конференция
РусКрипто

Ежегодная международная научно-практическая конференция

«РусКрипто'2024»

Форматно-логический контроль, как основа безопасности микросервисов

Юрьев Артемий Сергеевич, исполнительный директор, Департамент развития технологий защиты информации, АО Газпромбанк
Крибель Александр Михайлович к.т.н., директор проекта, Департамент развития технологий защиты информации, АО Газпромбанк
Ирхин Артем Александрович кафедра инфокоммуникационных технологий и систем связи (МТУСИ)
Алхимов Василий Юрьевич кафедра КБ-1 «Защита информации» (МИРЭА)

Содержание

- Угрозы ИБ, согласно OWASP;
- Задачи злоумышленника, обход сигнатур и методы обхода;
- Зависимости ClaimRate от FalsePositive;
- Суть и методы форматно-логического контроля;
- Методы защиты от атак на микросервисы в организациях;
- Особенности применения ФЛК и WAF в Fintech;
- Исследование эффективности инструментов и методов;
- Заключение и выводы.

Угрозы (по OWASP)

Open Web Application Security Project (OWASP) — это открытый проект обеспечения безопасности веб-приложений

OWASP Top 10:

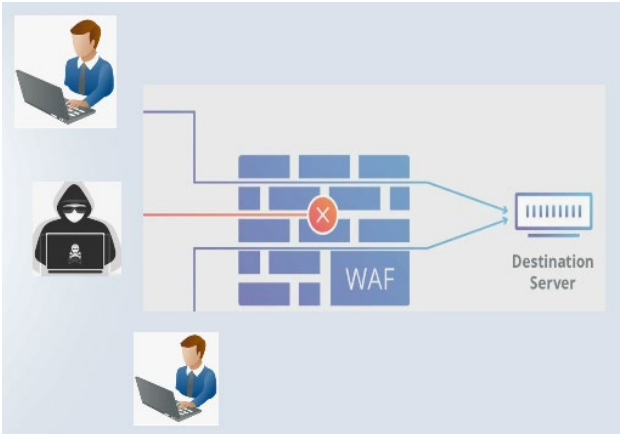
- Инъекции (Injections).
- Нарушенная аутентификация (Broken Authentication).
- Раскрытие критически важных данных (Sensitive Data Exposure).
- Внешние объекты XML (XXE) (XML External Entities (XXE)).
- Нарушенный контроль доступа (Broken Access control).
- Неправильная конфигурация безопасности (Security misconfigurations).
- Межсайтовый скриптинг (XSS) (Cross Site Scripting (XSS)).
- Небезопасная десериализация (Insecure Deserialization).
- Использование компонентов с известными уязвимостями (Using Components with known vulnerabilities).
- Недостаточно подробные журналы и слабый мониторинг (Insufficient logging and monitoring).

Определение WAF

Файрвол веб-приложений (Web application firewall, WAF) — совокупность мониторов и фильтров, предназначенных для обнаружения и блокировки сетевых атак на веб-приложение.

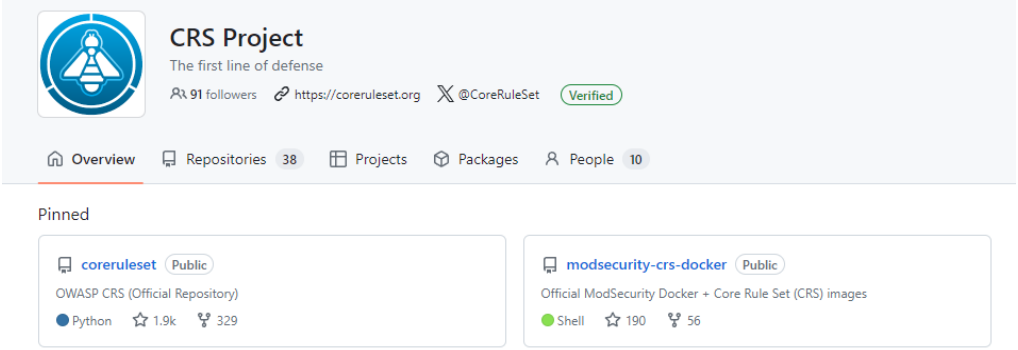
ИЛИ

WAF - это решение для фильтрации и блокировки вредоносного трафика на веб-сервисах.
*Примеры решений **WAF**:*
CloudFlare, AWS, Citrix, Akamai, Radware, Microsoft Azure, Barracuda, Sucuri, PT Application Firewall, Nemesida WAF, InfoWatch Attack Killer, ModSecurity, NAXSI, WebKnight, Coraza, Shadow Daemon, Lua-resty-waf, Vulture, IronBee, open-appsec, bunkerweb, janusec, OpenWAF



Методы WAF

– Регулярные выражениях (RegExp)
– Набор правил (например, OWASP ModSecurity Core Rule Set (CRS));
– Применение методов машинного
– обучения;



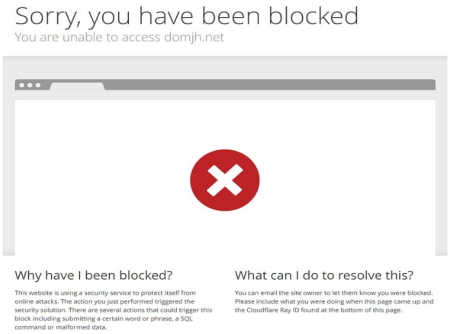
- PL 1: Baseline Security with a minimal need to tune away false positives. This is CRS for everybody running an HTTP server on the internet. If you encounter a false positive on a PL 1 system, please report it via GitHub.
- PL 2: Rules that are adequate when real customer data is involved. Perhaps an off-the-shelf online shop. Expect false positives and learn how to tune them away.
- PL 3: Online banking level security with lots of false positives. From a project perspective, false positives are accepted here, so you need to be able to help yourself by writing rule exclusions.
- PL 4: Rules that are so strong (or paranoid) they are adequate to protect the crown jewels. Use at your own risk and be prepared to get a large number of false positives.

