

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего
профессионального образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И
РАДИОЭЛЕКТРОНИКИ» (ТУСУР)
Кафедра комплексной информационной безопасности электронно-вычислительных систем
(КИБЭВС)

УТВЕРЖДАЮ

заведующий каф. КИБЭВС

_____ А.А. Шелупанов

« _____ » _____ 2016г.

ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ ПРОВЕДЕНИЯ СОРЕВНОВАНИЙ В ОБЛАСТИ
ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ
Отчет по групповому проектному обучению
Группа КИБЭВС-1502

Ответственный исполнитель

студент гр. 723

_____ Д.Е. Муковкин

« _____ » _____ 2016г.

Научный руководитель

аспирант каф. КИБЭВС

_____ А.И. Гуляев

« _____ » _____ 2016г.

РЕФЕРАТ

Отчет содержит 26 страниц, 10 рисунков, 5 источников, 1 приложение.

CTF, ATTACK-DEFENSE, СОРЕВНОВАНИЯ, KEVA, ЗАЩИТА ИНФОРМАЦИИ, ФЛАГИ, ПРИЕМКА ФЛАГОВ, SCOREBOARD, GIT, PYTHON, MONGODB, MONGOOSE, ZABVIX, ОРТПARSE, ЧЕКЕРЫ, HTML, CSS, JAVASCRIPT, КОМАНДА, API.

Цель работы — доработка программного комплекса, предназначенного для проведения соревнований в области информационной безопасности CTF.

Задачи, поставленные на данный семестр:

- доработка ядра и модулей платформы;
- развитие экосистемы платформы;
- тестирование программного комплекса.

Результаты работы в данном семестре:

– доработан существующий API и администраторская панель для управления соревнованиями;

- доработан модуль работы с чекерами;
- доработан модуль приемки флагов;
- доработан модуль таблицы результатов;
- созданы тестовые сервисы для тестирования платформы;
- внедрение системы мониторинга;
- произведено тестирование платформы в форме соревнований.

Пояснительная записка выполнена при помощи системы компьютерной вёрстки L^AT_EX.

Список исполнителей

Муковкин Д.Е. – программист, ответственный исполнитель.

Задачи:

- развертывание инфраструктуры;
- доработка модуля проверки сервисов, сдачи флагов;
- доработка панели администратора;
- разработка нового дизайна для таблицы результатов;
- разработка сервиса для платформы;
- настройка и тестирование платформы.

Койшинов Т.С. – программист.

Задачи:

- разработка API функций;
- разработка сервиса для платформы;
- разработка нового дизайна для таблицы результатов;
- тестирование платформы.

Засыпкин Г.В. – программист.

Задачи:

- внедрение системы мониторинга Zabbix;
- разработка сервиса для платформы;
- тестирование платформы.

Содержание

Введение	5
1 Назначение и область применения	6
2 Постановка задачи	6
3 Инструменты	6
3.1 Система контроля версий Git	6
3.2 Система компьютерной вёрстки T _E X	7
3.3 Язык программирования Python	7
3.4 База данных MongoDB	8
3.5 TrackingTime - Система управления проектами и задачами	9
3.6 Flask - микрофреймворк для Python	9
3.7 ArgParse - инструмент для создания интерфейсов командной строки	9
3.8 Zabbix - система мониторинга сервисов компьютерной сети	10
4 Технические характеристики	11
4.1 Требования к аппаратному обеспечению	11
4.2 Требования к программному обеспечению	11
4.3 Требования к сервисам	11
5 Разработка программного обеспечения	13
5.1 Архитектура	13
5.1.1 Ядро	13
5.2 Модуль: прием и проверка принятых флагов	13
5.2.1 Принцип работы	13
5.3 Модуль: таблицы результатов	15
5.3.1 Принцип работы	15
5.4 Модуль: организации работы с сервисами	17
5.4.1 Принцип работы	17
6 Развитие экосистемы	18
6.1 Сервис whoami	19
6.2 Сервис blozhik	21
6.3 Zabbix	22
7 Планы на будущее	23
Заключение	24
Приложение А Компакт-диск	26

Введение

CTF (Capture the flag, Захват флага) — это командные соревнования, целью которых является оценка умения участников атаковать и защищать компьютерные системы. По типу, соревнования делятся на два типа: task-based (квесты), attack-defense (классические соревнования).

Для проведения соревнований типа attack-defense, каждой команде выдается сервер, на котором имеется ряд сервисов, одинаковые у всех. Сервисами являются программы, которые должны быть запущены на протяжении всего времени соревнований. Раз в минуту жюри проверяет работу сервисов, присылает новый флаг, а также проверяет его наличие на сервере. В роли флага выступает случайно сгенерированная строка, определенной длины. Сервисы заведомо имеют уязвимости, через которые можно украсть флаг. Команда должна найти уязвимости в сервисах и закрыть их. Но также она должна эксплуатировать эти уязвимости на сервисах других команд, похищая флаги.

Команда Keva имеет большой опыт в проведении соревнований CTF. Во время проведения межрегиональных межвузовских соревнований в области информационной безопасности SibirCTF 2014, 2015 использовались наработки Екатеринбургской команды HackerDom. Однако их решение не подходит нам по некоторым критически важным параметрам. Поэтому было принято решение написать собственную платформу для проведения соревнований CTF.

Платформа должна отвечать следующим требованиям:

- Низкое потребление ресурсов сервера;
- Управление игрой через графический интерфейс;
- Возможность горизонтального масштабирования.

Новая платформа позволит нам расширить формат проведения соревнований, а также решать непридвиденные проблемы в кратчайшие сроки.

При разработке платформы должны быть использованы уже имеющиеся наработки, в частности, панель администратора и API для task-based проведения соревнований.

