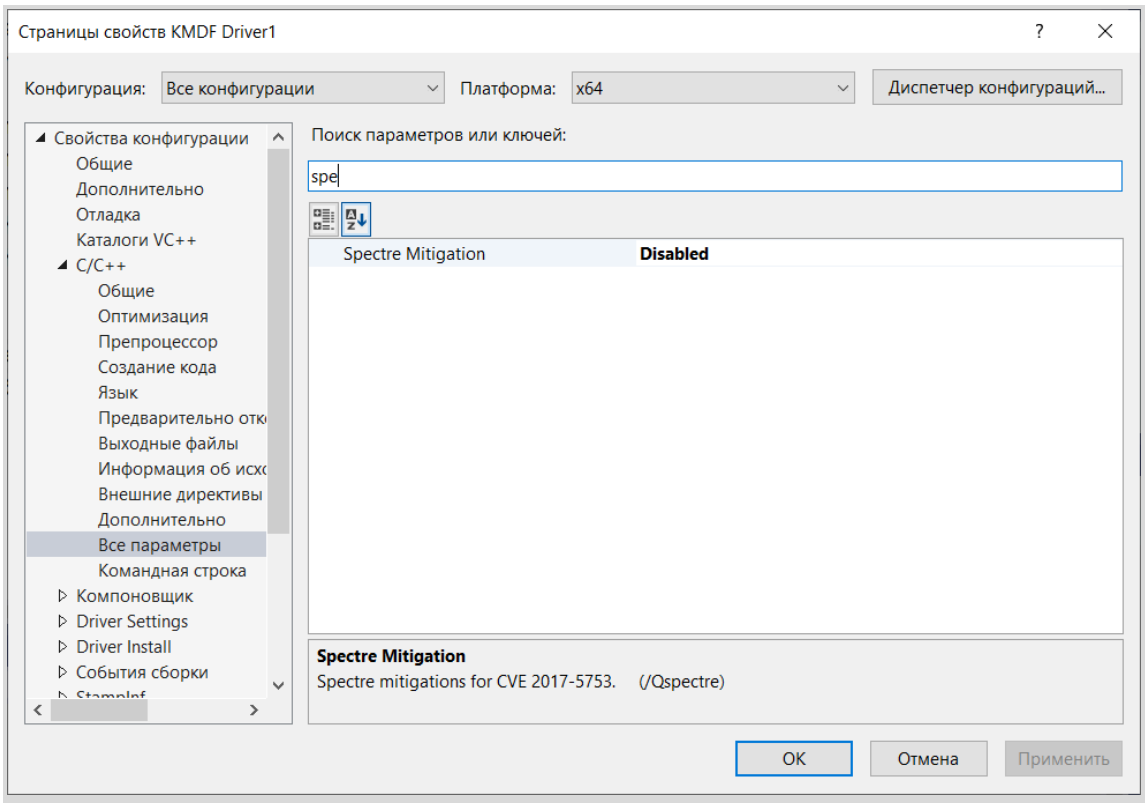


EaseFilter

Последовательность выполнения

Этап 1. Установка инструментов разработки и сборка тестового драйвера.

1. Установить на рабочую машину Visual Studio Community, Windows Driver Kit, рекомендуемую версию SDK для данной версии WDK.
2. Создать проект Visual Studio из шаблона драйвер-фильтра файловой системы (https://github.com/microsoft/Windows-driver-samples/tree/master/filesys/miniFilter/passThrough).
3. Отключить использование версий библиотек Си/C++, устраняющих риски Spectre/Meltdown (см. рис.), или доустановить требуемые библиотеки в инсталляторе Visual Studio - на усмотрение учащегося.



4. Минифильтр-драйвер из репозитория собирается с ошибками и требует доработки INF-файла согласно инструкции https://docs.microsoft.com/en-us/windows-hardware/drivers/develop/creating-a-primitive-driver и таблице:

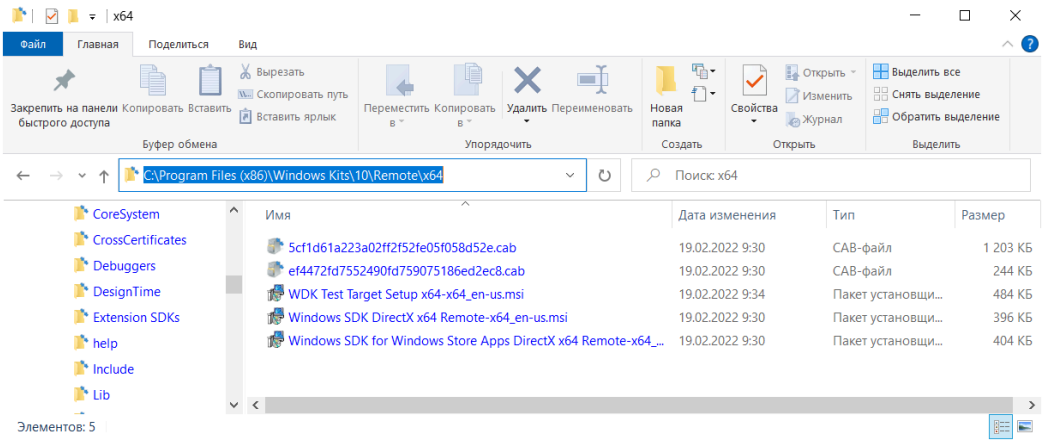
Фрагмент из оригинального репозитория	**Заменить на**
[DefaultInstall]	[DefaultInstall.NTamd64]
[DefaultInstall.Services]	[DefaultInstall.NTamd64.Services]
[DefaultUninstall]	[DefaultUninstall.NTamd64]
[DefaultUninstall.Services]	[DefaultUninstall.NTamd64.Services]
-	добавить LegacyUninstall=1 в секции

Фрагмент из оригинального репозитория	Заменить на
	[DefaultUninstall.NTamd64] и [DefaultUninstall.NTamd64.Services]

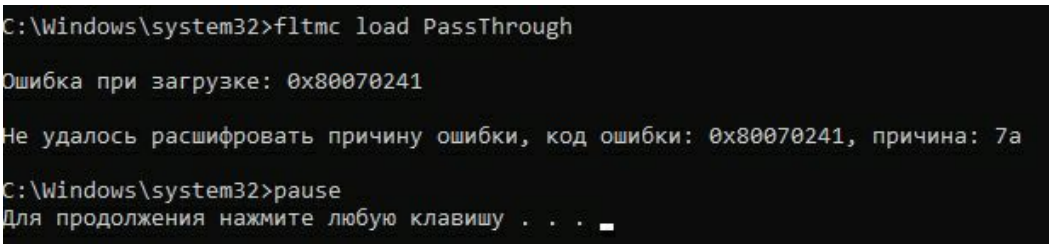
5. Добиться, чтобы проект создавался без ошибок, и при сборке создавались бы 3 файла, представляющие собой пакет для установки:
    a. passThrough.sys,
    b. passThrough.inf,
    c. passthrough.cat.

Этап 2. Настройка тестовой машины, развёртывание и запуск драйвера.

1. Установить на рабочую машину среду виртуализации и установить ОС на виртуальную машину.
2. Настроить виртуальную машину в соответствии с рекомендациями [https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/provision-a-target-computer-wdk-8-1], [https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/setting-up-network-debugging-of-a-virtual-machine-host]
    a. Установить в тестовую виртуальную машину дополнение гостевой ОС “WDK Test Target Setup” из каталога WDK



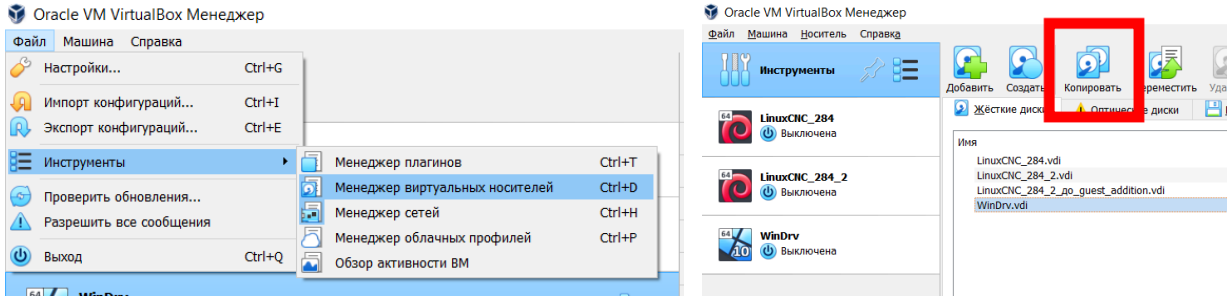
3. В командной строке гостевой ОС **с правами администратора** создать запись загрузчика ОС со специальными настройками для разработки и тестирования драйверов:
   - создание новой конфигурации загрузки ОС на основе существующей
     `bcdedit /copy {current} /d "Windows 10 with kernel debug"`
     При этом команда выведен в терминал UUID созданной конфигурации загрузки. Сохраните его, он потребуется для дальнейшей настройки.
   - включить отображение меню при загрузке и время его отображения, удобное для выбора при старте ВМ
     `bcdedit /set {bootmgr} displaybootmenu yes`
     `bcdedit /set {bootmgr} timeout 10`
   - разрешить загрузку драйверов без подписи Microsoft*
     `bcdedit /set {UUID-возвращённый-первой-командой} testsigning on`
     **в некоторых случаях по неустановленной причине ОС продолжает запрещать загрузку драйверов с тестовой подписью (см. рис. ниже).*



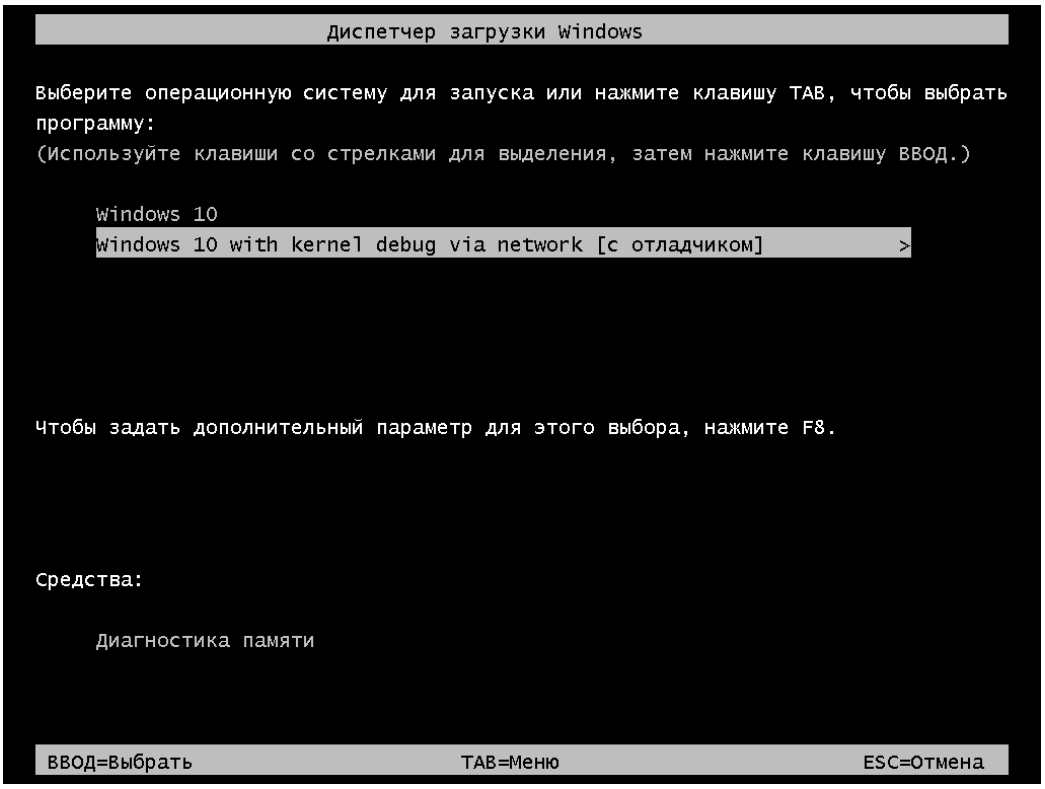
     В этом случае каждый раз при загрузке ОС в виртуальной машине необходимо вручную выбирать опцию “Отключить обязательную проверку подписей драйверов”, либо отключить SecureBoot в настройках виртуальной машины и от имени администратора ввести в консоли
     `bcdedit /set loadoptions DISABLE_INTEGRITY_CHECKS`

```
bcdedit                                          /set
{UUID-возвращённый-первой-командой}              /set
NOINTEGRITYCHECKS ON
bcdedit                                          /set
{UUID-возвращённый-первой-командой}
testsigning on
```

4. Выключить ВМ, сделать копию виртуального жёсткого диска:

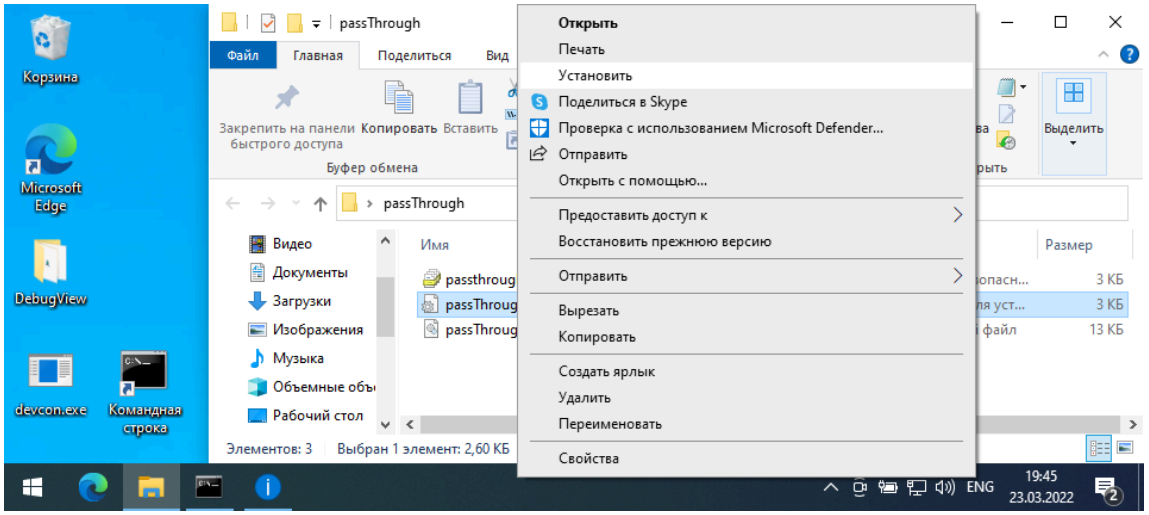


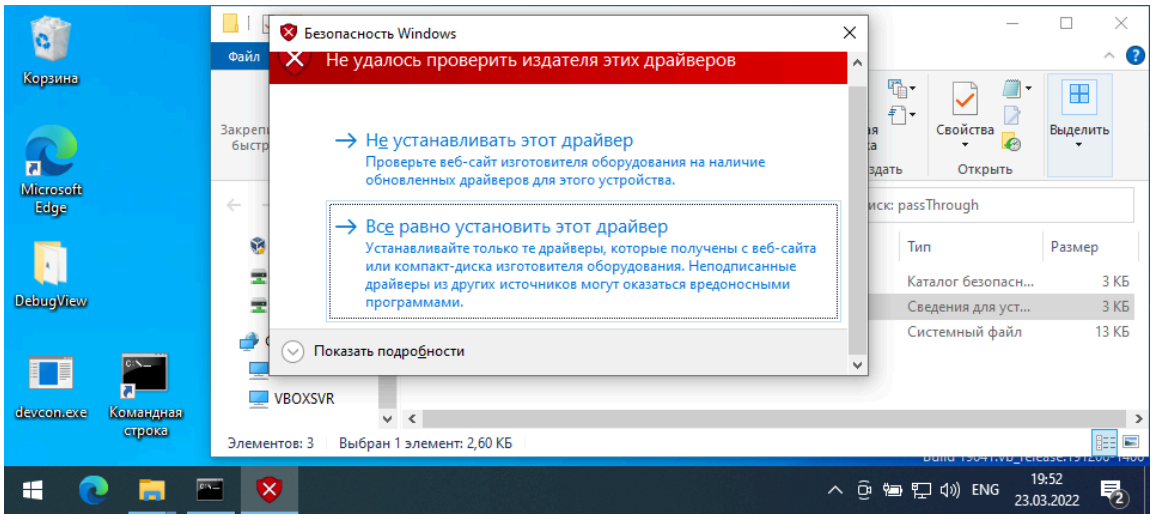
5. При загрузке выбрать опцию загрузки в только что созданной конфигурации.



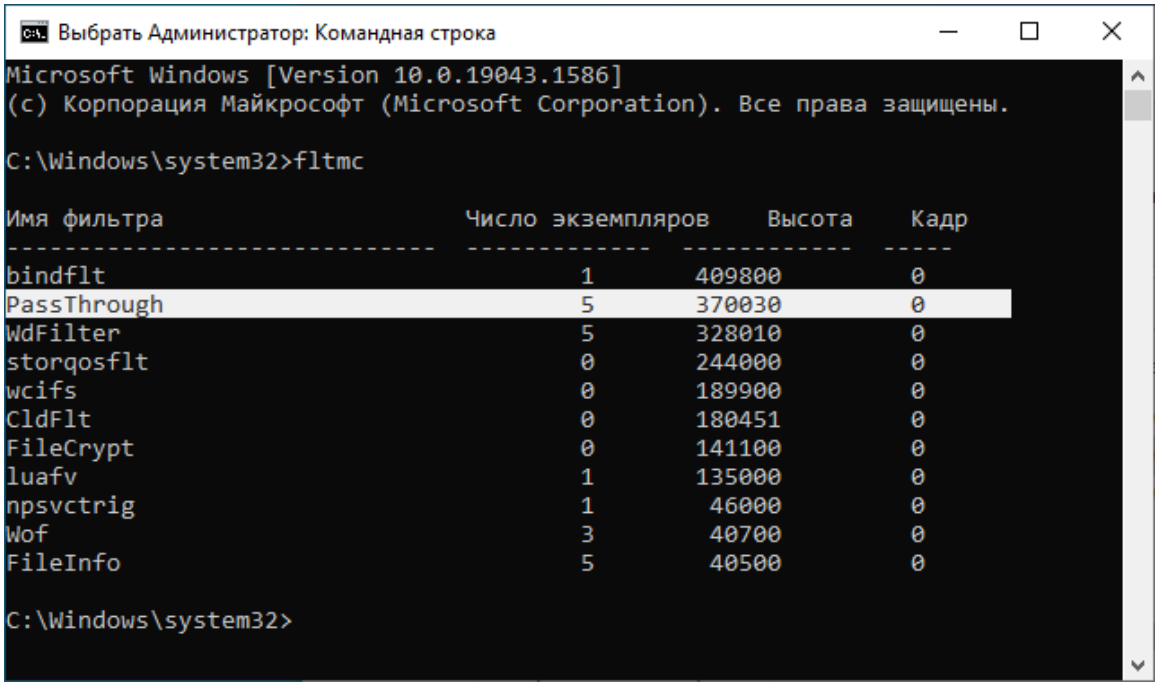
6. Перенести в тестовую виртуальную машину три файла (*.sys, *.inf и *.cat), созданных на этапе 1.

