Фонд оценочных средств
для промежуточной аттестации
по дисциплине
Системное программирование

Направление подготовки (специальность)
10.05.01 Компьютерная безопасность

Направленность (профиль)
специализация № 6 «Информационно-аналитическая и техническая экспертиза компьютерных систем»

Присваиваемая квалификация
специалист по защите информации

Форма обучения
очная

Челябинск 2025 г.

Содержание

1. ПАСПОРТ ФОНДА ОЦЕНОЧНЫХ СРЕДСТВ

Специальность10.05.01 Компьютерная безопасность.
Специализация № 6 «Информационно-аналитическая и техническая экспертиза компьютерных систем».
Дисциплина: **Системное программирование**.
Семестр (семестры) изучения: 6 семестр.
Форма (формы) промежуточной аттестации:
зачёт 6 семестр.
Используется балльно-рейтинговая система для оценивания результатов.

2. ПЕРЕЧЕНЬ ФОРМИРУЕМЫХ КОМПЕТЕНЦИЙ

2.1. Компетенции, закреплённые за дисциплиной

Изучение дисциплины «Системное программирование» направлено на формирование следующих компетенций:

Коды компетенции согласно ФГОС (ОПОП ВО)	**Содержание компетенций согласно ФГОС (ОПОП ВО)**	**Индикаторы достижения компетенции согласно ОПОП**	**Перечень планируемых результатов обучения по дисциплине**
1	**2**	**3**	**4**
ОПК-7	Способен создавать программы на языках высокого и низкого уровня, применять методы и инструментальные средства программирования для решения профессиональных задач, осуществлять обоснованный выбор инструментария программирова-	ОПК-7.1 Знает общие принципы построения, области и особенности применения языков программирования высокого и низкого уровня; знает язык программирования высокого и низкого уровня; знает общие сведения о методах проектирования, документирования, разработки, тестирования и отладки программного обеспечения. ОПК-7.2 Умеет работать с интегрированной средой разработки программного обеспечения; умеет разрабатывать и реализовывать на языке высокого и низкого уровня алгоритмы решения типовых профессиональных задач; умеет применять известные методы программирования и возможности базового языка про-	Знать: – архитектуру и программный интерфейс современных операционных систем. Уметь: – создавать прикладное и системное программное обеспечение для современных операционных систем. Владеть: – навыками реализации программного обеспечения любой сложности с использованием высокоуровневых и низкоуровневых языков программирования.

	ния и способов организации программ	граммирования для решения типовых профессиональных задач. ОПК-7.3 Владеет навыками разработки алгоритмов решения типовых профессиональных задач; владеет навыками разработки, документирования, тестирования и отладки программ.	
ОПК-13	Способен разрабатывать компоненты программных и программно-аппаратных средств защиты информации в компьютерных системах и проводить анализ их безопасности	ОПК-13.1 Знает общие принципы построения и использования современных языков программирования высокого и низкого уровня; современные технологии программирования; показатели качества программного обеспечения; ОПК-13.2 Умеет формализовать поставленную задачу, работать с интегрированными средами разработки программного обеспечения; разрабатывать эффективные алгоритмы и программы. ОПК-13.3 Владеет навыками разработки программных модулей, реализующих задачи, связанные с обеспечением безопасности операционных систем распространенных семейств; навыками разработки алгоритмов для решения типовых профессиональных задач.	Знать: – системную и программную архитектуру современных процессоров. Уметь: – создавать пользовательские программы; – создавать код режима ядра. Владеть: – навыками использования инструментальных средств создания кода под современные операционные системы.

3. СОДЕРЖАНИЕ ОЦЕНОЧНЫХ СРЕДСТВ ПО ДИСЦИПЛИНЕ

3.1 Виды оценочных средств

№п/п	Код компетенции / планируемые результаты обучения	Контролируемые темы/ разделы	Наименование оценочного средства для текущего контроля	Наименование оценочного средства на промежуточной аттестации/№ задания
1.	ОПК-7 ОПК-13	Раздел 1. Системное программирование для процессоров архитектур IA-32, x64, ARM, MIPS	Зачетные задания. Домашние задания. Аудиторные задания.	Вопросы на зачёте. Задания на зачёте.
2.	ОПК-7 ОПК-13	Раздел 2. Системное программирование в Windows NT	Зачетные задания. Домашние задания. Аудиторные задания.	Вопросы на зачёте. Задания на зачёте.
3.	ОПК-7 ОПК-13	Раздел 3. Системное программирование в UNIX	Зачетные задания. Домашние задания. Аудиторные задания.	Вопросы на зачёте. Задания на зачёте.

Типовые задания, критерии и показатели оценивания в рамках текущего контроля представлены в рабочей программе дисциплины (модуля). Полные комплекты оценочных средств и контрольно-измерительных материалов хранятся на кафедре.

3.2 Содержание оценочных средств

3.2.1 Темы для домашних, аудиторных и зачетных заданий

1. Системное программирование для процессоров архитектур IA-32, x64.
2. Системное программирование в Windows.
3. Системное программирование в Linux.
4. Руткиты.

3.2.2 Примеры зачетных заданий

I.

1 часть.

Написать загрузчик, располагающийся в MBR.

Загрузчик должен просматривать таблицу разделов, выводить информацию о всех найденных первичных и логических разделах (начало, размер, тип, признак активности) и ожидать от пользователя указания раздела, с которого грузиться. Ожидать необходимо в течении определённого промежутка времени, по истечении которого грузить первый активный раздел. Если пользователь выбирает раздел для загрузки, то необходимо внести изменения в таблицу разделов: пометить выбранный раздел как активный, чтобы при следующей загрузке он грузился по умолчанию.

При загрузке с раздела следует считать его первый сектор по адресу 0x7c00, предварительно скопировав себя по другому адресу, и передать ему управление.

Должна быть поддержка CHS и LBA адресов.

2 часть.

Написать мини ОС (ядро), работающую в защищённом режиме с виртуальной памятью.

Система вместе с загрузчиком записывается в отдельный раздел (загрузчик системы - в первый сектор раздела).

Загрузчик MBR загрузит загрузчик системы (первый сектор). Загрузчик системы должен загрузить основной код системы. Распределение обязанностей между загрузчиком системы и основным кодом системы произвести самостоятельно. Кто-то из них должен произвести начальную инициализацию системы:

- перепрограммировать контроллер прерываний;
- перевести процессор в защищённый режим;

- настроить IDT, GDT, TSS;
- настроить таблицы страниц;
- включить механизм страничной трансляции;
- передать управление основному коду системы, расположенному в старших 2 ГБ виртуального адресного пространства.

Виртуальное адресное пространство следует разделить на две части. Верхние 2 ГБ (с адреса 0x80000000) для кода и данных системы нулевого кольца. Нижние 2 ГБ (0x000000 - 0x7FFFFFFF) для кода и данных пользовательских программ третьего кольца. Верхние 2 ГБ должны быть защищены от доступа из третьего кольца.

Код системы должен работать в нулевом кольце. Должно быть не менее двух пользовательских процессов, исполняющихся в третьем кольце. Для эмуляции полезной работы один процесс должен бесконечно вызывать функцию вывода строк из статически заданного массива указателей на строки. Другой процесс - бесконечно вызывать функцию поиска минимального элемента в статически заданном массиве и выводить его. Для вывода строк процессы должны обращаться к системе через шлюз, так как непосредственного доступа к видеопамяти (которая должна быть отображена на верхние 2 ГБ) у них быть не должно. По прерыванию таймера (с некоторой периодичностью, например раз в 10 тиков таймера) управление должно переключаться с одного процесса на другой.

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой), 20 баллов.

Требования:

1 часть

1) Загрузчик в MBR должен анализировать таблицу первичных разделов (можно игнорировать расширенные разделы), находить загрузочный раздел и загружать загрузчик из него.

2) Должна быть поддержка CHS и LBA адресации.

2 часть

1) Загрузчик ОС должен загружать основной код системы.

2) ОС должна работать в защищённом режиме с виртуальной памятью (необходимо произвести соответствующую инициализацию).

3) Адресное пространство должно разделяться пополам между кодом третьего и нулевого кольца. Система должна находиться в верхних 2 ГБ виртуальной памяти.