Назначение и область применения

Разрабатываемый программно-аппаратный комплекс предназначен для проведения игр в области информационной безопасности Capture the flag по типу Attack-Defense.

Постановка задачи

На данный семестр были поставлены следующие задачи:

- определение индивидуальных задач для каждого участника проектной группы;
- исследование предметных областей в рамках индивидуальных задач;
- доработка программных модулей.

Задачи, связанные с разработкой программного комплекса:

- 1) доработка ядра платформы;
- 2) доработка модуля проверяющей системы;
- 3) доработка модуля сдачи флага;
- 4) доработка модуля рейтинговой таблицы;
- 5) доработка модуля администраторской панели.

Задачи, связанные с развитием экосистемы:

- 1) создание сервисов;
- 2) создание руководства пользователя;
- 3) создание руководства администратора;
- 4) продвижение платформы в CTF сообществе.

Задачи, связанные с тестированием программного комплекса:

- 1) проверка работы ядра;
- 2) проверка работы модуля проверяющей системы;
- 3) проверка работы модуля сдачи флага;
- 4) проверка работы модуля рейтинговой таблицы;
- 5) проверка работы модуля администраторской панели;
- 6) проверка работы всех модулей с помощью нагрузочного тестирования.

Инструменты

Система контроля версий Git

Для разработки программного комплекса для проведения соревнований в области информационной безопасности было решено использовать Git.

Git — распределённая система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux как противоположность системе управления версиями Subversion (также известная как «SVN») [1].

При работе над одним проектом команде разработчиков необходим инструмент для совместного написания, резервного копирования и тестирования программного обеспечения. Используя Git, мы имеем:

- возможность удаленной работы с исходными кодами;
- возможность создавать свои ветки, не мешая при этом другим разработчикам;

- доступ к последним изменениям в коде, т.к. все исходники хранятся на сервере `git.keva.su`;

- исходные коды защищены, доступ к ним можно получить лишь имея RSA-ключ;
- возможность откатиться к любой стабильной стадии проекта.

Основные постулаты работы с кодом в системе Git:

- каждая задача решается в своей ветке;
- необходимо делать коммит как только был получен осмысленный результат;
- ветка `master` мерджится не разработчиком, а вторым человеком, который производит вычитку и тестирование изменения;
- все коммиты должны быть осмысленно подписаны/прокомментированы.

Для работы над проектом был поднят собственный репозиторий на сервере `gitlab2.keva.su`.

Исходные файлы проекта:

`git@gitlab2.keva.su:sibirctf/sibirctf-attack-defense.git`

Система компьютерной вёрстки \TeX

\TeX — это созданная американским математиком и программистом Дональдом Кнудом система для вёрстки текстов. Сам по себе \TeX представляет собой специализированный язык программирования. Каждая издательская система представляет собой пакет макроопределений этого языка.

\LaTeX — это созданная Лэсли Лэмпортом издательская система на базе \TeX 'а [2]. \LaTeX позволяет пользователю сконцентрировать свои усилия на содержании и структуре текста, не заботясь о деталях его оформления.

Для подготовки отчётной и иной документации нами был выбран \LaTeX так как совместно с системой контроля версий Git он предоставляет возможность совместного создания и редактирования документов. Огромным достоинством системы \LaTeX то, что создаваемые с её помощью файлы обладают высокой степенью переносимости [3].

Совместно с \LaTeX часто используется \BibTeX — программное обеспечение для создания форматированных списков библиографии. Оно входит в состав дистрибутива \LaTeX и позволяет создавать удобную, универсальную и долговечную библиографию. \BibTeX стал одной из причин, по которой нами был выбран \LaTeX для создания документации.

Язык программирования Python

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций [4].

Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули.

Основными преимуществами языка программирования Python являются большое количество библиотек, кроссплатформенность, широкие возможности профилирования кода.

Язык обладает чётким и последовательным синтаксисом, продуманной модульностью и масштабируемостью, благодаря чему исходный код написанных на Python программ легко читаем. При передаче аргументов в функции Python использует вызов по соиспользованию.

Разработка языка Python была начата в конце 1980-х годов сотрудником голландского института CWI Гвидо ван Россумом.

База данных MongoDB

Для реализации программного комплекса для проведения соревнований в области информационной безопасности была использована база данных MongoDB.

MongoDB — документо-ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Написана на языке C++ [5].

Основные возможности MongoDB:

- документо-ориентированное хранение (JSON-подобная схема данных);
- достаточно гибкий язык для формирования запросов;
- динамические запросы;
- поддержка индексов;
- профилирование запросов;
- быстрые обновления «на месте»;
- эффективное хранение двоичных данных больших объёмов, например, фото и видео;
- журналирование операций, модифицирующих данные в базе данных;
- поддержка отказоустойчивости и масштабируемости: асинхронная репликация, набор реплик и распределения базы данных на узлы;
- может работать в соответствии с парадигмой MapReduce;
- полнотекстовый поиск, в том числе на русском языке, с поддержкой морфологии.

Архитектура:

СУБД управляет наборами JSON-подобных документов, хранимых в двоичном виде в формате BSON. Хранение и поиск файлов в MongoDB происходит благодаря вызовам протокола GridFS. Подобно другим документо-ориентированным СУБД (CouchDB и др.), MongoDB не является реляционной СУБД. В СУБД:

- нет такого понятия, как «транзакция». Атомарность гарантируется только на уровне целого документа, то есть частичного обновления документа произойти не может;
- Отсутствует понятие «изоляции». Любые данные, которые считываются одним клиентом, могут параллельно изменяться другим клиентом.