7. Установить минифильтр-драйвер в тестовой ВМ, выбрав в контекстном меню INF-файла “Установить”:



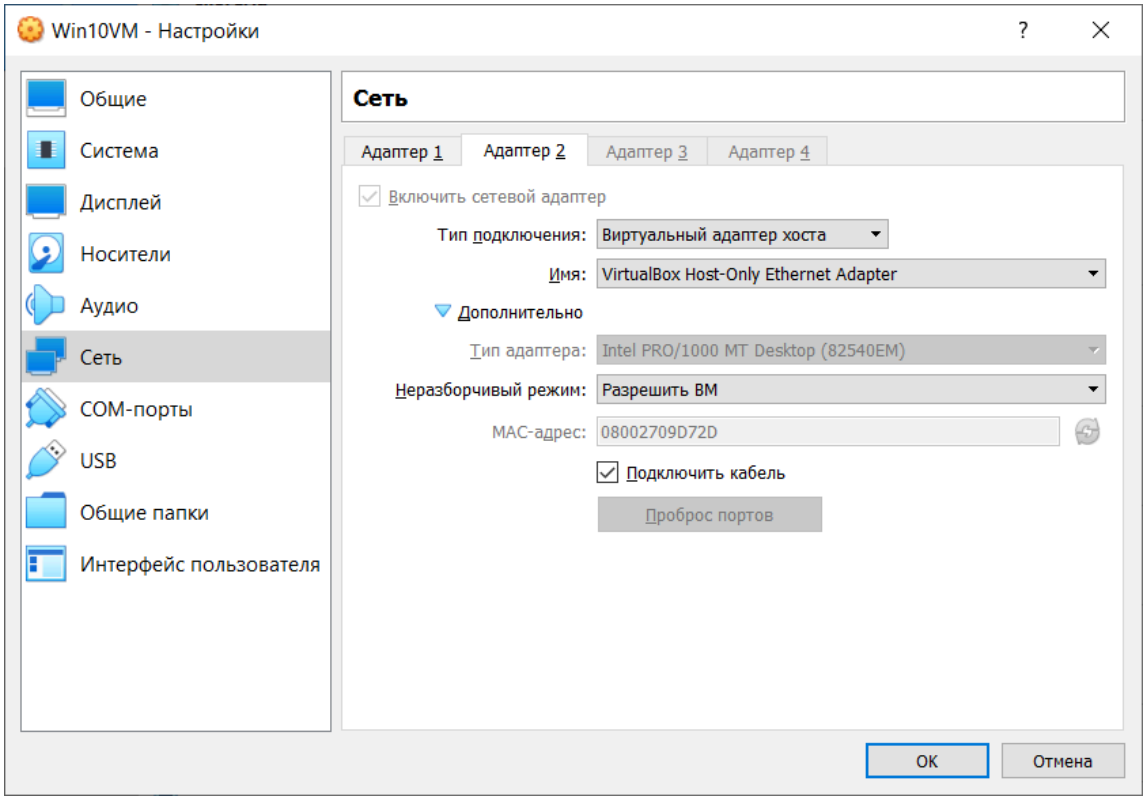


8. Запустить драйвер, запустив команду в консоли от имени администратора:
   `fltmc load PassThrough`
   Для последующей выгрузки драйвер-фильтра также от имени администратора в терминале следует ввести команду
   `fltmc unload PassThrough`
9. Убедиться в наличии загруженного драйвера в стеке с помощью команды `fltmc` в консоли Windows в режиме администратора, либо с помощью утилиты WinObj https://docs.microsoft.com/en-us/sysinternals/downloads/winobj:



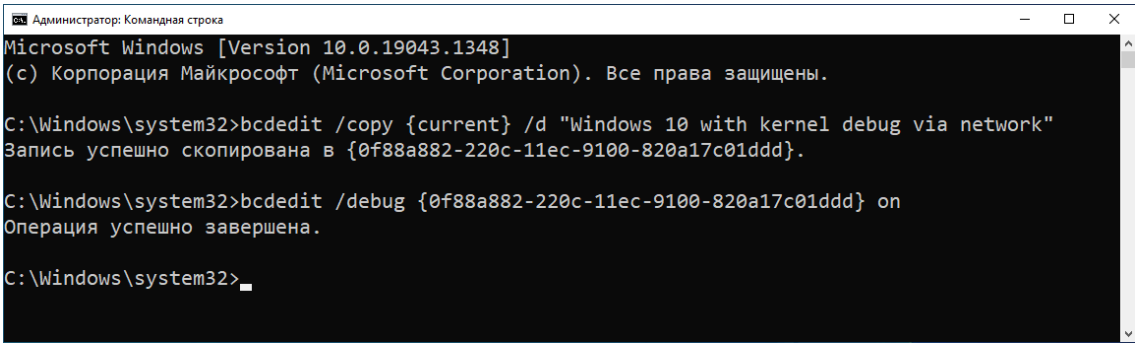
Этап 3. Настройка отладки драйвера на тестовой удалённой машине.

1. Настроить виртуальную машину для поддержки необходимых соединений (рис. ниже - пример для подготовки VM Virtual Box для соединения с отладчиком по tcp). Возможно, на основной ОС (хост) будет необходимо отключить блокировку данного IP, порта (50000), путём внесения исключения в сетевой фильтр (firewall).



2. Разрешить удалённую отладку в созданном варианте загрузки
`bcdedit /debug {UUID-возвращённый-первой-командой} on`

`bcdedit /set {UUID-возвращённый-первой-командой} debugtype net`

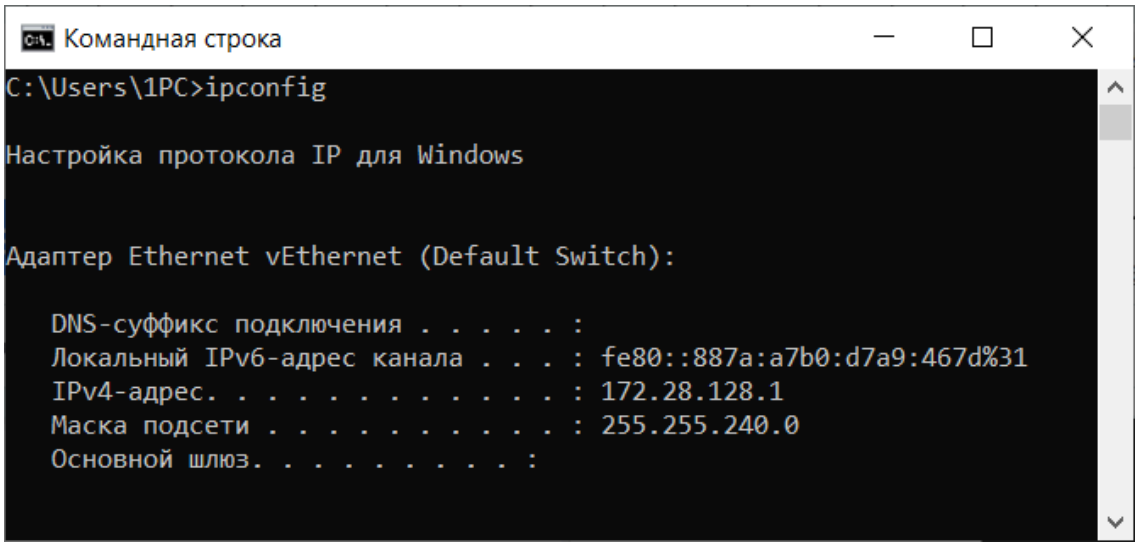


3. Для отладки по сетевому протоколу - сконфигурировать порт, ключ защиты и (опционально) IP (IP-адрес основной машины - хоста, который он имеет в локальной сети с виртуальной машиной)

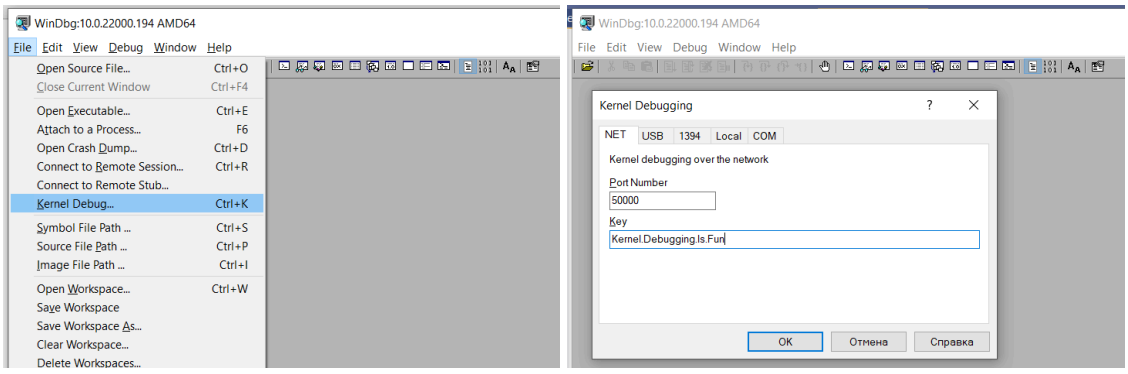
```
bcdedit /dbgsettings net hostip:ip_осн_машины port:50000 key:ключ_защиты
```

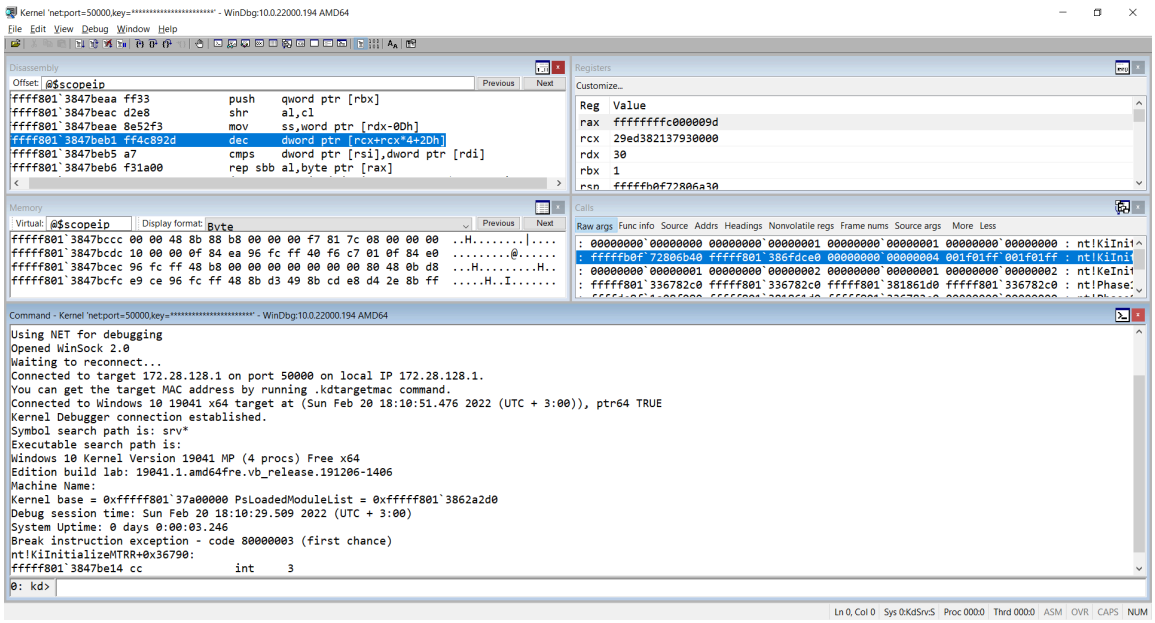
~~или без IP-адреса~~

```
~~bcdedit /dbgsettings net port:50000 key:ключ_защиты~~
```

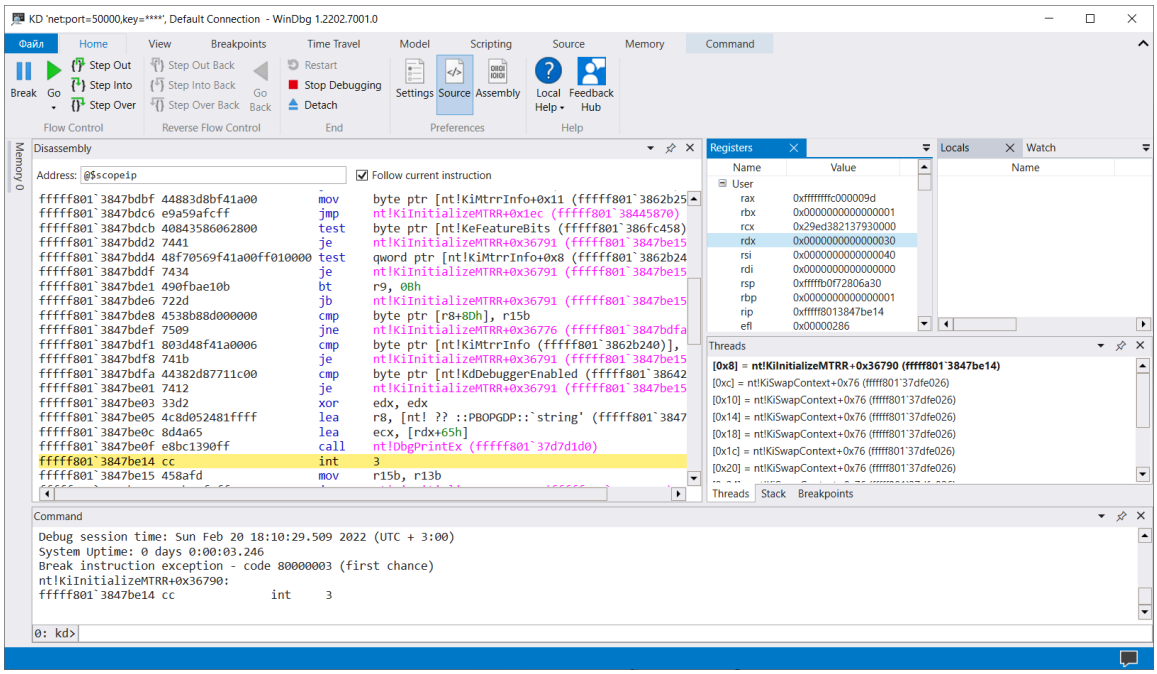


4. Выключить тестовую ВМ, сделать резервную копию виртуального жёсткого диска.
5. Перезапустить ВМ с опцией отладки.
6. ~~//TODO написать что после этого ОС не грузится без отладчика~~
7. Настроить WinDbg для установки соединения:





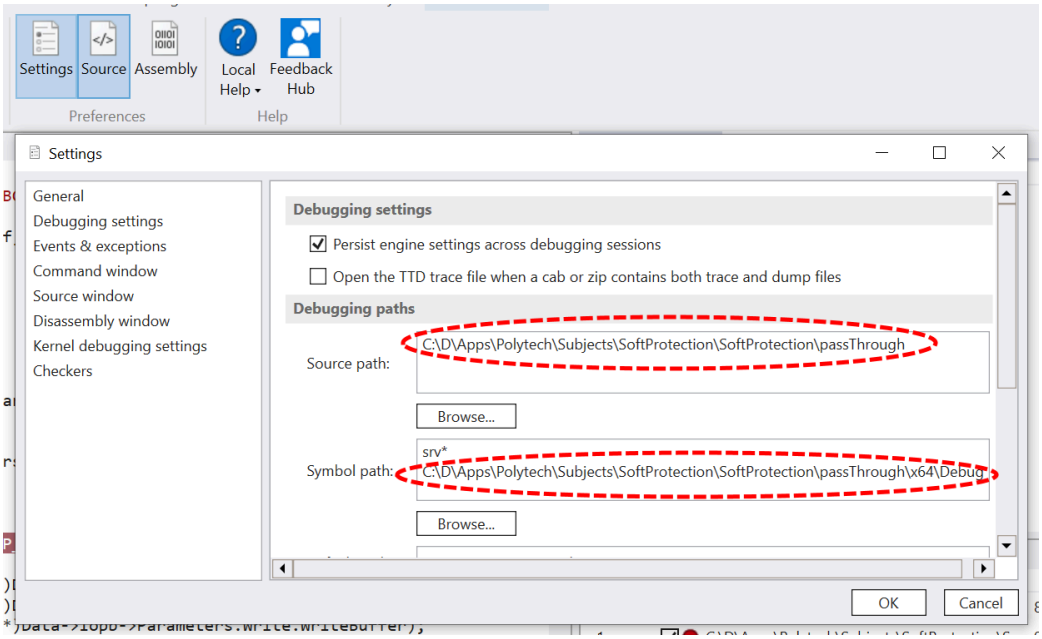
Помимо обычной версии WinDbg, поставляемой с WDK, можно воспользоваться WinDbg с обновлённым интерфейсом из Microsoft Store:



8. Добиться работы инструментов отладки: срабатывания установленной в исходном коде точки останова, пошагового выполнения, просмотра значений переменных в интерфейсе Visual Studio.

