**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

Факультет Инфокоммуникационных сетей и систем

Кафедра Защищенных систем связи

Дисциплина Основы информационной безопасности

**КУРСОВАЯ РАБОТА**

Разработка задания для командной игры Capture The Flag (CTF)

*(тема курсовой работы)*

Направление/специальность подготовки

10.05.02 Информационная безопасность телекоммуникационных систем

*(код и наименование направления/специальности)*

Студент:

Чезлов К.С., ИБС-12

*(Ф.И.О., № группы) (подпись)*

Преподаватель:

ст. преп. Пестов И.Е.

*(Ф.И.О. преподавателя) (подпись)*

Санкт-Петербург

2022

**СОДЕРЖАНИЕ**

**Введение3**

**Глава 1. Теоретические аспекты разработки задания PPC5**

1.1 Что такое PPC?5

1.2 Основная идея задания5

1.3 Средства для реализации и решения задания5

1.4 Создание карточки задания и расчет его стоимости7

**Глава 2. Реализация задания и его возможного решения8**

2.1 Создание программы-сервера8

2.2 Создание программы-клиента13

**Заключение20**

**Список использованных источников21**

**Приложение 122**

**Приложение 224**

**ВВЕДЕНИЕ**

Информация является неотъемлемой частью современного общества, она используется во всех сферах деятельности человека, поэтому нуждается в надежной защите. Для этого существует множество средств и способов, которые помогают значительно повысить уровень ее защищенности и сделать попытки получения данных незаконным способом значительно трудозатратнее и дороже.

Технологии не стоят на месте, они постоянно дорабатываются, улучшаются, поэтому людям, работающим с информацией, требуются новые знания и навыки, с помощью которых они будут способны обеспечить безопасность данных. Из этого следует, что особенно актуальна проблема обучения защите информации. Для этих целей была создана дисциплина в области информационной безопасности – CTF.

CTF (от англ. Capture the Flag-захват флага) – это командная игра, в которой нужно захватывать «флаги». Существует несколько форматов проведения:

* Task based – участникам предоставляют набор заданий на определенные темы или смесь тематик (osint – поиск информации в интернете, стеганография, криптография, сети, Linux, PPC – программирование и автоматизация, reverse – исследование программ без исходного кода, реверс-инжиниринг, web – эксплуатация веб-уязвимостей и др.), за выполнение которых они получают очки,
* Attack-defense – команды получают выделенный сервер или сеть для поддержания ее работоспособности и защиты от противников, получая очки за корректную работу сервисов своего защищаемого объекта или за кражу данных соперников. Во время соревнований участники обмениваются накопленным опытом и получают новый, повышают навыки при работе в команде и индивидуальное мастерство.

Таким образом, участвуя в командных соревнованиях CTF, игроки значительно повышают уровень своих знаний и практических навыков в области информационной безопасности, что и является объектом данной работы.

Предмет работы – создание задания для CTF в формате Task-based.

Целью курсовой работы является изучение теоретических аспектов создания задания вида PPC для CTF и практическая реализация навыков для его решения.

Задачи курсовой работы:

* Исследовать понятие сокет.
* Рассмотреть основные библиотеки Python, используемые для создания и решения заданий PPC.
* Создать задание, рассчитать его стоимость и сформировать макет карточки.

**ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ РАЗРАБОТКИ ЗАДАНИЯ PPC**

* 1. **Что такое PPC?**

PPC (от англ. Professional Programming and Coding) - профессиональное программирование и кодирование. Это вариация упражнений по спортивному программированию в прикладной форме.

Задания на эту тему могут предполагать написание программ, собирающих и анализирующих данные веб-ресурсов, скриптов для подбора паролей, различных ботов, выполняющих однотипные операции вместо человека.

* 1. **Основная идея задания**

Для реализации любого проекта нужно сформулировать техническое задание, четко описывающее все требования, которым должен соответствовать итоговый продукт.

Пусть существует сервер, где пользователь имеет пустые счета в долларах и биткоинах. При подключении к нему у игрока есть возможность отправлять три вида команд: получить 300 долларов, обменять все доллары на биткоины, купить флаг за 100 биткоинов. Помимо этого, команды уведомляются о том, что накопление некоторых неизвестных заранее сумм на счетах приводит к обрушению этих банковских систем и обнулению счетов. Вместе с тем задание усложняется изменением курса обмена валют, отслеживанием времени между отправкой команд для определения несовершенной программы, занимающейся добычей денежных средств вместо игрока и ограничением сеанса подключения к серверу.

Задание состоит в анализе возможностей сервера и создании программы, способной автоматически заработать достаточное количество средств для покупки «флага», избежав обнаружения.

* 1. **Средства для реализации и решения задания**

Создаваться серверная программа будет посредством языка программирования Python. Он имеет множество удобных встроенных инструментов для решения подобных задач. Помимо этого, Python отличается простотой синтаксиса и быстротой написания программ, что безусловно является важной особенностью для CTF, где скорость выполнения заданий может повлиять на положение команды в рейтинге.

Для связи сервера и игрока нужен какой-то программный интерфейс, обеспечивающий обмен информацией. Эту задачу возьмут на себя сокеты.

