Отчёт по проекту CTF

Тема: шифрование и базовые криптографические преобразования

Содержание

1 Введение

1.1 Цель работы

Разработать обучающий мини-CTF (Capture The Flag) по теме «Шифрование» в виде статического сайта, публикуемого через GitHub Pages. Участнику доступен только веб-интерфейс: задания выстроены в цепочку, а переход к следующему шагу выполняется только после ввода корректного ответа.

1.2 Идея и формат

Проект представляет собой последовательность криптографических мини-задач: Base64, ROT13, Атбаш, Цезарь, Виженер, XOR, SHA-256, повторное Base64, сборка ключа из промежуточных результатов, финальная проверка через AES-GCM. Такой формат позволяет закрепить базовые операции, часто встречающиеся в CTF.

1.3 Состав проекта (файлы)

Сайт статический и состоит из четырёх ключевых файлов:

- `index.html` — разметка страницы;
- `style.css` — минималистичное оформление;
- `stages.js` — данные заданий (тексты/подсказки) и проверочные значения;
- `app.js` — логика навигации и проверки ответов.

2 Развёртывание на GitHub Pages

1. Убедиться, что `index.html`, `style.css`, `stages.js`, `app.js` лежат в корне репозитория.
2. Закоммитить и запушить изменения в ветку `main`.
3. В репозитории открыть `Settings` → `Pages`.
4. В разделе `Build and deployment` выбрать:
   - `Source`: Deploy from a branch;
   - `Branch`: `main`;
   - `Folder`: `/(root)`.
5. Сохранить настройки. После деплоя появится ссылка.

3 Механика прохождения и проверка ответов

3.1 Навигация по токену

Каждый шаг доступен по URL-фрагменту `#token`. Переходы записываются в историю браузера, поэтому можно возвращаться назад/вперёд стандартными кнопками браузера.

3.2 Нормализация ответа

Перед проверкой ответ приводится к нижнему регистру и обрезаются пробелы по краям:

$$\texttt{answer} \leftarrow \texttt{trim(lower(answer))}$$

3.3 Проверка ответа на шаге (salt + SHA-256)

Для каждого шага в `stages.js` хранится `salt` и контрольная `hash`. Ответ считается корректным, если:

$$\mathrm{SHA256}(\texttt{salt} \,||\, \texttt{|} \,||\, \texttt{answer}) = \texttt{hash}$$

Где `SHA256` возвращает hex-строку длиной 64 символа.

3.4 Получение токена следующего шага

После верного ответа генерируется токен следующего шага:

$$\texttt{nextToken} = \mathrm{SHA256}(\texttt{token:} \,||\, \texttt{answer})[0..11]$$

То есть первые 12 hex-символов.

3.5 Финальная проверка (AES-GCM)

На финальном шаге вводится ключ-пароль `passphrase`, из которого формируется ключ AES-GCM:

$$K = \mathrm{bytes}(\mathrm{SHA256}(\texttt{passphrase}))$$

Затем выполняется расшифрование AES-GCM с заданными параметрами:

- `IV` (base64),
- `CT` (base64),
- `AAD` = `ctf2026`.

Если расшифрование успешно (аутентификация проходит), на сайте выводится сообщение «Поздравляем, вы добрались!». Сам открытый текст используется как внутренний маркер корректности ключа.

3.6 Замечание о «секретности»

Так как сайт полностью клиентский (весь код доступен в браузере), абсолютную «неизвлекаемость» ответов гарантировать нельзя. Однако ответы не хранятся в явном виде: проверка реализована через хэши (salt + SHA-256), что убирает прямое чтение и требует реального решения задач или перебора.

4 Описание заданий (10 шагов)

4.1 Шаг 1

Дано:

```
Y2lwaGVyX3RyYWlsX3N0YXJ0
```

Смысл: тренировка распознавания и декодирования Base64.
Как решается: выполнить Base64-decode и получить осмысленную строку (слово/фразу).
Что вводить: результат декодирования.

4.2 Шаг 2

Дано:

```
Arkg xrl: ebgngvba_xrl
```

Смысл: простой подстановочный шифр (ROT13).
Как решается: применить ROT13 к строке, извлечь значение после двоеточия.
Что вводить: ключ после двоеточия.