9. В некоторых случаях отладку быстрее и удобнее производить без отладчика с помощью отладочной печати (наподобие print_f(), qDebug() и std::cout). Для этого предусмотрена функция KdPrint()/DbgPrint(). Просмотр отладочных сообщений, полученных таким образом, удобно производить с помощью утилиты DebugView (https://docs.microsoft.com/en-us/sysinternals/downloads/debugview) .
10. Для отладки на уровне исходного кода, а не дизассемблированного листинга, в драйвер необходимо загрузить 1) файлы исходного кода с расширениями *.c, *.cpp. *.h 2) файл, связывающий исходный код с фрагментами исполняемого скомпилированного кода Program Data Base (*.pdb). Для необходимо
    a. запустить виртуальную машину с опцией отладки (при выборе этой опции загрузки ВМ остановится и будет ждать подключения отладчика),
    b. подключиться в отладчике к виртуальной машине,
    c. дождаться загрузки гостевой ОС,
    d. после любых изменений исходного кода требуется пересобрать драйвер, переустановить и перезапустить его в виртуальной машине,
    e. для загрузки исходного кода в интерфейсе отладчика вызвать пункт меню Файл → Open source file и выбрать файлы исходного кода, в которых требуется выставить точки останова (passThrough.c). Исходный код загрузится в отдельную панель интерфейса, удобно вывести эту панель вместо дизассемблированного листинга.
    f. **(в случае, если после загрузки с-файла точки останова не отображаются в коде и не срабатывают)** Иногда требуется, помимо файла исходного кода *.c, предоставить отладчику pdb-файлы, полученные в ходе сборки драйвера. Для этого открыть настройки (Settings) отладчика и записать путь к каталогу, в котором хранится файл PDB, полученный при компиляции для данного драйвера (для

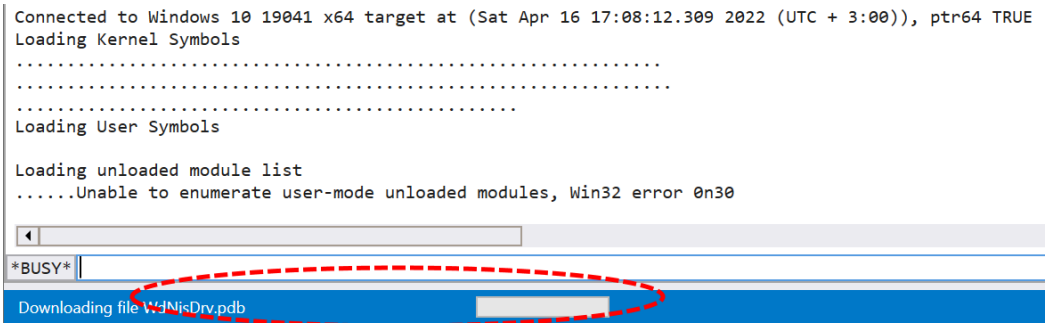
надёжности можно вписать несколько вложенных путей, где может лежать pdb-файл драйвера)



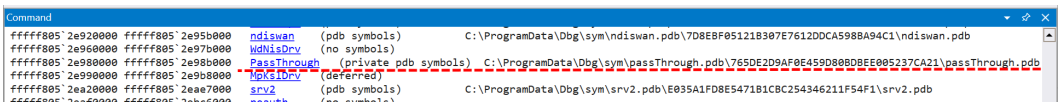
либо поставить отладку на паузу и ввести в консоли отладчика команды

```
.srcpath+ каталог\где\хранится\исходный\код
.sympath+ каталог\где\хранится_pdb
```

(На ПО 2023 года данный пункт не требуется, также не требуется при сборке драйвера в конфигурации Debug) *После этого необходимо дождаться окончания автоматической загрузки файлов отладчиком*



g. ***(На ПО 2023 года данный пункт не требуется, также не требуется при сборке драйвера в конфигурации Debug)*** *Для проверки готовности модуля к отладке на уровне исходного кода необходимо поставить отладку на паузу, и в консоли отладчика ввести команду* `lm`*. В полученном текстовом выводе найти строку, соответствующую интересующему модулю (passthrough) и убедиться, что напротив модуля указан путь к его pdb-файлу (а не “(no symbols)”).*



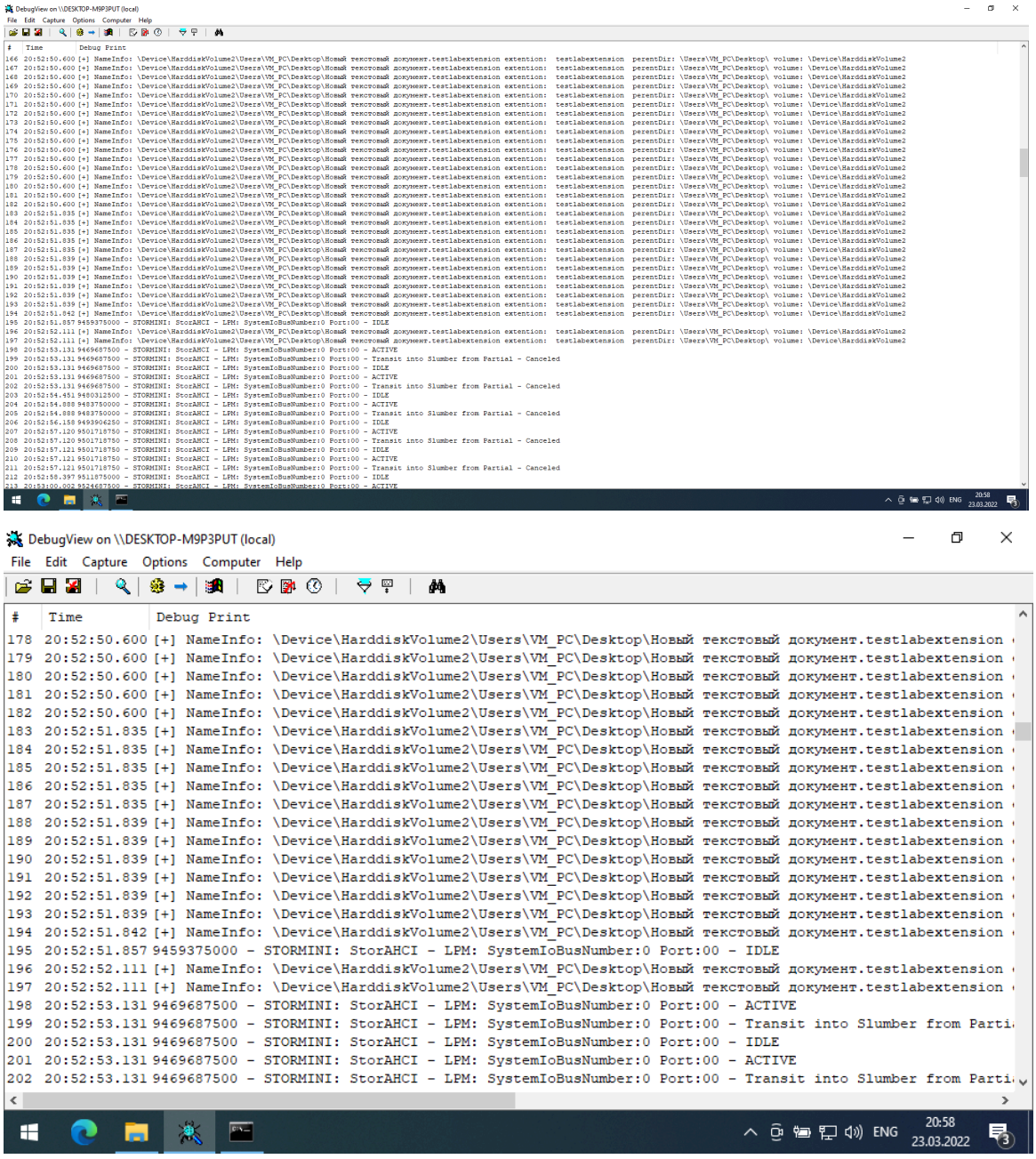
Этап 4. Разработка минифильтр-драйвера прозрачного шифрования и приложения для его тестирования.

1. Разработать простейшее приложение, использующее системные функции ReadFile и WriteFile (https://learn.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-readfile) для доступа к тестовому файлу и вызова срабатывания драйвер-фильтра. Функции могут быть обёрнуты в стандартную библиотеку языка C или C++. Нужно гарантировать
    - доступ к тестовому файлу с помощью системных функций чтения/записи, в то время как сторонние приложения (даже простейший Microsoft Notepad.exe) могут обращаться к файлу с помощью иных механизмов, например сопоставление виртуальной памяти с файлом (memory file mapping, https://docs.microsoft.com/en-us/windows/win32/memory/file-mapping); обработка других механизмов ввода/вывода, помимо ReadFile() и WriteFile(), значительно усложнит выполнение данной работы;

- чтение и запись всего файла за один вызов, без разбивки содержимого файла на несколько буферов, что также усложнило бы выполнение данной лабораторной работы.

Приложение должно реализовывать вывод в терминал содержимого файла и запись файла фиксированного размера. Рекомендуется реализовать приложение на языке Си, используя структуру FILE, функции fopen_s(), fread_s(), fseek(), fwrite() и fclose() (https://en.cppreference.com/w/c/io), т.к. они оборачивают необходимые WinAPI ReadFile и WriteFile. Рекомендуется протестировать приложение перед включением драйвера и убедиться в корректном чтении и записи содержимого файла.

2. Реализовать в драйвере внутри функции PtPostOperationPassThrough() ветку кода, срабатывающую при совпадении имени или расширения файла с именем или расширением, предусмотренным учащимся для демонстрационного файла. Внимание! У некоторых учащихся по неясной причине запись в буфер (шифрование и запись на диск, работа с WriteBuffer) успешно работает в PtPreOperationPassThrough(), а в PtPostOperationPassThrough() - только чтение (расшифровка при чтении с диска, работа с ReadBuffer). За справкой по работе с именем файла в драйвере обращаться к документации (https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/managing-file-names). Протестировать и убедиться, что созданная ветка действительно срабатывает только на файлах по признаку, указанному учащимся, и не испортит содержимое всех остальных файлов:



Листинг. Блок кода, который можно применять для извлечения расширения файла из параметров PtPostOperationPassThrough()

```
NTSTATUS status;
PFLT_FILE_NAME_INFORMATION NameInfo = NULL;
status = FltGetFileNameInformation(
     Data,
     FLT_FILE_NAME_NORMALIZED |
     FLT_FILE_NAME_QUERY_DEFAULT,
    &NameInfo);
UNICODE_STRING required_extension =
RTL_CONSTANT_STRING(L"testlabextension");
if (!NT_SUCCESS(status))
{
     //…
}
else {
     // Вариант 2022, начало
     if (
         RtlEqualUnicodeString(&required_extension,
             &NameInfo->Extension, FALSE)==TRUE) {
          // в целях отладки можно вставить DbgPrint(...)
     // Вариант 2022, конец

     /* Вариант 2023, начало

     PWSTR extensionStart = wcsrchr(NameInfo->Name.Buffer,
     L'.');
     if (extensionStart != NULL)
        {
            UNICODE_STRING extension;

            RtlInitUnicodeString(&extension,
extensionStart);

            if (RtlEqualUnicodeString(&required_extension,
&extension, FALSE)) {

     Вариант  2023, конец */

          if (Data->Iopb->MajorFunction == IRP_MJ_WRITE) {

               //здесь - шифрование
     Data->Iopb->Parameters.Write.WriteBuffer
```

```
            }

            else if (Data->Iopb->MajorFunction ==
      IRP_MJ_READ) {

                  //здесь - расшифровка
      Data->Iopb->Parameters.Read.ReadBuffer
            }

      }

}
```

* На сборке 19041 работает, на поздних сборках может не работать

3. Реализовать внутри функции PtPostOperationPassThrough() минифильтр-драйвера условную ветку кода, срабатывающую при чтении (Data->Iopb->MajorFunction == IRP_MJ_READ) и при записи (Data->Iopb->MajorFunction == IRP_MJ_WRITE).
4. В .inf-файле в проекте драйвера задать значение Instance1.Altitude в соответствии с диапазоном, рекомендуемым Microsoft для драйвер-фильтров, выполняющих шифрование файловой системы https://learn.microsoft.com/en-us/windows-hardware/drivers/ifs/allocated-altitudes#140000---149999-fsfilter-encryption.
5. Реализовать внутри созданных веток простейшую модификацию записываемых/считываемых данных (https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/accessing-user-buffers, Data->Iopb->Parameters.Read.ReadBuffer и Data->Iopb->Parameters.Write.WriteBuffer). Например, добавление какого-либо символа в начало и конец файла при записи, или другая модификация данных на выбор учащегося. Общий смысл данного этапа в том, чтобы убедиться на простейшем примере в работоспособности модификации буферов чтения и записи драйвера.

6. Реализовать в ранее созданных условных ветках записи и чтения соответственно шифрования и дешифровки данных по алгоритму AES-256 (допускается использовать header-only библиотеку языка Си, например https://github.com/kokke/tiny-AES-c) с постоянным ключом и расшифровку считываемых с диска данных. Убедиться, что приложение, работающее совместно с драйвером, записывает а затем считывает данные с диска без потерь, а также что данные хранятся на диске в зашифрованном виде и недоступны для считывания прочими приложениями и на другой машине либо без запущенного минифильтр-драйвера. Допускается работа минифильтр-драйвера с буфером (и файлом) фиксированного размера (например, 1кб), прописанного константой. Размер файла, создаваемого клиентским приложением, не должен превышать размер буфера драйвера.
7. Реализовать включение и отключение минифильтр-драйвера с помощью средств автоматизации команд (bat-файлы).

Visual Studio в настоящее время нельзя настроить для установки соединения с тестовой машиной и отладки драйвера:

- https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/performing-kernel-mode-debugging-using-visual-studio, “Important. This feature is not available in Windows 10, version 1507 and later versions of the WDK.”,
- также https://social.msdn.microsoft.com/Forums/azure/en-US/3159e8d4-56bd-4369-a564-043dea0bb6a6/is-it-really-that-using-of-vm-as-target-computer-for-driver-debugging-is-prohibited-on-windows-10?forum=wdk)

> ⓘ **Important**
>
> This feature is not available in Windows 10, version 1507 and later versions of the WDK.

1. Загрузить и запустить собранный пример драйвера на тестовой машине (рядом с inf-файлом на тестовой машине должны находиться сгенерированные Visual Studio также *.sys и *.cer-файлы или *.cat-файлы).

```
devcon install <название файла>.inf root\ECHO
```

~~Дополнительные задания (выполняются только по согласованию с преподавателем индивидуально)~~

1. ~~Разработка драйверов в VS Code~~
    a. ~~Поиск и подключение инструментов разработки (SDK и WDK)~~
    b. ~~Подключение к удалённому хосту и удалённая настройка (bcdedit и накатка WDK tools)~~
    c. ~~Перенос драйвера после сборки на тестовую машину, установка и запуск~~
    d. ~~Удалённая отладка с windbg~~
2. ~~Найти, протестировать и продемонстрировать, какие ещё комбинации инструментов разработки и отладки, помимо Visual Studio + WinDbg, удобны для полного цикла разработки драйверов режима яыдра на Windows (работа с исходным кодом + компиляция и сборка + развёртывание на удалённой машине + отладка).~~
3. [полная реализация данного задания оценивается в 50 баллов, задание выполняет Горский из 221-3210] Выполнить прозрачное избирательное шифрование файлов на основе драйвера ядра

Linux (если операционной системой это предусмотрено) (kdb вместо WinDbg).

   a. одинаковое решение засчитывается первому сдавшему в каждой группе
   b. ~~в 351 группе задание уже сделано через Fuse - https://en.wikipedia.org/wiki/Filesystem_in_Userspace, далее оцениваться будут только иные решения~~

4. ~~[полная реализация данного задания оценивается до 20 баллов] Реализовать управление драйвером из графического приложения пользовательского уровня, реализующего (несколько либо все функции):~~
   a. ~~[5 баллов] включение/отключение драйвер-фильтра по нажатию кнопки/меню (без вызова консольной команды fltmc приложением),~~
   b. ~~[10-15 баллов] передачу драйверу списка расширений и названий файлов, для которых необходимо производить шифрование и расшифровку,~~
   c. ~~[15 баллов] ведение лога файлов, зашифрованных и расшифрованных с помощью драйвер-фильтра,~~
   d. ~~[10 баллов] передачу драйверу ключа и инициализирующего вектора, задаваемых в графическом интерфейсе пользователем,~~
   e. ~~настройку типа алгоритма шифрования (в качестве примера реализовать и переключаться как минимум между 2 типами)~~
   f. ~~[5 баллов] автоматический запуск вместе с ОС и отображение иконки в системном меню панели задач (данное задание засчитывается только если реализовано одно из вышеприведённых заданий по взаимодействию с драйвером).~~
5. ~~[данное задание оценивается от 5 до 15 баллов в зависимости от объёма кода реализации] Выяснить причины двойного срабатывания событий IRP_MJ_READ и IRP_MJ_WRITE при чтении и записи содержимого файла. Изменить код драйвера~~

~~таким образом, чтобы из каждый пары повторяющихся событий он срабатывал только на нужное.~~

6. [требуется заранее записаться на задание] Доработать драйвер, устранив условности лабораторной реализации:
    a. Адаптировать драйвер-фильтр для работы с буфером переменного размера (на основе https://github.com/microsoft/Windows-driver-samples/tree/master/filesys/miniFilter/swapBuffers).
    b. Реализовать работу драйвер-фильтра с приложениями, использующими сопоставление памяти файловым адресам (memory mapped files, например, Microsoft Notepad.exe, подробнее https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/memory-mapped-files-in-a-file-system-filter-driver).
    c. Реализовать работу драйвер-фильтра с функцией seek(), частичным чтением и записью файла.

~~Последовательность проверки практического задания~~

~~1. Продемонстрировать содержимое тестового файла до загрузки драйвера.~~
~~2. Установить драйвер в ОС.~~
~~3. Загрузить драйвер.~~
~~4. Запустить тестовое приложение для чтения и модификации файла. Показать распечатку содержимого файла в приложении.~~
~~5. Показать срабатывание точек останова в исходном коде драйвера при расшифровке и шифровании файла.~~
~~6. Показать печать отладочных сообщений, соответствующую веткам кода, обрабатывающим IRP_MJ_READ и IRP_MJ_WRITE, и содержимое буфера.~~
~~7. Запустить тестовое приложение ещё несколько раз (если необходимо).~~
~~8. Выгрузить драйвер.~~
~~9. Открыть и продемонстрировать содержимое тестового файла.~~

Вопросы на защиту теоретической части

1. Описать схему работы виртуальной памяти в современных ОС. Назвать преимущества и недостатки виртуализации оперативной памяти. Назвать особенности организации виртуального адресного пространства и диапазоны адресов для модулей с привилегиями пользовательского уровня, и уровня ядра.
2. В чём заключаются отличия в режимах ядра и пользовательском режиме выполнения кода? Какие модули кода работают с привилегиями пользовательского уровня и уровня ядра.
3. Привести определение драйвера. Назвать 3 типа драйверов WDF и их функции.
4. Описать, как задаётся место установки драйвер-фильтра в стеке драйверов. На какое место требуется устанавливать драйвер-фильтр прозрачного шифрования, и обосновать, почему.
5. Назвать преимущества модели “драйвер-минидрайвер”. Назвать и найти в проводнике ОС базовый драйвер для драйвер-фильтров.
6. Какие допущения и условности лабораторной реализации драйвера препятствуют его широкому применению? Что следует доработать и какой функционал следует добавить, чтобы разработанный драйвер можно было использовать для защиты пользовательских и корпоративных файлов.