Задачи злоумышленника

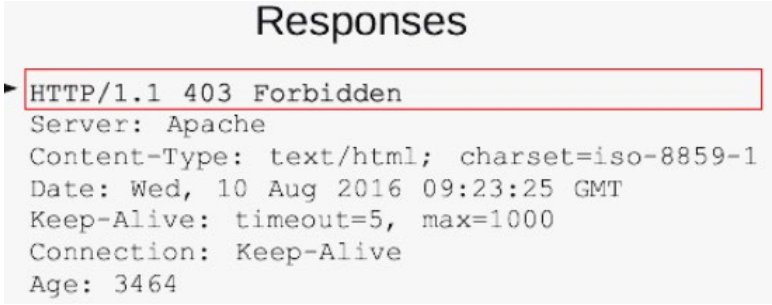
Определение типа WAF

xploit.in/?p4yl04d3=<script>alert(document.cookie)</script>.

HTTP Response может отличаться от ожидаемого для посещаемой страницы. WAF может вернуть свою собственную веб-страницу, подобную приведенной ниже, либо же другой код состояния сервера, обычно в диапазоне 400.
ЦЕЛЬ: Определение типа WAF позволяет применять необходимые методы для обхода защиты, включая известные CVE.



Как правило - 403 статус ответа

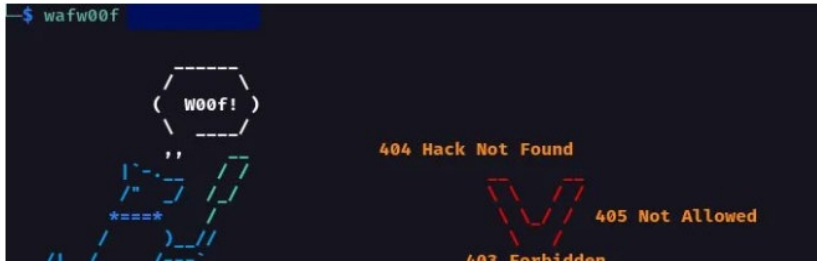


Еще признаки:

- Имя WAF может быть в заголовках ответа например: (*Server: OpenWAF)*
- Дополнительные заголовки ответов HTTP, связанные с WAF, например: (W*F-RAY: xxxxxxxxxxx*)
- Параметры Cookie, например: (*Set-Cookie: __cfduid=xxxxx)*
- Уникальный код ответа при отправке вредоносных запросов, например: (*412*)

Обход сигнатур WAF

```
$ wafw00f example.com
```



WAF Bypass Tool

WAF bypass Tool is an open source tool to analyze the securi
using predefined and customizable payloads. Check your WA
by Nemesida WAF team with the participation of community

```
##
# Target:       http://example.com
# Proxy:
# Timeout:      20s
# Threads:      50
# Block code:   403
# Exclude dirs:
# User-Agent:   Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) Applew
##

        FALSE NEGATIVE TEST
```

OWASP

GoTestWAF

Black Hat Arsenal USA 2022

GoTestWAF is a tool for API and OWASP attack simulation that supports a wide range of API protocols including REST, GraphQL, gRPC, WebSockets, SOAP, XMLRPC, and others.

It was designed to evaluate web application security solutions, such as API security proxies, Web Application Firewalls, IPS, API gateways, and others.

SQL Injection – Basic Concepts

There are two types of SQL Injection

- SQL Injection into a String/Char parameter
  Example: SELECT * from table where example = 'Example'
- SQL Injection into a Numeric parameter
  Example: SELECT * from table where id = 123

pentestit-team 17 янв 2023 в 07:44

Тестируем Web Application Firewall

Обход WAF с помощью методов машинного обучения (ML)

WafBrain	Recurrent Neural Network
ML-Based-WAF	Non-Linear SVM
ML-Based-WAF	Stochastic Gradient Descent
ML-Based-WAF	AdaBoost
Token-based	Naive Bayes
Token-based	Random Forest
Token-based	Linear SVM
Token-based	Gaussian SVM
SQLiGoT - Directed Proportional	Gaussian SVM
SQLiGoT - Directed Unproportional	Gaussian SVM
SQLiGoT - Undirected Proportional	Gaussian SVM
SQLiGoT - Undirected Unproportional	Gaussian SVM

```
1  admin' OR 1=1#
2  admin' OR 0X1=1 or 0x726!=0x726 OR 0x1Dd
     not IN/*(seleCt 0X0)>c^Bj>N]*/ ((SeLeCT
      476),(SELECT (SElEct 477)),0X1de) oR
     8308 noT lIkE  8308\x0c AnD truE OR '
     FZ6/q' LiKE 'fz6/qI' anD TRUE anD '>U'
     != '>uz'#t'%'03;Nd
```

WAF-A-MoLE: Evading Web Application Firewalls through Adversarial Machine Learning

Luca Demetrio
luca.demetrio@dibris.unige.it
Università di Genova

Andrea Valenza
andrea.valenza@dibris.unige.it
Università di Genova

Gabriele Costa
gabriele.costa@imtlucca.it
IMT School for Advanced Studies Lucca

Giovanni Lagorio
giovanni.lagorio@unige.it
Università di Genova

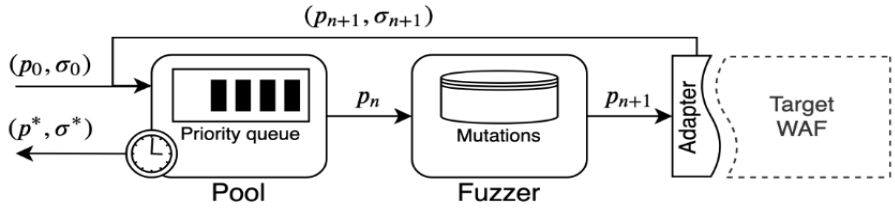
WAF-A-MoLE

A *guided mutation-based fuzzer* for ML-based Web Application Firewalls, inspired by AFL and based on the FuzzingBook by Andreas Zeller et al.

Given an input SQL injection query, it tries to produce a *semantic invariant* query that is able to bypass the target WAF. You can use this tool for assessing the robustness of your product by letting WAF-A-MoLE explore the solution space to find dangerous "blind spots" left uncovered by the target classifier.

