Фонд оценочных средств
для промежуточной аттестации
по дисциплине
Анализ уязвимостей программного обеспечения

Направление подготовки (специальность)
10.05.01 Компьютерная безопасность

Направленность (профиль)
специализация № 1 «Анализ безопасности компьютерных систем»

Присваиваемая квалификация
специалист по защите информации

Форма обучения
очная

Челябинск 2025 г.

Содержание

1. ПАСПОРТ ФОНДА ОЦЕНОЧНЫХ СРЕДСТВ

Специальность 10.05.01 Компьютерная безопасность.
Специализация № 1 «Анализ безопасности компьютерных систем».
Дисциплина: **Анализ уязвимостей программного обеспечения**.
Семестр (семестры) изучения: 8, 9 семестры.
Форма (формы) промежуточной аттестации:
зачет (8 семестр), экзамен (9 семестр).
Используется балльно-рейтинговая система для оценивания результатов.

2. ПЕРЕЧЕНЬ ФОРМИРУЕМЫХ КОМПЕТЕНЦИЙ

2.1. Компетенции, закреплённые за дисциплиной

Изучение дисциплины «Анализ уязвимостей программного обеспечения» 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-1.2	Способен оценивать корректность программных реализаций алгоритмов защиты информации	ОПК 1.2.1 Знает основные средства и методы защиты программного обеспечения от анализа и нарушения целостности; основные программные методы защиты данных от несанкционированного доступа. ОПК 1.2.2 Умеет проводить анализ программных средств, применяемых для контроля и защиты информации; проводить анализ программ и алгоритмов на предмет соответствия требованиям защиты информации.	Знать: – методы эксплуатации современных уязвимостей бинарного программного обеспечения; – методы поиска уязвимостей бинарного программного обеспечения; – требования и рекомендации по обеспечению безопасности бинарного программного обеспечения; – современные уязвимости аппаратного обеспечения; – современные защитные механизмы, противодействующие эксплуатации уязвимостей бинарного программного обеспечения. Уметь: – эксплуатировать классические и современные уязвимости бинарного программного обеспечения; – использовать базы данных уязвимостей при проведении анализа безопасности; – использовать лучшие практики по

			предотвращению появления уязвимостей в бинарном программном обеспечении; – эксплуатировать уязвимости аппаратного обеспечения; – использовать методы противодействия защитным механизмам. Владеть: – навыками создания эксплоитов для бинарного программного обеспечения; – навыками использования инструментальных средств поиска и эксплуатации уязвимостей; – навыками создания бинарного программного обеспечения с учётом требований безопасности; – навыками создания эксплоитов для уязвимостей аппаратного обеспечения и прошивок; – навыками создания эксплоитов с учётом защитных механизмов.

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.	ПК-3	Раздел 1. Уязвимости ПО в UNIX-подобных системах.	Зачетные задания. Домашние задания. Аудиторные задания.	Вопросы на зачёте. Задания на зачёте. Вопросы на экзамене. Задания на экзамене.
2.	ПК-3	Раздел 2. Уязвимости ПО в Windows NT	Зачетные задания. Домашние задания. Аудиторные задания.	Вопросы на зачёте. Задания на зачёте. Вопросы на экзамене. Задания на экзамене.
3.	ПК-3	Раздел 3. Уязвимости в аппаратном обеспечении	Зачетные задания. Домашние задания. Аудиторные задания.	Вопросы на зачёте. Задания на зачёте. Вопросы на экзамене. Задания на экзамене.
4.	ПК-3	Раздел 4. Уязвимости специализированного ПО	Зачетные задания. Домашние задания. Аудиторные задания.	Вопросы на зачёте. Задания на зачёте. Вопросы на экзамене. Задания на экзамене.

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. Шеллкоды для эксплоитов.
2. Уязвимости переполнения буфера в стеке.
3. Уязвимости переполнения буфера в кучу.
4. Уязвимости целочисленного переполнения.
5. Уязвимости переполнения кучи.
6. Защитные механизмы компиляторов и операционных систем.
7. Уязвимости аппаратного обеспечения и прошивок.