Литература

1. Йосифович П. Работа с ядром Windows. - 2021.
2. Write a Hello World Windows Driver (KMDF). - URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/writing-a-very-small-kmdf--driver.
3. How to write your first USB client driver (KMDF). - URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/usbcon/tutorial--write-your-first-usb-client-driver--kmdf-.
4. Windows Kernel Debugging Tips. - URL: https://www.virtualbox.org/wiki/Windows_Kernel_Debugging.

5. Setting Up a Windows 7+ Virtualbox VM for Kernel Mode Debugging. - URL: https://medium.com/@eaugusto/setting-up-a-windows-7-virtualbox-vm-for-kernel-mode-debugging-367911889316.
6. Setting up a Windows VM lab for kernel debugging. - URL: https://blahcat.github.io/2017/08/07/setting-up-a-windows-vm-lab-for-kernel-debugging/.
7. Driver samples for Windows 10. - URL: https://github.com/microsoft/Windows-driver-samples.
8. System setup for kernel development and debugging. - URL: https://codemachine.com/articles/system_setup_for_kernel_development.html.
9. Adventures in Windows Driver Development. - URL: https://research.nccgroup.com/2016/04/27/adventures-in-windows-driver-development-part-1/.
10. Driver samples for Windows 10. - URL: https://github.com/Microsoft/Windows-driver-samples.
11. Debug Windows Drivers - Step by Step Lab (Echo Kernel-Mode). - URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/debug-universal-drivers---step-by-step-lab--echo-kernel-mode-.
12. Процедуры драйвер-фильтра: https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/writing-preoperation-callback-routines, https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/managing-file-names, https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/accessing-user-buffers.
13. Windows Driver Development Tutorial for Beginners. - URL: https://www.youtube.com/playlist?list=PLZ4EgN7ZCzJyUT-FmgHsW4e9BxfP-VMuo.
14. https://www.apriorit.com/dev-blog/678-driver-development-minifilter-backup
15. https://www.apriorit.com/dev-blog/371-file-encryption-driver-development-with-per-process-access-restriction

16. https://sysprogs.com/legacy/virtualkd
17. http://article.nadiapub.com/IJSIA/vol8_no1/12.pdf
18. https://www.youtube.com/@nirlichtman
19. https://www.youtube.com/watch?v=CZkEHMRz-Ps

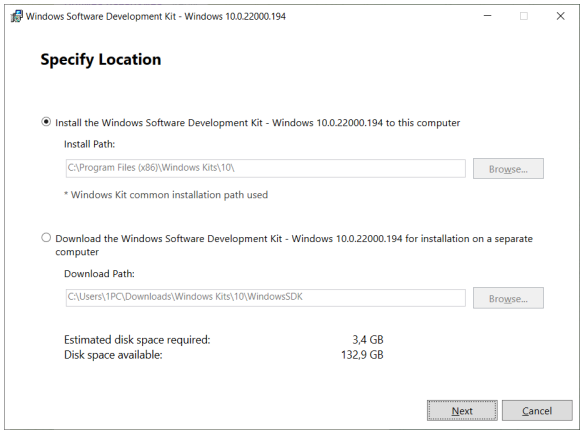
Необходимое ПО и инструкция по установке

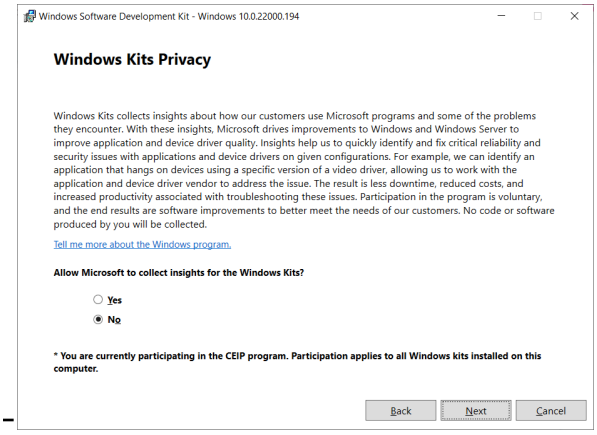
Список ПО, необходимого для выполнения лабораторной работы

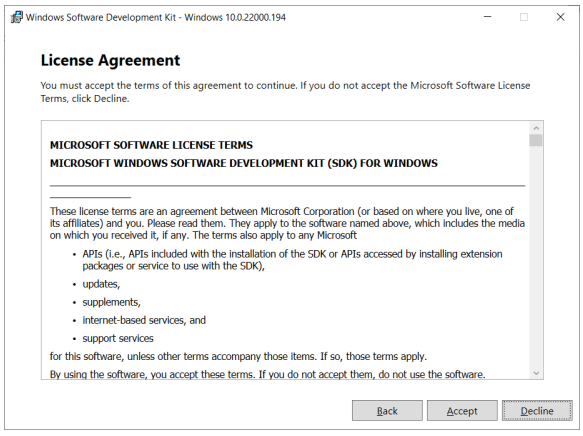
1. Visual Studio Community 2022 с комплектом разработки для C++ (установленный ранее для ЛР1).
2. WDK.
3. Виртуальная машина (VirtualBox, либо HyperV, либо др.) с гостевой ОС Windows 10.
4. WinDbg Preview.

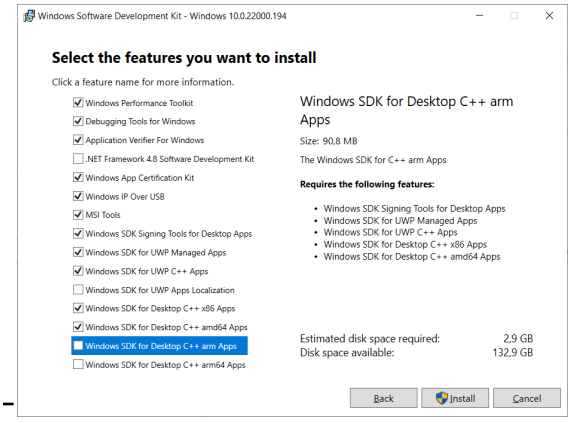
WDK

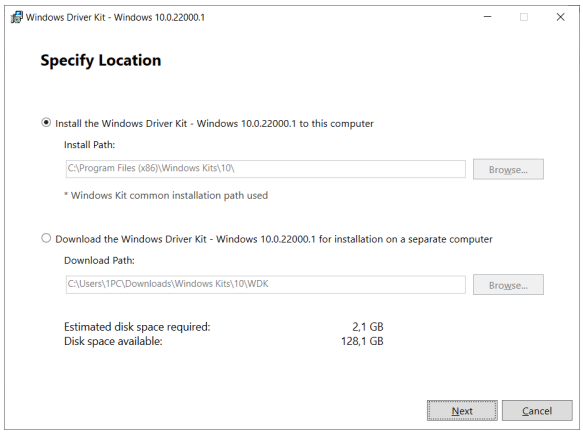
WDK (Windows Driver Kit) - комплект файлов исходного кода, библиотек и инструментов разработки драйверов для Windows. Перед установкой требует также установки специальной версии SDK, подробная и регулярно обновляемая инструкция по установке описана на https://learn.microsoft.com/en-us/windows-hardware/drivers/download-the-wdk. В некоторых случаях без видимой причины после успешной установки SDK+WDK Visual Studio отображает ошибку подключения заголовочных файлов при открытии и сборке проектов драйверов. В этом случае проблему может решить удаление SDK и WDK и последующая установка EWDK по той же инструкции (см. ссылку выше), либо установка Visual Studio, SDK и WDK на чистую виртуальную машину (например, ту же, на которой будет вестись тестирование драйвера).

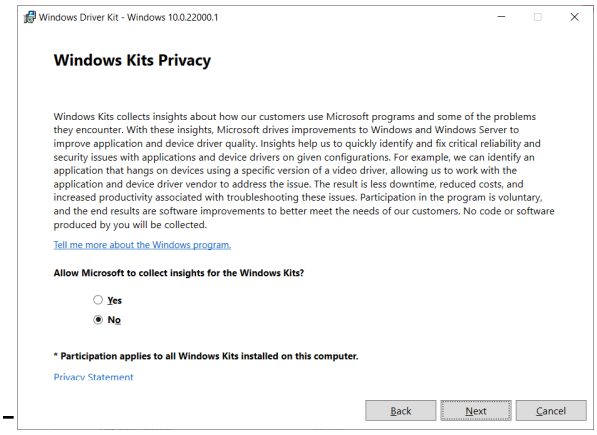

Windows Software Development Kit - Windows 10.0.22000.194
Specify Location
Install the Windows Software Development Kit - Windows 10.0.22000.194 to this computer
Install Path:
C:\Program Files (x86)\Windows Kits\10\
Browse...
* Windows Kit common installation path used
Download the Windows Software Development Kit - Windows 10.0.22000.194 for installation on a separate computer
Download Path:
C:\Users\1PC\Downloads\Windows Kits\10\WindowsSDK
Browse...
Estimated disk space required: 3,4 GB
Disk space available: 132,9 GB
Next
Cancel

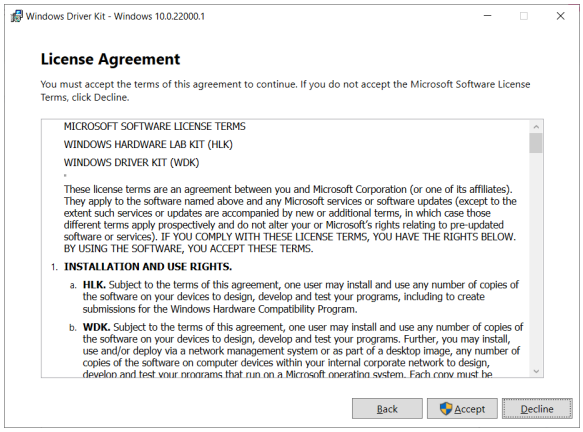

Windows Software Development Kit - Windows 10.0.22000.194
Windows Kits Privacy
Windows Kits collects insights about how our customers use Microsoft programs and some of the problems they encounter. With these insights, Microsoft drives improvements to Windows and Windows Server to improve application and device driver quality. Insights help us to quickly identify and fix critical reliability and security issues with applications and device drivers on given configurations. For example, we can identify an application that hangs on devices using a specific version of a video driver, allowing us to work with the application and device driver vendor to address the issue. The result is less downtime, reduced costs, and increased productivity associated with troubleshooting these issues. Participation in the program is voluntary, and the end results are software improvements to better meet the needs of our customers. No code or software produced by you will be collected.
Tell me more about the Windows program.
Allow Microsoft to collect insights for the Windows Kits?
Yes
No
* You are currently participating in the CEIP program. Participation applies to all Windows kits installed on this computer.
Back
Next
Cancel

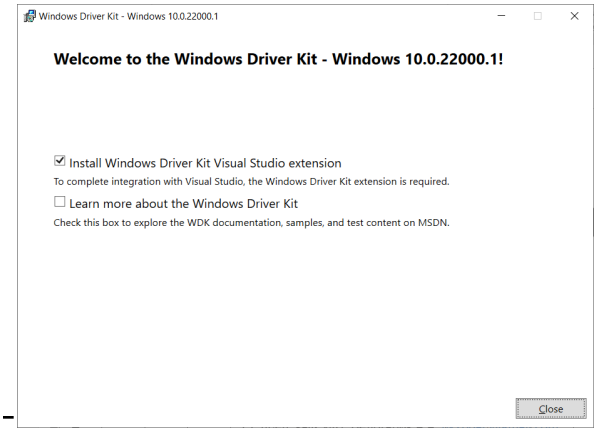

Windows Software Development Kit - Windows 10.0.22000.194
License Agreement
You must accept the terms of this agreement to continue. If you do not accept the Microsoft Software License Terms, click Decline.
MICROSOFT SOFTWARE LICENSE TERMS
MICROSOFT WINDOWS SOFTWARE DEVELOPMENT KIT (SDK) FOR WINDOWS
These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to the software named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft
APIs (i.e., APIs included with the installation of the SDK or APIs accessed by installing extension packages or service to use with the SDK),
updates,
supplements,
internet-based services, and
support services
for this software, unless other terms accompany those items. If so, those terms apply.
By using the software, you accept these terms. If you do not accept them, do not use the software.
Back
Accept
Decline

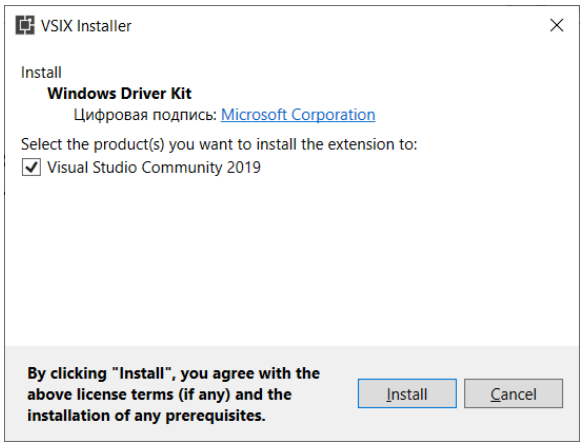

Windows Software Development Kit - Windows 10.0.22000.194
Select the features you want to install
Click a feature name for more information.
Windows Performance Toolkit
Debugging Tools for Windows
Application Verifier For Windows
.NET Framework 4.8 Software Development Kit
Windows App Certification Kit
Windows IP Over USB
MSI Tools
Windows SDK Signing Tools for Desktop Apps
Windows SDK for UWP Managed Apps
Windows SDK for UWP C++ Apps
Windows SDK for UWP Apps Localization
Windows SDK for Desktop C++ x86 Apps
Windows SDK for Desktop C++ amd64 Apps
Windows SDK for Desktop C++ arm Apps
Windows SDK for Desktop C++ arm64 Apps
Windows SDK for Desktop C++ arm Apps
Size: 90,8 MB
The Windows SDK for C++ arm Apps
Requires the following features:
Windows SDK Signing Tools for Desktop Apps
Windows SDK for UWP Managed Apps
Windows SDK for UWP C++ Apps
Windows SDK for Desktop C++ x86 Apps
Windows SDK for Desktop C++ amd64 Apps
Estimated disk space required: 2,9 GB
Disk space available: 132,9 GB
Back
Install
Cancel


Windows Driver Kit - Windows 10.0.22000.1
Specify Location
Install the Windows Driver Kit - Windows 10.0.22000.1 to this computer
Install Path:
C:\Program Files (x86)\Windows Kits\10\
Browse...
* Windows Kit common installation path used
Download the Windows Driver Kit - Windows 10.0.22000.1 for installation on a separate computer
Download Path:
C:\Users\1PC\Downloads\Windows Kits\10\WDK
Browse...
Estimated disk space required: 2,1 GB
Disk space available: 128,1 GB
Next
Cancel

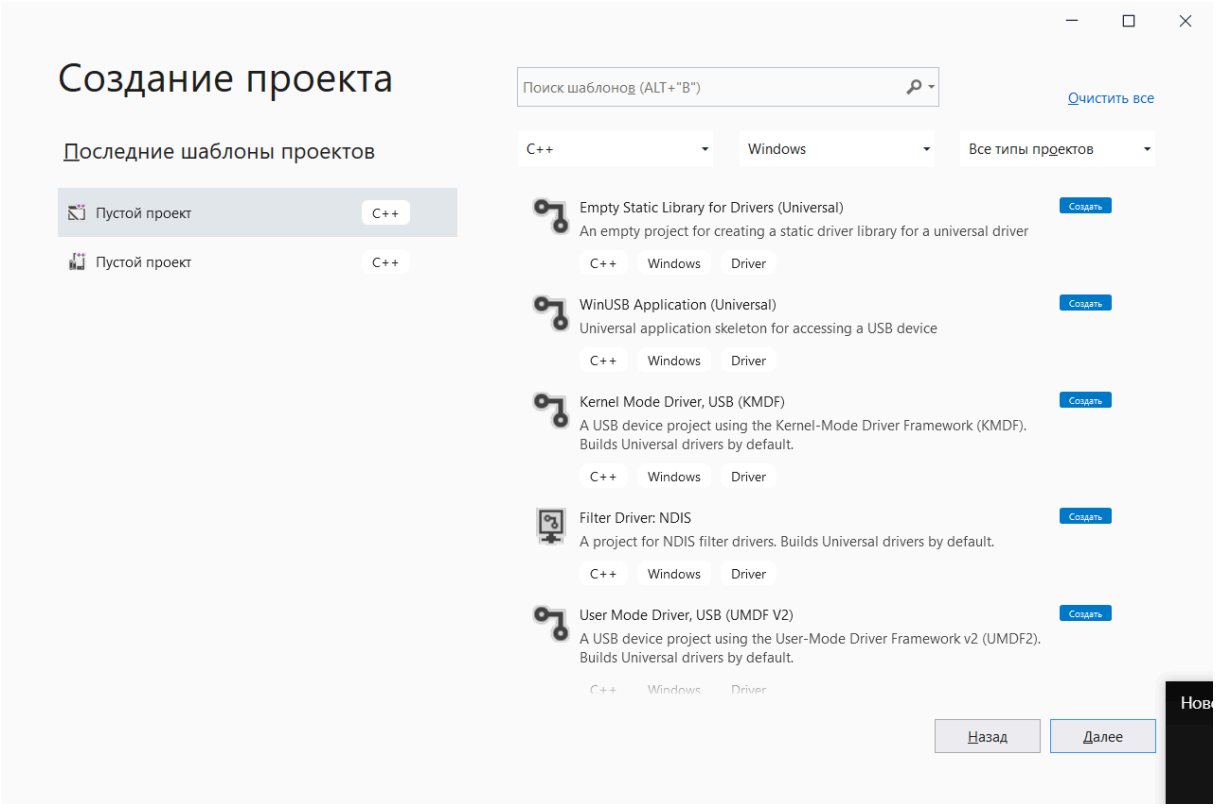

Windows Driver Kit - Windows 10.0.22000.1
Windows Kits Privacy
Windows Kits collects insights about how our customers use Microsoft programs and some of the problems they encounter. With these insights, Microsoft drives improvements to Windows and Windows Server to improve application and device driver quality. Insights help us to quickly identify and fix critical reliability and security issues with applications and device drivers on given configurations. For example, we can identify an application that hangs on devices using a specific version of a video driver, allowing us to work with the application and device driver vendor to address the issue. The result is less downtime, reduced costs, and increased productivity associated with troubleshooting these issues. Participation in the program is voluntary, and the end results are software improvements to better meet the needs of our customers. No code or software produced by you will be collected.
Tell me more about the Windows program.
Allow Microsoft to collect insights for the Windows Kits?
Yes
No
* Participation applies to all Windows Kits installed on this computer.
Privacy Statement
Back
Next
Cancel