Существуют серверные и клиентские сокеты. Первые прослушивают определенный порт, в то время как последние подключаются к нему. После успешной установки соединения начинается обмен данными.

В стандартной библиотеке Python есть модуль socket, обеспечивающий доступ к интерфейсу сокета BSD, поддерживаемого всеми современными операционными системами. С его помощью можно создать объект Socket, который будет обрабатывать канал данных.

Игрок может использовать стороннюю библиотеку под названием pwntools. Она разработана для более эффективного и удобного написания эксплойтов. Pwntools содержит API (Application programming interface) для работы с методами соединения и передачи информации, что также способствует более быстрому решению задачи.

Исходя из вышесказанного, программа-сервер будет прослушивать определенный порт, анализировать полученные команды от игрока и возвращать ему сформированный ответ. Клиентская сторона также предполагает скрипт, написанный на языке программирования, но существует программа, позволяющая устанавливать соединение по протоколу передачи данных TCP и взаимодействовать с сервером вручную – Netcat. Она поможет игрокам разобраться с принципами работы сервера.

Подключение производится с помощью команды «nc» в терминале, после которой следует указать нужный IP-адрес и порт. Далее Netcat начинает прослушивать этот порт. При этом у пользователя есть возможность отправить данные на сервер.

* 1. **Создание карточки задания и расчет его стоимости**

Карточка задания должна определять требования от игрока и содержать информацию или файлы, составляющие его или необходимые для решения. Сообщение о требованиях для решения задачи и поддерживаемых командах ради усложнения обработки данных, отправляемых сервером, и создания тематической атмосферы будут воспроизводиться сервером. Карточка будет содержать его IP-адрес, порт для подключения и информацию, намекающую на ограничение времени подключения.

Стоимость задания должна учитывать его сложность. Порог знаний для него выше, чем для стандартных заданий по стеганографии или криптографии, обычно, имеющих цену величиной 100. Здесь нужно знать основы программирования на Python, его модули для работы с протоколами передачи данных через интернет и временем, также необходим опыт работы с терминалом Linux. В связи с этим стоимость задания может иметь значение 300.

Готовая карточка задания приведена на рис. 1.

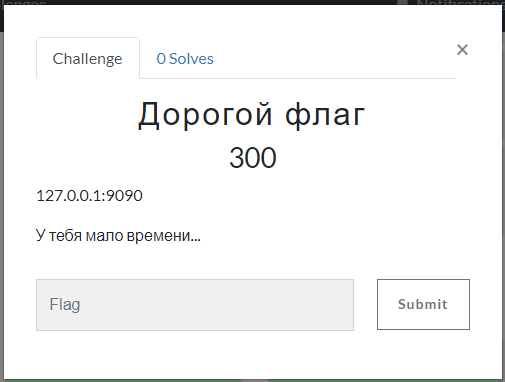


Рисунок 1. Карточка задания

**ГЛАВА 2. РЕАЛИЗАЦИЯ ЗАДАНИЯ И ЕГО ВОЗМОЖНОГО РЕШЕНИЯ**

* 1. **Создание программы-сервера**

Как говорилось ранее, сервер должен уметь работать с сокетами, генерировать случайные числа в некотором диапазоне для задания курса валют и работать с функциями, контролирующими время, значит нужно подключить модули, предоставляющие эти возможности (рис. 2).

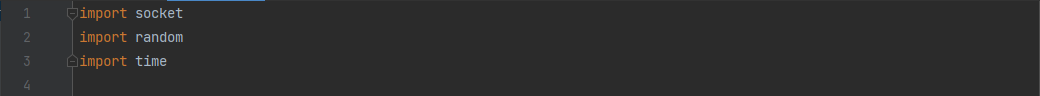


Рисунок 2. Подключение модулей

Следующим шагом нужно определить переменные для хранения IP-адреса, порта, «зарплаты», начисляемой за выполнение команды, стоимости флага, минимального времени между командами, чтобы определить несовершенную программу, времени, ограничивающего сеанс подключения, флага и списка, содержащего строки с приветствием и условиями (рис. 3).

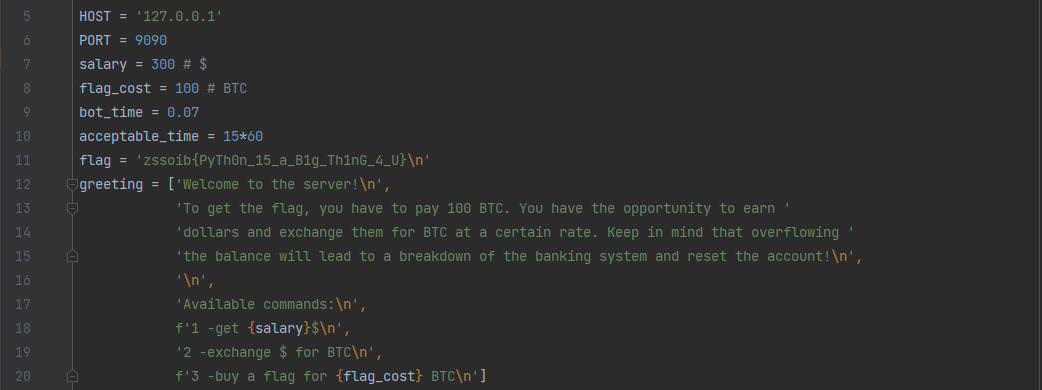


Рисунок 3. Определение переменных.