4) Реализовать шлюз, через который код 3 кольца сможет обращаться к ядру для вывода символов (через видеобуфер, отображённый в память

ядра).

5) Корректно обрабатывать прерывания (таймера, клавиатуры, шлюза системного вызова), не нарушая работоспособности прерванной программы.

6) Реализовать тестовый код третьего кольца с одной из указанных функциональностей.

Полная функциональность.

Требования (дополнительно к базовым):

1 часть

1) Анализировать расширенные разделы (предполагать возможность существования до 4-х расширенных разделов), допускать загрузку с логических разделов, запоминать выбор пользователя (изменяя признак активности выбранного раздела), ожидать выбор пользователя в течении определённого времени, по истечении которого загружать активный раздел. 10 баллов.

2 часть

1) Реализовать переключение между двумя процессами (корректно сохранять и восстанавливать контекст, отслеживать кванты времени). 5 баллов (+5 баллов за поддержку произвольного количества процессов).

2) Реализовать управление виртуальной памятью (выделение/освобождение памяти, учет виртуальных страниц и страничных фреймов). 10 баллов.

3) Реализовать драйвер клавиатуры, который будет сохранять в буфере нажатые клавиши. Через шлюз реализовать возможность для программ получения нажатых клавиш (блокирующее ожидание и неблокирующие считывание доступной в буфере клавиши). 5 баллов.

4) Реализовать драйвер жесткого диска, позволяющий считывать и записывать секторы диска. Предоставлять для программ через шлюз интерфейс к функциям этого драйвера. 10 баллов.

5) Реализовать поддержку файловой системы (своей или существующей). Предоставлять для программ через шлюз интерфейс для работы с файловой системой. 20 баллов.

II. Три задания по формату PE.

Так как задания опираются по сути на одни и те же знания, то необходимо выбрать не более двух из них: за одно баллы будут засчитаны по описанию задания, за второе - до 10 баллов (при выполнении всей функциональности).

При сдаче любого задания необходимо уметь объяснить все используемые особенности формата PE и программного окружения Windows.

Задание 1. Программа просмотра PE-файлов.

Повторить на языке ассемблера (x86 или x64) функциональность программы разбора PE-файла (формата PE32 и PE32+) из примера.

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой), 10 баллов.

Требования:

1) Поддерживать один из форматов PE32 или PE32+.

2) Вывод информации о заголовках, о секциях.

3) Вывод дампа указанных секций.

Полная функциональность.

Требования (дополнительно к базовым):

1) Вывод релоков. Для каждого выводимого релока добавить вывод значения корректируемого абсолютного адреса. 3 балла.

2) Вывод таблицы импорта. Для каждой функции добавить вывод адреса таблицы импорта, по котому будет проставлен адрес функции при загрузке. 3 балла.

3) Вывод таблицы экспорта. Обрабатывать перенаправленный экспорт. 3 балла.

4) Обрабатывать оба формата (PE32 и PE32+). При загрузке динамически определять формат. Всю реализуемую функциональность задания сделать для двух форматов. 3 балла.

Задание 2. Программа редактирования PE-файлов.

Реализовать консольную или графическую программу редактирования PE-файлов (форматы PE32 и PE32+).

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой), 10 баллов.

Требования:

1) Поддерживать один из форматов PE32 или PE32+.

2) Редактированиеосновныхполей NT-заголовка (сигнатура"PE", NumberOfSections, TimeDateStamp, SizeOfOptionalHeader, Characteristics, Magic, AddressOfEntryPoint, ImageBase, SectionAlignment, FileAlignment, SizeOfImage, SizeOfHeaders, Subsystem, NumberOfRvaAndSizes)

3) Редактирование всех полей элементов таблицы секций.

4) Обнуление элементов (адрес, размер) таблицы директорий.

Полная функциональность.

Требования (дополнительно к базовым):

1) Модификация таблицы релоков:

a) Модификация заголовка блока (размер, адрес), модификация элемента блока (тип, смещение). 2 балла.

b) Перенастройка всех корректируемых адресов на новый базовый адрес загрузки (с соответствующей корректировкой ImageBase). 2 балла.

c) Добавление нового блока, добавление нового элемента в блок. 2 балла.

2) Модификация таблицы импорта.

a) Модификация дескриптора импорта (все поля), модификация имени библиотеки, модификация имени функции. 2 балла.

+2 балла за возможность указать новые имена библиотек и функций длиннее исходных (необходимо найти новую память).

b) Добавление дескриптора импорта, добавление новой функции в дескриптор импорта. 5 баллов.

3) Модификация таблицы экспорта.

a) Модификация имени библиотеки, модификация имени функции, адреса функции (по имени, по ординалу). 2 балла. +2 балла за возможность указать новые имена библиотеки и функций длиннее исходных (необходимо найти новую память).

b) Добавление новой функции (в массив адресов), добавление нового имени функции (в массив имён). 2 балла.

c) Перенаправление функции в другую библиотеку. 2 балла.

4) Расширение размера файла (добавление новой секции либо расширение последней) для размещения добавляемых элементов. 5 баллов.

Задание 3. Загрузчик PE-файлов.

Необходимо реализовать загрузчик PE-файла (для Win32 и Win64), который должен формировать образ в памяти и передавать управление точке входа.

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой), 10 баллов.

Требования:

1) Поддерживать один из форматов PE32 или PE32+.

2) Предполагать, что модуль всегда грузится по ImageBase.

3) Обрабатывать таблицу импорта. Загружать библиотеки с помощью LoadLibrary, получать адреса экспортируемых функций с помощью GetProcAddress.

4) Выставлять правильные права доступа на секции.

Полная функциональность

Требования (дополнительно к базовым):

1) Корректно обрабатывать возможность загрузки модуля по адресу, отличному от ImageBase (при необходимости, обрабатывать релоки). 3 балла.

2) Самостоятельно загружать библиотеки и поддерживать в актуальном состоянии список загруженных библиотек в PEB. 3 балла.

3) Самостоятельно анализировать таблицу экспорта и получать адреса экспортируемых символов. 3 балла.

4) Обрабатывать оба формата (PE32 и PE32+): 32-х разрядная сборка должна поддерживать формат PE32, 64-х разрядная - PE32+. Всю реализуемую функциональность задания сделать для двух форматов. 3 балла.

5) Добавить возможность передавать загружаемой программе аргументы. Можно поменять содержимое памяти, содержащей аргументы, переданные загрузчику (указатель на эту память возвращается функцией GetCommandLine). Либо перехватывать функцию GetCommandLine (при импорте этой функции подставлять адрес своей функции). При запросе командной строки с аргументами возвращать указатель на память со своими аргументами.

Добавить возможность получать программой свой адрес загрузки. При запросе загружаемой программой своего адреса загрузки (может использоваться, например, при поиске ресурсов) возвращать адрес загрузки программы (вместо адреса загрузки загрузчика, который для системы и является текущей программой). Перехватывать функцию GetModuleHandle (вызывается для получения адреса загрузки библитек по имени, при передачи нуля вместо имени возвращает адрес загрузки самой программы). При вызове этой функции с нулевым аргументом возвращать адрес загрузки программы. 5 баллов.

III. Реализовать серверное и клиентское программное обеспечение, реализующие некоторый сетевой протокол. Приложения должны быть написаны на C/C++ под Windows с использованием WinAPI. Не должны использоваться никакие библиотечные функции языка Си - только WinAPI.

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой), 10 баллов.

Требования:

1) Реализовать свой сетевой протокол (за основу можно взять

существующий: HTTP, FTP, ...), позволяющий выполнять следующие действия: загрузка файлов с сервера, получение листинга.

2) Реализовать серверное и клиентское программное обеспечение с использованием WinAPI.

3) Хранить конфигурационные данные (IP-адреса, порты, каталоги, логины, пароли и т.д.) в ini-файлах или реестре.

4) Каждый новый клиентский запрос обрабатывать в новом потоке сервера.

Полная функциональность.

Требования (дополнительно к базовым):

1) Дополнительная функциональность протокола: выгрузка файлов на сервер, докачка файлов с сервера (т.е. с указанием файлового смещения), получение времени изменения файла (для определения обновления файла), возможность запуска скриптов на сервере с получением их стандартного вывода. 5 баллов.

