Задание 1

1. найти CVE, относящиеся к недостаткам REST API.

2. на предоставленном стенде найти хотя бы один недостаток из top-10 rest api

Описание/Пошаговая инструкция:

Часть I.
Необходимо среди CVE за последний год найти 3 CVE, относящиеся к недостаткам REST API, входящим в owasp top-10 rest api. Пояснить, в чем заключается недостаток и какова его причина.
Ответ - текстовый файл с описанием.
4 балла
Часть II.

1. Скачать уязвимый образ с docker.hub, выполнив команду из консоли docker pull ket9/otus-devsecops-owasp-rest

2. Запустить образ docker run -d -p 8080:8080 ket9/otus-devsecops-owasp-rest:latest

3. Зайти на http://localhost:8080/, посмотреть, какой функционал доступен в приложении.

4. Наполнить базу данных, перейдя по ссылке http://localhost:8080/createdb После этого вторую часть задания можно выполнить в двух вариациях: либо самостоятельно попробовать найти какой-то недостаток в приложении из owasp top-10 rest api, либо провести эмуляцию атаки (выполнить по шагам уже описанную уязвимость. Файл с описанием приложен к материалам лекции и называется homework-emu.txt) Вне зависимости от того, какой вариант вы выбираете, ответом на дз будет текстовый файл, содержащий:

5. ход ваших рассуждений, что пробовали делать, какой был ответ приложения, какие выводы вы из этого делали.

6. описание найденной уязвимости

7. к какому типу из 10 изученных относится недостаток

8. меры предотвращения 6 баллов

Задание 2

Провести симуляцию XSS-атаки
Цель:

1. найти CVE за 2021 год, содержащие XSS в компонентах, написанных на php.

2. на предоставленном стенде провести XSS-атаку и убедиться в выполнении JS-скрипта

Описание/Пошаговая инструкция выполнения домашнего задания:
Часть I.
Необходимо среди CVE за 2021 год найти те, которые содержат XSS в компонентах, написанных на php. Для двух из них описать, для чего конкретный компонент используется, какой функционал предоставляет. Также необходимо посчитать общее количество cve за 2021 год.
Ответ - текстовый файл.
5 баллов
Часть II.

1. Скачать уязвимый образ с docker.hub, выполнив команду из консоли docker pull ket9/otus-devsecops-xss

2. Запустить образ docker run -d -p 8080:80 ket9/otus-devsecops-xss:latest

3. Зайти на http://localhost:8080/

4. Провести XSS-атаку на приложение и убедиться, что ваш скрипт выполняется в контексте браузера клиента. В материалах к занятию также будет доступен файл homework-emu.txt, где будет описана атака, и к которому вы сможете обратиться в случае сложностей с практической частью дз. Ответом на дз будет текстовый файл, содержащий:

5. ход ваших рассуждений, что пробовали делать, какой был ответ приложения, какие выводы вы из этого делали.

6. тип найденной уязвимости

7. меры предотвращения (напишите подробно, каких символов хотите избегать, что будете для этого делать.) 5 баллов

Задание 3

Разобраться с уязвимостью XXE через XInclude.
Цель:
Часть I. - 5 баллов
Необходимо своими словами описать механизм XInclude, а также то, как возникает уязвимость XXE при его использовании.
Можно подсмотреть ресурс https://book.hacktricks.xyz/pentesting-web/xxe-xee-xml-external-entity

пример https://gist.github.com/jakekarnes42/effe052f1095532cda84307024b3d512
или самостоятельно попробовать пройти пример https://portswigger.net/web-security/xxe/lab-xinclude-attack.
Ответом будет являться текстовый файл с описанием:

1. механизма XInclude, как работает и для чего нужен

2. причина возникновения уязвимости

3. 2 примера того, к чему атака на XInclude может привести Часть II. - 5 баллов Придумать/найти атаку на механизм сериализации/десериализации, приводящую к частичному или полному отказу в обслуживании. Ответ - текстовый файл с описанием. Дополнительно для тех, кто хочет поизучать java-код - не оценивается. В материалах к занятию будет пример java-кода, содержащего уязвимость, необходимо понять, в чем она заключается и исправить ее, не нарушив при этом реализуемый функционал.