Теперь нужно сформировать сокет, который будет осуществлять обмен информацией. Для этого надо создать экземпляр класса socket, установить нужные параметры сокета и указать IP-адрес и порт для прослушивания (рис. 4).

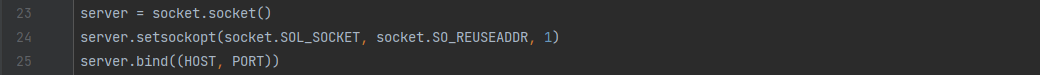


Рисунок 4. Настройка сокета сервера

Далее начинается основной цикл программы-сервера (рис. 5). Он будет ожидать подключения, после которого запишет в переменные сокет и адрес клиента для последующего взаимодействия с ним. Также сервер отправит приветственную речь и условия задания, обнулит все значения словаря, содержащего счета игрока и зафиксирует время начала программы в переменную start\_time для последующего контроля размера сеанса. Объект tic нужен для вычисления перерыва между командами.

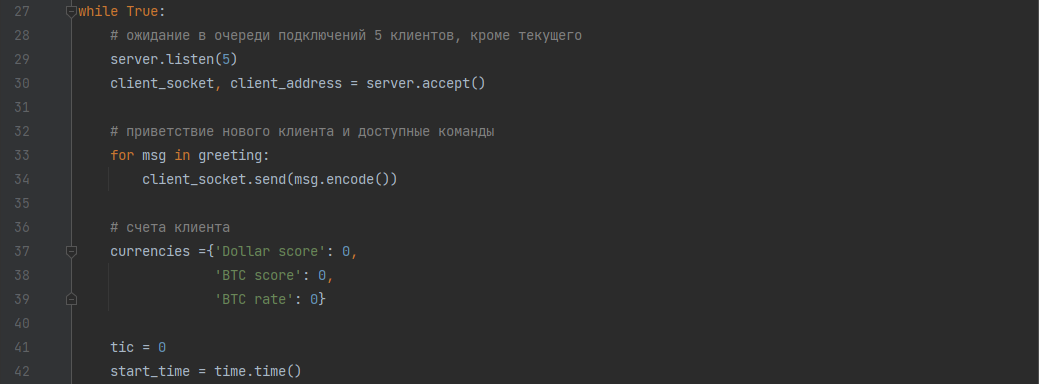


Рисунок 5. Начало основного цикла программы-сервера

Внутри основного цикла, по сути, нужного для ожидания нового клиента и его приветствия, будет работать еще один цикл. Он займется обработкой команд, посылаемых клиентом и обновлением переменных для курса валют и пределов значений на балансе.

Для получения случайных чисел из заданного диапазона используется метод randint модуля random (рис. 6). Концы этих диапазонов подобраны так, чтобы игрок не смог написать программу, выполняющую лишь команду для получения долларов, чтобы обменять сразу большую сумму на флаг. Она должна будет делать это постепенно в течение всего времени работы.

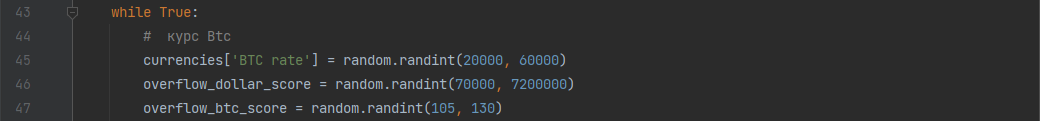


Рисунок 6. Задание случайных значений

Каждую итерацию цикла-обработчика команд нужно уведомлять клиента о состоянии счетов и курса обмена валют. Это можно реализовать с помощью метода send() (рис. 7). Однако обмениваться можно лишь данными, преобразованными в байты, что может сделать метод encode().

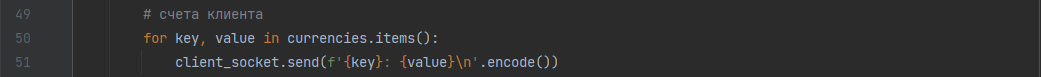


Рисунок 7. Отправка уведомлений о состоянии счетов и курса валют

Следующим шагом является принятие и обработка команд игрока. Для этого нужно воспользоваться методом recv(), считывающим заданное количество байт поступающих данных. Здесь же сервер должен определять время между получаемыми командами (execution\_time). Метод perf\_counter() позволяет получить более точное время, чем time().

Следует учесть, что игрок может случайно или намеренно отправить не поддерживаемую команду, поэтому нужно добавить этот блок кода в конструкцию try/except (рис. 8). Неверная команда вызовет исключение, которое прервет подключение. В ином случае программа продолжит работу в блоке else.