После успешной установки WDK и его интеграции с Visual Studio, в VS появятся шаблоны проектов для разработки драйверов:

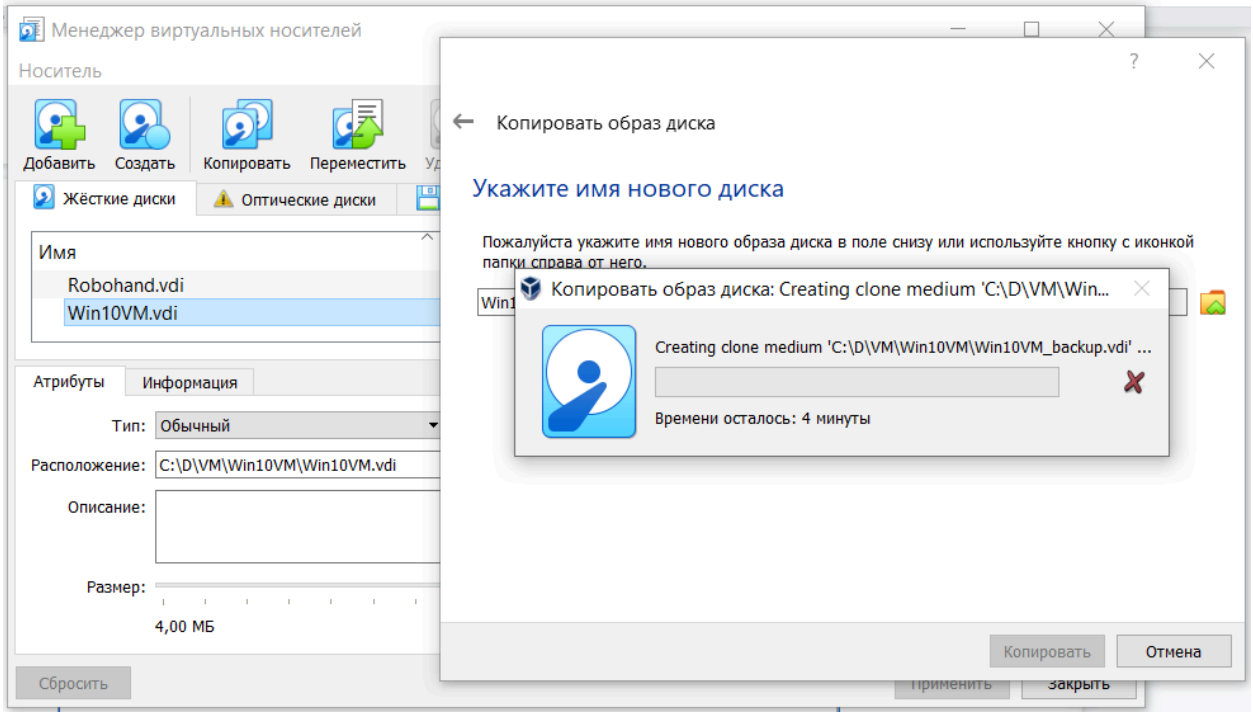


Windows VM на примере Oracle Virtual Box

Виртуальная машина Oracle Virtual Box https://www.virtualbox.org/.

Образ Windows 10 https://www.microsoft.com/ru-ru/software-download/windows10.

Бекап после установки и настройки VM:



WinDbg

Инструкция по установке и ссылка для скачивания на https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/.

DebugView

Инструкция по установке и ссылка для скачивания на https://learn.microsoft.com/en-us/sysinternals/downloads/debugview.

ЛР3. Защита автоматизированной системы на аппаратном уровне. Защита данных в доверенной среде выполнения

Цель

Получение навыков технических методов защиты информации с помощью аппаратных средств доверенной среды выполнения центрального процессора.

Задачи

1. Ознакомиться с основами работы доверенных сред выполнения и их многообразием в различных архитектурах современных центральных процессоров.
2. Реализовать пример защищённого хранилища информации в защищённом анклаве Intel SGX.
3. Реализовать клиентское приложение, запрашивающее данные в анклаве.
4. Проанализировать модель безопасности разработанного программного комплекса из анклава и клиента.

Необходимый теоретический материал

1. Безопасная среда выполнения (TEE). Основные концепции, схема функционирования, схема защиты данных в TEE, применение анклавов TEE в пользовательском и промышленном ПО.
2. Реализации TEE на распространённых аппаратных платформах ARM, Intel, AMD, PowerPC, MIPS.
3. Библиотечные ОС для TEE.

Последовательность выполнения

Фактически, данная лабораторная воспроизводит пример от Intel https://www.intel.com/content/www/us/en/developer/articles/guide/getting-started-with-sgx-sdk-for-windows.html с незначительными изменениями и значительными упрощениями.

Этап 1. Разработка и тестирование незащищённого клиентского приложения.

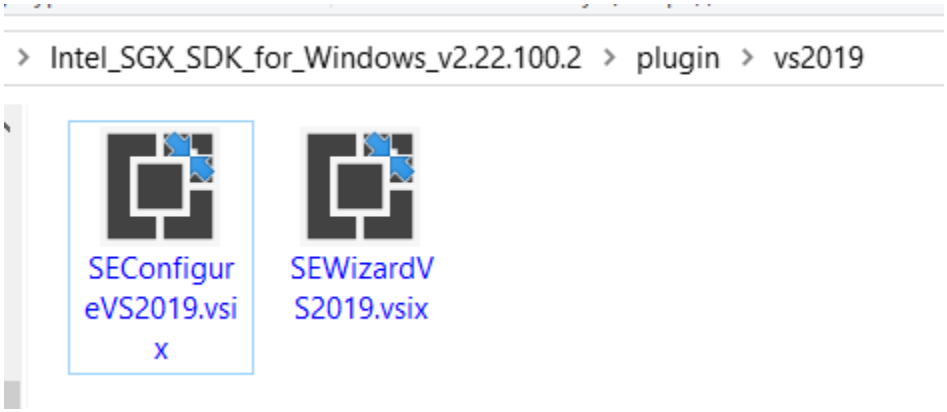
1. Реализовать в графическом либо консольном приложении простейший функционал по работе с данными:
    a. Табличное хранилище данных (к примеру, массив строк, который имитирует хранилище персональных данных, которые нужно защитить),

b. Функцию для получения элемента данных по индексу,
c. Простой интерфейс для запроса элемента данных пользователем по индексу и выводом данных в интерфейс. Предусмотреть обработку при запросе несуществующих элементов массива.

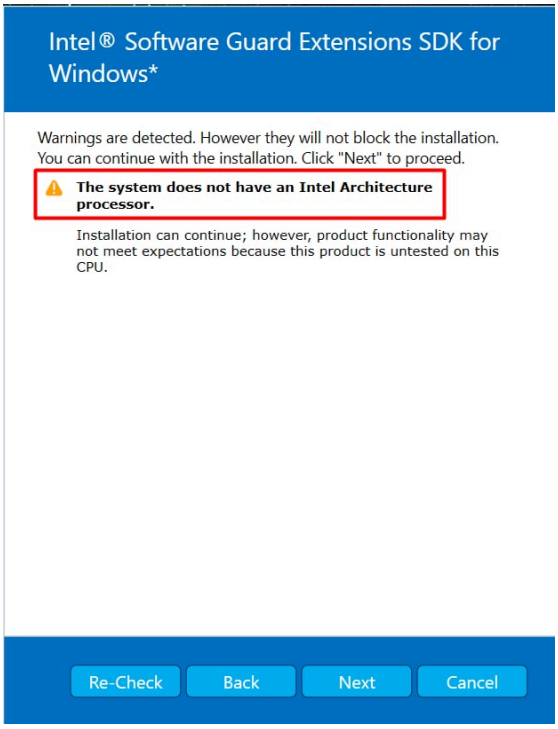
2. Протестировать приложение и убедиться в его работоспособности (без защиты SGX).

Этап 2. Перенос массива данных в защищённый анклав IntelSGX.

1. Установить из каталога plugins предоставленного архива Intel SGX SDK 2 расширения Visual Studio 2019 (следует устанавливать при выключенном Visual Studio):

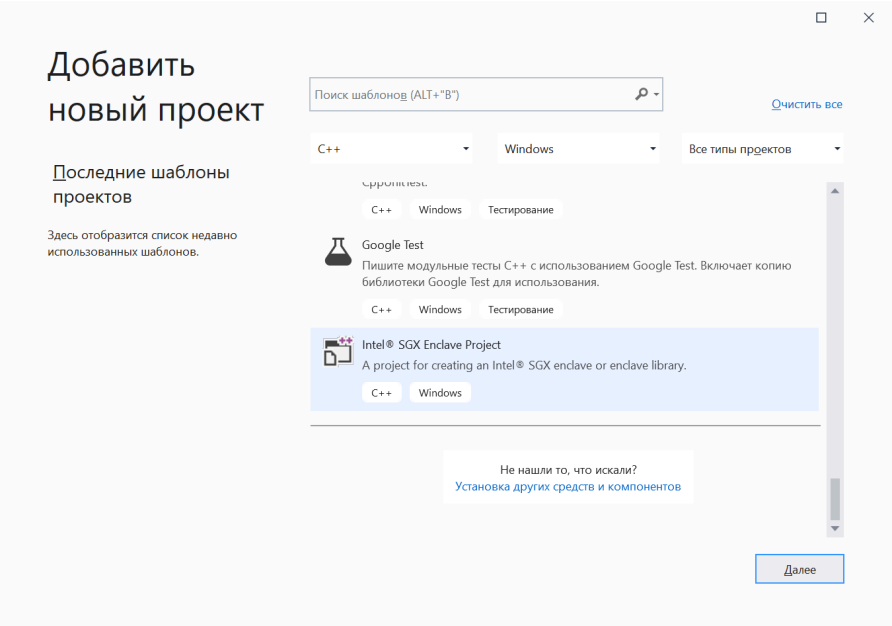


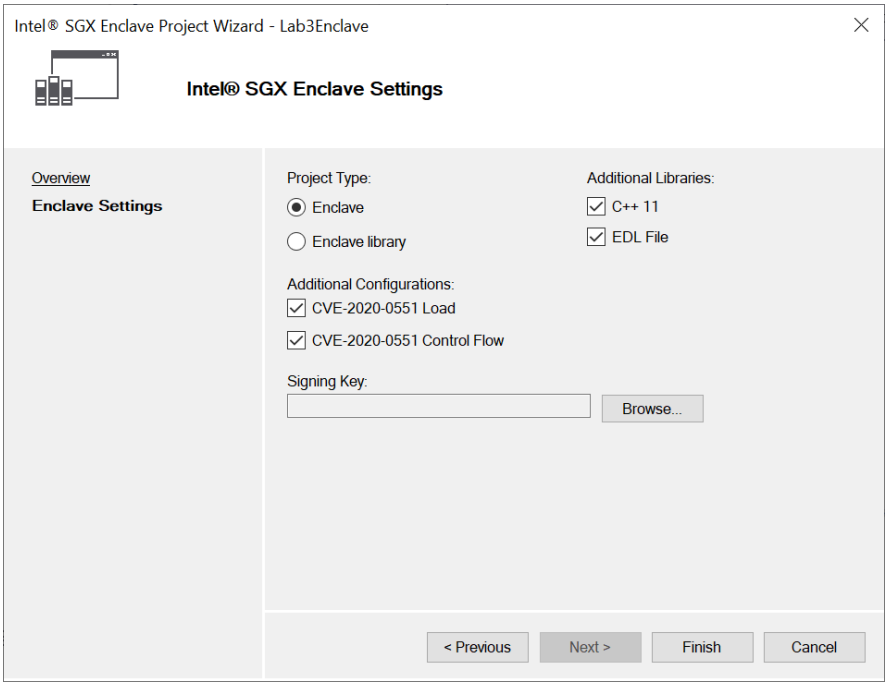
~~(устарело с 2023 года) Установить Intel SGX SDK (обязательно) и Intel SGX PSW (не обязательно, только при самостоятельном решении учащегося запустить код не в режиме симуляции, а в реальном аппаратном анклаве) (ссылка для скачивания https://registrationcenter.intel.com/en/forms/?productid=2614&pass=yes). В случае использования ЦП иного производителя, нежели Intel, инсталлятор может вывести предупреждение:~~



~~В этом случае всё равно необходимо продолжать установку, так как специфические инструкции Intel не потребуются и лабораторная будет запускаться в симуляционном режиме.~~

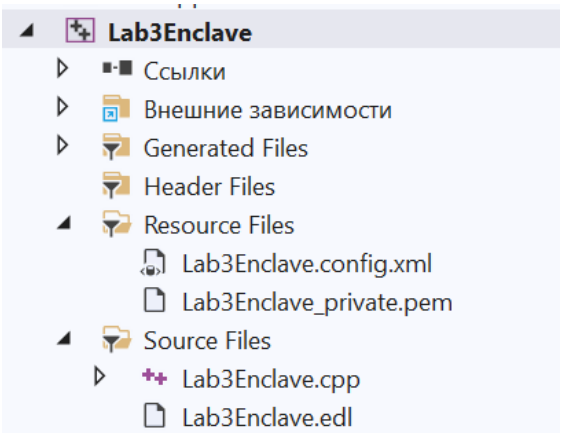
2. Запустить Visual Studio и в том же решении (solution) из этапа 1 создать второй проект “Intel SGX Enclave Project”:





В итоге будет создан проект, включающий как минимум 4 файла:

- Файл исходного кода,
- Файл с описанием анклава *.edl (Enclave Description Language),
- Файл свойств анклава *.xml,
- Криптографический RSA-сертификат *.pem (если пользователь не выбрал свой ключ шифрования, сертификат генерируется автоматически по умолчанию).



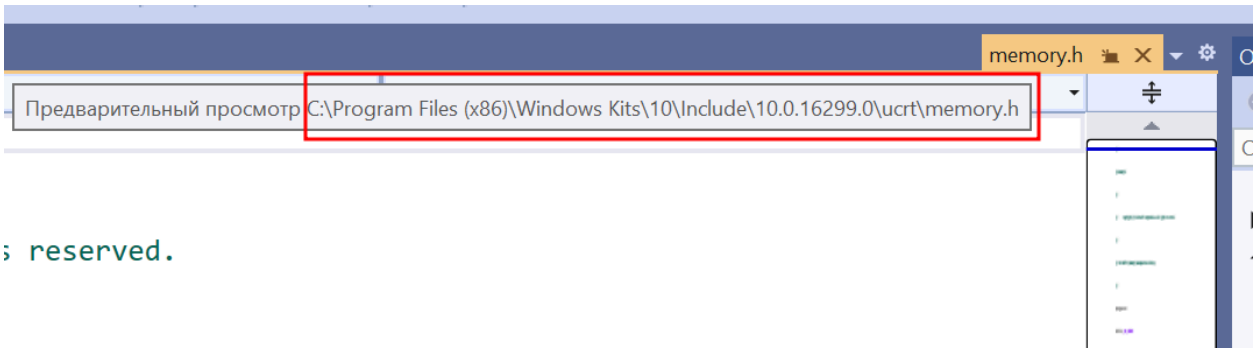
3. В файл исходного кода анклава перенести функцию для работы с защищенным хранилищем из проекта клиентского приложения.

