**Тестовое задание**

**Данные:**

* **ФИО:** Громов Максим Игоревич
* **Тел:** +79774987136
* **Email:** richard1703@mail.ru
* **Tg:** @plakl

**1 Вопросы для разогрева**

**1. Расскажите, с какими задачами в направлении безопасной разработки вы сталкивались?**

В период учебной практики (летом 2023 года) я занимался оценкой уязвимостей веб-приложений, искал способы для их смягчения или решения. Одной из основных задач было выявление потенциальных уязвимостей, таких как IDOR, SQL-инъекции, XSS.

**2. Если вам приходилось проводить security code review или моделирование угроз, расскажите, как это было?**

Да, приходилось проводить code review и статический анализа кода для выявления уязвимостей. Все происходило по тому же принципу, что в тестовом задании, на практике мне давали код, в нем была какая-то уязвимость, я пытался ее найти, проэксплуатировать и описать последовательность действия для этого.

**3. Если у вас был опыт поиска уязвимостей, расскажите, как это было?**

На практике мне давали уязвимый локальный сайт, я пытался найти уязвимость используя инструменты для автоматического сканирования, но в основном это было ручное тестирование, я вводил в поле регистрации или в поле поиска каких-либо товаров полезную нагрузку, которая должна была нарушить логику работы сайта, тем самым я выявлял уязвимость и комментировал, какой импакт может получить с этого злоумышленник.

**4. Почему вы хотите участвовать в стажировке?**

После пройденной практики меня заинтересовало направление безопасной разработки.

Сейчас я пишу диплом на тему Безопасности приложений. Мне нравится это направление, я хочу расти и развиваться в данной области, хочу работать с реальными задачами бизнеса, больше работать руками.

**2. Security code review**

**Часть 1. Security code review: GO**

Требуется провести анализ кода на GO с точки зрения безопасности и подготовить отчет по следующим пунктам:

1. Какие уязвимости присутствуют в этом фрагменте кода?
2. Указать строки, в которых присутствуют уязвимости.
3. К каким последствиям может привести эксплуатация найденных уязвимостей злоумышленником?
4. Описать способы исправления уязвимостей. Если уязвимость можно исправить несколькими способами, необходимо перечислить их, выбрать лучший по вашему мнению и аргументировать свой выбор.

Код 1:

package main

import (

"database/sql"

"fmt"

"log"

"net/http"

"github.com/go-sql-driver/mysql"

)

var db \*sql.DB

var err error

func initDB() {

db, err = sql.Open("mysql", "user:password@/dbname")

if err != nil {

log.Fatal(err)

}

err = db.Ping()

if err != nil {

log.Fatal(err)

}

}

func searchHandler(w http.ResponseWriter, r \*http.Request) {

if r.Method != "GET" {

http.Error(w, "Method is not supported.", http.StatusNotFound)

return

}

searchQuery := r.URL.Query().Get("query")

if searchQuery == "" {

http.Error(w, "Query parameter is missing", http.StatusBadRequest)

return

}

query := fmt.Sprintf("SELECT \* FROM products WHERE name LIKE '%%%s%%'", searchQuery)

rows, err := db.Query(query)

if err != nil {

http.Error(w, "Query failed", http.StatusInternalServerError)

log.Println(err)

return

}

defer rows.Close()

var products []string

for rows.Next() {

var name string

err := rows.Scan(&name)

if err != nil {

log.Fatal(err)

}

products = append(products, name)

}

fmt.Fprintf(w, "Found products: %v\n", products)

}

func main() {

initDB()

defer db.Close()

http.HandleFunc("/search", searchHandler)

fmt.Println("Server is running")

log.Fatal(http.ListenAndServe(":8080", nil))

}

**1. Какие уязвимости присутствуют в этом фрагменте кода?**

1. SQL-инъекция;
2. Захардкоженные креды

**2. Указать строки, в которых присутствуют уязвимости.**

1. SQL-инъекция в строке кода:

query := fmt.Sprintf("SELECT \* FROM products WHERE name LIKE '%%%s%%'", searchQuery)

rows, err := db.Query(query)