Python 3.7 | license MIT | docs failing

Architecture



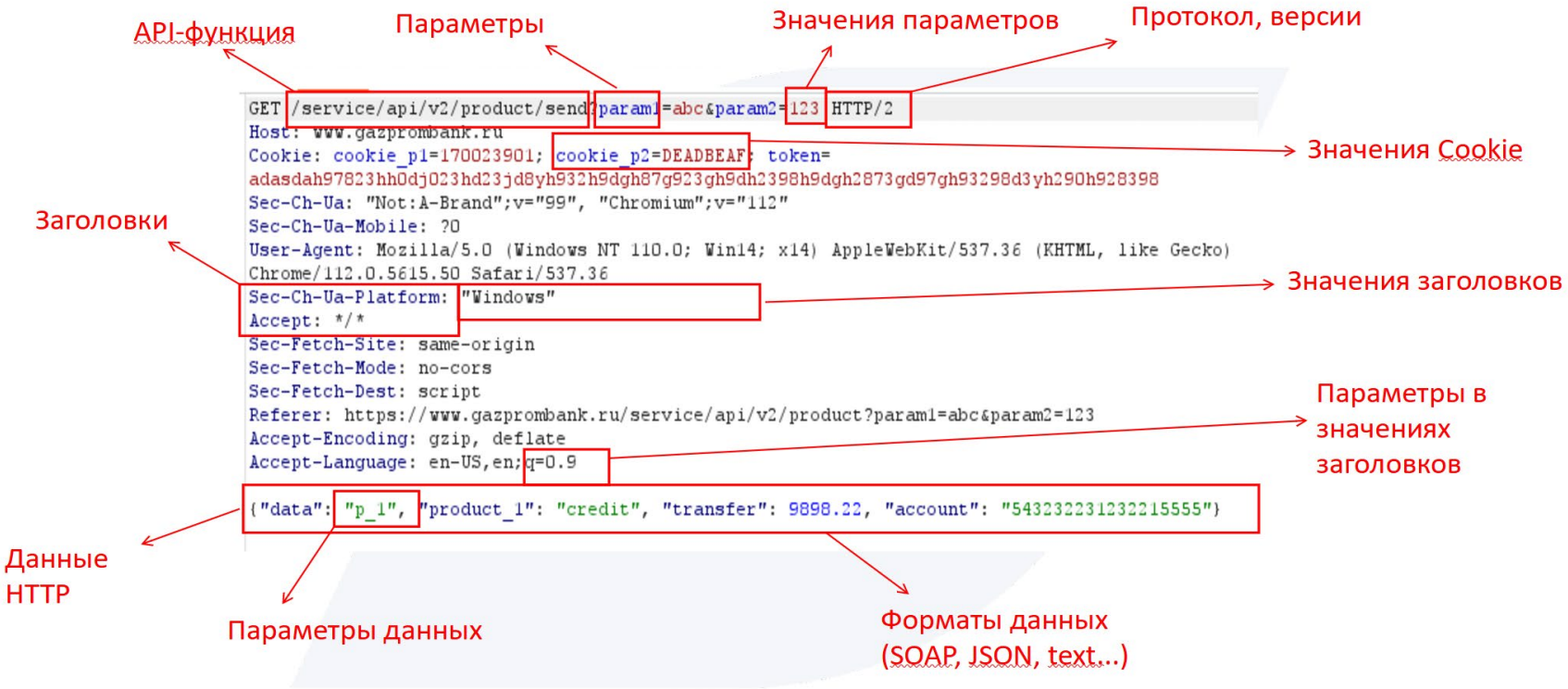
Методы обхода WAF

- Изменение регистра символов: <sCrIpT>alert(XSS)</sCriPt>
- Добавление знаков: <<script>alert(XSS)</script>, <iframe src=http://malicous.com <
- Модификация нагрузки: <script>alert(XSS)
- Изменение символов: <script>alert`XSS`</script>
- Изменение кодировок: java%0ascript:alert(1) #using encoded newline characters
- Изменение типов: <STYLE>.classname{background-image:url("javascript:alert(XSS)");}</STYLE>
- Удаление пробелов: <img/src=1/onerror=alert(0)>
- Переполнение: <a aa aaa aaaa aaaaa aaaaaa aaaaaaa aaaaaaaa aaaaaaaaa href=javascript:alert(1)>xss , ?page_id=null%0A/**//*!50000%55nIOn*//*yoyu*/all/**/%0A/*!%53eLEct*/%0A/*nnaa*/+1,2,3,4….
- Замена функций: Function("ale"+"rt(1)")(); , new Function`alt\`6\``;
- Кодирование: javascript:74163166147401571561541571411447514115414516216450615176,
- data:text/html;base64,PHN2Zy9vbmxvYWQ9YWxlcnQoMik+, %26%2397;lert(1)
- Добавление комментариев: /?id=1+un/**/ion+sel/**/ect+1,2,3--
- Переходы: <a src="%0Aj%0Aa%0Av%0Aa%0As%0Ac%0Ar%0Ai%0Ap%0At%0A%3Aconfirm(XSS)">
- Нестандартные данные: <BODY onload!#$%&()*~+-_.,:;?@[/|\]^`=confirm()>
- Обфускация: <%s%cr%u0131pt> == <script>, /path1/path2/ == ;/path1;foo/path2;bar/;

