ВСЕРОССИЙСКАЯ МЕЖДИСЦИПЛИНАРНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ
"НАЦИОНАЛЬНАЯ ТЕХНОЛОГИЧЕСКАЯ ОЛИМПИАДА"

Заключительный этап по профилю "Информационная безопасность"

ОТЧЕТ
о проделанной на финале работе

Подготовила комадна "Просто"

Номер команды: 17

Москва

20.03.2023-25.03.2023

Содержание

Состав команды

Попович Владимир

Фомин Тимофей

Седельников Данила

Тресунов Николай

Часть 1. Наступательная кибербезопасность

Web 10

Flag: nto{w3bs0ck3ts_plu5_xx3_1s_l0v3}

Файл с решением: web1.js

Описание решения:

Сервис представляет из себя сайт по расчёту стоимости страховки. Данные вводятся через форму и отправляются на сервер через вебсокет.

Уязвимость заколючается в том, что сервер отправляет обратно все полученные данные, а при отправке можно указать формат данных (по умолчанию - json). Если изменить тип на xml, то можно будет использовать xxe-уязвимость. Проще всего сделать это с полем countries.

Для того, чтобы получить флаг, находящийся в файле /flag.txt, можно воспользоваться расширением Resource override для браузера Firefox (Скачать: https://addons.mozilla.org/en-US/firefox/addon/resourceoverride/). Создадим новую группу правил (Tab URL: http://10.10.17.10:8080/calculate) и в ней новое правило (From: http://10.10.17.10:8080/static/js/script.js; To: web1.js). Обновим страницу и нажмём "Calculate!". в консоль выведутся данные, полученные с сервера, в том числе и флаг.

В скрипте web1.js сделана замена формата с json на xml и вставлен код, читающий файл с флагом с сервера (xml ENTITY).

Часть 1. Наступательная кибербезопасность

Web 20

Flag: NTO{request_smuggling_917a34072663f9c8beea3b45e8f129c5}

Описание решения:

Сервис 2 (порт 3001) принимает cookie-файлы от сервиса 1 (порт 3002), в которых находятся флаг и имя пользователя. После этого он сравнивает флаг, полученный в cookie, с правильным. Если флаги совпадают, то сервис 2 выводит надпись "Hello, <username>" (<username> - это имя пользователя). Если флаги не совпадают, то сервис выведет надпись "I don't trust you!".

Первая уязвимость находится во 2-м сервисе и заключается в том, что ошибки в запросах никак не обрабатываются, а просто выводятся на экран. В описании ошибки можно увидеть неправильную часть запроса (файл service2/app.py).

Вторая уязвимость находится в 1-м сервисе и заключается в том, что при формировании запроса в него подставляются не фильтруемые сырые данные от пользователя (файл service1/app.py, строка 74).

С помощью второй уязвимости можно вызвать ошибку в запросе, и сервис 2 выведет содержимое запроса. Для этого необходимо зарегистрировать пользователя с именем, содержащим символ новой строки (чтобы отправить его в запросе, необходимо закодировать с помощьюб urlencode: %0A). Это нельзя сделать через форму регистрации на сайте, но возможно через BURP, OWASP ZAP или другие аналогичные программы.

Пример вывода при неправильном запросе:

```
Bad Request Bare CR or LF found in header line  "Cookie: username=;flag=NTO{request_smuggling_917a34072663f9c8beea3b45e8f129c5}" (generated by waitress)
```

Часть 1. Наступательная кибербезопасность

Crypto 10

Flag: nto{5tr4ng3_gr0up_5tr4ng3_l0g_and_depressed_kid_zxc_ghoul}

Файл с решением: solve1.py

Описание решения:

Проходимся по каждому байту и хешируем его. Составляем таблицу. После этого хеш из задания прогоняем по таблице и получаем флаг.

Часть 1. Наступательная кибербезопасность

Crypto 20

Flag: nto{0h_n0_t1m1ng}

Файл с решением: solve2.py

Описание решения:

Из файла в таске понятно, что номер бита, который мы хотим получить из флага, либо случайное число, либо `pow(7, getPrime(300), n)`, понятно, что эти числа распределены неравномерно.

Тогда возьмем средние значение и сравним со значением, которое подберем, из мы можем понять, какой бит `1`, а какой `0`. Так получим флаг.