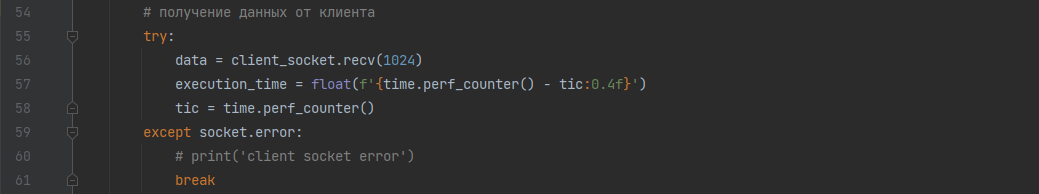


Рисунок 8. Получение данных

Полученные данные нужно декодировать из байтов в строковый тип с помощью метода decode() и удалить символ переноса строки для удобства обработки с помощью среза [:-1]. Снова надо учесть случай возможной ошибки, на этот раз из-за неверной кодировки. Исключение вызовет метод send() для отправки клиенту сообщения об ошибке и прервет подключение (рис. 9).

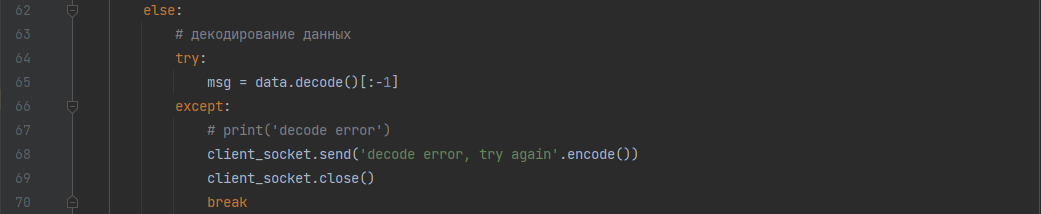


Рисунок 9. Получение данных сервером

Пришло время обработки инструкций клиента. Сервер имеет три вида команд, описанных ранее. Программа должна действовать согласно им. Значит нужен блок для проверки полученной инструкции с заданной командой. Отдельно будет обрабатываться пустое сообщение.

При отправке команды «1» к балансу долларового счета прибавляется вознаграждение salary и игроку отправляется сообщение об успешной операции (рис. 10).

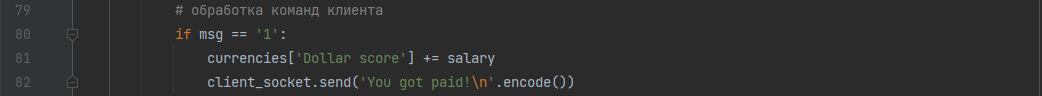


Рисунок 10. Обработка команды «1»

При отправке команды «2» производится проверка на наличие достаточного количества средств для обмена долларов на биткоины. В случае истины этого высказывания производится обмен (начисляется максимально возможное количество биткоинов и списываются доллары в соответствии с текущим курсом валют). В ином случае отправляется сообщение о недостаточном количестве средств (рис. 11).

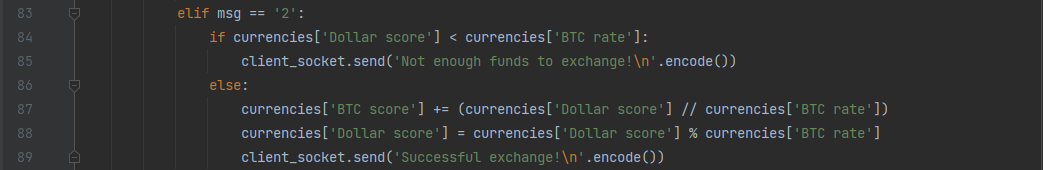


Рисунок 11. Обработка команды «2»

При отправке команды «3» производится проверка на наличие достаточного количества средств для покупки флага. В случае истины сервер отправляет его и прерывает соединение, иначе сообщает о недостаточном количестве средств (рис. 12).

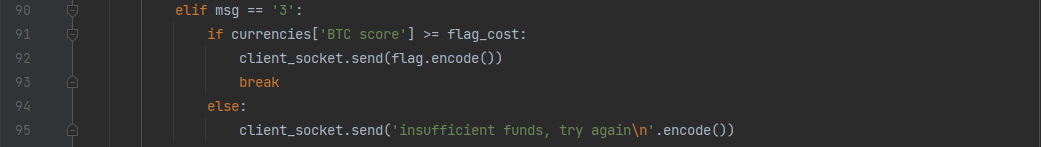


Рисунок 12. Обработка команды «3»

При отправке пустого сообщения сервер отключит игрока (рис. 13).

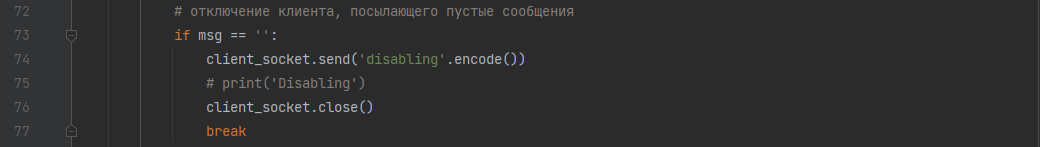


Рисунок 13. Обработка пустого сообщения

Поломка банковских систем будет реализована следующим образом: каждую итерацию цикла обработки команд будет сравниваться текущий баланс валют со сгенерированным случайным числом из заданного диапазона, заданного в начале, в случае его превышения данный валютный счет обнуляется независимо от второго (рис. 14).