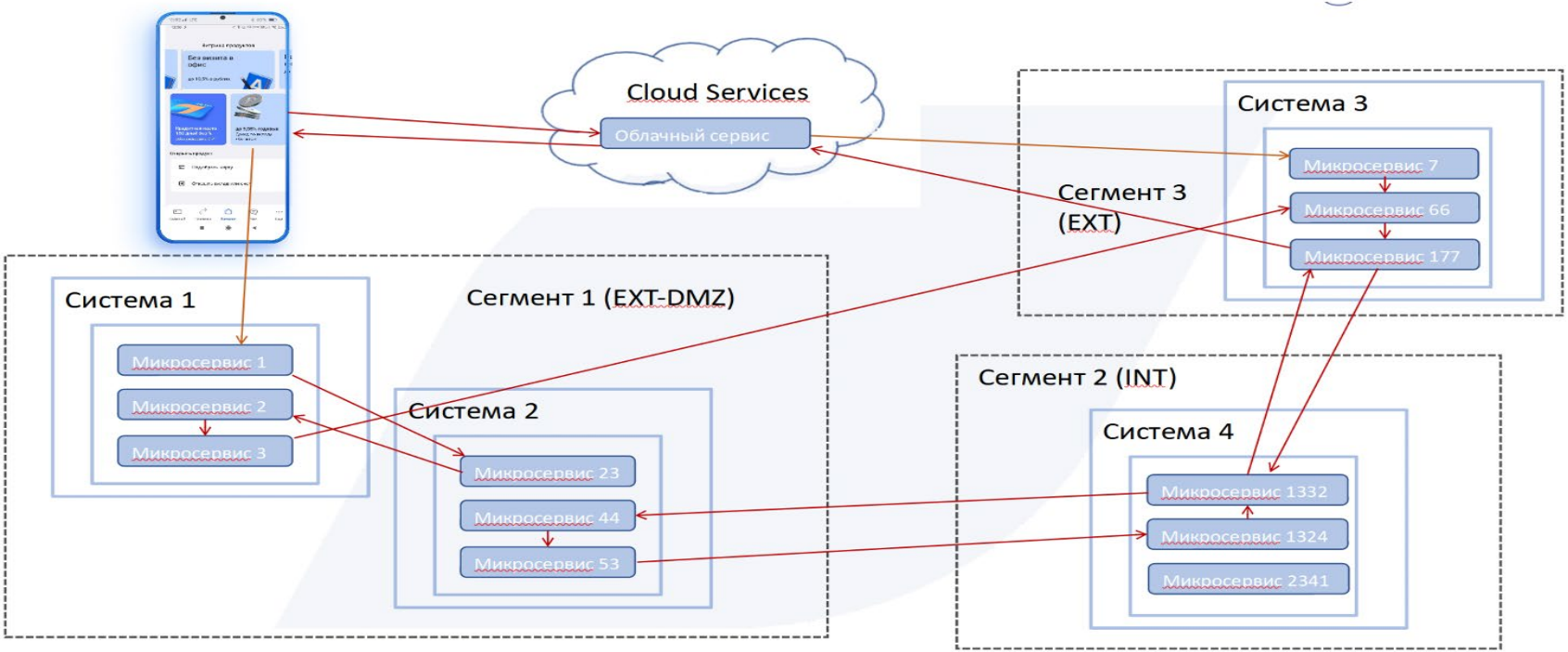
Атаки на регулярные выражения:

RegExp	Обход
(^a|a$)	%20a%20
http	hTtP
a.*b	a%0Ab
(a+)+	aaaaaaaaaaaaaaaaaaaa!
a{1,5}	aaaaaa (6 times)
[A-z] = [a-zA-Z] + [\]^_`	aaa[\]^_`aaa
a[^\n]*$	a\n? a\r?
a'\s+\d	a'5
a\s(not[whitespace]|and)\sb	a not b

Структура HTTP запроса. Варианты размещения полезной нагрузки (payloads)