**3. К каким последствиям может привести эксплуатация найденных уязвимостей злоумышленником? Описать способы исправления уязвимостей. Если уязвимость можно исправить несколькими способами, необходимо перечислить их, выбрать лучший по вашему мнению и аргументировать свой выбор.**

1. SQL-инъекция

Это может привести к выполнению злоумышленником злонамеренного SQL запроса, который будет использоваться в его интересах, например, злоумышленник сможет изменят или удалять записи в базу данных. Также это может привести к утечке данных.

Для того, чтобы решить эту проблему, необходимо использовать параметризованные переменные, либо ORM.

Как мне кажется, лучше всего использовать параметры, так как передача данных идет отдельно от самого запроса, что исключает возможность внедрения вредоносного SQL запроса. Так же параметры помогают избежать проблемы с экранированием спец. символов

**Часть 2: Security code review: Python**

Требуется определить тип уязвимости в примерах кода на Python и ответить на следующие вопросы:

1. Указать строки, в которых присутствуют уязвимости.
2. К каким последствиям может привести эксплуатация данных уязвимостей злоумышленником?
3. Описать способы исправления уязвимостей.
4. Если уязвимость можно исправить несколькими способами, необходимо перечислить их, выбрать лучший по вашему мнению и аргументировать свой выбор.

Код 2.1:

from flask import Flask, request

from jinja2 import Template

app = Flask(name)

@app.route("/page")

def page():

name = request.values.get('name')

age = request.values.get('age', 'unknown')

output = Template('Hello ' + name + '! Your age is ' + age + '.').render()

return output

if name == "main":

app.run(debug=True)

Код 2.2:

from flask import Flask, request

import subprocess

app = Flask(name)

@app.route("/dns")

def dns\_lookup():

hostname = request.values.get('hostname')

cmd = 'nslookup ' + hostname

output = subprocess.check\_output(cmd, shell=True, text=True)

return output

if name == "main":

app.run(debug=True)

**1. Указать строки, в которых присутствуют уязвимости.**

Для кода 2.1:

1. Уязвимость SSTI в строке:

output = Template('Hello ' + name + '! Your age is ' + age + '.').render()

Для кода 2.2:

1. Command Injection:

output = subprocess.check\_output(cmd, shell=True, text=True)

**2. К каким последствиям может привести эксплуатация данных уязвимостей злоумышленником?**

**Описать способы исправления уязвимостей.**

Для кода 2.1:

1. Уязвимость SSTI

Злоумышленник может выполнить произвольный код на сервере, изменив шаблон страницы, это позволит получить доступ ко всем данным на сервере, изменить содержимое веб-страницы, также есть возможность получить доступ к конфиденциальным данным на сервере, пароли, ключи API и так далее.

Для того, чтобы исправить данную проблему, необходимо использовать безопасные методы шаблонизации, вместо конкатенации строк для создания шаблона следует использовать безопасные методы шаблонизации такие как jinja2, который предоставляет инструменты для безопасного рендеринга шаблонов. Но перед использованием пользовательского ввода в шаблоне следует экранировать данные, чтобы предотвратить выполнение вредоносного кода. Ниже будет представлен код, в котором использован безопасные метод шаблонизации из jinja2:

from flask import Flask, request

from jinja2 import Environment, BaseLoader

app = Flask(\_\_name\_\_)

env = Environment(loader=BaseLoader())

@app.route("/page")

def page():

name = request.values.get('name')

age = request.values.get('age', 'unknown')

template = env.from\_string('Hello {{ name }}! Your age is {{ age }}.')

output = template.render(name=name, age=age)

return output

if \_\_name\_\_ == "\_\_main":

app.run(debug=True)

Здесь данные вставляются в шаблон без возможности выполнения произвольного кода.