Задание 4

Поиск уязвимостей в python-коде.
Цель:
Научиться находить распространенные недостатки в python-коде. Познакомиться с механизмами безопасности в django.
Описание/Пошаговая инструкция выполнения домашнего задания:
Часть 1.
Для приведенного фрагмента кода необходимо указать его недостаток, а также написать рекомендации по его устранению. Например, у такого кода:
with connection.cursor() as cursor:
cursor.execute(""" SELECT admin FROM users WHERE username = '%s' """ % username)

недостаток заключается в том, что введенные пользователем данные не проверяются и "as is" передаются в базу данных. В качестве мер противодействия можно либо корректно обрабатывать пользовательский ввод, либо использовать библиотеки, которые экранируют все опасные символы, что является более предпочтительным вариантов.
Фрагмент кода 1:
```
#!/usr/bin/env python3
from flask import Flask
from mod_api import mod_api
app = Flask('vuln_app')
app.config['SECRET_KEY'] =
'F0cUzh8BgYJSLXAU8qDmClM0dE8GJTpsiyVEl3BCqQMCABp1U
$f%'
app.register_blueprint(mod_api, url_prefix='/api')
```
Фрагмент кода 2:
```
from flask_wtf import Form
from wtforms import TextField
class LoginForm(Form):
username = TextField('username')
password = TextField('password')
```
Фрагмент кода 3:
```
def post(self):
username = self.get_argument('username')
password = self.get_argument('password').encode('utf-8')
email = self.get_argument('mail')
try:
username = username.lower()
email = email.strip().lower()
user = User({'username': username, 'password': password, 'email':
email, 'date_joined': curtime()})
user.validate()
save_user(self.db_conn, user)
except Exception, e:
return self.render_template("success_create.html")
```
Часть 2.
Изучить модель безопасности django.
Ответить на вопросы.

1. Какие наиболее распространенные атаки предотвращает django?

2. Описать, как устроено управление пользовательскими сессиями в django

Задание 5

Найти уязвимости в C/C++ коде
Цель:
Научиться использовать инструменты статического и динамического анализа для поиска уязвимостей в коде на языках C/C++.
Описание/Пошаговая инструкция выполнения домашнего задания:
Инструкция
Для запуска окружения установите docker - get docker

1. Скачайте docker образ sdukshis/devsecops_homework docker pull sdukshis/devsecops_homework

2. Запустите docker контейнер docker run --rm -ti sdukshis/devsecops_homework

3. Склонируйте репозиторий с примерами уязвимых программ git clone https://github.com/sdukshis/devsecops-homework.git cd devsecops-homework

4. С помощью code review, статического и динамического анализа найдите все уязвимости в директории src Используйте примеры и задания с вебинара

5. В чат по ДЗ скопируйте отчет в виде таблицы: имя файла | номер строки | тип уязвимости

Задание 6

Базовая настройка безопасности docker-контейнера
Цель:
Закрепить на практике материал с лекции и научиться на базовом уровне обеспечивать безопасность docker-контейнеров
Описание/Пошаговая инструкция выполнения домашнего задания:
Часть 1

1. скачать любой docker-образ из репозитория docker.io (можно использовать любой из предыдущих домашних заданий)

2. с помощью команды docker image ls посмотреть, какие образы есть на вашей машине

3. до запуска контейнера выполнить команду whoami

4. запустить командную оболочку sh внутри выбранного контейнера с помощью команды docker run -it <имя_образа_из_пункта_2> sh

5. **внутри контейнера выполнить команду whoami и посмотреть пользователя. Сделать скриншот с пунктами 3-5, он и будет ответом на первую часть домашнего задания.**
   Часть 2
6. напишите простейший dockerfile, который выполнит build контейнера на основе базового контейнера из части 1 (например, FROM ket9/otus-devsecops-owasp-rest:latest EXPOSE 8080)

7. дополните dockerfile необходимыми инструкциями, чтобы внутри контейнера приложение запускалось не от пользователя root

8. зайдите внутрь контейнера и выполните whoami. Подумайте, какие еще настройки безопасности можно указать в самом dockerfile и с какими флагами запускать контейнер. Укажите это в комментарии внутри Dockerfile.**Ответом ко второй части задания будет написанный Dockerfile и скриншот из пункта 2.**
Часть 3 Прогнать образ через один из сканеров безопасности: https://github.com/quay/clair https://github.com/aquasecurity/trivy Проанализировать его вывод. Ответ - ваши выводы про безопасность выбранного образа, а также скрин с выводом сканера.

Задание 7

Лабораторная работа по развертыванию Kubernetes-кластера
Цель:
В данной лабораторной работе вам предстоит настроить кластер k8s в облаке, а также сделать некоторые настройки безопасности.
Описание/Пошаговая инструкция выполнения домашнего задания:
Полная пошаговая инструкция лабораторной работы есть в материалах к занятию (Лабораторная работа.pdf)