Вариант организации микросервисной архитектуры



Применения WAF в зависимости от особенностей ИС

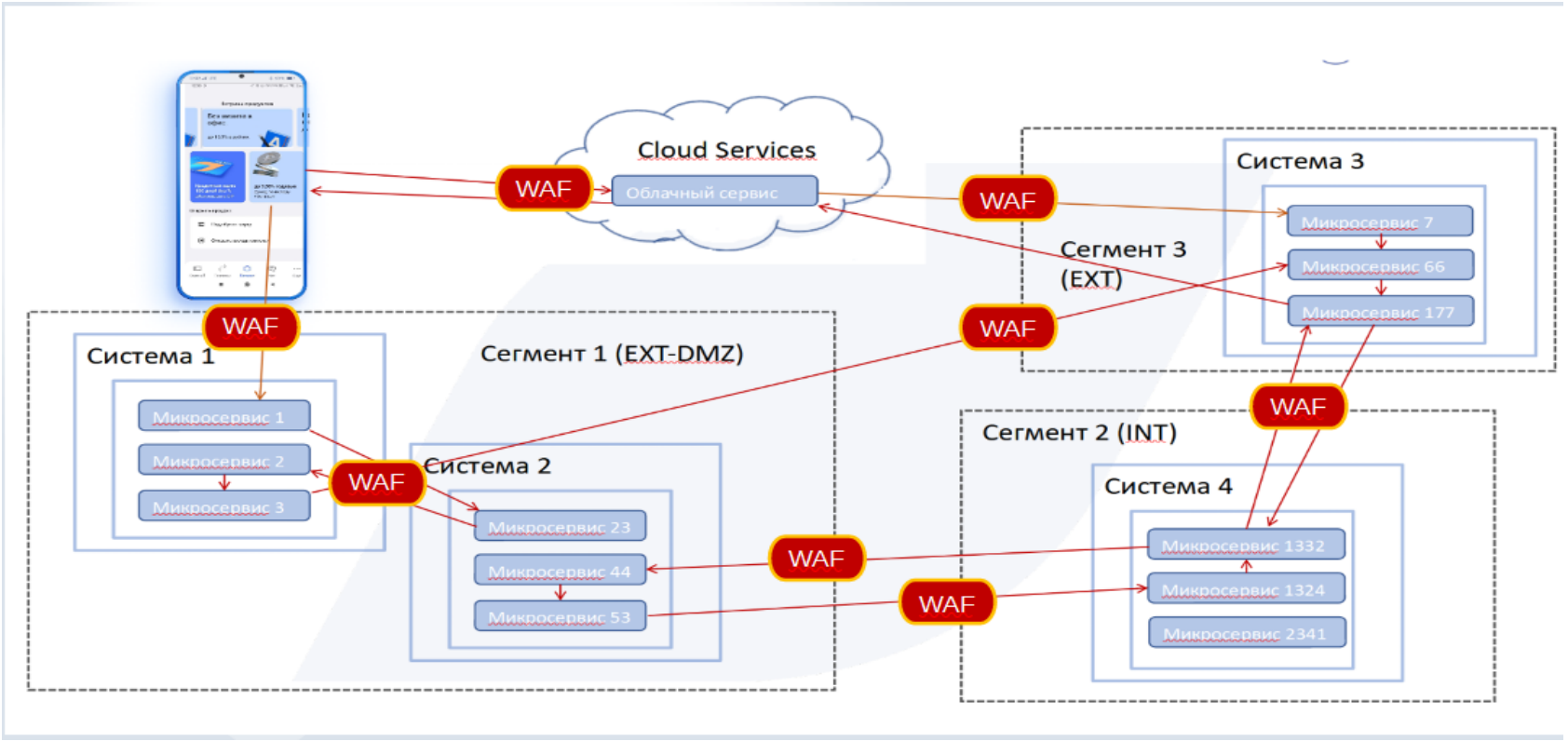


График. Зависимость ClaimRate от FalsePositive (FP)

Форматно-логический контроль (ФЛК)

Проблемы:

- Не существует общего стандартизованного определения ФЛК в рамках ИБ;
- ФЛК узко определен в разных организациях в зависимости от задач и бизнеса;

```java
public boolean isCyrillic(String s) {
    boolean result = false;
    for (char a : s.toCharArray()) {
        if (Character.UnicodeBlock.of(a) == Character.UnicodeBlock.CYRILLIC) {
            result = !result;
            break;
        }
    }
    return result;
}
```

Проверка ИНН

Вычисление контрольных цифр

Для 10-значного ИНН, присваиваемого юридическому лицу, контрольной является последняя, десятая цифра:

$$n_{10} = ((2n_1 + 4n_2 + 10n_3 + 3n_4 + 5n_5 + 9n_6 + 4n_7 + 6n_8 + 8n_9) \mod 11) \mod 10$$

Для 12-значного ИНН, присваиваемого физическому лицу, контрольными являются последние две цифры:

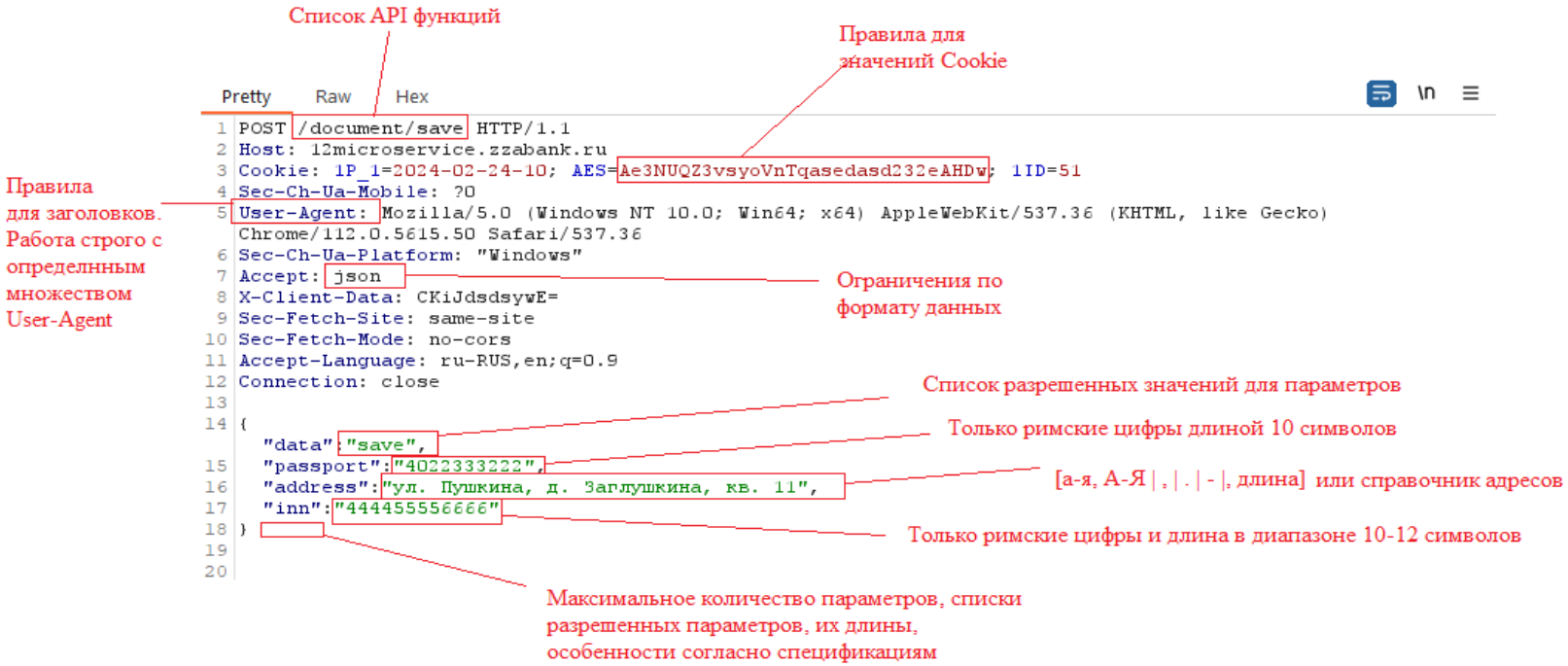
$$n_{11} = ((7n_1 + 2n_2 + 4n_3 + 10n_4 + 3n_5 + 5n_6 + 9n_7 + 4n_8 + 6n_9 + 8n_{10}) \mod 11) \mod 10$$
$$n_{12} = ((3n_1 + 7n_2 + 2n_3 + 4n_4 + 10n_5 + 3n_6 + 5n_7 + 9n_8 + 4n_9 + 6n_{10} + 8n_{11}) \mod 11) \mod 10$$

Предлагается использовать метод ФЛК для задач информационной безопасности, как отдельный класс решений ИБ