3.2.2 Примеры зачетных заданий

I. Программы task1 и backup содержат ошибки, приводящие к переполнению буфера в стеке. Используя уязвимость сетевой программы, выполнить шеллкод, который запустит уязвимую suid'ную программу и, воспользовавшись её уязвимостью, запустит шеллкод с её правами. Для демонстрации получения прав программы backup получить по сети файл key.

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

Требования:

1) Отдельно проэксплуатировать программы task1 и backup. Исполнить любой шеллкод. Можно полагаться на адреса в стеке, полученные под отладчиком.

Полная функциональность.

Требования (дополнительно к базовым):

1) Считать адреса в стеке случайными и не полагаться на них (5 баллов).

2) Проэксплуатировав уязвимость в программе task1, запуситить шеллкод, который проэксплуатирует уязвимость в программе backup (до 10 баллов).

3) Написать эксплоит, который через эксплуатацию уязвимостей в task1 и backup получит по сети файл key (5 баллов).

II. Проэксплуатировать уязвимость переполнения буфера в куче в task_heov.cpp и исполнить шеллкод, предоставляющий удалённую оболочку.

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

1) Проэксплуатировать уязвимость без обхода защитных механизмов.

Полная функциональность.

Требования (дополнительно к базовым):

1) Обход ASLR (5 баллов).

2) Обход DEP (до 10 баллов).

III. Сетевая программа task3 принимает для исполнения исходный код, компилирует его и запускает на виртуальной машине.

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

1) Написать эксплоит. Используя уязвимость, исполнить шеллкод,

предоставляющий удалённую оболочку.

Полная функциональность.

Требования (дополнительно к базовым):

1) Обход ASLR (до 10 баллов).

2) Обход DEP (до 15 баллов).

IV. Два задания по эксплуатации уязвимостей в приложениях под Windows. Только базовую функциональность можно сдать не более чем в одном из них.

Программа net.exe принимает входящие соединения и запускает на исполнение в новом процессе программу local.exe, перенаправив стандартные дескрипторы в сокет. Программа local.exe содержит уязвимость при обработке данных, получаемых из стандартного ввода (перенаправленного в сокет).

Задание 1. Программа local.exe собрана с защитой стека и опцией /safeseh. Проэксплуатировать уязвимость в программе local.exe и выполнить шеллкод, запускающий калькулятор.

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

10 баллов.

Требования:

1) Переполнить буфер, обойти защиту стека, выполнить шеллкод (адрес шеллкода можно узнать под отладчиком).

Полная функциональность.

Требования (дополнительно к базовым):

1) Обойти ASLR (т.е. использовать можно только адреса из самой программы). 3 балла.

2) Реализовать небольшой "Egg hunter" шеллкод с использованием SEH. 3 балла.

Задание 2. Программа local.exe собрана с защитой стека и опцией /safeseh. Проэксплуатировать уязвимость в программе local.exe и выполнить шеллкод, запускающий калькулятор.

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

Требования:

1. Переполнить буфер, обойти защиту стека с помощью SEH, выполнить шеллкод (адрес шеллкода можно узнать под отладчиком).

Полная функциональность.

Требования (дополнительно к базовым):

1. Обойти ASLR (т.е. использовать можно только адреса из самой программы). 10 балла.

V. Реализовать эксплоиты для двух уязвимых плагинов для браузеров. В плагинах присутствует ошибка Use-After-Free.

Плагин 1. Плагин собран без поддержки ASLR. С помощью уязвимости возможна утечка памяти и передача управления по произвольному адресу с двумя контролируемыми аргументами в стеке. Уязвимость можно проэксплуатировать во всех браузерах, поддреживающих плагины NPAPI, во всех версиях операционной системы (в IE в Windows 10 реализована защита от UAF, поэтому в этой системе можно проэксплуатировать в других браузерах). Необходимо обходить защитные механизмы ASLR (рандомизируются все модули кроме самого плагина) и DEP. Rop-цепочки желательно строить с помощью утечки памяти.