4. Добавить описание реализованной функции в файл EDL. Пример функции с интерфейсом void getPD(char * outbuffer, const int i) из Lab3Enclave.cpp в Lab3Enclave.edl:

```
enclave {
    from "sgx_tstdc.edl" import *;
    trusted {
        /* define ECALLs here. */
        public void getPD([out, size=len] char* outbuffer, size_t len,
const int i);
    };

    untrusted {
        /* define OCALLs here. */
    };
};
```

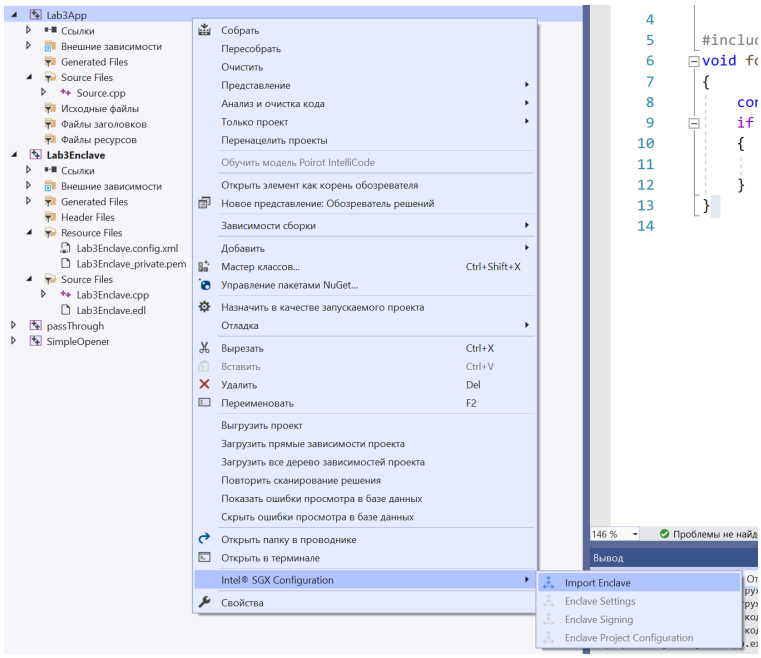
5. Собрать проект с настройками “Simulation” и “x64”. Убедиться, что проект собрался успешно и была сгенерирована библиотека имя_проекта.signed.dll.
6. Также следует обратить внимание, что из заголовочных файлов стандартной бибилотеки C, подключаемых в cpp-файл анклава, необходимо использовать только те, которые есть в IntelSGX SDK. Например, <string.h> подключается из IntelSGX SDK, а <memory.h> - нет:

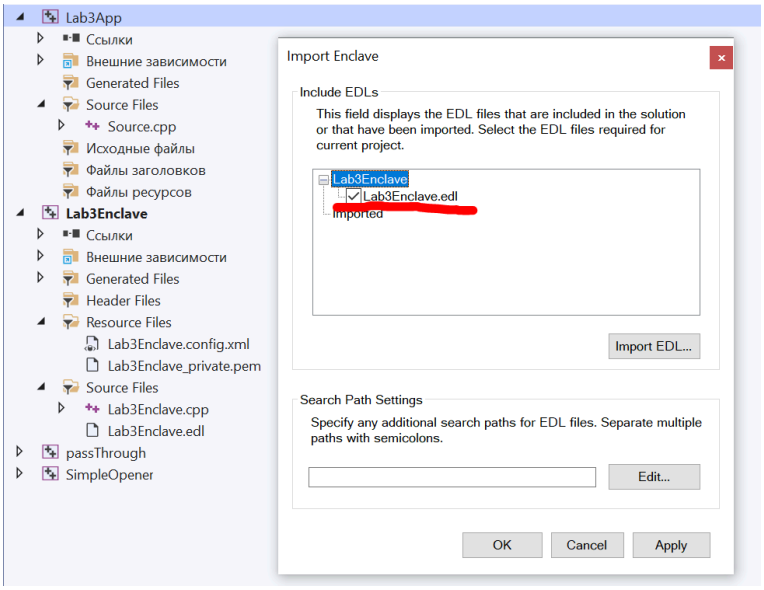
\



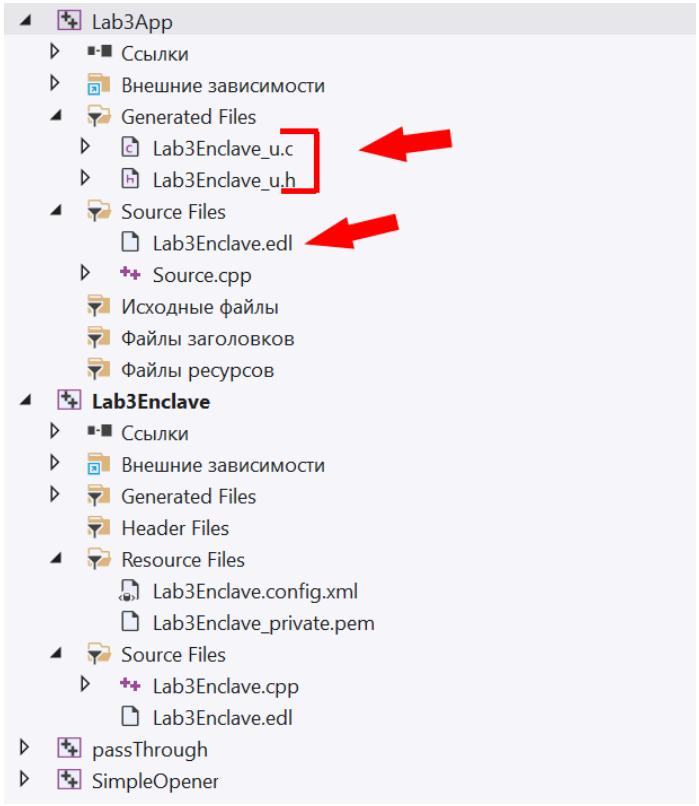
Этап 3. Интеграция анклава в клиентское приложение.

1. Подключить NuGet-пакет Intel SGX SDK в проект клиентского приложения так же, как он был подключен в проект анклава.
2. Вернуться к проекту клиентского приложения. Вызвать в контекстном меню проекта Visual Studio команду импорта анклава:





После успешного импорта в проекте клиентского приложения будет добавлено EDL-описание созданного анклава, а также два пустых файла *.h и *.c:



7. Доработать исходный кода клиентского приложения:

a. Включить в cpp автоматически сгенерированный заголовок “название_проекта_u.h” (название в дереве проекта, см. ниже),
b. Добавить код для инициализации и деинициализации анклава (см. листинг ниже),

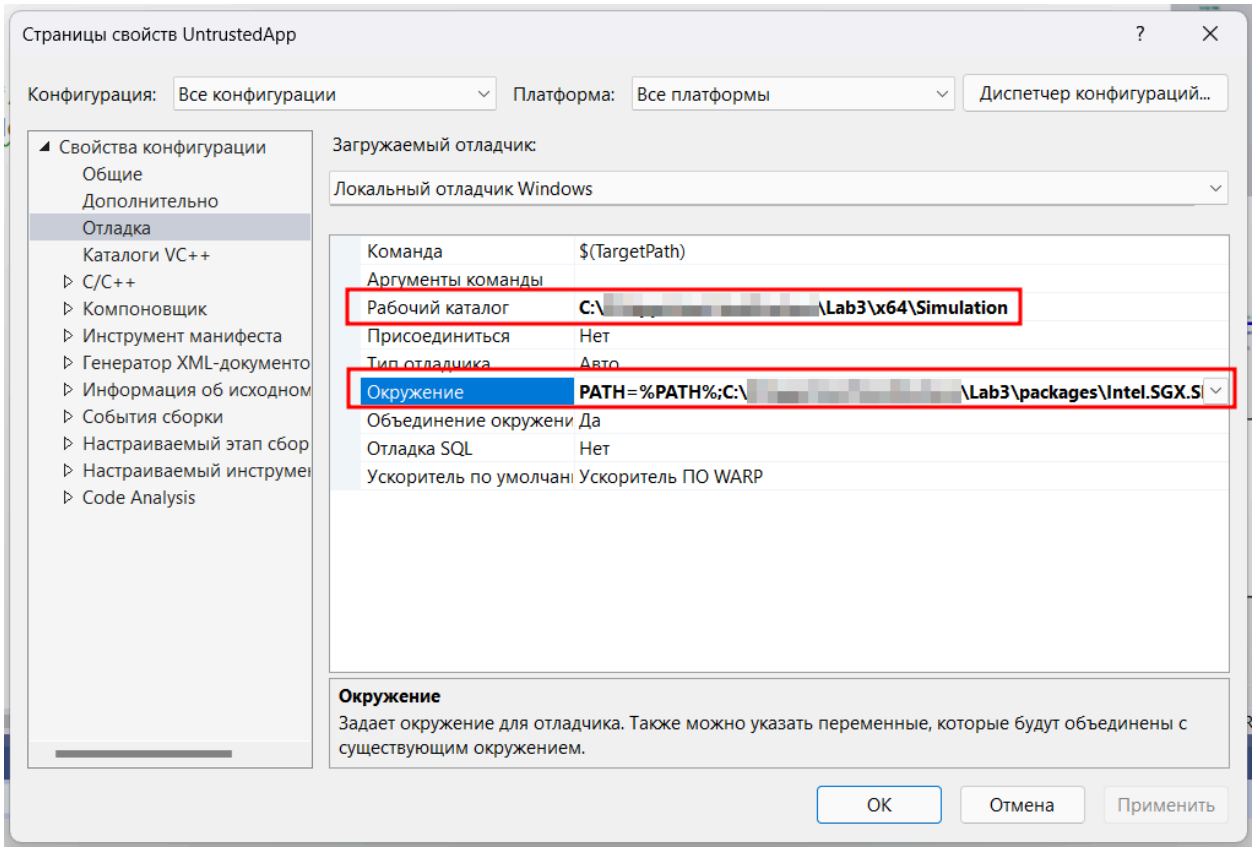
Код для инициализации и деинициализации анклава

```
/* 1. специфичные заголовки для использования анклава */
#include "sgx_urts.h" // вместо #include <string.h>
#include "sgx_tseal.h"
#include "название_проекта_u.h" // !!! вписать свой
#define ENCLAVE_FILE L"название_проекта.signed.dll" // !!! вписать
свой

…

main(){
   ….
   /* 2 активация анклава*/
   sgx_enclave_id_t eid;
   sgx_status_t ret = SGX_SUCCESS;
   sgx_launch_token_t token = { 0 };
   int updated = 0;

   ret = sgx_create_enclave(ENCLAVE_FILE, SGX_DEBUG_FLAG,
&token, &updated, &eid, NULL);
   if (ret != SGX_SUCCESS) {
      printf("App: error %#x, failed to create enclave.\n", ret);
      return -1;
   }
   …
   // далее код из этапа 1
   …
   /* 3. Выгрузить анклав */
   if (SGX_SUCCESS != sgx_destroy_enclave(eid))
      return -1;
```

```
}
```

c. Добавить первым параметров в вызов функции для обращения к хранилищу id создаваемого анклава.

8. В конфигурации сборки выставить Simulation и x64.
9. Добавить в переменную окружения PATH путь до DLL-библиотек Intel SGX SDK (настройки проекта приложения → “Отладка” → “Окружение” → PATH=%PATH%;C:\путь_до_SDK\build\native\bin\x64\Release(см. рис. ниже).
10. Добавить там же в “Рабочий каталог” путь до каталога, где после компиляции анклава хранится библиотека с расширением signed.dll (см. рис. ниже).



11. Собрать и запустить клиентское приложение приложение, убедиться в его работоспособности.

~~Дополнительные задания (выполняются только по согласованию с преподавателем индивидуально)~~

1. ~~На базе примера “SealUnseal” из Intel SGX SDK дополнить ЛР3 следующим функционалом:~~
    a. ~~При завершении работы клиентского приложения должно выполняться сохранение таблицы данных в анклаве в объект, защищённый шифрованием (seal), вывод из анклава в незащищённое клиентское приложение и сохранение объекта в файл.~~
    b. ~~При загрузке клиентского приложения должно выполняться считывание защищённого объекта из файла, передача его в анклав и расшифровка (unseal).~~
2. ~~В менеджере паролей (Лабораторная работа №2) реализовать расшифровку хранилища учётных записей в анклав, хранение и модификацию (добавление и удаление записей) в анклаве до завершения работы приложения.~~
3. ~~…~~

~~Вопросы на защиту~~

~~В день не более двух попыток.~~

1. Дать определение TEE, анклава, назвать реализации TEE для как минимум двух архитектур ЦП. Что может располагаться в анклаве.
2. В чём заключается защита, предоставляемая TEE коду и данным. Для размещения какого кода и данных предназначен TEE?
3. Дать определение LibOS для TEE. Назвать как минимум 2 примера LibOS. Назвать отличия LibOS от обычных ОС по предоставляемому функционалу.
4. Назвать аппаратные средства, обеспечивающие работу TEE:
    a. Каким образом CPU переключается на исполнение кода, размещённого в TEE,

b. Где расположен ключ шифрования для доступа к данным или коду, расположенному в TEE.

~~Вопросов по АСУ ТП нет)~~

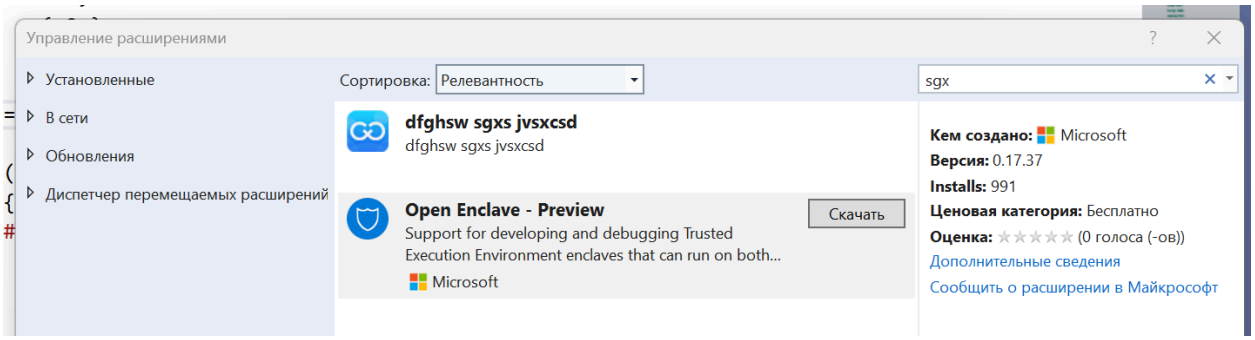
~~Литература~~

1. ~~Getting Started with Intel® Software Guard Extensions SDK for Microsoft* Windows* OS URL: https://www.intel.com/content/www/us/en/developer/articles/guide/getting-started-with-sgx-sdk-for-windows.html.~~
2. ~~Configure Intel SGX on Win10 - URL: https://dqdongg.com/windows/2020/05/31/Windows-sgx.html.~~
3. ~~Selectel. Конфиденциальные вычисления на базе технологии Intel® SGX - URL: https://selectel.ru/lab/anklav/.~~

~~Необходимое ПО и инструкции по установке~~

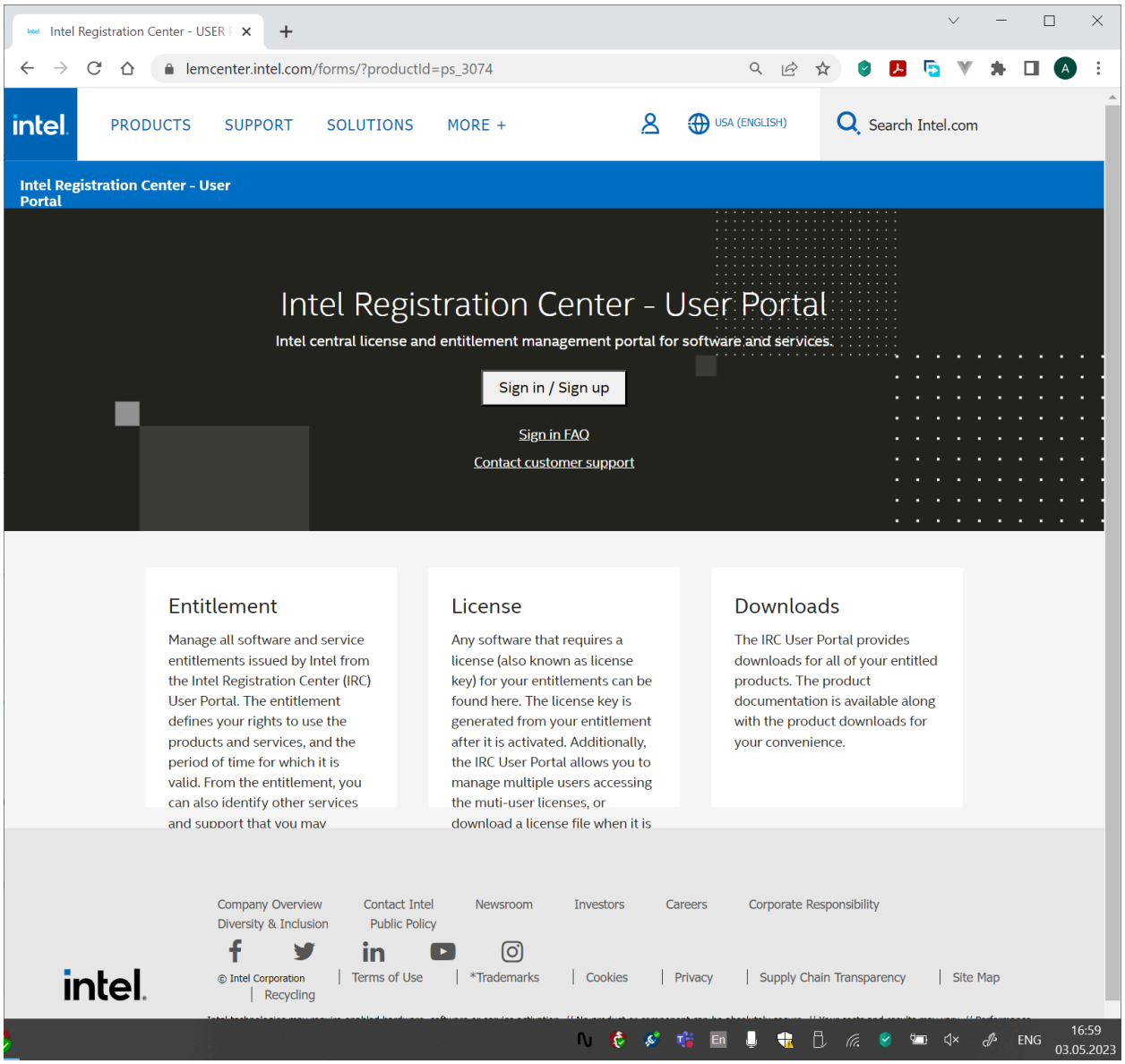
Ссылка для скачивания SGX SDK (**за исключением плагинов-расширений VS**): https://www.nuget.org/packages/Intel.SGX.SDK#supportedframeworks-body-tab.