В MongoDB реализована асинхронная репликация в конфигурации «ведущий — ведомый» (англ. master — slave), основанная на передаче журнала изменений с ведущего узла на ведомые. Поддерживается автоматическое восстановление в случае выхода из строя ведущего узла. Серверы с запущенным процессом `mongod` должны образовать кворум, чтобы произошло автоматическое определение нового ведущего узла. Таким образом, если не используется

специальный процесс-арбитр (процесс `mongod`, только участвующий в установке кворума, но не хранящий никаких данных), количество запущенных реплик должно быть нечётным.

TrackingTime - Система управления проектами и задачами

TrackingTime — это сервис, предоставляющий возможность управлять своими проектами, задачами, персоналом.

Основным преимуществом является удобный учет времени ответственного за задачу. Проект декомпозируется на этапы и задачи. Каждая задача назначается на ответственного работника, который в свою очередь при её выполнении устанавливает таймер.

Приложение доступно на всех основных платформах (Windows, Linux, OS X, iOS, Android).

Flask - микрофреймворк для Python

Flask - это микрофреймворк, написанный на языке программирования Python, основанный на Werkzeug и Jinja 2. Выпускается под BSD лицензией.

Слово «микро» означает, что цель написания фреймворка - сохранить ядро простым, но в то же время легко расширяемой. По умолчанию Flask не включает в себя уровень абстракции базы данных, валидацию форм или другие библиотеки, которые с легкостью можно подключить. Вместо этого Flask поддерживает механизм расширения существующего кода так, как будто он уже был подключен к нему. Множество расширений предоставляют интеграцию с базой данных, валидацию форм, поддержку загрузки файлов, аутентификации пользователя и другие полезные функции.

Особенности фреймворка

- удобная работа с URL;
- богатые возможности шаблонизатора;
- минимальные требования к ресурсам в сравнении с аналогичными фреймворками.

Flask используется в модуле таблицы рейтинга.

ArgParse - инструмент для создания интерфейсов командной строки

Argparse — это библиотека для обработки аргументов (ключей, параметров) командной строки для языка программирования Python.

Основные возможности Argparse:

- анализ документов `sys.argv`;
- конвертирование строковых аргументов в объекты программы и работа с ними;
- форматирование и вывод информативных подсказок;
- обработка позиционных аргументов;
- поддержка субкоманд.

Позиционные аргументы — это аргументы, влияющие на работу программы, в зависимости от порядка, в котором они в эту программу передаются. Простейший пример — программа `ср`, имеющая минимум 2 таких аргумента («`ср source destination`»).

Zabbix - система мониторинга сервисов компьютерной сети

Zabbix — свободная система мониторинга и отслеживания статусов разнообразных сервисов компьютерной сети, серверов и сетевого оборудования.

Архитектура Zabbix:

- Zabbix-сервер — это ядро программного обеспечения Zabbix. Сервер может удаленно проверять сетевые сервисы, является хранилищем, в котором находятся все конфигурационные, статистические и оперативные данные;
- Zabbix-прокси — собирает данные о производительности и доступности от имени Zabbix сервера. Все собранные данные заносятся в буфер на локальном уровне и передаются Zabbix серверу, к которому принадлежит прокси-сервер;
- Zabbix-агент — контроль локальных ресурсов и приложений (таких как жесткие диски, память, статистика процессора и т. д.) на сетевых системах. Эти системы должны работать с запущенным Zabbix агентом;
- Веб-интерфейс — интерфейс является частью Zabbix сервера, и, как правило (но не обязательно), запущен на том же физическом сервере, что и Zabbix сервер. Работает на PHP, требует веб-сервер (например, Apache).

Технические характеристики

Требования к аппаратному обеспечению

Необходимо не менее двух компьютеров, находящихся в локальной сети (компьютеры команд-участников), а также сервер, на котором запускается платформа.

Минимальные системные требования для сервера:

- процессор 1ГГц Pentium 4;
- оперативная память 512 Мб;
- место на жёстком диске – 9 Гб.

Минимальные системные требования для компьютеров команд-участников определяются исходя из написанных сервисов.

Требования к программному обеспечению

Для корректной работы разрабатываемого программного комплекса сервер должен работать под управлением ОС Ubuntu Linux. Также должны быть установлены следующие пакеты: Python 3.X, MongoDB, PHP 5, MySQL, Веб-сервер Apache 2 или Nginx.

Требования к ПО команд-участников выставляются уже при непосредственном проведении соревнований.

Требования к сервисам

Для каждого сервиса необходимо иметь следующие компоненты:

- сервис - программа, имеющая уязвимости;
- чекер - программа, реализующая интерфейс работы между сервисом и проверяющей системой.

Чекер должен содержать в себе 3 метода:

- 1) Check — проверка доступности сервиса. В параметрах передается адрес сервиса команды-участника;
- 2) Put — отправление флага на сервис команды-участника. В параметрах передается адрес сервиса, идентификатор флага и флаг;
- 3) Get — получение флага по его идентификатору. В параметрах передается адрес сервиса, идентификатор флага и флаг.

Вызов чекера происходит при помощи вызова исполняемого файла с необходимыми параметрами. Первым параметром является название метода. В ответ чекер должен возвращать одно из четырех состояний, обозначенного кодами 101-104:

- 101 — сервис команды работает и работает корректно;
- 102 — сервис команды работает, но на каком-то этапе отвечает некорректно;
- 103 — сервис команды отвечает, но не работает из-за какой-либо ошибки;
- 104 — сервис команды не отвечает на запрос.

