Бобровый дом | Easy | Web

Информация

Бобры пригласили тебя в гости. Но они потеряли 2 очень ценные вещи. Помоги им найти.

Выдать участинкам

https://vsosh.neimark-it.ru/web

Описание

Найти в html 2 части флага. Сложность в том, что к коду задания добавляется код от CTFd, из-за чего объём исследования увеличивается.

Решение

Исходное содержимое

1. Нажимаем в браузере F12.
2. Открываем Elements
3. Исследуем потихоньку каждый блок на странице
    - Первый способ - вручную раскрыть каждый блок
    - Второй способ - в задании сказано, что есть 2 части флага. Известно, что формат флага `NMARK{}`. Если нажать Ctrl+F и ввести `NMARK`, то получится найти первую часть флага. Далее можно найти `}` и увидеть вторую часть флага.

Флаг

`NMARK{B3av3R_B0br_0dN0_i_t0_zh3}`

Панкосвод | Easy | Crypto

Информация

Котик! Я тут тебе готовила салат, пробовала различные ингредиенты, смешивала. Да что-то так намешала, что получилось очень вкусно. Но вот проблема - я не

помню, что там было изначально. Помоги, пожалуйста, восстановить салат. Флаг был зашифрован приложенным скриптом: NMARK{zhml_rlf_buzhmll_jpwoly_huk_h_jvyylja_mshn_dpao_dvykz_nv_zbiitpa} В программе флаг был убран из кода, восстановите его из зашифрованных данных. В правильно расшифрованном флаге содержится слово correct.

Выдать участинкам

encryption.py

Решение

В приложенной программе реализован шифр Цезаря. Достаточно лишь перебрать небольшое количество возможных ключей и выбрать подходящий, используя условие: "В правильно расшифрованном флаге содержится слово `correct`"
Так как любой сдвиг букв в обратную сторону аналогичен некому обычному сдвигу, при переборе для расшифровки можно переиспользовать функцию из задания. Скрипт расшифровки приложен в solve.py.

Флаг

`NMARK{safe_key_unsafee_cipher_and_a_correct_flag_with_words_go_subbmit}`

Кризотютя | Easy | Crypto

Информация

Котик! Я тебе на 8 марта приготовил подарок, но он как-то перемешался по дороге в пакете. Я не знаю, как его восстановить! Может, ты сможешь? В результате шифрования приложенной программой получился следующий вывод: ciphertext = 6653a882e12bba271dcf307092be3b2aa331e51edb777c8c8fd831a02b1bd13073cbec6514a26aa10f hint = 6b7f87f0d33fbb6f10d90c6487af2855b436f90180 В программе флаг был убран из кода, восстановите его из зашифрованных данных.

Выдать участинкам

encryption.py

Решение

В приложенном скрипте релизован XOR-шифр (шифр Вернама/шифр Виженера с бинарным алфавитом).

Этот шифр (подобно лежащему в его основе оператору XOR) имеет симметрию: если `x ^ y = z`, то `x = y ^ z`, а `y = x ^ z`.
Для шифрования это означает, что достаточно "зашифровать" шифротекст используя открытый текст, как ключ, чтобы получить изначальный ключ. Другими словами, раз `plaintext ^ key = ciphertext`, то `ciphertext ^ plaintext = key`. Поэтому достаточно вычислить `xor_cipher("Can you decrypt this?".encode(), hint)`, чтобы получить ключ.
Подобным образом происходит и расшифровка. Все это приведено в скрипте solve.py.

Флаг

`NMARK{this_flag_could_be_random_e539ab41}`

mini-bobry | Medium | Reverse

Информация

Дорогой друг. Я спокойно шёл по лесу, собирал ягоды и грибы и вдруг встретил кое-что необычное. Это была семья мини-бобров. Они сказали, что не пропустят меня дальше, пока я не принесу им дерево. Да такое дерево, которое им понравится. Помоги мне узнать, какое дерево им нужно.

Вам помогут: uncompyle6 или pycdc

Формат флага - "NMARK{}"

Выдать участникам

main.pyc

Описание

Обычный .pyc. Сдекомпилить и потихоньку разбираться в проверке флага, которая состоит из 3 частей.