Пример требований (не ИБ) к разработке функций микросервисов

Сущность	Атрибут	Название маски поля	Описание маски поля
Юр. лицо - legalperson	ОГРН - ogrn	ОГРН организации	Цифровой код, 13 знаков. Только арабские цифры. С последним контрольным числом.
Юр. лицо - legalperson	ИНН - inn	ИНН	Цифровой код, 10 знаков. Только арабские цифры. С последним контрольным числом.
Гражданин - citizen	Имя - firstname	Имя	Слово или несколько слов на кириллице, разделенных пробелом или дефисом. Каждое слово начинается с заглавной буквы.
Гражданин - citizen	Фамилия - surname	Фамилия	Слово или несколько слов на кириллице, разделенных пробелом или дефисом. Каждое слово начинается с заглавной буквы.
Гражданин - citizen	Отчесво - patronymic	Отчество	Слово или несколько слов на кириллице, разделенных пробелом или дефисом. Каждое слово начинается с заглавной буквы.
Гражданин - citizen	пол - sex		символ, принимающий значение ‘M’ – мужской, ‘F’ - женский
Документы гражданина – citizen_doc		Серия паспорта РФ	Четырехзначный цифровой код. Только арабские цифры.
Документы гражданина – citizen_doc		Номер паспорта РФ	Шестизначный цифровой код. Только арабские цифры.
Документы гражданина – citizen_doc		СНИЛС	Цифровой код, только арабские цифры. Формат 00000000000. Две последних цифры – контрольное число.

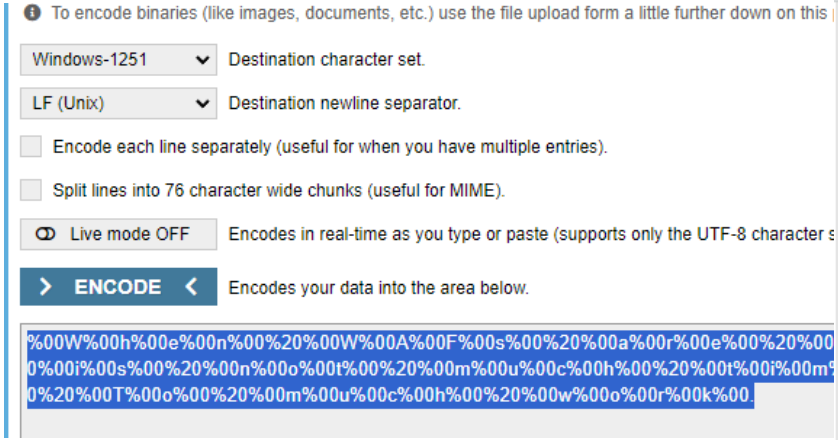
Применение ФЛК для микросервисов



Почему форматно-логический контроль (ФЛК) нужен

Проблемы. Когда WAF не справится.

- Шифрование параметров, Encrypt (AES, 3DES, Blowfish, RC5 и т.д.)
- Кодирование параметров (Base64, URLEncoding, Сериализация, Кириллица)
- Нестандартные преобразования (Кодирование, Таблицы Замен)




```
POST /apps/details HTTP/1.1
Host: 123fddfs3f3232ff32.com
Sec-Ch-Ua: "Not:A-Brand";v="99", "Chromium";v="112"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/51212.0.5625.50 Safari/537.36
Accept-Language: en-US,en;q=0.9
Connection: close
Content-Length: 105

data=
V2hlbiBXQUZzIGFyZSBleHBsb3JlZCBpdCBpcyBub3QgbXVjaCBOaW1lI
GZvciBsaWZlLiBUbyBtdWNoIHdvcmsuIA==&tag=123&russcrypto=
thebest
```

Правила ФЛК

В зависимости от задач микросервиса, правила могут содержать:

1) Проверку на типы;
2) Проверку состава сообщения на наличие обязательных полей и недокументированных в спецификации сервиса полей;
3) Проверку максимальной и минимальной длин полей запроса и значений полей сообщения;
4) Проверку содержимого полей запроса на предмет соответствия допустимым значениям и алфавитам, которые указаны в спецификации сервиса, и отсутствия недопустимых значений/символов;
5) Применение «белых списков» и «черных списков» к полям сообщений.

```
String regex = "[а-яёА-ЯЁ]+";
String str = "Работа не walk - работа work";
Pattern pattern = Pattern.compile(regex);
Matcher m = pattern.matcher(str);
if (m.find()){
   //делаем что-то
}
```

Пример для XML-сообщений (SOAP):

1) Проверка максимальной глубины вложенности элементов.
2) Проверка максимального количества XML-атрибутов.
3) Проверка максимального размера атрибутов XML-сообщения.
4) Проверка наличия внешних ссылок/IP-адресов в сообщении. (актуально для XXE атак)

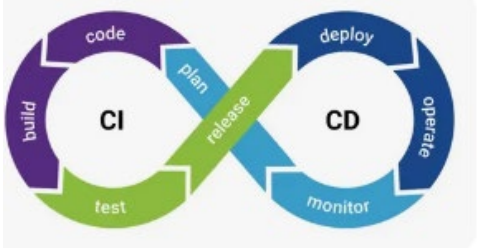
Правила ФЛК

Применение регулярных выражений для поиска дефектов ИБ

Blind SQL: (sleep\(.*\)|waitfor|delay).*(\#|\'|\"|--)
SQL LIKE % Match: ('|")[\s]*[\%]+
SQL escape sequence: '[\s]*(\;|\+|*|\=)+
MS SQL Commands: (exec)(\(|)
SQL Metacharacter:
(|union[\s](|all)|;)[\s]+(abort|analyze|begin|audit|checkpoint|close|comment|commit|copy|create|declare|delete|drop
|end|execute|fetch|move|noaudit|reindex|rename|reset|revoke|select|set|show|shutdown|start|truncate|unlisten|update)
Oracle Overflow: \b(tz_offset|to_timestamp_tz|bfilename)\b
SQL Logic: (\w*.=.*\w*[\s]*–)
XML\XXE:
(?<![<\[CDATA\[*\]{2,}])(?i)(.foo|iso|..entity|entity|xxe|datafldsystem|\Wsrc\W|script|<script|root|alert[^\)]|robots\.|methodcall|
methodname|base64|exec|upload|bash|\Wsh\W|shadow|passwd|etc|encode|xerosecurity|crowdshield|java.|\`)(?![^\[*\[]*])