Ниже приведен шаблон кода программы чекера. Шаблон написан на языке Python.

```
#!/usr/bin/python
# coding=utf-8

from sys import argv, exit

def check(hostname):
    return 101

def put(hostname, id, flag):
    return 101

def get(hostname, id, flag):
    return 101

if __name__ == '__main__':
    if len(argv) > 1:
        if argv[1] == "check":
            if len(argv) > 2:
                exit(check(argv[2]))
        elif argv[1] == "put":
            if len(argv) > 4:
                exit(put(argv[2], argv[3], argv[4]))
        elif argv[1] == "get":
            if len(argv) > 4:
                exit(get(argv[2], argv[3], argv[4]))
    exit(110)
```

Разработка программного обеспечения

Архитектура

Ядро

При запуске платформы ядро выполняет следующие действия:

- Получение инициализационных данных из API;
- Создание необходимых таблиц и заполнение их инициализационными данными;
- Сохранение структурированных данных в переменной;
- Подключение модуля, на основе входных параметров.

В настоящий момент доступны следующие модули:

- Прием и проверка принятых флагов;
- Таблицы результатов;
- Организации работы с сервисами.

Модуль: прием и проверка принятых флагов

В соревнованиях по информационной безопасности задача команд найти уязвимость и эксплуатировать её, добывая секретную информацию, в нашем случае флаги. Целью модуля является прием и проверка на валидность флагов.

Принцип работы

Программа реализована с использованием вебсокетов. На порту, полученном с API, программа сравнивает IP адрес клиента с данными о IP адресах клиента или его подсети в базе данных и при нахождении его, клиент определяется как одна из команд и может отправить серверу строку. Строка проверяется на длину символов. Так же флаг проверяется на наличие в базе данных, времени его жизни (флаги валидны определенное количество времени), принадлежность другой команде (свои флаги сдавать нельзя) и статус сервиса (аналогичный сервис сдающей команды должен быть поднят).

Ниже представлен алгоритм работы flags.py (Рисунок 5.1)

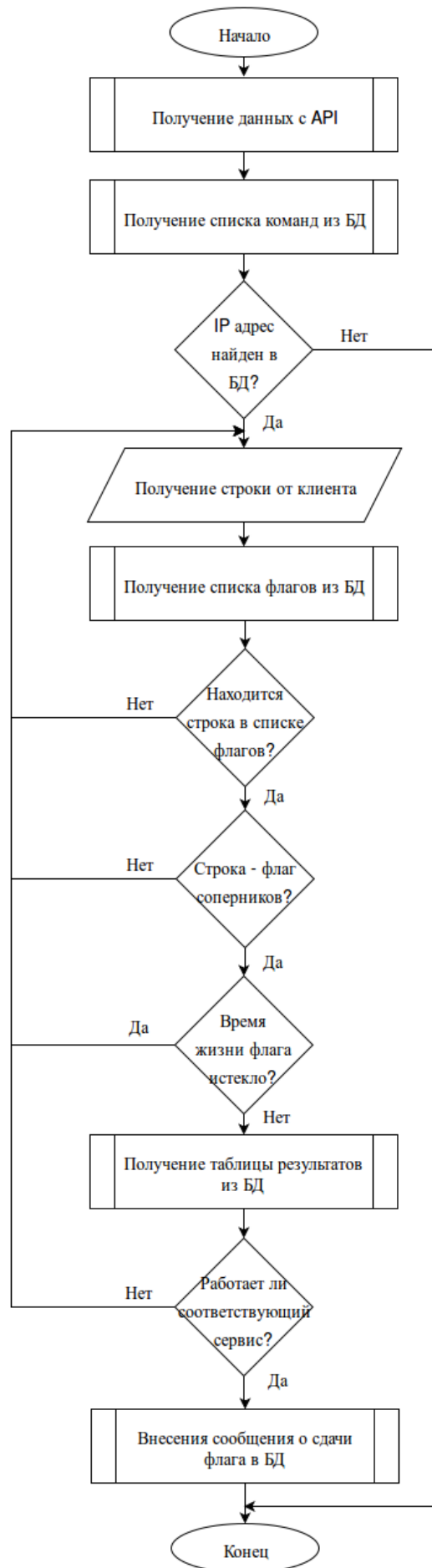


Рисунок 5.1 – Алгоритм модуля flags.py

Модуль: таблицы результатов

Целью работы в текущем семестре стало написание программного модуля, осуществляющего отображения состояния работы сервисов. Описание ошибок у сервисов показываются только для своей команды.

Принцип работы

Программа реализована с использованием микрофреймворка flask. Программный модуль получает данные из базы данных, выводит результат состояний работы сервисов каждой команды на html страницу.

Ниже представлена схема работы scoreboard.py (Рисунок 5.2).

Таблица результатов тренировочной игры

Сейчас идет 493 раунд

Всем удачной игры

Team	whoami	Blozhik
1 Dima Score: 234 10.53.254.155	48% 0 234	0% 0 0
2 Rita Score: 218 10.53.254.134	44% 0 218	0% 0 0
3 ADS Score: 155 10.53.254.125	0% 0 0	31% 0 155
4 Tima Score: 150 10.53.254.127	0% 0 0	30% 0 150
5 Dima N. Score: 123 10.53.254.163	0% 0 0	22% 1 122

Рисунок 5.2 – Результат работы scoreboard.py

Алгоритм работы модуля scoreboard представлен ниже (Рисунок 5.3).