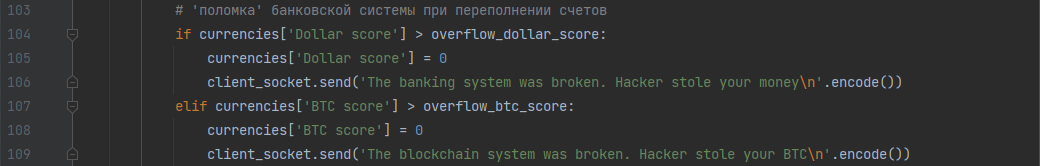


Рисунок 14. «Поломка банковских систем»

Определение несовершенной программы для выполнения команд вместо игрока будет производиться сравнением времени выполнения инструкции с заданным ограничением bot\_time. В случае, если execution\_time меньше, тогда сервер отправит игроку сообщение о ее обнаружении и прервет подключение (рис. 15).

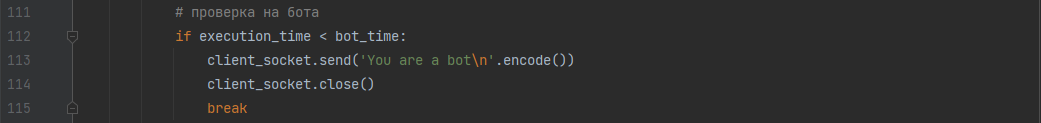


Рисунок 15. Определение несовершенной программы-бота

Ограничение времени сеанса реализовано подобным образом. Только здесь сравнение происходит посчитанного общего времени выполнения программы с допустимой продолжительностью сеанса (рис. 16).

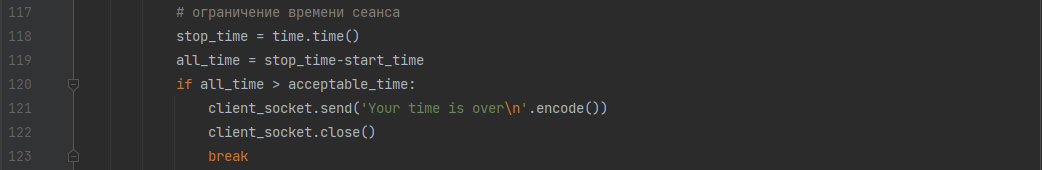


Рисунок 16. Ограничение продолжительности сеанса

Полный код программы-сервера приведен в Приложении 1.

* 1. **Создание программы-клиента**

Решение полученного задания будет зависеть от знаний и опыта игрока. Для начала участник CTF изучит карточку с описанием проблемы и увидит лишь адрес, порт и намек на ограничение времени.

Первым делом нужно попробовать подключиться к указанному серверу. Для этого нужно воспользоваться утилитой Netcat и командой «nc» (рис. 17).

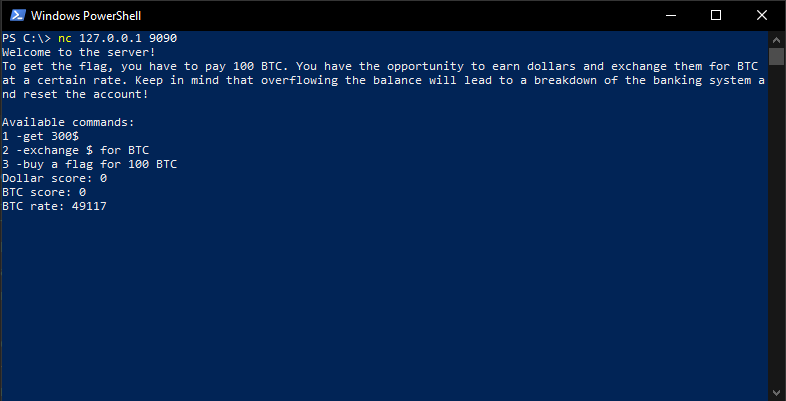


Рисунок 17. Подключение к серверу

Сервер отправил сообщение с формулировкой задачи, доступными командами, балансом счетов и курсом валют. Теперь игрок должен пробовать вводить команды вручную, чтобы выяснить возможности и механики игры.

Спустя время пользователь поймет, что курс валют меняется каждый раз, кода он отправляет команду, инструкция «1» добавляет некоторую постоянную сумму, инструкция «2» позволяет обменивать сразу все накопленные доллары на биткоины и не дает этого сделать при нехватке средств, а команда «3» работает подобно второй, только с покупкой флага. Если игрок оставит терминал открытым, не прекратив исполнение команды подключения или написав неэффективный код в последующих этапах выполнения задания, то увидит, что сервер отправил сообщение, на которое намекала карточка задания (рис. 18).