//TODO Изучить возможность использования плагина VS для разработки анклавов от MS из официального репозитория расширений (также https://marketplace.visualstudio.com/items?itemName=MS-TCPS.OpenEnclaveSDK-VSIX, https://aka.ms/openenclave-vs-extension, https://github.com/openenclave/openenclave):



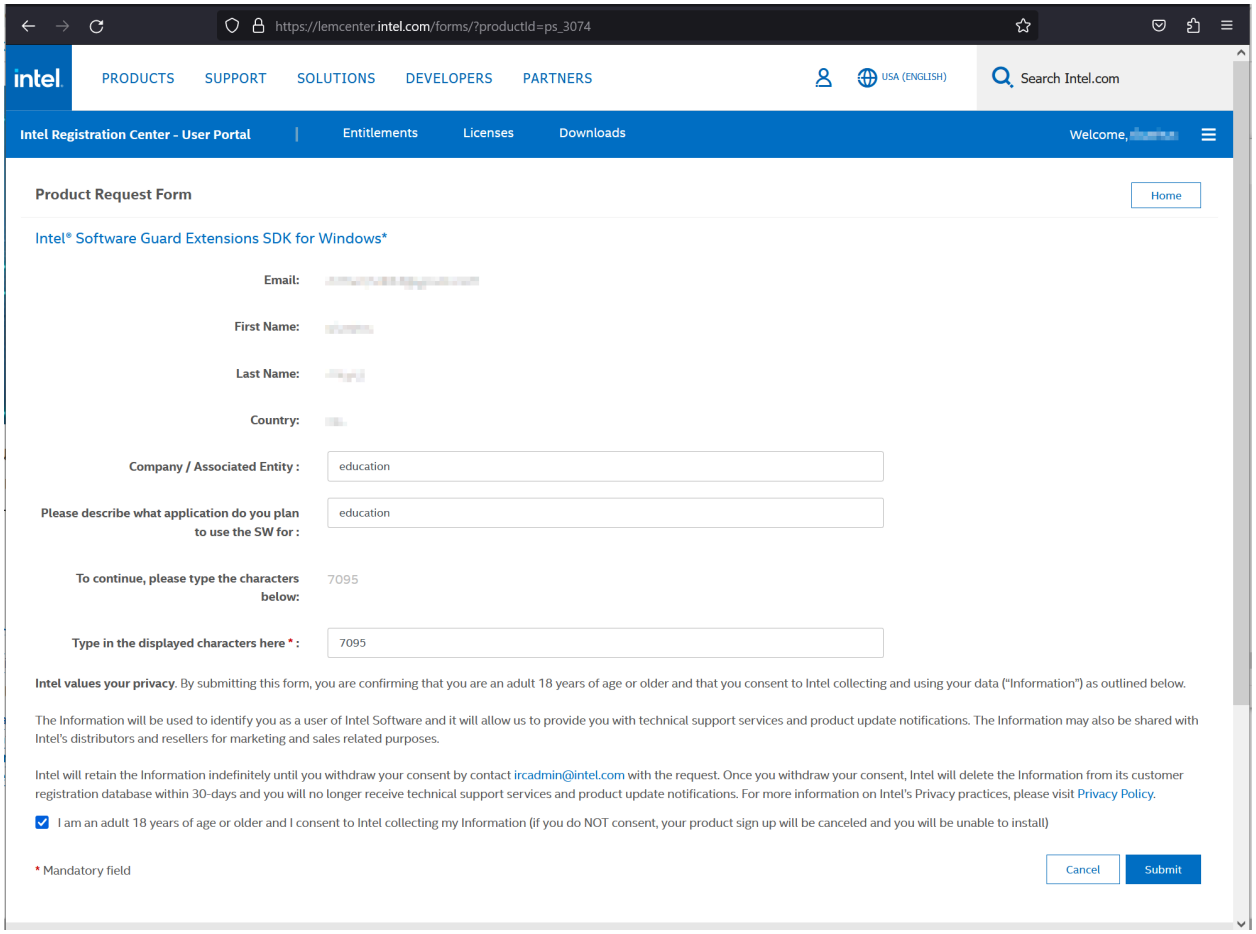
~~Для Windows:~~

1. ~~Перейти на https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/get-started.html, нажать Intel SGX SDK for Windows.~~
2. ~~Зарегистрировать аккаунт Intel.~~

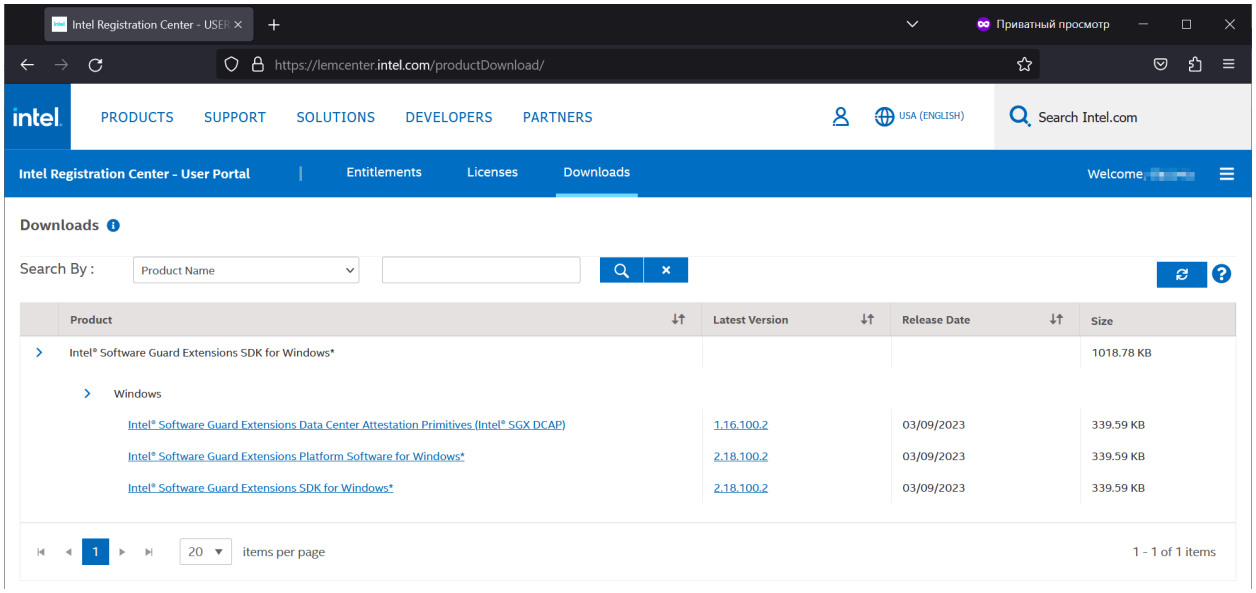


~~3. Снова перейти к закачке Intel SGX SDK for Windows.~~

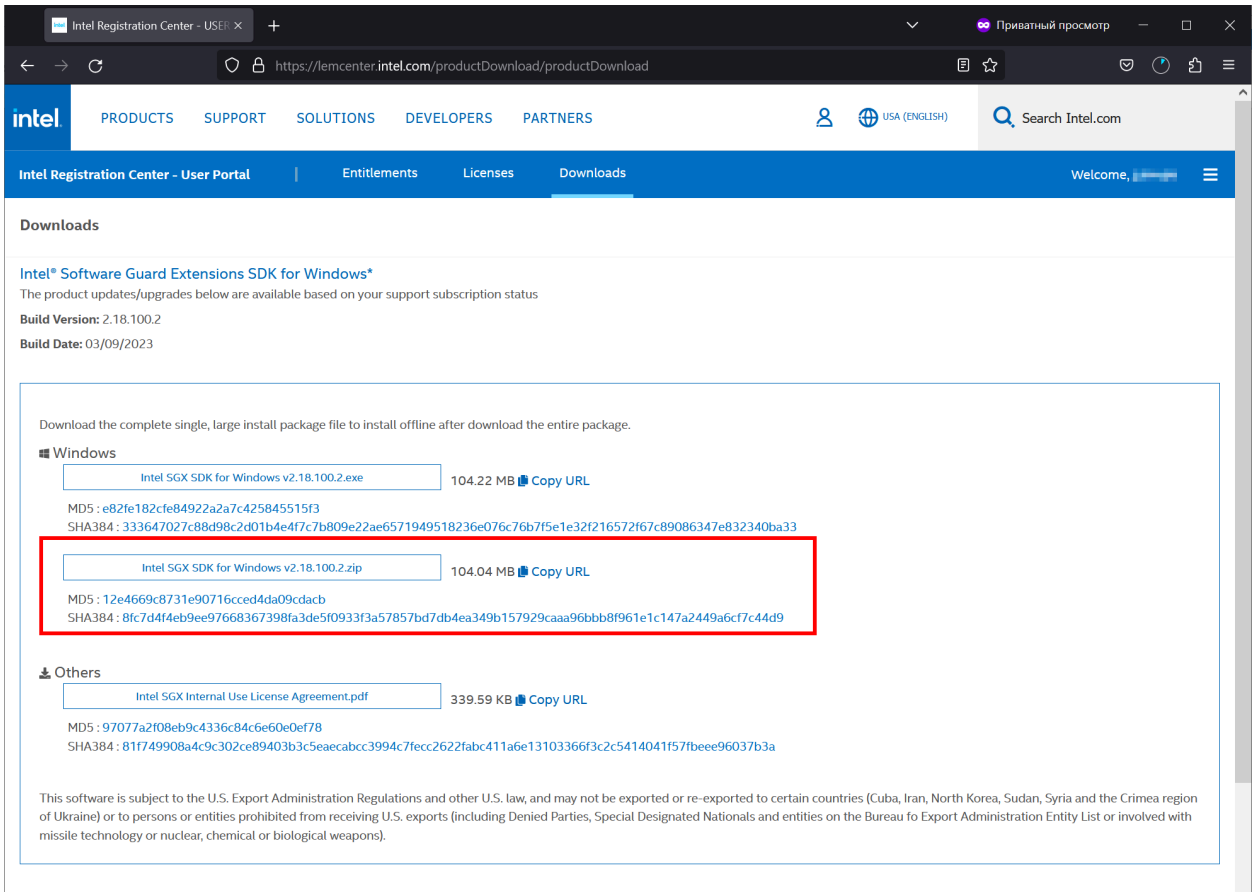
~~4. В поле запроса указать образовательный характер работы:~~



~~5. Дождаться письма от компании и появления в вашем личном кабинете запрошенных продуктов:~~



~~6. Скачать архив с SDK~~



~~Для Linux~~

1. ~~https://github.com/intel/linux-sgx#introduction. Всё =)~~

~~Задания на автомат в первом семестре~~

~~Задания назначены поимённо по вариантам. Автомат выставляется только на обычном экзамене или зачёте (не при пересдаче), при этом должны быть выполнены, защищены и размещены в удалённом репозитории git штатные лабораторные работы ЛР1-ЛР3.~~

~~Задачи на выбор для Экгардта, Рогожина и Коротаева~~

1. ~~(40 баллов каждому при 100% выполнении) Защитить в ЛР1 буфер обмена Windows: если менеджер пароля скопировал в буфер логин или пароль, другие приложения не должны иметь~~

~~доступ к этим данным до тех пор, пока буфер обмена не будет очищен. Приблизительные варианты:~~

~~a. Сделать отдельный буфер обмена (не системный)~~

~~b. Сделать отдельный процесс, который следит за буфером обмена, кто в него что скопировал, запрещает обращение сторонних процессов если там лежит логин/пароль, и стирает содержмое по таймеру~~

~~c. То же самое, но драйвером~~

~~**d. Вариант с имитацией ввода с клавиатуры из менеджера паролей в сторнюю программу уже сделан, этот вариант костыльный и не подходит**~~

~~e. Предложить и обсудить заранее свои варианты перед тем, как приступать к ним~~

~~2. (40 баллов каждому при 100% выполнении) ЛР2, но для memory mapping: добиться, чтобы обращение к файлу через memory mapping также защищалось бы прозрачным шифрованием **через драйвер**. **Продемонстировать защиту**.~~

~~3. (40 баллов каждому при 100% выполнении) ЛР3, но с ARM Thrust Zone (на 25 баллов) + OpTEE (ещё 15 баллов): разработать приложение с защищённым анклавом, как в ЛР3, но оно должно быть не для IntelSGX, а для ARM ThrustZone (в сумме 20 баллов), и, на доп. Баллы, работать в анклаве в доверенной ОС, например, OpTEE или иной (в сумме 40 баллов автоматом - отл)~~

1. (2024) Дубовской. Выполнение ЛР3 в ARM Thrust Zone на RPi 3

1. Выяснить по спецификациям на плату и чип SoC : BCM2837, поддерживает ли он ARM Thrust Zone.
2. Найти, как работать с анклавами на RPi.
3. Запустить на RPi простейший пример приложения в Op-TEE, убедиться, что пример запускается успешно и совместим с RPi.
4. Написать анклав, запускаемый в OpTEE с аппаратной защитой.

5. Написать фронтенд (консольное приложение, выполняющее функцию, аналогичною ЛР3) для обращения к анклаву и получения из него данных.
6. Провести демонстрацию, включающую:
    a. Подключение к RPi по SSH.
    b. Запуск консольного приложения фронтенда.
    c. Успешный вывод из анклава запрошенных данных (элемент массива определённого индекса, введённого во фронтенд) во фронтенд.
7. Подробно объяснить и защитить исходный код.

Ласурова, Маслова

При выполнении - каждой по 20 баллов.

1. Разработать на языке высокого уровня приложение или скрипт для хостовой машины, позволяющий после компиляции драйвера (три файла *.inf, *.sys и *.cat) автоматически перенести их на тестовую виртуальную машину без использования общего каталога. Рекомендуется использовать сетевое соединение между хостом и гостевой ОС и один из протоколов, позволяющий передавать файлы (например, http).
2. Разработать на языке высокого уровня приложение или скрипт для виртуальной машины, автоматически устанавливающее и запускающее принятый с хоста драйвер. Рекомендуется реализовать приложение как сервер, принимающий файлы, сохраняющий их, устанавливающий и запускающий.

Солдатов. Выполнение ЛР3 в ARM Thrust Zone на ПК

1. Выяснить по спецификациям на ноутбук и ЦП, поддерживается ли Thrust Zone. И можно ли его включить в UEFI (и включен ли он в уже).
2. В случае, если архитектура и настройки позволяют воспроизвести ЛР3 - воспроизвести её.

3. План Б. В случае, если располагаемый компьютер не позволяет запустить реальный защищённый анклав ThrustZone - то в симуляционном режиме выполнить ЛР3, но используя SDK thrustzone или ОС Op-TEE.

Тютичкин в гр. 221-329, Абдулаева и Шакиров в гр. 221-3210 (25+25 в случае вы)

Бонусные баллы:

- Тютичкин (работает один) - 50 баллов в случае успешного выполнения
- Абдулаева и Шакиров - по 25 баллов в случае успешного выполнения

1. Реализовать в ЛР1 расшифровку файла учётных данных с помощью биометрических данных лица. Пример: автор должен представить лицо в веб-камеру, и разработанное ПО должно из биометрических параметров лица рассчитать ключ для расшифровки файла учётных данных. **Ключевое условие: между биометрическими параметрами лица и ключом для расширования должна быть математическая связь!!!**
2. Реализовать без использования готовых API и биометрических функций операционной системы (таких как биометрическая разблокировка в Android или Windows Hello), а с помощью библиотек машинного зрения и машинного обучения.
3. У Тютичкина, и у команды из Абдулаевой и Шакирова должны использоваться различные библиотеки и не прослеживаться общие заимствования.
4. Все прочие элементы работы (архитектура, подходы, библиотеки, особенности реализации) следует перед реализацией обсудить с преподавателем.
5. Для защиты:
    a. Провести демонстрацию, включающую разблокировку файла учётных данных лицом автора, а также запрет на вход в менеджер паролей чужим лицом

b. Описать подробно работу своего кода

~~1. Размещение защищённого хранилища в библиотечной ОС в TEE~~

~~191-351: Давлетбаев (задание довольно простое, третий не нужен)~~

~~191-331: Борисенко, Мамонова (задание довольно простое, третий не нужен)~~

~~1) Выбрать одну из библиотечных ОС для Intel SGX либо ARM Thrust Zone (см. ссылки в слайдах). При выборе руководствоваться критериями:~~

~~a. возможности тестового запуска без специфического оборудования (эмуляция или симуляция),~~

~~b. интенсивности поддержки проекта разработчиками,~~

~~c. наличия примеров.~~

~~2) Организовать в библиотечной ОС хранилище данных (с помощью самостоятельно разработанного приложения либо готового решения) и интерфейс для запроса данных по индексу.~~

~~3) Разместить в хранилище таблицу с данными (численными либо текстовыми).~~

~~4) Разработать клиентское ПО для RichOS, выполняющее запрос данных с выбранным индексом к хранилищу, получение и отображение данных. Продемонстрировать в режиме симуляции при отсутствии аппаратной поддержки на располагаемой машине.~~

~~2. Защита учётных данных от перехвата в буфере обмена~~

191-351: Лагутов, Лунин, Ефремов’99 (можно кого-то третьего, хотя не обязательно)
191-331: Горшков, Гудков (можно кого-то третьего, хотя не обязательно)

1) Найти методы защиты данных от кражи из буфера обмена:
   a) С помощью ловушек API (Windows API Hook) https://www.apriorit.com/dev-blog/166-clipboard-protection5.
   b) Другие методы на своё усмотрение, например, драйвер уровня ядра.
   c) Колхозный способ, но используется как альтернатива буферу обмена, для которого нет системных средств защиты: создать глобальное сочетание клавиш (https://www.evileg.com/en/post/165/, https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-registerhotkey), например, Alt+C, и при нажатии этого сочетания имитировать ввод с клавиатуры в том месте, где стоит курсор (https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-sendinput).
2) Реализовать запрет на вставку пароля из буфера обмена куда-либо кроме браузера. Продемонстрировать работу техники.
3) Реализовать в менеджере паролей очистку буфера обмена в течение 20 с после копирования туда пароля в открытом виде.

3. Защита процесса от дампа памяти с помощью драйвер-фильтра

191-351: Филиппович, Фаградян, Барышников
191-331:
181-331: Балабанова, Савушкин, Шумаков

1) Найти API-функцию, ответственную за снятие дампа

- https://improsec.com/tech-blog/user-mode-api-hooks-and-bypasses

- https://stackoverflow.com/questions/51558429/detect-block-read-writeprocessmemory-calls-from-a-driver, https://www.unknowncheats.me/forum/anti-cheat-bypass/271733-driver-aka-kernel-mode.html;
- https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-readprocessmemory;
- https://docs.microsoft.com/en-us/windows/win32/api/minidumpapiset/nf-minidumpapiset-minidumpwritedump.

2) Найти, к каким функциям ядра она обращается, и какие при этом задействуются драйверы и устройства

3) Реализовать драйвер-фильтр, предотвращающий дамп процесса с заданным именем

- https://github.com/EquiFox/KsDumper

4) Продемонстрировать невозможность сбросить дамп памяти менеджера пароля с помощью диспетчера задач, x64dbg/Scylla и ProcDump (https://docs.microsoft.com/ru-ru/sysinternals/downloads/procdump).

4. Реализовать графическое приложение для запуска/останова драйвер-фильтра с помощью вызова API-функции

191-351: Петренко, Щеголькова (задание довольно простое, третий не нужен)
191-331: Медникова, Потапова (задание довольно простое, третий не нужен)
181-331: Москальчук

1) Выяснить, какая API соответствует команде fltmc (https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/loading-and-unloading, https://community.osr.com/discussion/83101/minifilter-filterload-from-application).