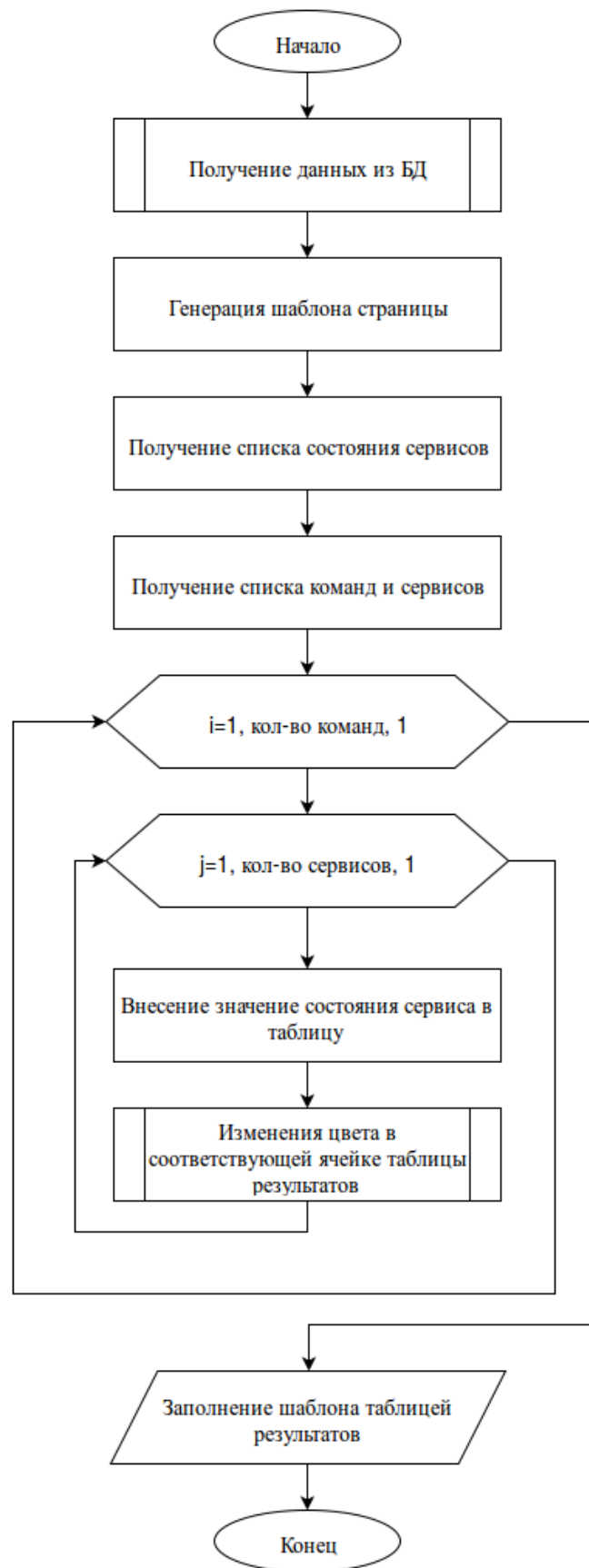


Рисунок 5.3 – Схема работы scoreboard.py

Модуль: организации работы с сервисами

Основное назначение платформы - общение с сервисами команд-участников посредством отправки и проверки специально сгенерированных флагов на сервере. Предназначение этого модуля - работа с интерфейсом сервисов (чекерами).

Принцип работы

Игра делится на раунды, каждый раунд, обычно, длится 1 минуту (настраивается при инициализации). Интерфейс для работы модуля с сервисом называется чекер. За этот период модуль опрашивает с помощью чекеров все сервисы команд-участников. Структура взаимодействия модуля с сервисами представлена на рисунке 5.4.

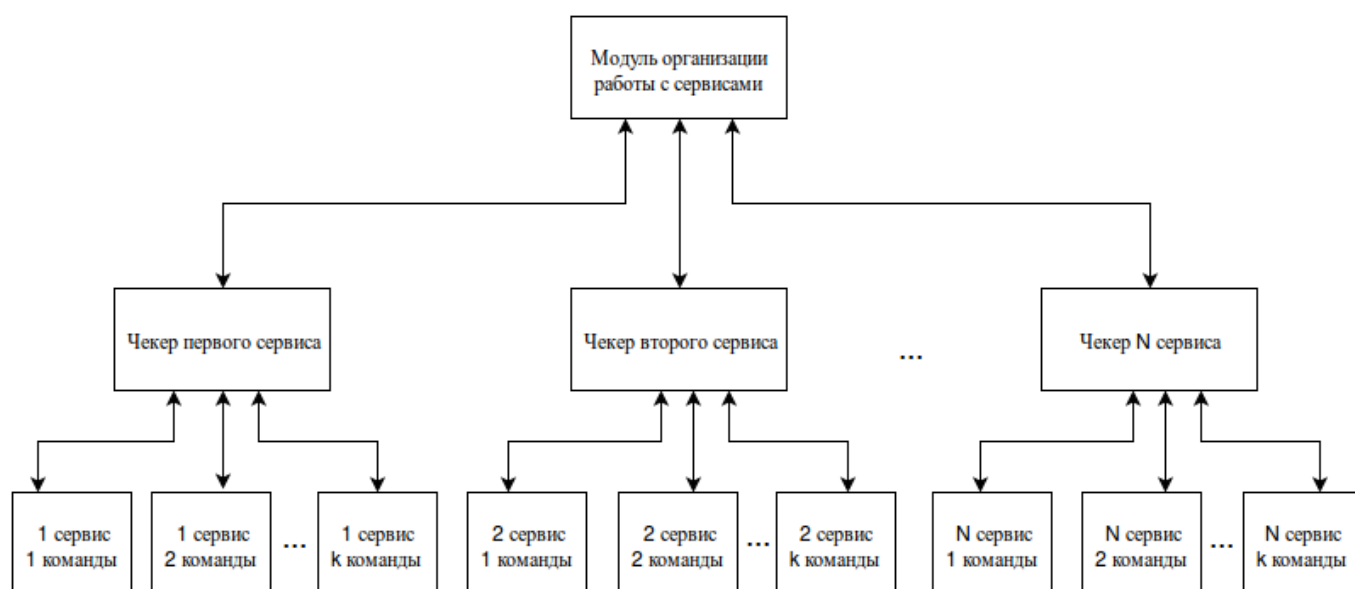


Рисунок 5.4 – Иерархическая структура взаимодействия модуля с сервисами

Опрос происходит в три этапа:

- 1) Проверка работоспособности сервиса команды-участника;
- 2) Отправление флага на сервис;
- 3) Проверка того, что флаг сохранен.