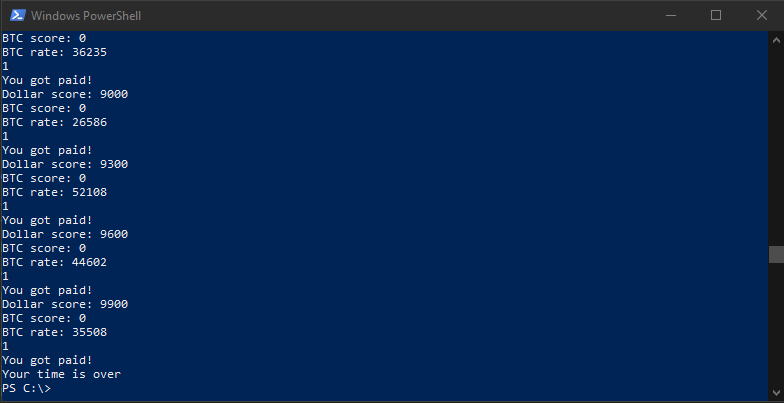


Рисунок 18. Сообщение об израсходованном времени

Следует обратить внимание на то, что после каждой команды сервер всегда отправляет ответ о проделанной операции, ошибке при обмене валют или покупке флага в одной строке, а после три строки с данными о состоянии счетов и курсом валют. Это поможет при написании программы-клиента.

Для начала нужно подключить библиотеку для работы с методами соединения и передачи информации и модуль itertools для создания бесконечного итератора (рис. 19).



Рисунок 19. Подключение модулей

Теперь нужно создать переменные для IP-адреса, порта, балансов валют и курса обмена (рис. 20).



Рисунок 20. Создание необходимых переменных

Следующим шагом является установка соединения с сервером с помощью класса connect(), принадлежащего библиотеке pwntools и чтение первых семи строк, отправляемых сервером и содержащих приветствие и доступные команды с помощью метода recvline(), не забыв декодировать из байтов в строки (рис. 21).



Рисунок 21. Создание подключения и чтение данных

Теперь нужно оформить основной цикл программы-клиента, обрабатывающий сообщения сервера и отправляющий команды в определенных ситуациях.

Из опыта работы с этим сервером следует, что сначала нужно прочитать и записать в переменные для дальнейшей обработки числа, составляющие баланс счетов и курс валют. Сделать это можно с помощью команды вида int(s.recvline().decode()[14:-1]). Срез отделит цифры от текста, а int() превратит строку их в целое число. Также можно вывести полученный результат с помощью форматированной строки (рис. 22).

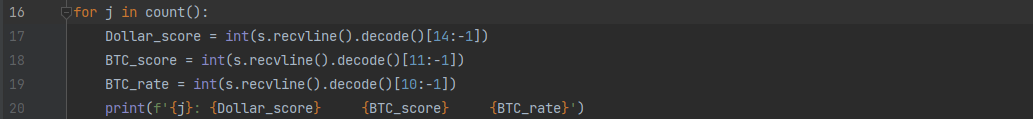


Рисунок 22. Чтение данных пользователя

Далее следует обработка команд. Если баланс счета биткоинов превышает число, заданное в условии, отправленном сервером, то программа должна отправить команду «3», считать флаг и завершить цикл (рис. 23).



Рисунок 23. Обработка команды «3»

Если баланс счета с долларами меньше, чем курс обмена, то программа должна отправить команду «1» и считать ответ сервера об успешной операции, который также можно вывести на экран. В противном случае она должна отправить команду «2» для обмена долларов на биткоины (рис. 24), чтобы избежать случаев с обрушением банковских систем в случае накопления больших сумм как на рис. 25.



Рисунок 24. Обработка команд «1» и «2»

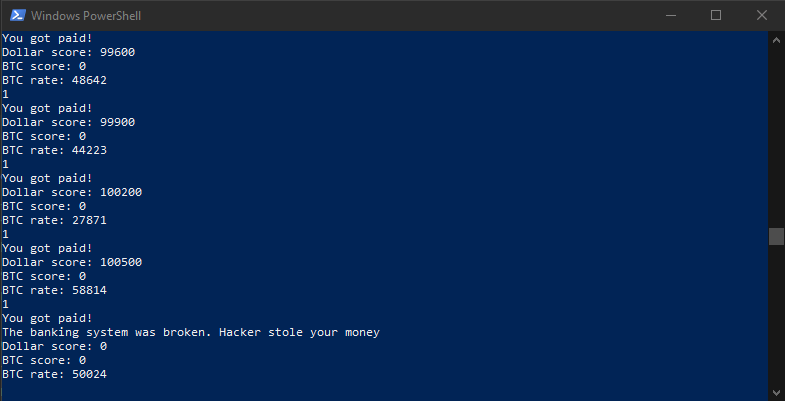


Рисунок 25.– Ошибка при переполнении счета

При запуске этого кода игрок увидит ошибку (рис. 26). Python сообщает о том, что обнаружил неподходящий литерал для числа в считываемой строке. Чтобы выяснить почему это произошло, следует выводить на экран считывание из строки с Dollar\_score, но не забыть присвоить ему любое число (рис. 27).