2) Реализовать приложение, при запуске отображающее значок в системной области (“system tray”) с пунктами контекстного меню «Загрузить драйвер», «Выгрузить драйвер» и «Выход» (https://doc.qt.io/qt-5/qtwidgets-desktop-systray-example.html)

3) Продемонстрировать с помощью команды fltmc и вывода списка загруженных драйвер-фильтров, что приложение появляется в трее, отображает меню, а загрузка и выгрузка драйвер фильтра действительно работает (возможно, для этого потребуется запустить приложение от имени администратора).

5. Передача драйверу списка расширений, для которых необходимо производить шифрование и расшифровку, из клиентского приложения

191-351: Тохсыров, Сиплатов (задание довольно простое, третий не нужен)
191-331: Малышева, Юрин (задание довольно простое, третий не нужен)
191-331: Мхоян, Умерзакова (то же задание, но передавать не список расширений, а список имён файлов)
181-331: Фурман
181-331: Петряев (то же задание, но передавать ключ шифрования)

1) Найти функции для передачи информации драйверу (https://docs.microsoft.com/en-us/windows/win32/api/fltuser/).

2) Реализовать графическое приложение, которое позволяет ввести строку со списком расширений (разделённых, к примеру, пробелом) и передать их запущенному драйвер-фильтру.

3) Продемонстрировать, что при изменении списка расширений драйвер изменяет своё поведение, обрабатывает файлы из списка и игнорирует остальные (возможно, для этого потребуется запустить приложение от имени администратора).

6. Биометрия (разблокировка приложения при распознавании личности по изображению)

1) Выбрать библиотеку для захвата, обработки и классификации изображений:

   a. OpenCV (метод Eigenfaces – класс EigenFaceRecognizer, метод Fisherfaces - класс FisherFaceRecognizer, метод Local Binary Patterns Histograms – класс LBPHFaceRecognizer)
   https://docs.opencv.org/5.x/da/d60/tutorial_face_main.html,
   https://techvidvan.com/tutorials/face-detection-recognition-opencv-python/,
   https://medium.com/geekculture/i-tried-opencv-face-recognition-in-c-d072e4eac3ab.

   b. TensorFlow Lite
   https://www.youtube.com/watch?v=u6XktO-w4Y4,
   https://github.com/Seeed-Studio/Seeed_Python_reTerminal_QT5_Facerec.

   c. Другие примеры на своё усмотрение.

2) При необходимости, найти в сети дополнительные статьи, уроки и демонстрации по использованию выбранной библиотеки.

3) Обучить классификатор изображений по требуемому количеству своих фото (если с выбранной библиотекой требуется такая операция).

4) Встроить распознавание личности по камере в свой менеджер паролей:

   a. Чтобы наряду с вводом пин-кода приложение можно было разблокировать с помощью биометрической аутентификации по изображению лица с камеры.

b. Получаемое с камеры изображение должно отображаться в панели/окне предварительного просмотра.

c. При распознавании лица должна отображаться вероятность корректного распознавания и никнейм пользователя.

d. После успешного распознавания с вероятностью 0,8 хранилище паролей разблокируется.

5) Продемонстрировать собранный датасет, лог обучения, показать в отладке пошагово изменение состояния приложения при срабатывании распознавания лица.

7. Реализация функций управления хранилищем учётных данных (добавление, удаление, бекап, печать)

191-351: Пономарёв, Ефремов'01 (задание довольно простое, третий не нужен)

191-331: Горшков, Травкин (задание довольно простое, третий не нужен)

181-331: Васюткин

191-351: Балдуев (Rust)

1) Шифрование хранилища паролей должно быть двухслойным:

a. Файла хранилища должен быть зашифрован на диске целиком при выключении приложения

b. После расшифровки всего файла поле с паролем внутри каждой учётной записи требует отдельной расшифровки и не представлено в открытом виде.

2) Реализовать в менеджере паролей функцию добавления учётной записи

   a. кнопку «Добавить», при нажатии на которую вызывается описанное ниже окно/панель,

   b. диалоговое окно/панель для ввода логина, пароля, URL веб-ресурса и кнопку «Сохранить»,

   c. при нажатии кнопки «Сохранить» введённый пароль шифруется, учётные данные добавляются к отображаемому списку, выполняется резервная копия старого файла, а новый зашифрованный файл-хранилище перезаписывается с добавлением новой учётной записи.

3) Реализовать в менеджере паролей функцию удаления учётной записи

   a. Реализовать в интерфейсе приложения необходимые графические элементы для выбора и удаления учётной записи.

   b. Реализовать окно/панель с подтверждением либо отменой удаления учётной записи.

   c. После подтверждения удаления реализовать резервное копирование прежнего файла, удаление выбранной учётной записи из отображаемого списка и перезапись файла-хранилища.

4) Резервное копирование должно производиться при добавлении или сохранении в отдельный каталог, к имени файла должна добавляться дата и время копирования.

5) Печать (сохранение hard-copy учётных записей)

a. В интерфейсе реализовать кнопку для отправки учётных записей на печать.

b. После нажатия кнопки отображать диалог настройки печати (https://doc.qt.io/qt-5/qprintdialog.html)

c. Расшифровать учётные записи в и отправить их на печать без сохранения в какой-либо вспомогательный файл (в Qt классы QPrinter и QPainter).

d. При отсутствии принтера печать демонстрировать в PDF.

8. Защита от присоединения отладчика пользовательского уровня с помощью драйвера уровня ядра

191-351: Поляков, Новикова
191-331: Михайлов Илья
181-331:

1) Выяснить, какие API-функции стоят за присоединением отладчика в usermode (
   a) https://blog.xpnsec.com/anti-debug-openprocess/;
   b) https://books.google.ru/books?id=irTvDwAAQBAJ&lpg=PA68&ots=LmWSC7oj5m&dq=how%20anticheats%20protect%20debugging%20driver&hl=ru&pg=PA68#v=onepage&q=how%20anticheats%20protect%20debugging%20driver&f=false;
   c) https://docs.microsoft.com/ru-ru/windows/win32/debug/process-functions-for-debugging;
   d) https://link.springer.com/chapter/10.1007/978-3-030-52683-2_4
   e) https://docs.microsoft.com/ru-RU/samples/microsoft/windows-driver-samples/obcallback-callback-registration-driver/
   f) https://github.com/notscimmy/libelevate
   g) https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-obregistercallbacks)
2) Выяснить, какой стек драйверов и API-функций ядра задействован в п.1
3) Реализовать минимальный демонстрационный пример - драйвер (возможно типа “драйвер-фильтр”), препятствующий

присоединению с помощью отладчика к процессу с заданным именем.

9. Разработка драйвера ядра Linux для прозрачного шифрования (без FUSE)

191-331: Кузнецов, Шорников (можно кого-то третьего)
181-331:

1) Найти описание стека ввода/вывода Linux https://wxdublin.gitbooks.io/deep-into-linux-and-beyond/content/io.html
2) Определить, чем в стеке Linux можно заменить драйвер-фильтры (либо какой из драйверов будет проще всего модифицировать), какой эквивалент есть у IRP и где пакеты лучше обрабатывать.
3) Реализовать шифрование/расшифровку файлов на уровне ядра со всеми условностями, с которыми выполнялась ЛР1 (привязка к расширению/имени/процессу, ограниченный объём, только разовые read/write).

10. Разработка плагина для бекенд-фреймворка для автоматизации проверки параметров API из запроса клиента по схеме

181-331: Кутузов

1) Разработать плагин для Django REST, обеспечивающий возможность задания (разметки) разработчиком параметров API, их типа, обязателен/не обязателен, min/max, множества возможных значений и другие параметры, задаваемые спецификацией OpenAPI 3.0. Т.е. плагин должен обеспечить задание разработчиком схемы (schema) для параметров API.
    - Как один из вариантов для описания схемы параметров в коде - декоратор. Другие варианты могут быть предложены студентом и обсуждаемы.

2) Автоматизацию проверки параметров, поступающих в request, по заданной пользователем разметке (п.1)

- Если параметры прошли проверку и соответствуют ограничениям - происходит штатная отработка get/post методов в классе APIView.
- Если параметры, полученные бэкендом, не соответствуют схеме, автоматическая проверка плагина генерирует исключение Django типа 404 Bad Request (или с более детальной дифференциацией по кодам и описанию, из числа тех, что уже объявлены в Django и Django REST).

11. Передача шифрованных данных в/из анклава (автомат на оценку 3)

191-351: Давыткина, Валикова
Совместить ЛР3 и пример Intel SGX SDK “SealUnseal”

12. Совместить ЛР2 и ЛР3 (автомат на оценку 3)

191-351: Ализаде, Валявский
Модифицировать ЛР2 таким образом, чтобы расшифровка производилась в анклаве. Допущения и условности:

- можно оставить только один слой шифрования
- можно использовать шифрование анклава, а не то, которое использовалось в ЛР2
- файл с учётными записями неизменный (можно отключить функционал по добавлению/удалению записей)

Темы (в разработке)

1. Законодательство

Модель нарушителя по РД Гостехкомиссии
Классификация угроз безопасности в соответствии с ПП 1119
Альтернативные классификации угроз безопасности
Классификации уязвимостей системы

Этапы формирования модели угроз, создание модели защиты системы.

Аттестация объектов информатизации (ОИ). Схема организации и проведения работ по аттестации ОИ. Функции организации-заявителя, ФСТЭК России и органов по аттестации ОИ. Содержание заявки на проведение аттестации ОИ.

2. Жизненный цикл

3. Моделирование при разработке защищённых АС.

Существующие точки зрения и подходы к моделированию. Модель и классификация нарушителей. Классификация угроз безопасности. Классификация уязвимостей. Модель угроз и защиты.

4. Администрирование и эксплуатация защищенной АС.

Средства обеспечения отказоустойчивости автоматизированной системы. Порядок выполнения обязанностей администратора автоматизированной системы. Управление рисками и инцидентами управления безопасностью. Эксплуатационная документация защищенной автоматизированной системы. Методы мониторинга и аудита, выявления угроз ИБ

ЛР 4. Безопасная разработка автоматизированных систем. Модульное тестирование

Цель

Получение навыков составления автоматических модульных тестов и включения их в процесс сборки и интеграции.

Задачи

1. Реализовать автоматические модульные тесты на одном из популярных фреймворков.
2. Включить тестирование в процесс сборки.

Необходимый теоретический материал

1. Базовые термины и определения по безопасной разработке (DevSecOps).
2. Базовые термины и определения по тестированию ПО.
3. Классификация методов тестирования.
4. Обзор современных популярных фреймворков для модульного тестирования.

Последовательность выполнения

1. Выбрать модуль кода объёмом не менее 100 строк на Python или Си/C++, включающий как минимум 2 функции/метода. Возможно использование кода из какой-либо предыдущей лабораторной работы. Выбранные модули должны различаться у всей группы.
2. Описать для каждой функции п.1 по 2 позитивных и 2 негативных тестовых случая.
3. Реализовать тестовые кейсы для модуля исходного кода на языке высокого уровня на одном из популярных фреймворков автоматизации модульного тестирования.
4. Настроить систему сборки таким образом, чтобы помимо обычной сборки и запуска были реализованы 2 опции:
    a. Сборка и тестирование.
    b. Тестирование.
5. Подготовить отчёт, включающий
    a. Титульный лист.
    b. Раздел “Исходный модуль”, с характеристикой кода, подвергаемого анализу:
        i. его происхождение и назначение.
        ii. Если исходный код модуля умещается на одном листе А4, то привести его целиком с сохранением синтаксической подсветки и нумерации строк. Если исходный код для анализа занимает больший объём - привести ссылку на его репозиторий.

c. Раздел “Тесты” с описанием и исходным кодом модульных тестов (если код тестов самодокументирован, а именно содержит в комментариях или докстринге развёрнутое описание объекта проверки теста и ожидаемый результат, то отдельное описание в тексте отчёта не требуется).
d. Раздел “Интеграция” с описанием последовательности действий по интеграции тестирования в процесс сборки проекта в IDE.

Дополнительные задания (выполняются только по согласованию с преподавателем индивидуально)

1. (10 баллов) Реализовать тесты на четырёх различных
2. …

Вопросы на защиту

1. …
2. …

Литература

1. …
2. …

ЛР 5. Безопасная разработка автоматизированных систем. Тестирование приложения методом фаззинга

Цель

Получение навыков автоматизированного обнаружения ошибок и уязвимостей в коде кода методом фаззинга.

Задачи

1. Реализовать модуль кода или приложение для тестирования.

2. Допустить или намеренно заложить ошибки в некоторые функции приложения.
3. Провести тестирование приложения методом фаззинга.
4. Составить отчёт.

Необходимый теоретический материал

1. Базовые термины и определения по безопасной разработке (DevSecOps).
2. Базовые термины и определения по тестированию ПО.
3. Классификация методов тестирования.
4. Номенклатура популярных фаззеров.

Последовательность выполнения

1. Разработать модуль кода или приложение, исходный код которого будет подвергаться тестированию, и разместить его в публичном репозитории. В качестве примера можно предложить:
    a. Простейший веб-сервер с двумя API для авторизации; один API реализован средствами фреймворка и содержит защиту от ошибок на высоком уровне; второй - аналогичный, но реализован учащимся и не содержит защиты.
    b. Функция, реализующая математическую модель с двумя-тремя входными параметрами.
    c. Задания с https://github.com/mykter/afl-training (по вариантам)
2. Развернуть из open-source репозитория фаззер на выбор студента.
3. Запустить фаззер на тестируемом приложении.
4. *Для использующих MS Restler fuzzer: проанализировать содержимое файлов:
    a. /ResponseBuckets/errorBuckets.json (включает наиболее полный лог, какие запросы отправлялись на сервер и какой был получен ответ, с URL, заголовками и телами запросов и ответов),

b. /ResponseBuckets/runSummary.json (включает краткий лог, какие коды были получены от сервера и ссылку на полный лог).

5. Составить отчёт в электронном виде, включающий
   a. Титульный лист.
   b. Краткое описание разработанного тестового модуля либо приложения, и ссылку на репозиторий с тестовым приложением.
   c. Снимок или текст отчёта, полученного фаззером. Его краткая интерпретация (половина страницы А4, своими словами):
      i. Получены ли ошибки сервера (код 5XX)
      ii. Чем они обусловлены
      iii. Как их исправить
6. Шаблон названия отчёта (файла): “LR4_Lastname_181_331.docx” (не забывайте подставлять номер лабы и фамилию)

Дополнительные задания (выполняются только по согласованию с преподавателем индивидуально)

1. …
2. …

Вопросы на защиту

1. Дать определение тестирования, ошибки (программиста), кейса.
2. Назвать основные типы тестирования (не менее 10).
3. Дать определение функциональным и нефункциональным требованиям к ПО. Назвать виды функционального и нефункционального тестирования, описать, в чём они заключаются.
4.

Литература

1. …

2. …

ЛР 6. Безопасная разработка автоматизированных систем. Статический анализ исходного кода

Цель

Получение навыков обнаружения ошибок и уязвимостей в коде методом статического анализа.

Задачи

1. Ознакомиться с теоретическим материалом по статическому анализу и анализаторам.
2. Провести обработку модуля исходного кода одним из открытых анализаторов.
3. Проанализировать результаты вывода статического анализатора
4. Внести исправления в исходный модуль.

Необходимый теоретический материал

1. Базовые термины и определения по безопасной разработке (DevSecOps).
2. Базовые термины и определения по тестированию ПО.
3. Классификация методов тестирования.
4. Перечень современных статических анализаторов.

Последовательность выполнения

1. Загрузить и установить открытый статический анализатор по варианту.
2. Выбрать модуль кода объёмом не менее 100 строк на языке, входящем в список поддерживаемых статическим анализатором языков. Возможно использование кода из какой-либо предыдущей лабораторной работы. Выбранные модули должны различаться у всей группы.
3. Провести анализ выбранного модуля с помощью статического анализатора.

4. Интерпретировать результаты анализа
    a. Перечислить, какие ошибки или предупреждения найдены и чем они обусловлены,
    b. Привести гипотезы, насколько они критичны для работы модуля,
    c. Привести гипотезы, как их исправить.
5. Исправить как минимум одну ошибку в исходном коде, снова подвергнуть его анализу статического анализатора и убедиться, что количество ошибок уменьшилось.
6. Подготовить отчёт, включающий:
    a. Титульный лист по образцу (см. ссылку в начале файла).
    b. Раздел “Исходный модуль”, с характеристикой кода, подвергаемого анализу:
        i. его происхождение и назначение.
        ii. Если исходный код модуля умещается на 1 листе А4, то привести его целиком с сохранением синтаксической подсветки и нумерации строк. Если исходный код для анализа занимает больший объём - привести ссылку на его репозиторий.
    c. Раздел “Статический анализ”, содержащий
        i. команды запуска и распечатку вывода статического анализатора (если она имеет форматирование с цветом или стилем шрифта - то с сохранением исходного форматирования, например, скриншотом)
        ii. Интерпретацию результатов работы анализатора (см. выше п. 4)
    d. Раздел “Исправление ошибок”, включающий описание, что было изменено в исходном коде для устранения ошибок, и новую распечатку статического анализа, доказывающую, что количество ошибок уменьшилось.
7. Шаблон названия отчёта (файла): “LR4_Lastname_181_331.docx” (не забывайте подставлять номер лабы и фамилию)
8. Защитить устно очно отчёт, ответив на вопросы к отчёту и теоретические вопросы.

~~Дополнительные задания (выполняются только по согласованию с преподавателем индивидуально)~~

1. ~~[+10 баллов] Провести дополнительное исследование по анализу одного и того же модуля исходного кода всеми 4-мя рекомендуемыми открытыми статическими анализаторами. Включить в отчёт по лабораторной работе дополнительный раздел “Сравнение статических анализаторов”, включающий:~~
    a. ~~Сравнение количества ошибок и предупреждений, предложенных статическими анализаторами,~~
    b. ~~Другие бросающиеся в глаза различия (разнообразие типов ошибок, информативность вывода и пр.)~~
    c. ~~Вывод о строгости проверки различными анализаторами.~~
2. ~~…~~

~~Вопросы на защиту~~

3. ~~…~~
4. ~~…~~

~~Литература~~

3. ~~…~~
4. ~~…~~

ЛР X. Шаблон главы

Цель

Задачи

5. …
6. …

Необходимый теоретический материал

Последовательность выполнения

9. …
10. …

Дополнительные задания

3. …
4. …

Вопросы на защиту

5. …
6. …

Литература

5. …
6. …

Инструкция по установке необходимого ПО