## Задание

Дана REST служба с API указанным ниже. Для аутентификации в службе пользователь должен передать её токен через метод ValidateRawToken.

Получить зашифрованный токен можно используя метод GetEncryptedToken.

Проверить корректность зашифрованного токена можно используя метод ValidateEncryptedToken, расшифровывающий токен, и проверяющий его корректность.

Проверить корректность незашифрованного ("сырого") токена можно используя метод ValidateRawToken, проверяющий корректность переданного токена.

Задача - вызвав метод ValidateRawToken, получить сообщение "Raw Token decoded and validated. Wellcome to secretNet!".

Шифрование токенов производится в режиме Mac-Then-Encrypt, CBC-AES, HMAC.

## Ход работы.

### Тестирование

<userId> = имя аккаунта на GitHub (или фамилия студента)

<challengeId> = debug

1. Проверить работоспособность контроллера с помощью метода GET <host>/api/IvIsTime
2. Получить зашифрованный дебаговый токен с помощью метода GET <host>/api/PaddingOracle/<userId>/debug/GetEncryptedToken
3. Восстановить дебаговый токен используя вызовы метода GET <host>/api/PaddingOracle/<userId>/debug//ValidateEncryptedToken/{encryptedToken}
4. Убедиться, что расшифрованный дебаговый токен равен следующей последовательности байт:  
    1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32

<userId> = имя аккаунта на GitHub (или фамилия студента)

<challengeId> = 1

1. Получить зашифрованный токен токен с помощью метода GET <host>/api/PaddingOracle/<userId>/<challengeId>/GetEncryptedToken
2. Восстановить токен используя вызовы метода GET <host>/api/PaddingOracle/<userId>/<challengeId>/ValidateEncryptedToken/{encryptedToken}
3. Аутентифицироваться с использованием метода GET <host>/api/PaddingOracle/<userId>/<challengeId>/ValidateRawToken/<{rawToken}>, получив сообщение "Raw Token decoded and validated. Wellcome to secretNet!"

### Сдача лр

шаги 5 - 7 этапа тестирования аналогично для 10 различных <challengeId>.

## Описание API

Rest запросы, в заголовке выставлен Content-Type: application/json; charset=utf-8.

### Описание методов

## GET <host>/api/PaddingOracle

Проверка работоспособности контроллера. Возвращает operating. Ответ не кодируется в BASE64.

|  |  |
| --- | --- |
| Параметр | Описание |
| <host> | имя хоста веб службы |

## GET <host>/api/PaddingOracle/{userId}/{challengeId}/GetEncryptedToken

Генерирует, подписывает с помощью MAC, и зашифровывает токен в режиме CBC на фиксированном для задания для задания ключе.  
Ответ не кодируется в BASE64.  
Ответ представляет собой hex строку.

|  |  |
| --- | --- |
| Параметр | Описание |
| <host> | имя хоста веб службы |
| <userId> | идентификатор студента |
| <challengeId> | идентификатор задания |

## GET <host>/api/PaddingOracle/{userId}/{challengeId}/ValidateEncryptedToken/{encryptedToken}

Расшифровывает указанный токен и проверяет его валидность.

|  |  |
| --- | --- |
| Параметр | Описание |
| <host> | имя хоста веб службы |
| <userId> | идентификатор студента |
| <challengeId> | идентификатор задания |
| <encryptedToken> | зашифрованный токен, в виде hex строки |

## GET <host>/api/PaddingOracle/{userId}/{challengeId}/ValidateRawToken/{rawToken}

Проверяет переданный "сырой" (незашифрованный) токен и аутентифицирует пользователя.  
Возвращает строку Raw Token decoded and validated. Wellcome to secretNet! в случае корректности токена.

|  |  |
| --- | --- |
| Параметр | Описание |
| <host> | имя хоста веб службы |
| <userId> | идентификатор студента |
| <challengeId> | идентификатор задания |
| <rawToken> | "сырой" (незашифрованный) токен в виде hex строки |

### Реально возможный use-case

Пусть аутентификация на сайте осуществляется с помощью cookie. Часто cookie зашифрованы, и не прозрачны для пользователя, но могут нести  
конфиденциальную информацию. Например, cookie могут содержать персональные данные пользователя. При этом аутентификация производится по зашифрованной cookie.

Если злоумышленник перехватил простроченную cookie пользователя, он не сможет использовать её после аутентификации. Но если время действия хранится внутри cookie  
(как оно часто и бывает), сервер всё равно будет вынужден расшифровать cookie для её проверки. Таким образом противник получает оракул расшифрования, аналогичный методу ValidateEncryptedToken,  
который он может использовать для извлечения информации из cookie. По "сырой" coookie обычно не аутентифицируют, так что метод ValidateRawToken служит в лр  
для валидации результата расшифрования. Иными словами ральная атака происходит не на аутентификацию, а на конфиденциальность информации в cookie.