2) Реализация сервера в качестве службы. 3 балла.

3) Использовать для обработки клиентских запросов пул потоков (с фиксированным количеством потоков): обработка очередного пользовательского запроса назначается некоторому потоку из пула. Использовать порты завершения ввода/вывода для асинхронной обработки файловых и сетевых операций. До 10 баллов.

4) Аутентификация пользователя средствами Windows. Имперсонация клиента. 3 балла.

5) Использование криптографии для шифрования трафика, передаваемого между клиентом и сервером (асимметричная криптография для передачи сеансового ключа и симметричная криптография для шифрования данных на сеансовом ключе), и для аутентификации передаваемых данных (HMAC). 5 баллов.

6) Вынесение логических функциональных модулей (обработка сетевого протокола, функции-обёртки над WinAPI и т.д.) в отдельные динамические библиотеки. В случае реализации, в отдельные библиотеки выносить аутентификацию, криптографию. 5 баллов.

IV. Написать драйвер, эмулирующий ссылку на каталог. При попытке доступа (чтение, запись, получение размера) к устройству драйвера запросы должны перенаправляться к файлам соответствующего каталога. Например, если символьная ссылка на устройство драйвера называется Driver, а каталог, на который ссылаемся \??\C:\, то запрос на чтение:

> red \\.\Driver\file.txt

(драйверу будет передано имя \file.txt) должен перенаправляться к файлу \??\C:\file.txt.

Необходимо написать программу, которая будет устанавливать, запускать, останавливать, удалять драйвер и управлять им с помощью ioctl-кодов. Должен быть предусмотрен специальный ioctl-код, с помощью которого будет задаваться/изменяться целевой каталог. Программа управления драйвером должна запросить этот каталог у пользователя, передать его драйверу и дальше ожидать от пользователя команд (выход, останов драйвера, запрос логов, смена каталога).

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой), 10 баллов.

Требования:

1) Написать драйвер, который запросы (чтение, запись, запрос размера) к "файлам" своего устройства будет перенаправлять к файлам некоторого фиксированного каталога. В случае записи в несуществующий файл он должен создаваться.

Полная функциональность.

Требования (дополнительно к базовым):

1) Написать программу, которая будет загружать/выгружать драйвер (подключаться к загруженному) и посылать ему команды на изменение каталога. 3 балла.

2) При обращении к файлам внутри подкаталогов с запросом на запись все несуществующие подкаталоги и файл должны создаваться (ZwCreateFile). 3 балла.

3) Драйвер должен вести внутренний лог всех обработанных операций (чтение, запись, запрос информации, создание файлов и каталогов) и отдавать лог программе при соответствующем ioctl-запросе. 3 балла.

V. Написать драйвер, обрабатывающий запросы на чтение и запись к своему устройству. Данные, записанные в файл на устройство во время последнего запроса на запись, должны сохраняться в памяти и возвращаться пользовательскому процессу при операции чтения из этого файла. После записи размер файла должен быть равен количеству записанных байт, чтобы программа чтения с устройства могла прочитать все записанные данные. Если при чтении запрашивается больше данных, чем было записано на устройство, то возвращать только записанные данные. Иначе возвращать столько байт, сколько запрошено. Память должна выделяться динамически.

При записи память, выделенная во время предыдущей операции записи, должна освобождаться. Также память должна освобождаться при выгрузке драйвера.

Именем файла является часть пути после имени устройства. Например при обращении к устройству \\.\Driver по имени \\.\Driver\file1 именем файла будет \file1. Имя файла сохраняется в объекте файла, указатель на который сохраняется в пришедшем IRP. Таким образом можно понять к какому файлу на устройстве обращались.

Для записи на устройство и чтения с устройства использовать соответственно

driver_work/tools/wrd/wrd.exe и driver_work/tools/red/red.exe.

Имя устройства передаётся в качестве параметра командной строки:

> wrd.exe \\.\Driver\file1

> red.exe \\.\Driver\file2

В драйвере необходимо организовать список, который имени файла будет сопоставлять его содержимое при запросе на чтение. А при запросе на запись будет создаваться новый элемент списка, или обновляться существующий (если в файл с таким именем уже писали).

10 баллов.

VI. Написать драйвер и программу управления драйвером. Программа должна осуществлять запуск/останов драйвера и осуществлять взаимодействие с драйвером: посылать команды и получать результат.

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой), 10 баллов.

Требования:

1) Программа должна общаться с драйвером через ioctl-коды к устройству драйвера.

2) Программа должна получать от драйвера содержимое IDT и GDT и осуществлять их форматированный вывод.

Полная функциональность.

Требования (дополнительно к базовым):

1) Поддерживать дополнительные способы взаимодействия с драйвером:

- добавить шлюз вызова в GDT;

- добавить шлюз прерывания/ловушки в IDT;

- добавить системный вызов (в существующую таблицу или в новую таблицу)

За каждый реализованный способ 1 балл.

2) Поддерживать дополнительные команды, посылаемые программой драйверу:

- добавить/модифицировать записи IDT/GDT (добавлять шлюзы и проверять их работоспособность);
- чтение/запись памяти по физическим/виртуальным адресам (без PAE и с PAE);
- модифицировать любую запись PDE/PTE (демонстрация на примере выставления бита U/S в 768 PDE без PAE и 1536-1539 PDE с PAE - тогда получим доступ из пользовательского режима к каталогам страниц).

За каждую реализованную команду по 3 балла.

VII. Реализовать руткит режима ядра. Вся функциональность должна быть реализована в драйвере. Должна быть поддержка всех 32-х разрядных ОС с Windows XP до Windows 10.

Руткит должен осуществлять скрытие своего присутствия (собственно руткит-составляющая) и выполнять полезную нагрузку.

Все действия должны настраиваться по сети через некоторый интерфейс управления.

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой), 25 баллов.

Требования:

1) Драйвер должен предоставлять доступ к системе по сети (подключаться к удалённому серверу и/или ожидать входящих соединений). Предполагается использование TDI и/или WSK.

2) Должна быть реализована полезная нагрузка:

- получение команд по сети;
- загрузка и выгрузка файлов;
- кейлоггер;
- дамп http-запросов (на стандартных портах);
- запуск программы на исполнение (можно после перезагрузки).

3) Выбрать метод запуска руткита:

- поставить драйвер;
- поставить службу;
- заразить системный файл (+10 баллов);
- поместить в автозагрузку программу.

Полная функциональность

Требования (дополнительно к базовым):

1) Полезная нагрузка:

- подмена http-трафика, внедрение js-кода (простое внедрение - 3 балла, до 10 баллов за универсальный алгоритм)

2) Функциональность руткита:

- скрывать файлы (3 балла);
- скрывать процессы (3 балла);
- скрывать ключи реестра (3 балла);
- скрывать сетевые подключения (3 балла);
- скрывать драйвер (3 балла);
- повышать привилегии указанного процесса до указанного пользователя либо заменять на системный (2 балла);
- скрывать факт скрытия (до 20 баллов).

3) Сетевое взаимодействие через NDIS, т.е. самостоятельная реализация стека TCP/IP (30 баллов).

4) Реализация руткита в виде буткита (30 баллов).

5) Реализация руткита в виде биоскита (50 баллов).

5) Реализация руткита в виде вирткита с использованием аппаратной виртуализации (70 баллов).

VIII. Реализовать антируткит (с компонентами в пользовательском режиме и на уровня ядра). Должна быть поддержка всех 32-х разрядных ОС с Windows XP до Windows 10. Для реализации функциональности может потребоваться (в зависимости от выбранных методов) снимать перехваты и восстанавливать код модифицированных модулей.

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой), 15 баллов.

Требования:

1) Обнаружение скрытия в пользовательском режиме. Для этого необходимо получать информацию в пользовательском режиме через документированный прикладной интерфейс (это та информация, которую показывают пользователю прикладные программы) и получать информацию через системные вызовы (непосредственно обращаясь в ядро). Необходимо сигнализировать об обнаруженных отличиях.

Осуществлять поиск скрытых процессов (ZwQuerySystemInformation), файлов (ZwQueryDirectoryFile) и ключей реестра (ZwQueryKey, ZwEnumerateKey, ZwEnumerateValueKey).