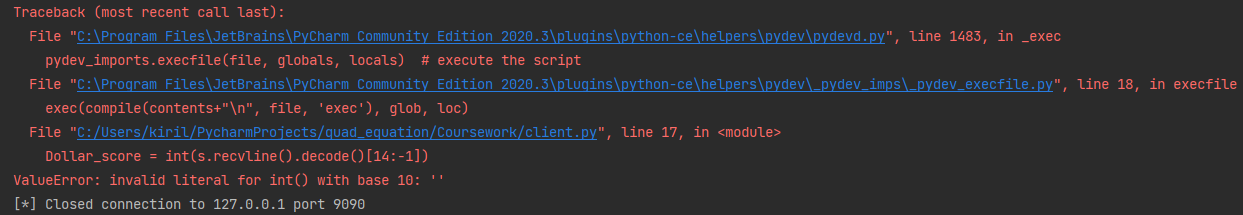


Рисунок 26. Ошибка при запуске программы-клиента



Рисунок 27. Выяснение причины ошибки

Таким образом выясняется, что слишком быстрая скорость выполнения операций обнаруживает скрипт игрока, поэтому нужно делать небольшой перерыв между итерациями цикла (время паузы подбирается путем проб). В этом поможет метод sleep() модуля time() (рис. 28), который нужно подключить в начале программы.



Рисунок 28. Добавление пауз между итерациями

Осталось вывести флаг, и программа-клиент будет завершена и готова к использованию (рис. 29).



Рисунок 29. Вывод флага

Запустив программу, по истечении времени игрок увидит сообщение об успешной покупке флага и сам флаг, который нужно будет ввести в ответ (рис. 30).

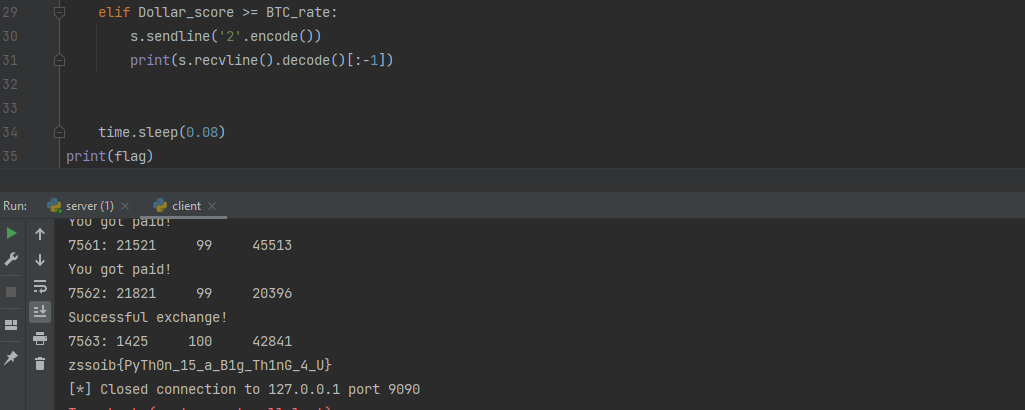


Рисунок 30. Полученный флаг

Полный код программы-клиента приведен в Приложении 2.

**ЗАКЛЮЧЕНИЕ**

Игра в CTF – это отличный способ получения новых знаний и опыта в области информационной безопасности, а также повышения эффективности работы в команде, что становится все актуальней с каждым годом.

В первой главе данной работы было сформировано задание вида PPC для Task-based CTF, рассчитана возможная стоимость и приведены средства для его реализации и решения.

В процессе написания второй главы были созданы программы, составляющие код сервера и клиента, являющиеся заданием и решением задания соответственно.

В ходе решения, игроки CTF получат новый опыт в написании программ для работы с обменом данных посредством сокетов, улучшат навыки программирования – откроют для себя множество инструментов и библиотек, а также изучат основы работы с утилитой Netcat и терминалом Linux. Это задание поможет приобрести понимание работы системы сервер-клиент, его программной части для дальнейшего применения знаний на практике в области информационной безопасности.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Документация Kali Linux : официальный сайт. – 2022. URL: <https://www.kali.org/docs> (дата обращения 01.05.2022).
2. Документация pwntools : официальный сайт. – 2022. URL: <http://docs.pwntools.com> (дата обращения 01.05.2022).
3. Документация Python : официальный сайт. – 2022. URL: <https://docs.python.org> (дата обращения 01.05.2022).
4. Лутц, М. Изучаем Python, том 1, 5-е изд. : Пер. с англ. – СПб. : ООО «Диалектика», 2019. – 832 с. : ил. – Парал. тит. англ. – ISBN 978-5-907144-52-1 (рус., том 1). – ISBN 978-5-907144-51-4 (рус., многотом.).
5. Среда разработки для языка программирования Python PyCharm. : официальный сайт. – 2022. URL: <https://www.jetbrains.com> (дата обращения 01.05.2022).

**ПРИЛОЖЕНИЕ 1**

Листинг 1 – код программы-сервера

1. **import** socket
2. **import** random
3. **import** time
5. HOST **=** '127.0.0.1'
6. PORT **=** 9090
7. salary **=** 300 # $
8. flag\_cost **=** 100 # BTC
9. bot\_time **=** 0.07
10. acceptable\_time **=** 1**\***60
11. flag **=** 'zssoib{PyTh0n\_15\_a\_B1g\_Th1nG\_4\_U}\n'
12. greeting **=** ['Welcome to the server!\n',
13. 'To get the flag, you have to pay 100 BTC. You have the opportunity to earn '
14. 'dollars and exchange them for BTC at a certain rate. Keep in mind that overflowing '
15. 'the balance will lead to a breakdown of the banking system and reset the account!\n',
16. '\n',
17. 'Available commands:\n',
18. f'1 -get {salary}$\n',
19. '2 -exchange $ for BTC\n',
20. f'3 -buy a flag for {flag\_cost} BTC\n']

