# Сервис публикации мемов (MemeZ3) - Go:

## Функционал сервиса:

1. регистрация + логин
2. сброс пароля через отправку сообщения с кодом на почту
3. загрузка картинок
4. непосредственная публикация картинок
5. их (рас)шифрование
6. настройка эффектов у пикч
7. просмотр мемов
8. отправка комментариев
9. просмотр метаданных у мемов
10. МОДЕРАТОР проверка мемов согласно выбранной политике

флагсторы:

1. надписи на мемах - ксор 2 пикч с плохим шифрованием
2. метаданные - написать коммент и на экспорт (тоже ямл) + через сброс пароля
3. комментарии модераторов -

критерии вирусной картинки:

1. там есть надпись `Толя`
2. там есть флаг (?) – пусть называется как-то `Раскрытие чужих персональных данных`
3. мем не смешной – можно просто рандомные пикчи без флагов банить)))

что делает чекер by илья:

1. регистрация - логин - пост пикчи (+ проверка сразу и через время)
2. регистрация - логин - пост зашифрованной пикчи (+ проверка сразу и через время)
3. регистрация - логин - написать куда-то комментарий
4. фантазия таво все короч

## Чекер (Рита):

1. put
   * Регистрация (user)
   * Логин

Флагстор 1

* загружаем и публикуем белую картинку без флага (без эффектов с шифрованием)
* добавляем той же картинке флаг
* вычисляем хеш
* загружаем и публикуем (без эффектов с таким же шифрованием на том же ключе)

Флагстор 2

* добавляем флаг в метаданные
* загружаем и публикуем картинку (случайный эффект и шифрование)

Флагстор 3

* загружаем и публикуем вирусную картинку
* заводим новую сессию
* регистрация (moderator)
* логин
* оставляем комментарий с флагом (только для модераторов) к загруженной картинке

flag\_id = {memeId, email, password, data (хеш), key}

1. get
   * Логин (email, password из flag\_id)

Флагстор 1

* Расшифровываем мем
* Проверяем, что хеш совпадает

Флагстор 2

* из /info/{memeId} получаем метаданные
* Проверяем, что флаг есть

Флагстор 3

* + из /info/{memeId} получаем комментарии модератора
  + Проверяем, что флаг есть

1. check

User:

* Регистрация
* Логин
* Загружаем мем (случайно вирусный или нет)
* get/{memeId}
* Генерируем новый тег
* Публикуем с тегом
* Если шифровали, расшифровываем /decrypt/{memeId}
* Получаем список тегов /tags, проверяем, что наш там есть
* если картинка вирусная, проверяем отчет в /info/{memeId}
* Сброс пароля
* Выбираем случайный тег
* Выбираем случайный мем, у которого есть выбранный тег
* Комментируем
* В /info/{memeId} проверяем, что комментарий есть

Moderator: (в процессе)

* Получение сертификата
* Регистрация
* Логин
* Находим вирусную картинку, комментируем (только для модераторов)
* Проверяем комментарий
* Добавляем метаданные к мему
* Загружаем и публикуем
* Проверяем в /info/{memeId}, что метаданные есть
* Сбрасываем пароль

## fuplan:

**A. Regular user**

1. Signup
2. Login
3. Upload [malicious?] file for check (contains **flag1 - filename, see “Flag stores”**)
4. Logout

**B. Security analyst**

1. Get professional certificate / diploma
2. Signup as analyst
3. Login
4. DOWNLOAD PRIVATE FILE???? WITH FLAG???
5. View malicious file reports
6. Comment something on all safe files (contains **flag2**)
7. Comment something on all malicious files (contains **flag3)**
8. Logout

Flag stores:

1. name of uploaded for scanning file / file contents (***ASSUME*** only visible to owner)
2. comments on others user file scan (***ASSUME*** only visible to owner IF AND ONLY IF file is SAFE)
3. comments on MALICIOUS files (***ASSUME*** only visible to analysts)

Vulnerabilities:

**SON** P1 registration check bypass on password reset => **flag1, flag2~~,~~** ~~possibly~~ **~~flag3~~**

~~(DOES NOT WORK) P2 ldd binary execution =>~~ **~~flag1~~**

**VITYA** (PoC) P3 yaml injection (reference) => **flag2, flag3**

**RITA** (WIP) P3 md5 collision - on static PNG files => **flag3**

**VITYA** (WIP) P4 - CBC/CFB vs CTR/OFB encryption mode when difference is private data => **flag1**

=> requires different flag store (image bytes of memes) + see <https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation>

**VITYA** (WIP) P4 homomorphic RSA sign => **flag1,** **flag2, flag3** // simple SPA with diploma RSA signed example (homomorphic gadget)

# Почта (Memeil.pogg) - Go:

## Функционал сервиса:

1. регистрация + логин
2. отправка сообщений
3. создание папок
4. перемещение сообщений между папками (можно убрать из папки просто)
5. удаление сообщений
6. получение всех входящих
7. получение всех исходящих
8. получение всех папок (только названия)
9. получение содержимого всей папки
10. получение сообщения по его ID
11. экспорт сообщения по формату
12. просмотр логов ошибок (tcp)

флагсторы:

1. имена пользователей
2. название папок
3. содержимое писем

что делает чекер:

1. просто проверка функционала сервиса (кроме tcp)

чекер на логи:

1. проверяет, что там в ошибках при регистрации выдается почта или `Новый пошел нахуй`
2. в логах при взаимодействии с папками должны быть сообщения об ошибках, то есть после фикса эти сообщения все еще должны быть, просто имени файла не будет

## Son’s checker IS HERE NOW:

1. Registration:

* generate randomly username and password

2. Login

* login , extract and store the authentication token for further requests

3. Send mail

* send mail to registered user
* verify the successful sending of mail by checking the response

4. Retrieve mail

* get all mail from the inbox
* get all sent mails
* get mail with its ID to verify its content

5. export mail

* export mail using randomly chosen format . The script uses a list of predefined formats for exporting emails. Each time an email is exported, a random format is selected from this list to ensure variability
* the formats list includes different templates to format the exported email content in various ways.

6. Retrieve folder

* create a new folder with random name
* add the sent mail to the newly created folder
* replace the mail from one folder to another

# 

## Tfuplan:

1. Signup (contains **flag2 - username, see “Flag stores”**) (regex for password)
2. Sign in
3. Send email (contains **flag1**)
4. Create *folder* (contains **flag3**)
5. Add to *folder*
6. Get or Delete emails from *folder* (by id, all emails, read and unread)
7. Get inbox (same)
8. Add contact (**flag2**) // P4 - contact functionality
9. Change username
10. Logout

*Scenario 1:*

1. Sign[up/in]
2. Send email
3. Create folder
4. Add email to folder
5. Logout

*Scenario 2:*

1. Login
2. Read inbox
3. Logout

Scenario 3:

1. Sign in
2. Read emails in folder
3. Delete some messages
4. Check Trash folder // P4 - trash functionality

*Scenario 4:*

1. Sign in
2. Check/change username // Checker should not remove valid (not expired) - **flag2**
3. Logout

Scenario 5:

1. Sign in
2. Subscribe to Mailing List (рассылка мемов с флагами - **flag4**)
3. Logout

Flag stores:

1. email body (***ASSUME*** only visible to sender / recipient)
2. username (***ASSUME*** only visible to owner and contacts)
3. *folder* names (***ASSUME*** only visible to owner)
4. image content (***ASSUME*** only visible to senders)

Vulnerabilities:

**VIKTOR2** P1 path-traversal (folder == “../..” or id == “../..”) => **flag1**

**SON** P1 bad password => **flag1, flag2, flag3**

**RITA** (PoC) P2 ssti on email export => **flag3**

**VIKTOR2** P3 - expvar => **flag2, flag3** // logger leaks {username, folder} as JSON base64 encoded in JWT

P5 - PKI / TLS cert

P5 - pprof mem / heap dump

—--------------------------------------

P1 - priority first

P2,3 - test first then if OK implement

P4,5 - do if some time left

—----------------------------------------

# Старое (архив):

*ушел в небытие OpenAPI from Gin Framework*

**OAPI 2.0** <https://github.com/swaggo/gin-swagger>

**OAPI 3.0** <https://github.com/wI2L/fizz>

“надо разобраться че каво куда и как” (с) Илья  
“нужно чтобы суть была в том что суть” (с) Витя (Илья)

Суть:

хз тут без афобазола (пол литра) не разберешься

Глоссарий:

**больница** - КОНЦЕПЦИЯ СЕРВИСОВ короче из “микро”сервисов (разных бэкэндов (жсон апи)), с одним гига фронтом

**аптека** - сервис 1

**препарат** - то что можно добавить в БД, проверить что он “валидный”, по идее то что можно купить

**ЭЦП** - подделать подпись гомоморфно будто это новый товар с “валидной” подписью

(id товара, подпись производителя, id экземпляра товара, подпись сайта продаж)

**врачевальня** - сервис 2

**рецепт** - должен быть одноразовым, но его можно переиспользовать по умолчанию

(умным способом верифицировать рецепт от другой команды в их API)

**врач** - кто-то кто выписывает рецепт, хоть Вася Пупкин

**пациент** -

в больнице планировал именно смежные специальности + соседние кабинеты, по которым либо врачи будут получать снимки законно (хирург-ортопед), либо по соседним кабинетам или серийники аппаратов

к чекеру нужен свой богдан чекер с тупыми вопросами аля а че это у меня пися не стоит или когда я встаю утром хочу писять блин че делать

### Сервис 1 - Сайт для регистрации/учета фарм-продуктов

##### Идея:

Надо что-то **мемное** генерировать вроде (редард, пофигин и т.п.): Предлагаю конечно чекер сделать креативный если получится…

На каждый продукт регистрируется название, слогон, изображение и генерируется DataMatrix код для аутентификации.Коды верифицируются в приложении и открывается описание продукта только если код верный. Также можно добавить механику оценки товаров и лидерборда товаров. При оценке будет только описание (товары где лежат флаги будут содержать кодовое слово FLAGSHIP). Чекер сам изначально рандомно выводит в топ лайками/дизлайками товары, так что их можно будет увидеть на главной странице.

Чекер естественно сам умеет регаться/логиниться загружать пикчи/текст через форму добавки продукта, сохранять//проверять коды для старых продуктов.

##### Уязвимости:

1. Генерация подписи с ошибкой - RSA без padding (гомоморфные свойства кастомной имплементации) по id, то есть если мы знаем id гомоморфно прибавим разницу и сгенерируем новую подпись зная id в БД.
2. ~~Тайминг атака на быстрый алгоритм RSA при проверке (идея Ильи по статье) - по времени ответа (какие и какого размера параметры нужны чтобы это заметить???) выявлять биты степени в которую возводим. {еще я предлагал~~ [~~это~~](https://www.cryptologie.net/article/371/fault-attacks-on-rsas-signatures/)~~}~~
3. Логическая ошибка с отрицательным рейтингом - добавить механику оценки лекарств - лайк/дизлайк и те у которых наибольший рейтинг выводить в топ для анонимных пользователей на главной странице (делать открытыми). Прикол в том, что если создавать аккаунты (будет прилетать запрос оценки) и дизлайкать все, то новые будут первыми. Вопрос только в том, как чекер то будет проверять и тогда функционал дизлайков в каком виде можно оставить.

##### API (JSON):

/register <= {“username”:String,”password”:String}

/login <= {“username”:String,”password”:String}

/addProduct <= {“name”:String,”image”:Base64/FileUpload,”description”:String} => {“id”:Int/UUID, “signature”:Base64}

/removeProduct <= {“id”:...}

/vote <= {“id”:Int/UUID, “positive”:Boolean}

##### Концепция:



### Сервис 2 - Аналог Virustotal (по смыслу файлообменник с авторизацией)

##### Идея:

Сервис для аналитиков безопасности, можно регаться, загружать файлы. Можно смотреть отчеты на файлы, которые ты загрузил - если ты загрузил такой же файл то видно комменты других пользователей.