Полная функциональность.

Требования (дополнительно к базовым):

1) В пользовательском режиме осуществлять поиск скрытых сетевых подключений (ZwDeviceIoControlFile). 5 баллов.

2) В режиме ядра осуществлять поиск скрытой информации. Эталонный список получать от компонента пользовательского режима и различными методами получать объекты, скрытые из этого списка. Необходимо осуществлять поиск следующих скрытых объектов:

- процессов/потоков;
- файлов;
- ключей реестра;
- сетевых соединений.

В зависимости от полноты реализованных методов за каждый пункт из вышеприведенного списка до 10 баллов.

3) Обнаружение перехватов в ядре (IDT, GDT, обработчик sysenter, SSDT, импорт/экспорт модулей ядра, обработчики IRP, драйверы фильтры, и др.). До 20 баллов.

4) Обнаружение модификаций кода модулей ядра. До 10 баллов.

IX. Реализовать руткит режима ядра и управляющую программу под Linux. Управляющая программа по сети получает управляющие команды и передаёт их драйверу (через файл устройства или файловую систему procfs). Руткит должен осуществлять скрытие своего присутствия (скрывать драйвер, управляющий процесс и файлы драйвера и программы) и выполнять полезную нагрузку. Все действия должны настраиваться по сети через некоторый интерфейс управления.

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой), 20 баллов.

Требования:

1) Программа должна предоставлять доступ к системе по сети (подключаться к удалённому серверу и/или ожидать входящих соединений).

2) В программе должна быть реализована полезная нагрузка:

- получение команд по сети;
- загрузка и выгрузка файлов;
- запуск программы на исполнение;
- запуск/останов драйвера руткита (системные вызовы init_module, delete_module).

3) Выбрать метод запуска руткита:

- зарегистрировать драйвер (/etc/modules);
- зарегистрировать демон;

- заразить системный файл (+20 баллов);
- поместить в автозагрузку программу.

4) Функциональность руткита:
- скрытие драйвера руткита (из файла /proc/modules);
- дамп http-запросов (на стандартных портах).

Полная функциональность.

Требования (дополнительно к базовым):

1) Полезная нагрузка:
- подмена http-трафика, внедрение js-кода (до 10 баллов).

2) Функциональность руткита:
- скрывать файлы (5 баллов);
- скрывать процессы (каталоги в /proc) (5 баллов);
- скрывать сетевые подключения (из файла /proc/net/tcp) (5 баллов);
- замена содержимого файлов (через интерфейс управления должна быть возможность задавать в каких файлах какое содержимое на что менять) (до 10 баллов).

X. Написать драйвер и программу управления драйвером. Программа должна осуществлять запуск/останов драйвера, взаимодействие с пользователем и взаимодействие с драйвером: посылать команды и получать результат.

n - номер студента в списке.

a=11

b=13

Необходимо выбрать от 2 до 5 задач из указанных ниже. Выполнение задачи подразумевает взаимодействие программы и драйвера через некоторый интерфейс. Типы интерфейсов представлены ниже. Для i-ой задачи тип интерфейса для её подзадач равен (n*a+i*b)%6.

Если потребуется использовать какой-то тип интерфейса несколько раз (для разных задач или для разных подзадач в рамках одной задачи), то в рамках этого типа необходимо реализовать отдельный интерфейс (отдельные системные вызовы, прерывания и др.) для каждой реализуемой подзадачи (а не реализовывать один, к примеру, системный вызов, который с помощью отдельного аргумента будет определять запрашиваемую подзадачу). Если для определения конкретного интерфейса (например, номера прерывания) требует номер подзадачи, то самостоятельно упорядочить и пронумеровать (с 0) все подзадачи, требующие этого типа интерфейса.

Типы интерфейсов взаимодействия программы с драйвером:

0) ioctl-запросы к устройству драйвера. Разные ioctl-коды для разных подзадач.

1) Запросы на чтение и запись к "файлам" устройства драйвера. Разные имена "файлов" для разных подзадач.

2) Перехват прерывания с помощью модификации соответствующего шлюза (прерывания) в IDT: заменять системный обработчик на свой, который будет выполнять запрошенную функциональность и возвращать управление программе либо передавать управление системному обработчику. Признаком обращения к интерфейсу (а не просто случайной генерации исключения) может служить уникальное значение в регистре. Для i-ой подзадачи перехватывать прерывание с номером 0x30+5*n+i.

3) Добавление системного вызова в существующий дескриптор в таблицах KeServiceDescriptorTable и KeServiceDescriptorTableShadow. Системные вызовы с номерами 0-0xFFF добавляются в нулевой дескриптор, с номерами 0x1000-0x1FFF - в первый. Для i-ой подзадачи реализовать системный вызов с номером 0x1000*(i%2)+500+((n*a+i*b)%37).

4) Добавление системного вызова в новый дескриптор (с номерами 2,3) в таблицы KeServiceDescriptorTable и KeServiceDescriptorTableShadow. Системные вызовы с номерами 0x2000-0x2FFF добавляются во второй дескриптор, с номерами 0x3000-0x3FFF - в третий. Для i-ой подзадачи реализовать системный вызов с номером 0x1000*(2+i%2)+((n*a+i*b)%37).

5) Добавить шлюз вызова в GDT (при необходимости, создать новую большую GDT). Для i-ой подзадачи добавить шлюз в элемент GDT с индексом 300+((n*a+i*b)%37).

Задачи:

0) Дамп GDT и IDT. Программа должна получить от драйвера содержимое GDT и IDT, отобразить подробную информацию о каждом дескрипторе. Две подзадачи: дамп GDT, дамп IDT.

3 балла.

1) Перехват всех прерываний с помощью модификации шлюзов в IDT (тип шлюзов меняться не должен). Для каждого прерывания должен вестись счетчик количества срабатываний этого прерывания. Программа запрашивает эту информацию у драйвера и отображает для каждого прерывания его счетчик. Две подзадачи: активация/деактивация перехвата, запрос информации.

5 баллов.

2) Перехват всех прерываний с помощью модификации шлюзов в IDT

(тип шлюзов меняться не должен). Для каждого прерывания хранить регистровый контекст на момент каждого срабатывания прерывания (необходимо реализовать соответствующие контейнеры). Для исключений класса fault сохранять еще и инструкцию, вызвавшую исключение. Программа запрашивает эту информацию у драйвера и отображает. Две подзадачи: активация/деактивация перехвата, запрос информации.

5 баллов.

3) Перехват обработчика инструкции sysenter. Для каждого системного вызова (индекс которого передается в eax) сохранять аргументы каждого обращения к этому системному вызову (необходимо реализовать соответствующие контейнеры). Программа запрашивает эту информацию у драйвера и отображает. Две подзадачи: активация/деактивация перехвата, запрос информации.

5 баллов.

4) Чтение/запись физической памяти (без PAE и с PAE). Драйвер должен предоставлять интерфейс, позволяющий указать адрес и размер читаемой/записываемой памяти. Для демонстрации завести в программе уникальную строку, сканированием всей физической памяти найти ее физический адрес, изменить ее по этому адресу и вывести эту строку в программе (она должна измениться). Две подзадачи: чтение по указанному адресу, запись по указанному адресу.

5 баллов.

5) Чтение портов ввода/вывода. Драйвер должен предоставлять интерфейс, позволяющий указать адрес в пространстве ввода/вывода (номер порта) и количество считываемых байт. Программа должна выводить шестнадцатиричный дамп пространства ввода/вывода (размером 64 КБ). Одна подзадача: чтение по указанному адресу.

3 балла.

Пример кода для определения номеров:

```
n = 1
a = 11
b = 13
for task_index in range(6):
    print 'interface index: ' + str((n*a+task_index*b)%6)
for subtask_index in range(2):
    print 'subtask ' + str(subtask_index)
    print '3 syscall index: ' + hex(0x1000*(subtask_index%2)+500+((n*a+subtask_index*b)%37))
```

```
print '4 syscall index: ' + hex(0x1000*(2+subtask_index%2)+((n*a+subtask_index*b)%37))
print '5 idt index: ' + str(300+((n*a+subtask_index*b)%37))
```

XI. Реализовать под Windows руткит режима ядра и программу управления. Вся руткит-функциональность должна быть реализована в драйвере. Программа должна получать команды от пользователя по сети, через некоторый интерфейс передавать их драйверу и отдавать пользователю по сети результат.

n - номер студента в списке

a=11

b=13

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой).