На каждом этапе модуль ожидает в ответ один из четырех чисел: 101, 102, 103, 104. Эти числа также называются статусом сервиса команды-участника. Алгоритм работы представлен на рисунке 5.5.

Число 101 соответствует успешной работе сервиса команды-участника. Число 102 означает, что сервис команды работает, но на каком-то этапе отвечает некорректно. Число 103 означает, сервис команды отвечает, но не работает из-за какой-либо ошибки. Число 104 означает, что сервис команды не отвечает на запрос.

Если на каком-то этапе чекер возвращает число отличное от 101, дальнейшее выполнение программы прекращается, а статус чекера записывается в базу данных.

На 1 этапе каждому чекеру посылается адрес сервера команды-участника. На 2 этапе каждому чекеру посылается адрес сервера, идентификатор флага и сам флаг. На 3 этапе

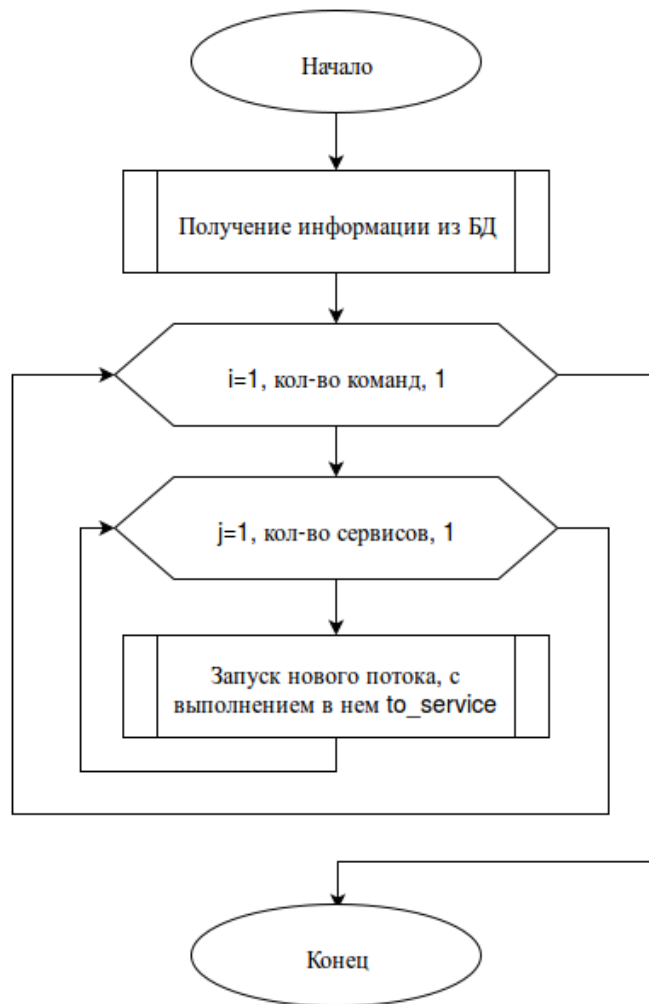


Рисунок 5.5 – Схема создания потоков

каждому чекеру посылается адрес сервера, идентификатор флага и сам флаг.

Алгоритм работы представлен на рисунке 5.6.

Для увеличения производительности, каждый опрос сервисов команд-участников помещается в новый поток. Это позволяет работать в асинхронном режиме. Поэтому нестабильная работа сервиса команды-участника или ошибка в работе чекера не повлияет на опрос других сервисов. Также каждому потоку задается ограничение по времени работы. Это позволяет своевременно завершать процессы, тем самым уменьшается нагрузка на процессор и на потребление памяти.

Развитие экосистемы

У организаторов всегда возникает проблема в выборе платформы для проведения соревнований.

Основные критерии при выборе платформы:

- стабильность работы;
- легкость установки и запуска;
- наличие примеров работы сервиса.

Первые два пункта учитывались на этапе проектирования архитектуры платформы и

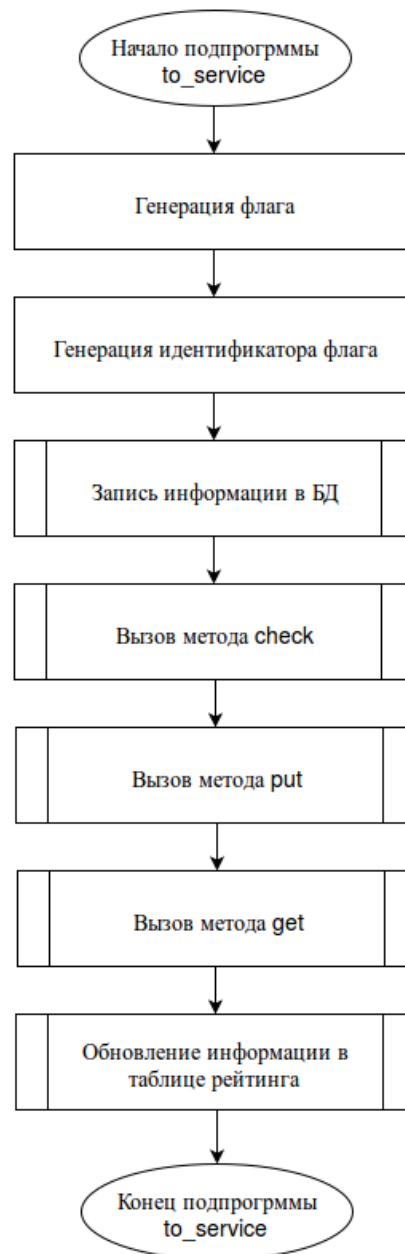


Рисунок 5.6 – Блок-схема работы с каждым сервисом каждой команды-участника

её разработке, эти пункты описаны в руководстве администратора. Ниже описаны сервисы, использующиеся как пример работы сервиса и позволяющие глубже понять логику работы платформы.