Для кода 2.2:

1. Command Injection

Это может привести к тому, что злоумышленник сможет ввести в качестве параметра команду, которую выполнит сервер. Это приведет к выполнению произвольных команд на сервере с правами приложения, что позволит получить несанкционированный доступ, повредить систему, провести атаки на другие серверы, либо изменить данные.

Для того, чтобы решить данную проблему, необходимо избегать использование командной оболочки “shell=True”, так как это опасно из-за возможности инъекции команд злоумышленника. Чтобы это исправить, лучше всего, по моему мнению, использовать массив такого вида, снизу представлен псевдокод:

cmd = [‘nslookup’, hostname]

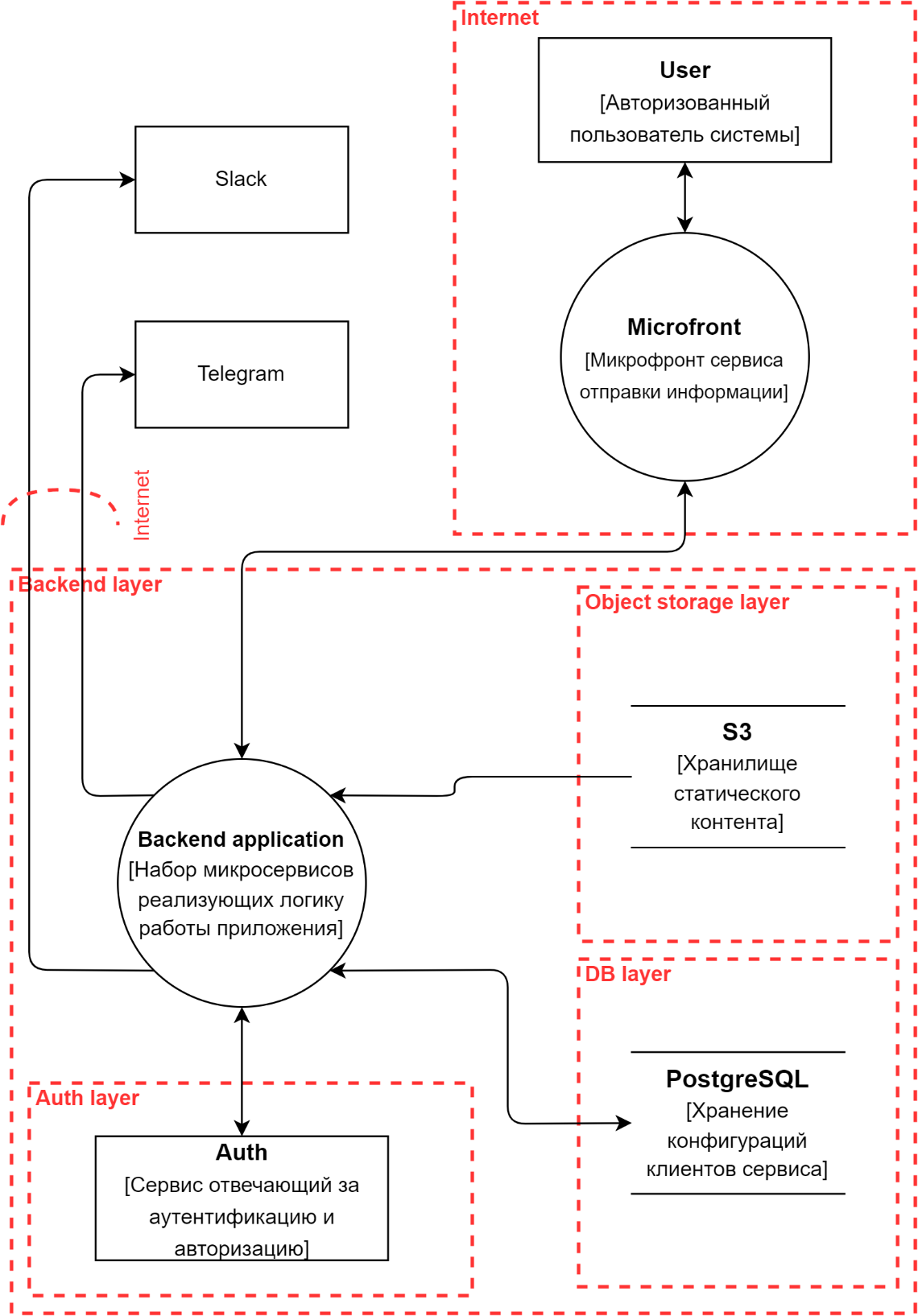
output = subprocess.check\_output(cmd, text=True)