15 баллов + баллы за выбранные задачи.

Вся реализуемая функциональность должна поддерживаться для 32-х разрядных ОС Windows XP, Windows 7, Windows 10 без перекомпиляции (т.е. во время исполнения необходимо определять версию системы и подстраиваться под неё).

Программа осуществляет загрузку и выгрузку драйвера. Для взаимодействия программы с драйвером перехватывать в драйвере системный вызов с номером 317*n%200 (подменять адрес в таблице системных вызовов). Программа должна вызывать этот системный вызов, передав в некотором аргументе предопределённое значение, по которому драйвер поймет, что это не обычный вызов системного вызова, а запрос от управляющей программы. В остальных параметрах программа может передавать аргументы для команды (в зависимости от реализиуемого интерфейса управления). Любой интерфейс управления можно организовать, задействуя два аргумента перехваченного системного вызова: в одном передаётся предопределённое значение, во втором - указатель на структуру с описанием команды. Если вдруг у системного вызова недостаточное количество аргументов, то можно увиличить число аргументов для этого вызова в таблице системных вызовов.

Вся реализуемая функциональность должна настраиваться через интерфейс управления без количественных ограничений (т.е. не должно, к примеру, быть ограничения на количество скрываемых процессов).

При выгрузке драйвер должен корректно завершать свою работу, возвращая систему к первоначальному состоянию.

Руткит должен корректно работать под Driver Verifier.

Ниже представлены 4 вида задач. Для каждого вида перечислены конкретные задачи и способы перехвата. Необходимо выбрать от 2 до 4 задач (не больше одной из каждого вида) и соответствующий им способ перехвата. Доступные для выбора задача и способ перехвата для i-го (i от 0 до 3) вида определяется следующим образом.

Пусть N - число задач этого вида, M - число способов перехвата для этого вида.

Задача - (a*n+b*i)%N.

Способ перехвата - (a*n+b*i)%M.

За каждую задачу 5 баллов.

Виды задач:

0. Модификация списка процессов, возвращаемого системным вызовом ZwQuerySystemInformation

Задачи:

0) Скрытие процесса (по PID'у и по имени).

1) Добавление в список (несуществующего) процесса с указанными пользователем именем и PID'ом.

2) Изменение у указанного (по PID'у и по имени) процесса имени на указанное пользователем.

3) Вместо оригинального списка процессов возвращать список процессов с указанными пользователем PID'ами и именами.

Способы перехвата:

0) Подмена адреса в таблице системных вызовов.

1) Сплайсинг системного вызова.

1. Модификация списка сетевых соединений, возвращаемого сетевым драйвером через системный вызов ZwDeviceIoControlFile.

Задачи:

0) Скрыть сетевое соединение (по номеру локального порта, по номеру удаленного порта, по номерам локального и удаленного портов).

1) Добавить в список сетевое соединение с указанными пользователем данными.

2) Изменить у указанного сетевого соединения (по номеру локального порта, по номеру удаленного порта, по номерам локального и удаленного портов) локальные и удалённые адреса и порты на указанные пользователем.

3) Вместо оригинального списка сетевых соединений возвращать список с указанными пользователем данными.

Способы перехвата:

0) Подмена адреса в таблице системных вызовов.

1) Сплайсинг системного вызова.

2) Подмена адреса в таблице IRP-обработчиков сетевого драйвера (зависит от версии системы).

3) Сплайсинг IRP-обработчика сетевого драйвера.

2. Модификация списка файлов, возвращаемого системным вызовом ZwQueryDirectoryFile.

Задачи:

0) Скрыть файл (по имени, по абсолютному NT-пути).

1) Добавить в список для указанной директории файл с указанным именем.

2) Изменить в списке для указанной директории имя файла.

3) Вместо оригинального списка файлов для указанной директории возвращать список с указанными пользователем данными.

Способы перехвата:

0) Подмена адреса в таблице системных вызовов.

1) Сплайсинг системного вызова.

2) Подмена адреса в таблице IRP-обработчиков драйвера файловой системы (зависит от версии системы).

3) Сплайсинг IRP-обработчика драйвера файловой системы.

3. Модификация списка ключей реестра, возвращаемого системным вызовом ZwEnumerateKey.

Задачи:

0) Скрыть ключ реестра (по имени, по пути).

1) Добавить в список для указанного ключа подключ с указанным именем.

2) Изменить в списке для указанного ключа имя подключа.

3) Вместо оригинального списка подключей для указанного ключа возвращать список с указанными данными.

Способы перехвата:

0) Подмена адреса в таблице системных вызовов.

1) Сплайсинг системного вызова.

Дополнительные задания.

Выбрать не более двух дополнительных заданий.

1. Подмена http-трафика: внедрение js-кода.

Простое внедрение - 3 балла, до 15 баллов за универсальный алгоритм с пересборкой сетевых пакетов.

2. Скрывать указанный процесс путём удаления его из всех списков и таблиц: списка процессов, списка таблиц директорий, списка объектов в объекте типа, таблицы (хендлов) идентификаторов процессов и потоков.

5 баллов.

3. Изменять права доступа указанного процесса на права указанного пользователя.

3 балла.

4. Перехватывать запросы на чтение к драйверу клавиатуры. Внедрять в поток нажатых клавиш указанные пользователем последовательности и комбинации клавиш. Продемонстрировать с помощью этого механизма выполнение команд по скачиванию и запуску некоторого исполняемого файла.

5 баллов.

5. Все перехваты осуществлять максимально незаметно. Осуществлять скрытие перехватов и скрытий от антируткитов режима ядра.

До 20 баллов.

6. Для загрузки драйвера заразить какой-нибудь системный загружаемый драйвер.

20 баллов.

7. Сетевое взаимодействие через NDIS, т.е. самостоятельная реализация стека TCP/IP.

30 баллов.

8. Реализация загрузки руткита с помощью буткита.

30 баллов.

9. Реализация загрузки руткита с помощью биоскита.

50 баллов.

10. Реализация руткита в виде вирткита с использованием аппаратной виртуализации.

70 баллов.

Пример кода для определения номеров:

```
n = 1
a = 11
b = 13
for n in range(1,10):
    print 'student ' + str(n)
    print '\tsyscall index: ' + str(317*n%200)
    i = 0
    print '\ttype '+str(i)+': ' + 'task '+str((a*n+b*i)%4) + '\thook ' +
```

```
str((a*n+b*i)%2)
        i = 1
        print '\ttype '+str(i)+': ' + 'task '+str((a*n+b*i)%4) + '\thook ' + str((a*n+b*i)%4)
        i = 2
        print '\ttype '+str(i)+': ' + 'task '+str((a*n+b*i)%4) + '\thook ' + str((a*n+b*i)%4)
        i = 3
        print '\ttype '+str(i)+': ' + 'task '+str((a*n+b*i)%4) + '\thook ' + str((a*n+b*i)%2)
```

XII. Реализовать под Windows антируткит (с компонентами в пользовательском режиме и на уровня ядра).

Программа осуществляет взаимодействие с пользователем, передаёт команды драйверу, показывает результат.

Для реализации функциональности может потребоваться (в зависимости от выбранных методов) снимать перехваты и восстанавливать код модифицированных модулей.

Основной задачей антируткита является обнаружение каких-либо скрытий/модификаций/перехватов в режиме ядра и отображение этой информации.

Вся функциональность должна быть реализована на работающей системе при активном противодействии со стороны руткита.

n - номер студента в списке

a=11

b=13

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой).

10 баллов + баллы за выбранные задачи.

Вся реализуемая функциональность должна поддерживаться для 32-х разрядных ОС Windows XP, Windows 7, Windows 10 без перекомпиляции (т.е. во время исполнения необходимо определять версию системы и подстраиваться под неё).

Программа осуществляет загрузку и выгрузку драйвера. Необходимо предусмотреть интерфейс взаимодействия программы с драйвером.