23. server **=** socket.socket()
24. server.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)
25. server.bind((HOST, PORT))
27. **while** True:
28. # ожидание в очереди подключений 5 клиентов, кроме текущего
29. server.listen(5)
30. client\_socket, client\_address **=** server.accept()
32. # приветствие нового клиента и доступные команды
33. **for** msg **in** greeting:
34. client\_socket.send(msg.encode())
36. # счета клиента
37. currencies **=** {'Dollar score': 0,
38. 'BTC score': 0,
39. 'BTC rate': 0}
41. tic **=** 0
42. start\_time **=** time.time()
43. **while** True:
44. #  курс Btc
45. currencies['BTC rate'] **=** random.randint(20000, 60000)
46. overflow\_dollar\_score **=** random.randint(70000, 7200000)
47. overflow\_btc\_score **=** random.randint(105, 130)
49. # счета клиента
50. **for** key, value **in** currencies.items():
51. client\_socket.send(f'{key}: {value}\n'.encode())

54. # получение данных от клиента
55. **try**:
56. data **=** client\_socket.recv(1024)
57. execution\_time **=** float(f'{time.perf\_counter() - tic:0.4f}')
58. tic **=** time.perf\_counter()
59. **except** socket.error:
60. # print('client socket error')
61. **break**
62. **else**:
63. # декодирование данных
64. **try**:
65. msg **=** data.decode()[:**-**1]
66. **except**:
67. # print('decode error')
68. client\_socket.send('decode error, try again'.encode())
69. client\_socket.close()
70. **break**
72. # отключение клиента, посылающего пустые сообщения
73. **if** msg **==** '':
74. client\_socket.send('disabling'.encode())
75. # print('Disabling')
76. client\_socket.close()
77. **break**
79. # обработка команд клиента
80. **if** msg **==** '1':
81. currencies['Dollar score'] **+=** salary
82. client\_socket.send('You got paid!\n'.encode())
83. **elif** msg **==** '2':
84. **if** currencies['Dollar score'] < currencies['BTC rate']:
85. client\_socket.send('Not enough funds to exchange!\n'.encode())
86. **else**:
87. currencies['BTC score'] **+=** (currencies['Dollar score'] **//** currencies['BTC rate'])
88. currencies['Dollar score'] **=** currencies['Dollar score'] **%** currencies['BTC rate']
89. client\_socket.send('Successful exchange!\n'.encode())
90. **elif** msg **==** '3':
91. **if** currencies['BTC score'] >**=** flag\_cost:
92. client\_socket.send(flag.encode())
93. **break**
94. **else**:
95. client\_socket.send('insufficient funds, try again\n'.encode())
96. **else**:
97. client\_socket.send('Command error\n'.encode())
98. # print('Command error, disabling')
99. client\_socket.close()
100. **break**

103. # 'поломка' банковской системы при переполнении счетов
104. **if** currencies['Dollar score'] > overflow\_dollar\_score:
105. currencies['Dollar score'] **=** 0
106. client\_socket.send('The banking system was broken. Hacker stole your money\n'.encode())
107. **elif** currencies['BTC score'] > overflow\_btc\_score:
108. currencies['BTC score'] **=** 0
109. client\_socket.send('The blockchain system was broken. Hacker stole your BTC\n'.encode())
111. # проверка на бота
112. **if** execution\_time < bot\_time:
113. client\_socket.send('You are a bot\n'.encode())
114. client\_socket.close()
115. **break**
117. # ограничение времени сеанса
118. stop\_time **=** time.time()
119. all\_time **=** stop\_time**-**start\_time
120. **if** all\_time > acceptable\_time:
121. client\_socket.send('Your time is over\n'.encode())
122. client\_socket.close()
123. **break**

**ПРИЛОЖЕНИЕ 2**

Листинг 2 – код программы-клиента

1. **from** itertools **import** count
2. **from** pwn **import** **\***
3. **import** time
5. HOST **=** '127.0.0.1'
6. PORT **=** 9090
8. s **=** connect(HOST, PORT)
9. **for** i **in** range(7):
10. print(s.recvline().decode()[:**-**1])
12. Dollar\_score **=** 0
13. BTC\_score **=** 0
14. BTC\_rate **=** 0
16. **for** j **in** count():
17. Dollar\_score **=** int(s.recvline().decode()[14:**-**1])
18. BTC\_score **=** int(s.recvline().decode()[11:**-**1])
19. BTC\_rate **=** int(s.recvline().decode()[10:**-**1])
20. print(f'{j}: {Dollar\_score}     {BTC\_score}     {BTC\_rate}')
22. **if** BTC\_score >**=** 100:
23. s.sendline('3'.encode())
24. flag **=** s.recvline().decode()[:**-**1]
25. **break**
26. **if** Dollar\_score < BTC\_rate:
27. s.sendline('1'.encode())
28. print(s.recvline().decode()[:**-**1])
29. **elif** Dollar\_score >**=** BTC\_rate:
30. s.sendline('2'.encode())
31. print(s.recvline().decode()[:**-**1])

34. time.sleep(0.08)
35. print(flag)