Логика в том, что мы удалим “shell=True” и будем передавать команды и аргументы как массив.

Но, конечно, лучше использовать внутренние возможности языка Python вместо команд ОС.

**3. Моделирование угроз**

Изучите диаграмму потоков данных (Data Flow Diagram, DFD) сервиса, обеспечивающего отправку информации в Telegram и Slack:



Краткое описание компонентов сервиса:

* User - авторизованный пользователь системы. Может настраивать отправку уведомлений и загружать изображения для дальнейшего использования при отправке уведомлений;
* Microfront - микрофронт, которые позволяет взаимодействовать с сервисом отправки информации;
* Backend application - набор микросервисов реализующих бизнес-логику приложения и обеспечивающих взаимодействие со всеми внешними сервисами;
* Auth - сервис отвечающий за аутентификацию и авторизацию клиентов сервиса отправки информации;
* S3 - объектное хранилище, предназначенное для хранения статического контента сервиса отправки информации;
* PostgreSQL - база данных, предназначенная для хранения пользовательских конфигураций сервиса отправки информации.
* Проанализируйте диаграмму потоков данных приложения и ответьте на следующий вопросы:

1. Расскажите, какие потенциальные проблемы безопасности существуют для данного сервиса?
2. Расскажите, к каким последствиям может привести эксплуатация проблем, найденных вами?
3. Расскажите, какие способы исправления уязвимостей и смягчения рисков вы можете предложить по отмеченным вами проблемам безопасности?
4. Напишите список уточняющих вопросов, которые вы бы задали разработчикам данного сервиса?

**1. Расскажите, какие потенциальные проблемы безопасности существуют для данного сервиса?**

1. Пользователь может загружать в хранилище статического контента нелегитимный контент, например, видео, xml-файлы и файлы большого объема;
2. Сервис Auth должен быть защищен от SQL-инъекций и от прочих атак, которые могут привести к компрометации чужих учетных данных и конфигураций;
3. Для безопасного хранения токена Telegram бота следует хранить его в переменных окружения, либо в хранилище секретов.

**2. Расскажите, к каким последствиям может привести эксплуатация проблем, найденных вами?**

1. Отказ в обслуживании сервиса S3, так как хранилище статического контента будет полностью забито файлами большого объема. Также возможность загружать файлы любого расширения может привести к различным уязвимостям;
2. Атаки на сервис Auth могут привести к компрометации чужих учетных данных и конфигураций;
3. Захардкоженный токен Telegram бота может быть скомпрометирован и использован для злонамеренных целей.

**3. Расскажите, какие способы исправления уязвимостей и смягчения рисков вы можете предложить по отмеченным вами проблемам безопасности?**

1. Запретить загрузку файлов отличных от разрешений картинок и ограничить размер загружаемых файлов;
2. Использовать ORM или параметризированные переменные для защиты от SQL-инъекций;
3. Использовать переменные окружения или хранилище секретов.

**4. Напишите список уточняющих вопросов, которые вы бы задали разработчикам данного сервиса?**

1. Какие разрешения файлов доступны для загрузки в хранилище статического контента?
2. Какого объема файлы доступны для загрузки в хранилище статического контента?
3. Каким образом сервис Auth защищен от SQL-инъекций и других атак, которые могут позволить злоумышленнику скомпрометировать учетные данные и конфигурации клиентов сервиса?
4. Каким образом хранится токен Telegram бота?
5. Существует ли white-list пользователей, которые могут получать уведомления в Telegram?