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

Требования:

1) Написать эксплоит с обходом DEP с помощью ROP-цепочки.

2) Так как при срабатывании узвимости нами контролируются два двойных слова (аргументы функции) в стеке, можно построить ROP-цепочку из гаджетов вида:

pop; pop ebp; ret

mov ebp, esp; ret

Эти гаджеты присутствуют в самом плагине по фиксированным адресам.

3) Допускается использование гаджетов из библиотек по фиксированным адресам (т.е. не обходить ASLR).

4) Допускается возможность использования heap spray.

Полная функциональность.

Требования (дополнительно к базовым):

1) Гаджеты в ROP-цепочке не должны использовать тот факт, что при передаче управления нами контролируются два двойных слова. Использовать только тот факт, что определённые регистры указывают на подконтрольную нам область памяти (как часто и бывает в реальных уязвимостях). Для этого понадобятся гаджеты, перекидывающие стек на эти регистры, которых нет в самом плагине. Эти гаджеты надо искать в системных библиотеках. Соответственно, для каждой версии системы ROP-

цепочка строится независимо (по 3 балла для каждой версии системы: Windows XP, Windows 7, Windows 10).

2) Поиск библиотек, гаджетов, адресов нужных функций с помощью разбора образов загруженных PE-файлов (до 10 баллов)

3) Не использовать heap spray (5 баллов).

Плагин 2. Плагин собран с поддержкой ASLR. С помощью уязвимости можно читать и писать память по произвольному адресу. Необходимо проэксплуатировать уязвимость под ОС Windows XP, Windows 7, Windows 10. Необходимо обходить защитные механизмы ASLR и DEP. Rop-цепочки строить с помощью утечки памяти.

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

Требования:

1) Написать эксплоит с обходом DEP и ASLR с помощью ROP-цепочки.

2) Поиск библиотек, гаджетов, адресов нужных функций с помощью разбора образов загруженных PE-файлов.

3) Допускается возможность использования heap spray.

Полная функциональность.

Требования (дополнительно к базовым):

1) Не использовать heap spray (3 балла).

VI. Модифицировать публичный эксплоит для уязвимости MS13-037.

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

Требования:

1) Переписать генерацию цепочек так, чтобы легко было изменить адрес 0x0c0c0c0c на любой другой (просто вычислять все адреса относительно единого базового адреса загрузки).

2) Отказаться от получения адреса загрузки ntdll через чтение значения по адресу 0x7ffe0300. Вместо этого искать адрес загрузки через таблицы импорта: получить адрес внутри библиотеки vgx.dll (например адрес таблицы виртуальных функций какого-нибудь объекта) -> получить адрес загрузки vgx.dll -> в её таблице импорта найти адрес внутри библиотеки kernel32.dll -> найти адрес загрузки kernel32.dll -> в её таблице импорта найти адрес внутри ntdll.dll -> найти адрес загрузки ntdll.dll.

Полная функциональность.

Требования (дополнительно к базовым):

1) Отказаться от привязки к конкретной версии ntdll. Вместо этого

через утечку памяти найти адреса нужных гаджетов (по сигнатурам) и адрес функции ZwProtectVirtualMemory (10 баллов).

2) В примере реализована поддержка связки Win7+IE8. Особенности эксплуатации могут изменяться от версии браузера и версии системы. Добавить поддержку других сочетаний версии системы и браузера. Например, WinXP+IE6, WinXP+IE8, Win7+IE9 (в предоставленных виртуальных машинах есть не все возможные сочетаний, поэтому понадобится настроить нужное окружение самостоятельно). По другому могут располагаться объекты в куче и по другому (с использованием других регистров) вызываться функции из таблицы виртуальных методов. Эти моменты следует выяснить под отладчиком (нужные функции упомянуты в описании).

За каждое сочетание версии системы и браузера (если для него необходима уникальная ROP-цепочка) помимо Win7+IE8 по 5 баллов. Версию системы и браузера необходимо определять в эксплоите динамически.