Сервис whoami

При разработке сервиса было поставлено несколько целей:

- Тестирование программного комплекса;
- Проведение тренировочной игры для команды;
- Упаковка сервиса в «стартовый набор» для ознакомления.

Сервис написан на языке Python с использованием базы данных MongoDB и микрофреймворка Flask.

Тестирование платформы заключалось в тестировании следующих функций:

- Работа чекера (методы check, get, put);
- Работа модуля сдачи флагов;
- Работа платформы в условиях медленного «ответа» сервиса.

Смысл сервиса состоит в том, что при регистрации пользователь указывает свою электронную почту, пароль и имя. Если пользователь уже регистрировался, то он просто входит в систему, при условии, что логин и пароль совпадают (Рисунок 6.1). Далее пользователь видит приветственное сообщение в формате «Привет, <имя пользователя>»

В сервисе содержится несколько уязвимостей:

- Открытый порт БД (незащищенный паролем);
- Уязвимость в данных cookie.

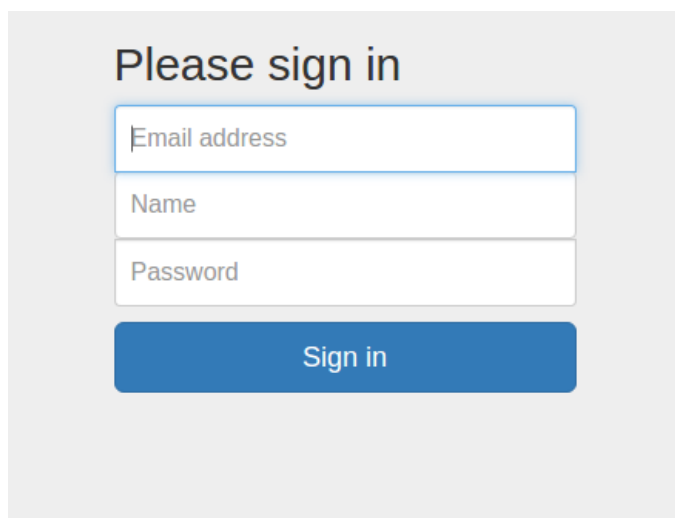
The image shows a login interface with a light gray background. At the top, the text "Please sign in" is displayed in a dark font. Below this, there are three stacked input fields. The first field is labeled "Email address" and has a blue border. The second field is labeled "Name" and the third is labeled "Password". Below these fields is a blue button with the text "Sign in" in white.

Рисунок 6.1 – Форма авторизации сервиса

Сервис blozhik

Сервис был написан для:

- проверки корректности работы платформы;
- выявление возможных проблем, которые могут произойти на соревнованиях;
- проведения тренировочной игры.

Сервис писался на языке Python с использованием микрофреймворка Flask.

Тестирование платформы заключалось в тестировании следующих функций:

- Работа чекера (методы check, get, put);
- Работа модуля сдачи флагов.

Смысл сервиса состоит в том, что в веб-приложении находится форма регистрации, которая сохраняет эл. почту и пароль в текстовый файл. На следующей странице есть форма, в которую нужно ввести эл. почту (введенную ранее в форму регистрации), после чего сервис выдает соответствующий пароль.

Несмотря на простоту сервиса, в нем содержится несколько уязвимостей:

- секретный URL;
- backdoor, с проверкой хеш-суммы эл. почты;
- уязвимость path-traveling (Рисунок 6.2).