4.3 Шаг 3

Дано:

```
XRKSVIYLLP
```

Смысл: классическая моноалфавитная подстановка Атбаш.
Как решается: заменить A↔Z, B↔Y, ..., и получить слово.
Что вводить: расшифрованное слово.

4.4 Шаг 4

Дано:

```
L ORYH FUBSWRJUDSKB
```

Смысл: шифр Цезаря (сдвиг).
Как решается: подобрать сдвиг (по смыслу), расшифровать строку и взять итоговое слово.
Что вводить: итоговое слово.

4.5 Шаг 5

Дано (подсказка к ключу + шифротекст):

```
When the tide rises
A secret becomes visible
Vary the alphabet
Encrypt with a key

ribijemi_hatin
```

Смысл: шифр Виженера + извлечение ключа из первых букв строк.
Как решается:

- взять первые буквы строк: получается ключ `WAVE`;
- расшифровать Виженером шифротекст; символ `_` не изменяется.

Что вводить: полученную строку (с подчёркиванием).

4.6 Шаг 6

Дано (hex-строка):

```
140009041c1c
```

Смысл: XOR с повторяющимся ключом, практика работы с hex/байтами.
Как решается:

- перевести hex в байты;
- взять ключ как ASCII-байты (ответ прошлого шага);
- выполнить XOR побайтно (ключ циклический);
- результат интерпретировать как ASCII-строку.

Что вводить: расшифрованное слово.

4.7 Шаг 7

Дано:

```
A:B:C
A = шаг 3
B = шаг 5
C = шаг 6
```

Смысл: хэширование как способ фиксации/проверки данных.
Как решается:

- собрать строку `A:B:C` из уже найденных ответов;
- посчитать SHA-256 от строки;
- взять первые 10 hex-символов.

Что вводить: первые 10 hex символов хэша.

4.8 Шаг 8

Дано:

```
Wm1sdVlXeHJaWGs9
```

Смысл: распознавание «многослойного» Base64.
Как решается: декодировать Base64 несколько раз, пока не получится осмысленная строка.
Что вводить: итоговое слово.

4.9 Шаг 9

Дано:

```
1) step3[0]
2) step5.split('_')[1][0]
3) step6[0]
4) step8[0]
```

Смысл: сборка ключа из артефактов предыдущих шагов (типичный CTF-приём).
Как решается: взять указанные символы и склеить в строку длиной 4.
Что вводить: полученную 4-буквенную строку.

4.10 Шаг 10 (финал)

Дано:

```
binary|finalkey|clbf

IV: p+f2K1CQgFp3t+0f
CT: +0IWUyBONSwQaXMOEfg4t03B+pL4k/WjzYwqz7EWaoa4QyfMvxV2Kg==
AAD: ctf2026
```

Смысл: полный цикл «получение ключа → симметричное шифрование с аутентификацией».
Как решается:

- посчитать SHA-256 (в байтах) от строки `binary|finalkey|clbf`;
- закодировать результат в Base32 (RFC4648) без символов `=`;
- взять первые 16 символов (в нижнем регистре) как `passphrase`;
- использовать `passphrase` для расшифрования AES-GCM с параметрами IV/CT/AAD.

Что вводить: `passphrase`. При успехе сайт выводит сообщение о завершении.

5 Ответы

5.1 Таблица ответов

Шаг	Ответ
1	`cipher_trail_start`
2	`rotation_key`
3	`cipherbook`
4	`cryptography`
5	`vigenere_layer`
6	`binary`
7	`e28e8519d2`
8	`finalkey`
9	`clbf`
10	`3ozfgpym32dexyry`

5.2 Чистый блок ответов (для копирования)

```
1: cipher_trail_start
2: rotation_key
3: cipherbook
4: cryptography
5: vigenere_layer
6: binary
7: e28e8519d2
8: finalkey
9: clbf
10: 3ozfgpym32dexyry
final_plaintext: CTF{crypto_is_not_magic}
```

A Токены шагов (для проверки навигации)

Токен — это значение после символа # в URL.

Шаг	Token
1	`start`
2	`8f7aa9a64915`
3	`56f0ce4808ae`
4	`f5599f779fa3`
5	`8ef3c6ec3984`
6	`135e61c6e3cc`
7	`c2f0d343e007`
8	`ebc7cebaf82e`
9	`e4d2f335387a`
10	`70b3be9a6158`