Чекер - регается/логинится, загружает файлы на проверку, оставляет комментарии.

##### Уязвимости:

1. Основная подстава будет в использовании md5 хэшей так что их нельзя выпилить, а эксплуатируются коллизии md5. То есть механика - найти md5, сделать коллизию загрузить от себя вирус на проверку и тогда появится доступ к комментам / флагам.
2. Файлообменник предполагает возможность загружать что угодно, надо придумать как будут работать “движки антивирусов” и реализовать в них ошибку парсинга/type confusion/XXE (чтобы фиксилось но нельзя было выпилить этот функционал).

~~!!! (УГАДАЙ КТО) я еще предлагал вариант с проверкой уникальности через crc, чтобы им коллизии искать пришлось и затирать флаги так можно было бы (ну или вдруг достать их так получится, типо при отправке файла с тем же кодом, тебе выводится уже существующий и ты такой хобана)~~

1. Все вирусы загружаются и исполняются в Docker from scratch контейнере с Golang лаунчером, так что в лежащих в одной папке “вирусных” файлах могут быть флаги, но для этого надо скомпилить статический exe и заслать его туда, а также обойти фильтры вывода персональных данных.

##### Концепция:

Выглядеть должно примерно также:

ы

ы

ы

ы

ы

ы

ы

ы

ы

ы

ы

ы

ы

### Сервис 3 - Protonmail без стероидов

##### Идея:

Сервис для отправки защищенной почты - можно регаться, отправлять почту.

Чекер - регается/логинится, отправляет почту (да, не супер полезно)

##### Уязвимости:

1. Поможет изучить особенностей перегенерации сертификатов (PKI). Уязвимостью в начальной настройке сертификатов - возможно читать траффик / расшифровывать зашифрованные сообщения на сервере. Либо возможно реализовать подмену данных при известном корневом сертификате (сгенерировать сообщение на своем сервере и отправить на другой как бы от него).
2. Закладка с генерацией id или еще чего во внешнем пакете (Python - PyPI / JS - NPM pkg@Github / Golang - pkg@Github).

##### Концепция:

Ну название Fast Breachless Invincible Mail (FBImail) должно соответствовать внешнему виду / в названии Browserless тогда TCP сервис!

### Сервис 4 - Photo Gallery Gram Reimagined TM

##### Идея:

Сервис для загрузки фоточек - можно регаться, отправлять фотки (файлом или сделать скрин с вебки и запрос местоположения), видеть ленту по местоположению или подпискам.

Чекер регается/логинится, загружает файлы (фото), подписывается/создает альбомы.

##### Уязвимости:

1. Метаданные определяют местоположение и записываются в БД для поиска ближайших фоточек. Либо подмена в файле который загружается, либо подмена местоположения в браузере.
2. Сделать флаги в фоточках перебираемыми в альбоме пользователя (они по умолчанию открытые или предсказуемые), а в фоточках флаги только OCR (или еще веселее чтобы нельзя было распарсить ахахахах).
3. Аналитика, которая собирает нажатия клавиш (условно все события так чтобы не сразу заметно было) и кидает их в массив, а это все складывается в открытую БД / дебаг интерфейс. Но надо хитрый чекер, который знает секрет, регистрируется и в БД все ок получается с самого начала. А вот потом он посылает запросы, но токены не проверяются в запросах на дебаг интерфейс…
4. сюда еще можно добавить атаку через фиксацию сессии, было бы прикольно, если сделать через чекерные сессии

##### Концепция:

На вид не как инстраграм, отвечаю.

### Что делать

Сюда писать идею, логику работы, потянете ли в одиночку написать и на каком ЯП можете, также советую накидать диаграммы с помощью <https://excalidraw.com>