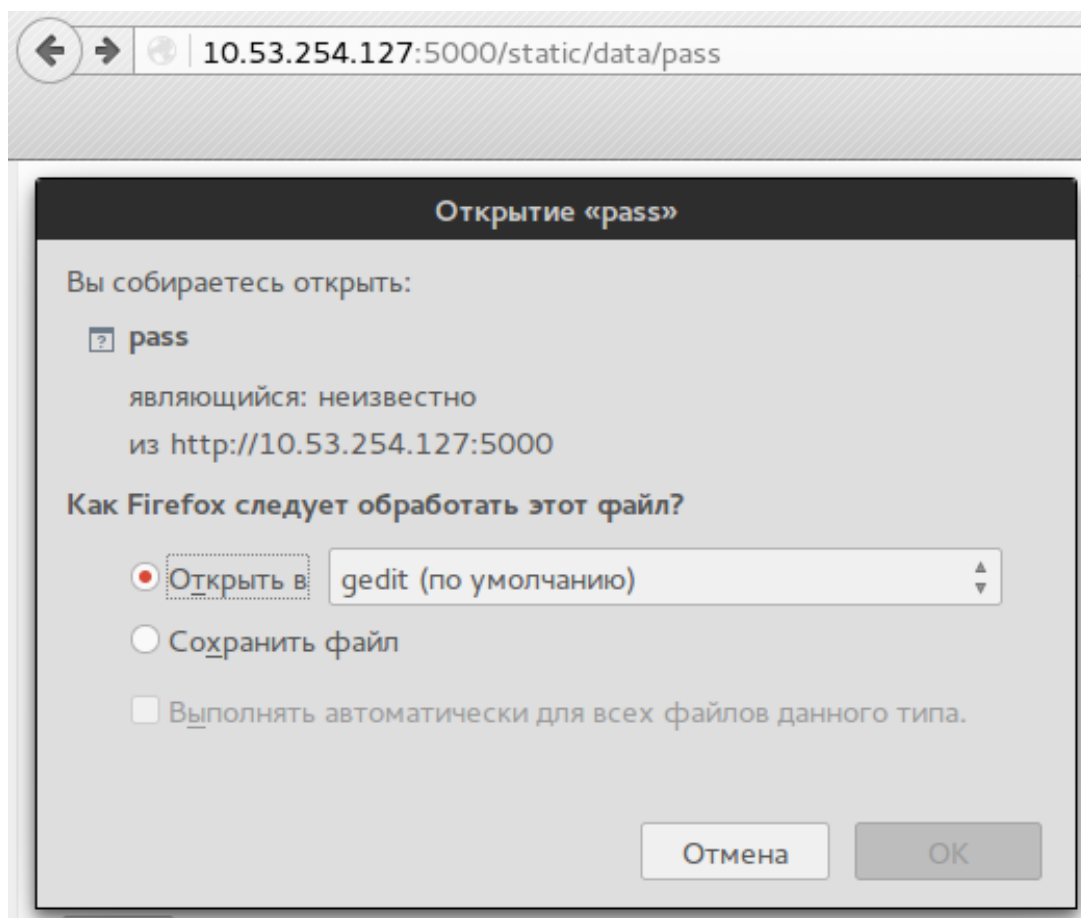


Рисунок 6.2 – Эксплуатирование уязвимости path-traveling

Zabbix

В нашем проекте Zabbix был использован для:

- мониторинга состояния сети;
- контролирование ресурсов сервера.

Была проделана следующая работа:

- был установлен и настроен Zabbix-сервер;
- для роутера и свитча были написаны шаблоны для получения данных по SNMP;
- для сервера был установлен и настроен Zabbix-агент, для получения данных с агента используется стандартный шаблон.

Items	Ubuntu
Agent ping	1
Available memory	415.05 MB
Checksum of /etc/passwd	1861087935
Context switches per second	183 ops
CPU idle time	73.12 %
CPU interrupt time	0 %
CPU iowait time	11.87 %
CPU nice time	0 %
CPU softirq time	0.13 %
CPU steal time	0 %
CPU system time	2.08 %
CPU user time	74.71 %

Рисунок 6.3 – Просмотр данных о сервере

Работа над шаблонами:

- Для начала были получены с устройств строки OID;
- Для каждой строки OID было написано правило обнаружения в шаблоне;
- Далее создавались элементы данных и триггеры;
- Были созданы несколько графиков.

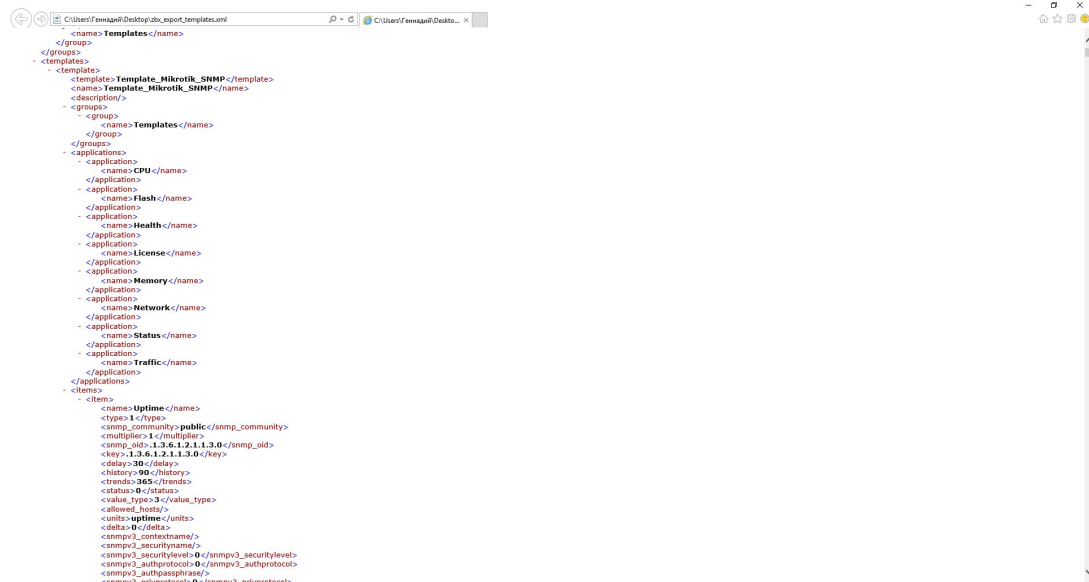


Рисунок 6.4 – Пример готового шаблона

Планы на будущее

- 1) Составление руководств пользователя и администратора;
- 2) Проведение соревнований SibirCTF2016;
- 3) Привлечение к использованию данного программного комплекса других команд.

Заключение

В данном семестре нашей группой была выполнена часть работы по созданию платформы для проведения соревнований в области информационной безопасности, определены основные векторы развития и поставлены задачи для дальнейшего развития платформы.

Список использованных источников

- 1 Scott Chacon. Pro Git : professional version control. 2011. URL: <http://progit.org/ebook/progit.pdf>.
- 2 С.М. Львовский. Набор и вёрстка в системе \LaTeX . МЦНМО, 2006. С. 448.
- 3 И. А. Чеботаев, П. З. Котельников. $\text{\LaTeX} 2_{\epsilon}$ по-русски. Сибирский Хронограф, 2004. 489 с.
- 4 Python. [Электронный ресурс] // [ru.wikipedia.org:\[сайт\]](https://ru.wikipedia.org/wiki/Python). 2015. URL: <https://ru.wikipedia.org/wiki/Python>.
- 5 The MongoDB 3.2 Manual. [Электронный ресурс]. 2015. URL: <https://docs.mongodb.org/manual/>.

Приложение А
(Обязательное)
Компакт-диск

Компакт-диск содержит:

- Электронную версию пояснительной записки в форматах *.tex и *.pdf;
- Актуальную версию программной платформы для проведения соревнований по информационной безопасности;
- Платформа панели администратора и API к ней.