VII. Реализовать эксплоит для сетевого приложения, реализованного в виде драйвера. В драйвере запускается системный поток, ожидающий входящих соединений на сокетах ядра (WSK). На каждое подключение создаётся системный поток, обрабатывающий это подключение в функции ProcessingRequest. Предоставляется функциональность сохранения и получения двойных слов по индексам.

Необходимо проэксплуатировать уязвимость и выполнить в ядре шеллкод.

Эксплоит должен быть работоспособным для ОС Windows 7+.

Необходимо обходить все защитные механизмы (ASLR, DEP, WP, GS, SMEP,...).

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

15 баллов.

Требования:

1) Написать эксплоит, позволяющий исполнить какой-нибудь шеллкод. Нужные адреса можно узнать под отладчиком. Не обходить DEP/WP: можно воспользоваться тем, что данные ядра исполняемые, либо записать шеллкод в исполняемую память (предполагая, что CR0.WP = 0).

После исполнения полезной нагрузки допускается падение системы.

Полная функциональность

Требования (дополнительно к базовым):

1) Эксплуатация должна происходить без нарушения работоспособности драйвера и системы.

2) Обходить DEP/WP с помощью модификации PTE. 3 балла.

3) Все необходимые адреса получать через утечку памяти в драйвере.

Поиск библиотек, гаджетов, адресов нужных функций с помощью разбора образов загруженных PE-файлов. Оптимизация скорости поиска гаджетов и поиска функций, нечеткий поиск rop-гаджетов, рекурсивный поиск всех библиотек, использование библиотек для дизассемблирования и ассемблирования.

До 15 баллов.

4) Реализовать шеллкод, загружающий драйвер, хранящийся в теле шеллкода (драйвер необходимо сохранить на диск и воспользоваться вызовом ZwLoadDriver).

5 баллов.

5) Реализовать шеллкод, внедрящий произвольный шеллкод в пользовательский процесс (находить привилегированный процесс, либо повышать права произвольному процессу, в который будет осуществляться внедрение).

5 баллов.

6) Реализовать функциональность предыдущих шеллкодов без исполнения шеллкода в ядре, т.е. только с помощью ROP-цепочек, чтения/записи памяти и общения с эксплоитом через сокет.

До +10 баллов за каждый из шеллкодов.

Указания и рекомендации.

Уязвимость в обработчике соединения позволяет читать и записывать некоторую память. При этом нужно добиться момента (с помощью небольшого перебора), когда это будет нужная память. Для этого следует создать несколько соединений.

После этого появится возможность писать и читать любую память и передавать управление (через формирование стекового фрейма с ROP-цепочкой).

Для изменения прав доступа к страницам необходимо непосредственно модифицировать элементы PTE этих страниц. Это может понадобиться для исполнения шеллкода в изначально неисполняемых страницах (например, в стеке) или для записи в память, доступную только для чтения (например, перезапись секции кода).

Содержащийся в шеллкоде исполняемый модуль можно сохранить на

диск (ZwCreateFile, ZwWriteFile, ZwClose). После этого драйвер с диска можно загрузить вызовом ZwLoadDriver. Этому вызову нужен ключ реестра, описывающий драйвер. Шеллкод должен создать и инициализировать эток ключ самостоятельно (RtlCreateRegistryKey, RtlWriteRegistryValue, ZwCreateKey, ZwSetValueKey). Вызывать ZwLoadDriver необходимо из процесса, обладающего соответствующими привилегиями. Можно поймать момент, когда драйвер будет выполняться в контексте привилегированного процесса, либо произвольному процессу добавить в маркер доступа привилегию загрузки драйверов (или подменить его маркер доступа на маркер доступа привилегированного процесса).

Сокет соединения можно получить через утечку памяти. Этот сокет можно передавать в функции драйвера Send, Receive и получить канал связи с эксплоитом. Вызов Receive может быть элементом ROP-цепочки, тогда можно будет записать произвольные данные (считанные из сокета) произвольного размера по произвольному адресу. Вызов Send в ROP-цепочке позволит читать произвольную память.