При выгрузке драйвер должен корректно завершать свою работу, возвращая систему к первоначальному состоянию.

Антируткит должен корректно работать под Driver Verifier.

Программа в пользовательском режиме через прикладной интерфейс осуществляет получение проверяемой информации, а драйвер пытается получить эту же информацию в ядре. Сравнение полученной информации позволит выявить модификикацию возвращаемых пользователю данных.

Ниже представлены 4 вида задач по обнаружению модификации информации, возвращаемой системными вызовами. Драйвер должен получать эту информацию в ядре, обращаясь к соответстсвующим Nt-вызовам (если они экспортируются) либо Zw-вызовам (в противном случае).

Для каждого вида перечислены конкретные задачи. Необходимо выбрать 2 задачи (не больше одной из каждого вида). Для выполнения доступны 2 вида с номерами (a*n+b)%17%4 и (a*n+b+2)%17%4 (в указанном порядке: второе можно выбрать только если сделано первое). Доступная для выбора задача i-го (i от 0 до 3) вида определяется следующим образом.

Пусть N - число задач этого вида.

Задача - ((a*n)%(i+2)+b)%N.

За каждую задачу 5 баллов.

Виды задач:

0. Поиск модификаций списка процессов, возвращаемого системным вызовом ZwQuerySystemInformation

Задачи:

0) Поиск скрытых процессов.

1) Поиск добавленных несуществующих процессов.

2) Обнаружение изменения в возвращаемом списке имени процесса.

1. Поиск модификаций списка сетевых соединений, возвращаемого сетевым драйвером через системный вызов ZwDeviceIoControlFile.

Задачи:

0) Поиск скрытых сетевых соединений.

1) Поиск добавленных несуществующих сетевых соединений.

2. Поиск модификаций списка файлов в некоторой директории, возвращаемого системным вызовом ZwQueryDirectoryFile.

Задачи:

0) Поиск скрытого файла.

1) Поиск добавленных несуществующих файлов.

3. Поиск модификаций списка подключей некоторого ключа реестра, возвращаемого системным вызовом ZwEnumerateKey.

Задачи:

0) Поиск скрытых ключей.

1) Поиск добавленных несуществующих ключей реестра.

Дополнительные задания.

Выбрать не более двух дополнительных заданий.

1. Осуществлять разбор pdb-файлов модулей ядра с отладочной информацией для поиска адресов (функций, глобальных переменных), недокументированной информации (например, смещений во внутренних структурах).

Во всех методах использовать эту полученную информацию вместо, например, вызова экспортируемых функций ядра (которые могут быть перехвачены в таблицах).

До 15 баллов.

2. В режиме ядра осуществлять поиск скрытой (разными методами) информации. Эталонный список получать от компонента пользовательского режима. Предложить и реализовать методы поиска информации, скрытой перехватом в режиме ядра Nt-функций (и более низкоуровневыми методами). Необходимо осуществлять поиск следующих скрытых объектов:

- процессов/потоков;
- файлов;
- ключей реестра;
- сетевых соединений.

В зависимости от полноты реализованных методов за каждый пункт из вышеприведенного списка до 15 баллов.

3. Обнаружение (с обходом противодействия со стороны руткита) перехватов в ядре (IDT, GDT, обработчик sysenter, SSDT, импорт/экспорт модулей ядра, обработчики IRP, драйверы фильтры, и др.).

До 30 баллов.

4. Обнаружение (с обходом противодействия со стороны руткита) модификаций кода модулей ядра.

До 20 баллов.

Пример кода для определения номеров:

```
n = 1
a = 11
b = 13
typesSize = [3, 2, 2, 2]
for n in range(1,15):
    print 'student ' + str(n)
    i = (a*n+b)%17%4
    print '\ttype '+str(i)+': ' + 'task '+str(((a*n)%(i+2)+b)%typesSize[i])
```

```
i = (a*n+b+2)%17%4
print '\ttype '+str(i)+': ' + 'task '+str(((a*n)%(i+2)+b)%typesSize[i])
```

XIII. Реализовать под Linux руткит режима ядра и программу управления. Вся руткит-функциональность должна быть реализована в драйвере. Программа должна получать команды от пользователя по сети, через некоторый интерфейс передавать их драйверу и отдавать пользователю по сети результат.

n - номер студента в списке

a=11

b=13

Базовая функциональность (к сдаче не принимаются задания с функциональностью меньше базовой).

15 баллов + баллы за выбранные задачи.

Программа осуществляет загрузку и выгрузку драйвера (системные вызовы init_module, delete_module).

Драйвер должен предоставлять для программы некоторый интерфейс взаимодействия для управления руткит-функциональностью:

0) запись на устройство драйвера с определённым младшим номером (MINOR(inode.i_rdev));

1) ioctl-коды к устройству драйвера;

2) запись в файл в файловой системе procfs;

3) перехват системного вызова с номером 293*n%200 (подменять адрес в таблице системных вызовов).

4) перехват системного вызова с номером 239*n%200 (сплайсинг системного вызова).

Вся реализуемая функциональность должна настраиваться через интерфейс управления без количественных ограничений (т.е. не должно, к примеру, быть ограничения на количество скрываемых процессов).

При выгрузке драйвер должен корректно завершать свою работу, возвращая систему к первоначальному состоянию.

Ниже представлены 4 вида задач по скрытию информации от пользовательских программ. Скрытие реализуется с помощью некоторого перехвата и модификации возвращаемой информации. Для каждого вида перечислены конкретные задачи и способы перехвата. Необходимо выбрать от 2 до 4 задач (не больше одной из каждого вида) и соответствующий им способ перехвата. Доступные для выбора задача, способ перехвата и интерфейс для i-го (i от 0 до 3) вида определяется следующим образом.

Пусть N - число задач этого вида, M - число способов перехвата для этого вида.

Задача - (a*n+b*i)%N.

Способ перехвата - (a*n+b*i)%M.

Для управления функциональностью задачи выбрать интерфейс с номером (a*n+b*i)%5.

За каждую задачу 5 баллов.

Виды задач:

0. Модификация списка файлов, возвращаемого системными вызовами readdir, getdents, getdents64.

Задачи:

0) Скрыть файл (по имени, по абсолютному пути).

1) Добавить в список для указанной директории файл с указанным именем.

2) Изменить в списке для указанной директории имя файла.

3) Вместо оригинального списка файлов для указанной директории возвращать список с указанными пользователем данными.

Способы перехвата:

0) Подмена адреса в таблице системных вызовов.

1) Сплайсинг системного вызова.

2) Подмена адреса в struct file_operations (файловый объект получать по имени "/").

3) Сплайсингобработчикаиз struct file_operations.

1. Модификация информации о процессах: список возвращается при листинге каталога /proc, информация о процессе получается чтением файлов в подкаталоге процесса (с именем равным PID'у) в каталоге /proc.

Задачи:

0) Скрытие процесса (по PID'у, по имени).

1) Изменение у указанного (по PID'у, по имени) процесса имени на указанное пользователем (модификация содержимого при чтении файла cmdline).

Способы перехвата:

0) Подмена адреса в таблице системных вызовов.

1) Сплайсинг системного вызова.

2) Подмена адреса в соответствующем поле struct file_operations (файловый объект получать по имени "/proc").

3) Сплайсингобработчикаиз struct file_operations.

2. Модификация списка сетевых соединений, возвращаемого сетевым

драйвером при чтении из файла /proc/net/tcp.

Задачи:

0) Скрыть сетевое соединение (по номеру локального порта, по номеру удаленного порта, по номерам локального и удаленного портов).

1) Добавить в список сетевое соединение с указанными пользователем данными.

2) Изменить у указанного сетевого соединения (по номеру локального порта, по номеру удаленного порта, по номерам локального и удаленного портов) локальные и удалённые адреса и порты на указанные пользователем.

3) Вместо оригинального списка сетевых соединений возвращать список с указанными пользователем данными.

Способы перехвата:

0) Подмена адреса в таблице системных вызовов.

1) Сплайсинг системного вызова.

2) Подмена адреса в struct file_operations (файловый объект получать по имени "/proc").

3) Сплайсингобработчикаиз struct file_operations.

3. Модификация списка модулей ядра, возвращаемого при чтении из файла /proc/modules.