Решение

1. Используем uncompyle6 или pycdc для декомпиляции байт-кода в исходный код `pycdc main.pyc -o main.py` - `-o` позволяет задать файл, в который записывается результат или `uncompyle6 main.pyc -o main.py` - `-o` позволяет задать файл, в который записывается результат Так как это был Python 3.7, то uncompyle6 справился лучше Исходный код
2. Ищем код, который начинает выполняться при запуске. Он в строках 67-68. Если запущен именно этот файл, выполняется функция `main_bobr()`
3. В `main_bobr()` видим `try`, `except FileNotFoundError`, `except Exception as e`. Начинает выполняться код в блоке try. Открывается файл `tree.txt` и из него читается строка в переменную `tree`. Если файл не был найден, выполняется блок `except FileNotFoundError`, который выводит сообщение о неудаче, и выполнение завершается. Придругих ошибках выполнение переходит в блок `except Exception as e`, выводится полученное исключение и выполнение завершается.
4. Проверяется, что длина `tree` (А это и есть флаг) равна 52. Иначе сообщение о неудаче
5. Далее вызывается функция `mother_bobr(get_crown(tree))`. В неё передаётся `get_crown(tree)` - содержимое флага до `{`.
   Функция `mother_bobr()` проверяет, что содержимое флага до `{` - это `NMARK`.
   Если не так, выводится сообщение о неудаче и выполнение завершается.
6. Далее вызывается функция `older_mini_bobrik(get_trunk(tree))`. В неё передаётся `get_trunk(tree)` - содержимое флага между `{ т _`.
   Функция `older_mini_bobrik()` выполняет несколько проверок
   - Длина содержимого равна 20
   - С помощью `encode()` кодирует содержимое в байты. Далее проходится по соответствующим символам байтовых строк `constnant` и `trunk`. Функция ксорит поочерёдно все элементы этих строк и формирует из получившихся байтов новую байтовую строку.
   - Далее с помощью `decode("utf-8")` байты в получившейся строке становятся читаемыми символами
   - Получившаяся строка сравнивается с `C001M1n1B0br1k1W0oow`.
   - Если что-то было не так, выводится сообщение о неудаче и выполнение завершается.
   - Чтобы получить из этих данных содержимое флага, нужно строку `C001M1n1B0br1k1W0oow` поксорить с `constnant`. Код приведён в solve.py
7. Далее вызывается функция `younger_mini_bobrik(get_root(tree))`. В неё передаётся `get_root(tree)` - содержимое флага между `_` и `}`.
   Функция `younger_mini_bobrik()` выполняет несколько проверок
   - Длина содержимого равна 24
   - Делит строку на 6 частей по 4 символа

    - Переставляет 4-символьные блоки в порядке, как в списке `indexes = [2, 4, 3, 5, 0, 1]`. То есть сначала идёт второй блок, потом четвёртый и тд.
    - Сравнивает полученный результат со строкой `uuur1B0b1M1nrrssD0N0tHuu`.
    - Чтобы получить содержимое флага, нужно обратно переставить блоки. То есть `uuur` был вторым блоком, `1B0b` был четвёртым блоком и тд. Код приведён в solve.py
8. Если все проверки пройдены, выводится сообщение об успехе
9. При желании можно протестировать полученный флаг. Для этого нужно в текущей диреткории создать файл `tree.txt` и выполнить `python3 main.pyc` либо `python main.pyc` в зависимости от системы.

Флаг

`NMARK{W0w1t1sG1ftF0rB0br1k_D0N0tHuuuuur1M1n1B0brrss}`

human reverse | Hard | Reverse

Информация

Товарищ! Валерия Николаевна сконструировала робота-женщину-ревёрсера, которая поможет разревёрсить Bombombini Gusini. Она отправила тебе помощницу вместе с инструкцией. К сожалению, инструкция потерялась, а ревёрсить надо. А без активации не получится. Придётся тебе разревёсить и активировать помощницу. Для активации нужно отправить какие-то команды. Разберись, какие и в каком формате.

Вам помогут: Ghidra

Формат флага - "NMARK{}"

Выдать участинкам

human

Описание

Читает из переменной окружения флаг и проверяет в 4 этапа.

Решение

Исходный код

1. Читает флаг из переменной окружения `HUMAN_COMMAND`
2. Проверяет, что длина флага равна 27
3. Узнаёт адрес `{`. Затем считает длину содержимого от начала до `{`. Затем выделяет память и копирует содержимое до `{` в отдельное место. Это содержимое - пароль.
4. Передаёт пароль в `brain()`. Так как состояние изначально 0, то, чтобы его изменить, надо попасть в `unlock()`. Для этого длина пароля должна быть равна 5. Если всё так, вызывается `unlock()`, которая проверяет, что пароль - `NMARK`. Состояние меняется на 1.
5. Если состояние поменялось, продолжаем. Вычисляем адрес `}`. Считаем длину содержимого между фигурными скобками - это набор команд, разделённых запятыми. Выделяем память и копируем их в отдельное место.
6. Далее функция `strtok()` ставит `\0` вместо первой запятой и возвращает указатель на начало первой команды. Удаляются пробелы в начале и в конце. Команда копируется в массив words[].
7. Аналогичные операции со второй и третьей командой.
8. Проверяем, что команд 3
9. Выводим эти три команды
10. Передаём каждую команду в `brain()`. Каждая должна изменить состояние на 1
11. Когда передали первую команду, состояние - 1. Значит, её длина должна быть 5 символов. Тогда вызывается `hand()`, которая проверяет, что команда - `HaNd3`, и меняет состояние на 2
12. Для состояния 2 длина команды должна быть 4. Тогда вызывается функция `leg()`. Функция `leg()` проверяет, что команда - `LeG4`, но сравнивает нулевой символ полученной команды с последним символом строки `4GeL`. Затем - первый и предпоследний символ соответственно и тд. Если всё хорошо, меняет состояние на 3.
13. Для 3 состояние длина должна быть 9. Тогда вызывается функция `stomach()`, которая проверяет, что команда - `StOMAcH52`. Сначала функция проверяет чётные символы. Затем проверяет нечётные символы.
14. Объединив результаты, получаем флаг
15. Можно себя проверить. Чтобы установить переменную окружения, делаем `export HUMAN_COMMAND="NMARK{HaNd3,LeG4,StOMAcH52}"`, затем `./human`

Флаг

`NMARK{HaNd3,LeG4,StOMAcH52}`