Малый размер стека ядра может не позволить выполнить все действия в рамках одного подключения (одной ROP-цепочки). Поэтому действия по чтению памяти, получению информации, записи значений и др. можно выполнять за несколько этапов в разных подключениях.

Также облегчить создание ROP-цепочки может возможность активного взаимодействия (чтение/запись) с эксплоитом через сокет.

3.2.3 Примеры домашних и аудиторных заданий

1. Реализовать шеллкоды без нулевых байт.
2. Исследовать бинарное программное обеспечение, найти уязвимость и проэксплуатировать её.
3. Проэксплуатировать уязвимости и выполнить шеллкод.
4. Переполнить буфер и исполнить шеллкод.
5. Проэксплуатировать уязвимости сетевых приложений.
6. Проэксплуатировать уязвимости форматных строк.
7. Проэксплуатировать уязвимости переполнения в куче.
8. Проэксплуатировать уязвимости и обойти защиту стека.
9. Проэксплуатировать уязвимости и обойти DEP.
10. Проэксплуатировать уязвимости и обойти ASLR.
11. Проэксплуатировать уязвимость с использованием методов противодействия защитным механизмам.
12. Реализовать эксплоит с использованием ROP.

13. Обойти DEP с помощью ROP и выполнить шеллкод.
14. Проэксплуатировать уязвимости в плагинах для браузеров.
15. Написать шеллкоды для ядра Windows NT.
16. Проэксплуатировать уязвимости в драйверах под Windows NT.
17. Написать шеллкоды для ядра Linux.
18. Проэксплуатировать уязвимости в драйверах под Linux.
19. Проэксплуатировать уязвимость в аппаратном обеспечении.
20. Проэксплуатировать уязвимость в прошивке.

3.2.4 Темы заданий на зачёте

1. Шеллкоды для эксплоитов.
2. Уязвимости переполнения буфера в стеке.
3. Уязвимости переполнения буфера в кучу.
4. Уязвимости целочисленного переполнения.
5. Уязвимости переполнения кучи.
6. Защитные механизмы компиляторов и операционных систем.

3.2.5 Примеры заданий на зачёте

1. Реализовать шеллкоды без нулевых байт.
2. Исследовать бинарное программное обеспечение, найти уязвимость и проэксплуатировать её.
3. Проэксплуатировать уязвимости и выполнить шеллкод.
4. Переполнить буфер и исполнить шеллкод.
5. Проэксплуатировать уязвимости сетевых приложений.
6. Проэксплуатировать уязвимости форматных строк.
7. Проэксплуатировать уязвимости переполнения в куче.
8. Проэксплуатировать уязвимости и обойти защиту стека.
9. Проэксплуатировать уязвимости и обойти DEP.
10. Проэксплуатировать уязвимости и обойти ASLR.
11. Проэксплуатировать уязвимость с использованием методов противодействия защитным механизмам.
12. Реализовать эксплоит с использованием ROP.
13. Обойти DEP с помощью ROP и выполнить шеллкод.
14. Проэксплуатировать уязвимости в плагинах для браузеров.

3.2.6 Примеры вопросов на зачёте

1. Шеллкоды для эксплоитов.
2. Техника создания эксплоитов.

3. Уязвимости переполнения буфера в стеке.
4. Уязвимости переполнения буфера в кучу.
5. Уязвимости целочисленного переполнения.
6. Уязвимости переполнения кучи.
7. Защитные механизмы компиляторов.
8. Защитные механизмы операционных систем.
9. Защитные механизмы аппаратного обеспечения.
10. Методы противодействия защитным механизмам.
11. Методы поиска уязвимостей.
12. Методы безопасного программирования.