Задачи:

0) Скрыть модуль ядра (по имени).

1) Добавить в список модуль с указанными пользователем данными.

2) Изменить в списке для указанного модуля (по имени) имя.

3) Вместо оригинального списка модулей возвращать список с указанными пользователем данными.

Способы перехвата:

0) Подмена адреса в таблице системных вызовов.

1) Сплайсинг системного вызова.

2) Подмена адреса в struct file_operations (файловый объект получать по имени "/proc").

3) Сплайсингобработчикаиз struct file_operations.

Дополнительные задания.

Выбрать не более трёх дополнительных заданий.

1. Для указанного процесса (по PID'у, по имени) поменять права (uid, euid, gid, egid) до прав указанного пользователя и группы.

5 баллов.

2. Подмена http-трафика: внедрение js-кода.

Простое внедрение - 3 балла, до 15 баллов за универсальный алгоритм с пересборкой сетевых пакетов (полностью будет учитываться только в одном рутките).

3. Замена содержимого файлов: через интерфейс управления должна быть возможность задавать в каких файлах какое содержимое на что менять.

Стоит учитывать, что чтение файла возможно через файловую проекцию.

До 10 баллов.

4. В операциях чтения с устройства драйвера возвращать информацию о функционировании руткита (скрываемые файлы, процессы, сокеты и т.д., список фактически скрытых файлов, процессов, сокетов и т.д., внутренняя статистика и другое).

Тип запрашиваемой информации определять из имени файла устройства. Например, через файл hook_proc_list возвращать список скрываемых процессов (идентификаторы, имена).

До 10 баллов.

5. Перехватывать запросы на чтение к драйверу клавиатуры. Внедрять в поток нажатых клавиш указанные пользователем последовательности и комбинации клавиш. Продемонстрировать с помощью этого механизма выполнение команд по скачиванию и запуску некоторого исполняемого файла.

5 баллов.

6. Все перехваты осуществлять максимально незаметно. Осуществлять скрытие перехватов и скрытий от антируткитов режима ядра.

До 20 баллов.

7. Для загрузки драйвера заразить какой-нибудь системный загружаемый драйвер (или само ядро).

20 баллов.

8. Реализация загрузки руткита с помощью буткита.

30 баллов.

9. Реализация загрузки руткита с помощью биоскита.

50 баллов.

10. Реализация руткита в виде вирткита с использованием аппаратной виртуализации.

70 баллов.

Указания и рекомендации.

Один из интерфейсов для взаимодействия программы с драйвером

является перехват в драйвере системного вызова. Программа должна вызывать этот системный вызов, передав в некотором аргументе предопределённое значение, по которому драйвер поймет, что это не обычный вызов системного вызова, а запрос от управляющей программы. В остальных параметрах программа может передавать аргументы для команды (в зависимости от реализуемого интерфейса управления). Любой интерфейс управления можно организовать, задействуя два аргумента перехваченного системного вызова: в одном передаётся предопределённое значение, во втором - указатель на структуру с описанием команды.

Перехват файловых операций (помимо перехвата системных вызовов) возможен с помощью модификации указателей на функции (либо самих функций) в структуре struct file_operations. Эта структура определяется драйвером файловой системы. Указатель на неё хранится в структуре struct file для каждого файла из этой файловой системы.

Как происходят вызовы можно посмотреть, например, в функциях vfs_readdir, sys_old_readdir, sys_getdents, sys_getdents64 (в файле readdir.c) либо vfs_read, sys_read (в файле read_write.c).

Получить объект файла по имени можно с помощью функции filp_open (удалить - filp_close).

Пример кода для определения номеров:

```
n = 1
a = 11
b = 13
for n in range(1,15):
    print 'student ' + str(n)
    print '\tinterface 3 syscall index: ' + str(293*n%200)
    print '\tinterface 4 syscall index: ' + str(239*n%200)
    i = 0
    print '\ttype '+str(i)+': ' + 'task '+str((a*n+b*i)%4) + '\thook ' + str((a*n+b*i)%4) + '\tinterface ' + str((a*n+b*i)%5)
    i = 1
    print '\ttype '+str(i)+': ' + 'task '+str((a*n+b*i)%2) + '\thook ' + str((a*n+b*i)%4) + '\tinterface ' + str((a*n+b*i)%5)
    i = 2
    print '\ttype '+str(i)+': ' + 'task '+str((a*n+b*i)%4) + '\thook ' + str((a*n+b*i)%4) + '\tinterface ' + str((a*n+b*i)%5)
    i = 3
    print '\ttype '+str(i)+': ' + 'task '+str((a*n+b*i)%4) + '\thook ' +
```

```
str((a*n+b*i)%4) + '\tinterface ' + str((a*n+b*i)%5)
```

3.2.3 Примеры домашних и аудиторных заданий

1. Написать обработчик прерывания.
2. Реализовать загрузочный код MBR.
3. Обрабатывать исключения защищённого режима.
4. Обрабатывать в загрузчике виртуальную память.
5. Реализовать элементы функциональности операционной системы (управление виртуальной памятью, обработка прерываний, переключение процессов и др.).
6. Реализовать программу под Windows с использованием WinAPI.
7. Реализовать функции разбора PE-файлов.
8. Реализовать загрузку PE-файлов.
9. Реализовать экспорт из исполняемого PE-файла.
10. Реализовать механизм плагинов с помощью динамических библиотек.
11. Реализовать двусвязный список в драйвере под WindowsNT.
12. Написать драйвер под Windows NT, обрабатывающий запросы на чтение и запись к своему устройству.
13. Реализовать в драйвере под Windows NT работу с аппаратными структурами процессора.
14. Написать драйверы под Windows NT, перехватывающие системные вызовы.
15. Написать драйвер под Windows NT, выводящий список процессов.
16. Написать драйвер под Windows NT, скрывающий процесс.
17. Написать драйверы под Windows NT, использующий нотификаторы ядра.
18. Реализовать в драйвере под Windows NT перехват с помощью сплайсинга.
19. Реализовать кейлоггер режима ядра под Windows NT.
20. Реализовать в драйвере под Windows NT скрытие портов.
21. Реализовать в драйвере под Windows NT перехват и модификацию трафика.
22. Реализовать в драйвере под Windows NT повышение привилегий процессов.
23. Реализовать в драйвере под Windows NT синхронизацию с помощью примитивов синхронизации.
24. Реализовать утилиты под Linux с использованием системных

вызовов.

25. Модифицировать системные вызовы в ядре Linux.
26. Реализовать модули ядра под Linux.
27. Реализовать перехват и модификацию сетевого трафика в модуле ядра под Linux.
28. Реализовать руткит.

3.2.4 Темы заданий на зачёте

1. Системное программирование для процессоров архитектур IA-32, x64.
2. Системное программирование в Windows.
3. Системное программирование в Linux.

3.2.5 Примеры заданий на зачёте

1. Написать обработчик прерывания.
2. Реализовать загрузочный код MBR.
3. Обрабатывать исключения защищённого режима.
4. Обрабатывать в загрузчике виртуальную память.
5. Реализовать элементы функциональности операционной системы (управление виртуальной памятью, обработка прерываний, переключение процессов и др.).
6. Реализовать программу под Windows с использованием WinAPI.
7. Реализовать функции разбора PE-файлов.
8. Реализовать загрузку PE-файлов.
9. Реализовать экспорт из исполняемого PE-файла.
10. Реализовать механизм плагинов с помощью динамических библиотек.
11. Реализовать двусвязный список в драйвере под WindowsNT.
12. Написать драйвер под Windows NT, обрабатывающий запросы на чтение и запись к своему устройству.
13. Реализовать в драйвере под Windows NT работу с аппаратными структурами процессора.
14. Написать драйверы под Windows NT, перехватывающие системные вызовы.
15. Написать драйвер под Windows NT, выводящий список процессов.
16. Написать драйвер под Windows NT, скрывающий процесс.
17. Написать драйвер под Windows NT, использующий нотификаторы ядра.
18. Реализовать в драйвере под Windows NT перехват с помощью

сплайсинга.

19. Реализовать кейлоггер режима ядра под Windows NT.
20. Реализовать в драйвере под Windows NT скрытие портов.
21. Реализовать в драйвере под Windows NT перехват и модификацию трафика.
22. Реализовать в драйвере под Windows NT повышение привилегий процессов.
23. Реализовать в драйвере под Windows NT синхронизацию с помощью примитивов синхронизации.
24. Реализовать утилиты под Linux с использованием системных вызовов.
25. Реализовать модули ядра под Linux.

3.2.6 Примеры вопросов на зачёте

1. Системная архитектуры процессоров IA-32, x64.
2. Программные интерфейсы Windows NT.
3. Интерфейсы ядра Windows NT.
4. Драйверы режима ядра для Windows NT.
5. Формат исполняемых файлов PE.
6. Программные интерфейсы Linux.
7. Интерфейсы ядра Linux.
8. Формат исполняемых файлов ELF.

4. ПОРЯДОК ПРОВЕДЕНИЯ И КРИТЕРИИ ОЦЕНИВАНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

4.1. Порядок проведения промежуточной аттестации

За своевременное и самостоятельно выполнение учебных работ в течение семестра студент получает рейтинговые баллы. Сумма за выполнение основных заданий в полном объёме - 100. Сверх этой суммы могут начисляться баллы за выполнение дополнительных заданий.

Основные баллы выставляются за выполнение объемных заданий (будем называть их зачётными), которые выполняются дома и сдаются в течение семестра. Сумма в 100 баллов делится между зачётными заданиями (не обязательно равномерно). При выдаче зачётного задания определено сколько баллов выставляется за реализацию определённой функциональности. Для зачётных заданий может быть определена функциональность повышенной сложности, за выполнение которой

выставляются дополнительные баллы. Также дополнительные баллы могут быть выставлены по усмотрению преподавателя за особо примечательную реализацию.

По пройденному материалу выдаются небольшие задания для выполнения дома и/или во время семинарских занятий. За эти задания выставляются небольшие дополнительные баллы. Сдавать их можно либо в день выдачи либо на следующем занятии.

Пропуск по неуважительной причине одной пары влечет вычет 1 балла из итоговой суммы за семестр.

При нехватке баллов преподавателем может быть предоставлено дополнительное задание или возможность доделать задание, в котором была оценена не вся функциональность.

Итоговая оценка за дисциплину выставляется по результатам выполнения заданий текущего контроля. При необходимости во время зачёта может быть предоставлена возможность получить дополнительные баллы (не более 20), выполнив дополнительные задания и ответив (в устной форме) на вопросы.

4.2. Критерии оцениванияпромежуточной аттестации по видам оценочных средств.

4.2.1 Критерии оценивания зачётных заданий

Выполнение заданий предполагает некоторую программную реализацию, к которой предъявляются обычные требования по качеству кода. Код должен быть удобочитаемым, хорошо структурированным, расширяемым, удобным в сопровождении, написанным в едином стиле. Должна быть проведена функциональная декомпозиция (на подпрограммы, модули, пакеты и т.д.), реализованы необходимые программные абстракции. Реализации алгоритмов должны быть логичными и понятными. Иначе возможна сбавка до 5 баллов. За программные ошибки, приводящие к неработоспособности кода для некоторых возможных случаев, возможна сбавка до 10 баллов (в зависимости от критичности ошибки). За программные ошибки, приводящие к аварийному некорректному завершению программы, возможна сбавка до 10 баллов (в зависимости от критичности ошибки).

При сдаче зачётного задания производится опрос по техническим деталям реализации и по теории, используемой при выполнении заданий. Неудовлетворительный ответ будет означать несамостоятельность

выполнения задания, что влечёт выставление 0 баллов за соответствующую функциональность. Если для одного задания это повторяется более 2 раз, то за всё задание выставляется 0 баллов без возможности повторной сдачи.

4.2.2 Критерии оценивания домашних и аудиторных заданий

Домашние и аудиторные задания – это небольшие задания, за которые обычно выставляется 1-2 балла. Они оцениваются атомарно: либо задание выполнено (выставляется указанное при выдаче задания количество баллов), либо не выполнено (0 баллов).

4.2.3 Критерии оценивания заданий на зачёте

Дополнительные задания, выдаваемые на зачёте являются относительно объемными, за них выставляется до 10-15 баллов. Поэтому к ним применимы описанные выше критерии оценивания зачётных заданий с соответствующей корректировкой баллов: сбавка за некорректную работу до 5 баллов, за аварийное завершение – до 5 баллов.

4.2.4 Критерии оценивания ответа в устной форме

Ответ оценивается по трём параметрам 1) построение ответа (структура ответа, грамотность речи, последовательность и т.д.); 2) фактическая полнота ответа; 3) собственный анализ излагаемого материала его оценка в контексте взаимодействия с другими областями, умение применять на практике.

1	Студент самостоятельно правильно выстраивает структуру ответа, изложение последовательное, речь грамотная без оговорок.	2
	Изложение студента непоследовательное и обрывочное, взаимосвязи частей ответа не всегда прослеживаются. Раскрытие сути ответа невозможно без уточняющих вопросов.	1
	Студент испытывает существенные трудности при самостоятельном построении ответа, способен только давать краткие ответы на конкретные вопросы.	0
2	Студент правильно ответил на теоретический вопрос билета. Показал отличные знания в рамках усвоенного учебного материала. Ответил на все дополнительные вопросы.	4
	Студент ответил на теоретический вопрос билета с	3

	небольшими неточностями. Показал хорошие знания в рамках усвоенного учебного материала. Ответил на большинство дополнительных вопросов.	
	Студент ответил на теоретический вопрос билета с существенными неточностями. Показал удовлетворительные знания в рамках усвоенного учебного материала. При ответах на дополнительные вопросы было допущено много неточностей.	2
	При ответе на теоретический вопрос билета студент продемонстрировал недостаточный уровень знаний. При ответах на дополнительные вопросы было допущено множество неправильных ответов.	1
	Студент не ответил на вопрос.	0
3	Студент ясно осознаёт место излагаемого материала в общей структуре профессионального знания, знает стандартные примеры использования и предлагает свои, даёт собственные компетентные оценки.	4
	Студент осознаёт взаимосвязи и знает стандартные примеры использования. Допускает неточности при самостоятельном анализе.	3
	Студент в общих чертах осознаёт взаимосвязи и знает стандартные примеры использования. При проведении самостоятельного анализа нуждается в уточняющих вопросах, при этом допускает существенные неточности.	2
	Студент знает основные стандартные примеры использования. Не осознаёт взаимосвязей с другими областями.	1
	Студент не осознаёт взаимосвязей и практическое приложение излагаемого материала.	0

4.3. Результаты промежуточной аттестации и уровни сформированности компетенций

Баллы, полученные за выполнение заданий текущего контроля и промежуточной аттестации, переводятся в оценки за зачет следующим образом:

0-60 баллов – не зачтено;

61-75 баллов – зачтено;
76-90 баллов – зачтено;
более 90 – зачтено.

Особенности проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья обозначены в рабочей программе дисциплины (модуля).

Уровни сформированности компетенций определяются следующим образом:

Оценка	зачтено	зачтено	зачтено	незачтено
Баллы	более 90 баллов	76-90 баллов	61-75 баллов	0-60 баллов
Уровень освоения проверяемых компетенций	высокий	средний	базовый	недостаточный

Специальность 10.05.01 Компьютерная безопасность
Специализация №6 «Информационно-аналитическая и техническая экспертиза компьютерных систем»
Фонд оценочных средств по дисциплине «Системное программирование»
2025 год набора, очная форма обучения

Фонд оценочных средств дисциплины (модуля) одобрен и рекомендован:
Проректор по учебной работе утверждено 24.02.25 А.А. Саламатов

Ученым советом математического факультета

Протокол заседания № 6 от 20.02.2025

Председатель Ученого совета
математического факультета согласовано Е.А. Сбродова

Заседанием кафедры компьютерной безопасности и прикладной алгебры

Протокол заседания № 8 от 01.02.2025

Заведующий кафедрой согласовано А. Н. Ручай

Автор (составитель) И. А. Маткин

Структура рабочей программы соответствует приказу ректора ФГБОУ ВО «ЧелГУ» от «13» апреля 2021 г. № 247-1