3.2.7 Темы заданий на экзамене

1. Шеллкоды для эксплоитов.
2. Уязвимости переполнения буфера в стеке.
3. Уязвимости переполнения буфера в кучу.
4. Уязвимости целочисленного переполнения.
5. Уязвимости переполнения кучи.
6. Защитные механизмы компиляторов и операционных систем.
7. Уязвимости аппаратного обеспечения и прошивок.

3.2.8 Примеры заданий на экзамене

1. Реализовать шеллкоды без нулевых байт.
2. Исследовать бинарное программное обеспечение, найти уязвимость и проэксплуатировать её.
3. Проэксплуатировать уязвимости и выполнить шеллкод.
4. Переполнить буфер и исполнить шеллкод.
5. Проэксплуатировать уязвимости сетевых приложений.
6. Проэксплуатировать уязвимости форматных строк.
7. Проэксплуатировать уязвимости переполнения в куче.
8. Проэксплуатировать уязвимости и обойти защиту стека.
9. Проэксплуатировать уязвимости и обойти DEP.
10. Проэксплуатировать уязвимости и обойти ASLR.
11. Проэксплуатировать уязвимость с использованием методов противодействия защитным механизмам.
12. Реализовать эксплоит с использованием ROP.
13. Обойти DEP с помощью ROP и выполнить шеллкод.
14. Проэксплуатировать уязвимости в плагинах для браузеров.
15. Написать шеллкоды для ядра Windows NT.

16. Проэксплуатировать уязвимости в драйверах под Windows NT.
17. Написать шеллкоды для ядра Linux.
18. Проэксплуатировать уязвимости в драйверах под Linux.
19. Проэксплуатировать уязвимость в аппаратном обеспечении.
20. Проэксплуатировать уязвимость в прошивке.

3.2.9 Примеры вопросов на экзамене

1. Шеллкоды для эксплоитов.
2. Техника создания эксплоитов.
3. Уязвимости переполнения буфера в стеке.
4. Уязвимости переполнения буфера в кучу.
5. Уязвимости целочисленного переполнения.
6. Уязвимости переполнения кучи.
7. Защитные механизмы компиляторов.
8. Защитные механизмы операционных систем.
9. Защитные механизмы аппаратного обеспечения.
10. Методы противодействия защитным механизмам.
11. Уязвимости аппаратного обеспечения
12. Уязвимости прошивок.
13. Методы поиска уязвимостей.
14. Методы безопасного программирования.

3.2.10 Пример билета

1. Проэксплуатировать уязвимость с использованием методов противодействия защитным механизмам.
2. Уязвимости переполнения кучи.

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 и экзамена может быть предоставлена возможность получить дополнительные баллы (не более 20), выполнив дополнительные задания и ответив (в устной форме) на вопросы.

4.2. Критерии оценивания промежуточной 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 и экзамене

Дополнительные задания, выдаваемые на зачёте и экзамене 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.

Параметры оценивания	**Критерии оценивания**	**Баллы**
1.Построение ответа	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.Фактическая	Студент правильно ответил на теоретический вопрос билета. Показал отличные знания в рамках усвоенного учебного материала. Ответил на все дополнительные вопросы.	4

полнота ответа	Студент ответил на теоретический вопрос билета с небольшими неточностями. Показал хорошие знания в рамках усвоенного учебного материала. Ответил на большинство дополнительных вопросов.	3
	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.Собственный анализ излагаемого материала	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 экзамен и зачет следующим образом.

Перевод рейтинговых баллов в оценки за зачет:

61 и более баллов – зачтено;

60 и менее баллов – не зачтено.

Перевод рейтинговых баллов в оценки за экзамен:

91 и более баллов – отлично;

76 - 90 баллов – хорошо;

61 - 75 баллов – удовлетворительно;

60 и менее баллов – неудовлетворительно.

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	**Отлично/ зачтено**	**Хорошо/ зачтено**	**Удовлетворительно/ зачтено**	**Неудовлетворительно/ не зачтено**
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 № 1 «Анализ безопасности компьютерных систем»
Фонд оценочных средств по дисциплине «Анализ уязвимостей программного обеспечения»»
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