Методы и решения для защиты данных (валидаторы, санитайзинг)

- **OpenAPI**
- **Решения API Security**
- **Pydantic**
- **FastAPI,**
- **Cerberus,**
- **Voluptuous,**
- **Marshmallow**




```python
from pydantic import BaseModel

class User(BaseModel):
    username: str
    age: int
    email: str
    password: str
```

OWASP

OWASP API Security Project

```python
bowie = dict(name="David Bowie")
album = dict(artist=bowie, title="Hunky Dory", release_date=date(1971, 12, 17))

schema = AlbumSchema()
result = schema.dump(album)
pprint(result, indent=2)
# { 'artist': {'name': 'David Bowie'},
#   'release_date': '1971-12-17',
#   'title': 'Hunky Dory'}
```

```
>>> v = Validator({'name': {'type': 'string'}})
>>> v.validate({'name': 'john doe'})
True
```

```
>>> from voluptuous import Schema
>>> schema = Schema({
...     'q': str,
...     'per_page': int,
...     'page': int,
... })
```

README Apache-2.0 license

build passing | npm v1.16.3 | semantic-release | chat on gitter | commitizen friendly | CLAs signed 43

OpenAPI Validator

The IBM OpenAPI Validator lets you validate OpenAPI 3.0.x and OpenAPI 3.1.x documents for compliance with the OpenAPI specifications, as well as IBM-defined best practices.

Использование сторонних решений при разработке

<'/> | libInjection

```c
#include <stdio.h>
#include <strings.h>
#include <errno.h>
#include "libinjection.h"
#include "libinjection_sqli.h"

int main(int argc, const char* argv[])
{
    struct libinjection_sqli_state state;
    int issqli;

    const char* input = argv[1];
    size_t slen = strlen(input);

    /* in real-world, you would url-decode the input, etc */

    libinjection_sqli_init(&state, input, slen, FLAG_NONE);
    issqli = libinjection_is_sqli(&state);
    if (issqli) {
        fprintf(stderr, "sqli detected with fingerprint of '%s'\n", state.fingerprint);
    }
    return issqli;
}
```

```java
public class Main {

    public static void main(String[] args) {
        /* test a string */
        Libinjection a = new Libinjection();
        boolean issqli = a.libinjection_sqli("admin' OR 1=1--");
        System.out.println(issqli);

        /* test a file and output its results to another file, with options to urldecode and time (in m
        Test t = new Test();
        t.testfile("data/sqli.txt", "data/resultsfile", true, false);
    }
}
```

Вопросы к ФЛК

- Требуется разработка применительно к каждой бизнес-логике. Разработчик должен знать суть бизнес-задачи;
- Возможна низкая производительность, но если реализовать внутри самого сервиса?
- Сможет ли ФЛК покрыть все уязвимости бизнес логики?

Перевод 100 рублей другу за обед:

{ send: "100", cur: "RUB" }

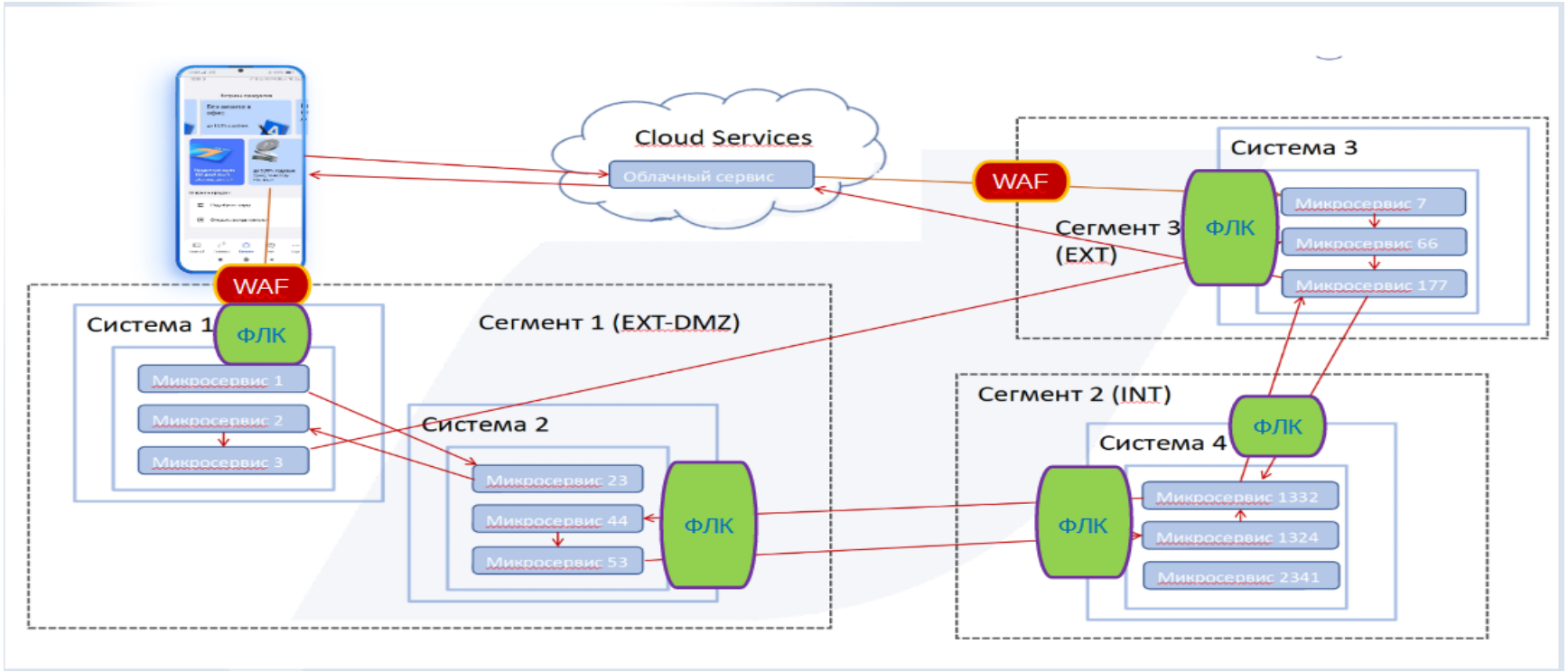
Злоумышленник:

{ send: "100", cur: "USD" }

А если файл?

```
POST /json HTTP/2
Host: ru.123.microservice
Cookie: PHPSESSID=ksuromp5fpeuqoal5i3mps449o
Content-Length: 487
Sec-Ch-Ua: "Not:A-Brand";v="99", "Chromium";v="112"
Accept: application/json
Content-Type: multipart/form-data; boundary="----WebKitFormBoundaryBhenyQZECTNs1wEu
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/112.0.5615.50 Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

------WebKitFormBoundaryBhenyQZECTNs1wEu
Content-Disposition: form-data; name="source"; filename="avatar-c0abc97c6a11bd03233cc7c17e7a2295.jpg"
Content-Type: image/jpeg

□□□□JFIF□□C□□C□□"□□
□□□□□
□□|□;F1D(□m□/L7Q□□□□□q□j□6□□□□tV>)S□=D□□□4□^Q□bq□v?*□□□\□□□J□9~□□u]%□cZ□)□R'_(N□1□□+□v□□□E□□□gM□□□1□□□□□ä□□
Q□□)Y□+□=□)□gu)v□:□□W>□Q'□Imu□□TU□{□I□□D□□□□□□□□rDp□□3□m=□□PV':□N5□g□□Z□□□□\□W□W□□□Xv/y+□□YY□□)□□□□□□/□□d□□
SLgX□□7,□□□□St□□□SCm□□□NvK3□□□U□
9□E.□□□|□□Y□n□ae%s6cD5%M□□□eq7"□□□Gn□(qX□T□□□n□□□w□z([□□mb□M□Ih
□F□z□□?(,□gQ,□□□~9□□□gi□□□□□M□QSL□9Q.q□7E□□kDp2n□u□?2(□□t□□□□□□GM□□□?b□□□T□1□□ok□h□#□□□P□)(□p□□□ś□n○○6□1
Y□%□□□□□(□□<)dU□K□□|□□□□[cOMJ□□mO□□□hD6?Ut□□K□□B□□b@□□□□$□□□□□□□7□□gD-□□□Y□)□k□□'Ut□;e5□.□i□□□□□□□U□□□□□3"
□□M□qQq□□5□s□□*□□□□:□eR8d.□□□d□]^□y9*D□`□:□□?□x□G□□fc5J5z□□'□□□6□□□I5□|-□Z□□e□□□□□EKUe□□Ge□□t'□)<□□u□L(□i7
8□<□□]=□□ .□□□rLqJDL<V□e□□b□Pt□M□'=7Q□□□□S□9□|□□7□*□□□?□N□□□□□_G8#□e□m□%□□$□□□h□□Fj;J;E5^□□□□o8&□□(□&□□
:E~□Vu3A□□□x□9eF_□□□□□□\□1m□□□□m□d□□f<□□□□8¿5□,□□c□□□(□A□[WR"□t□□>æ[*□R□ □F TQRT□%□k^m□□□é
g□X□□r□□P□□□□□□iD]H^□□□□□K1□□cg□iþ"□□□[□e□□□□□□U□□□□h□X□T□t□□□"□E□□□x□□□"□□[{dmB□□t2□m□□v]□Zu\|□□_□□□e|jC□□□
□&□-)*□□u□□□(□□J□9+□□%□□/□□□□/□□□ILQvD2□$□□□□□B□^`□□□uD).K,(□□□□^□□6□yf□icd□d□pJK"□□□`□□□□□□□8□)(9^n[□?
8)*□□x□
□yCN□Wr□%□□1□□Tk□□□8□"□Nb□□□□□□xd[P□'5¿□□□□□□□x□□□□□|yX□)*□|□□+□□□rV>\uvL□□Nf□)□4□□k□S□□□□L'V□□□^□□□IY□
K□□=□|□*Md`--□□s□)□□□
□%>J□□□8□(□2□□+□*□□l%7z□*□□□@,'xG□□R□5W□=□□t",□m□□□a□□□y2BX+$m7□□□S/PA□□□i>;B A□Dhw□□□□?□
□□F□W□□□□□□□□□S□□□□□L%□□□□□□,□□d□□A□□□IOJ□F□□h□.□□□□□□□□6□b□%Ub□□□t□V□X□□>□□P□□h□□<□N4 □□□G□□u□□□□□é
~□□4□;|vD□d□□
;□□)□PÂ□□□□□□□□q□□□□□□□D$h□□□)beRi□□□'□@.F.□u□$□□F□□□:e□□D□ar□S□□j□
$V□□'%□□M□j□□□_□□i□□Dg□hsh□□□□U@AD□aCaVB!v□b□□□□□□□>o□$□□'□□□G□@*uEb□B`*DN,□j□□□aD?$ÿ□□□S□□l□□□□'□l□□
43□RV□=□1Z5 cX□8□Q/,□U_□+^|□)iO□□□7□□Q□□□uJf□K±□□□@W□q□□□`x9[I□□TB$□'O&□□□X
)M□□gp%□□2b|uK%DJ□□=B□8□uw`□/□□8é□.□6□□□□)□H□YA□G□tf@□□□□□□fDv7q□□F/□ $$□M□□%□□□□U<□?□□
------WebKitFormBoundaryBhenyQZECTNs1wEu
Content-Disposition: form-data; name="type"

file
------WebKitFormBoundaryBhenyQZECTNs1wEu
Content-Disposition: form-data; name="action"

upload
------WebKitFormBoundaryBhenyQZECTNs1wEu
Content-Disposition: form-data; name="timestamp"

1708885350106
------WebKitFormBoundaryBhenyQZECTNs1wEu
Content-Disposition: form-data; name="auth_token"

327123bbbbaaaaff842c4bc3fee0
------WebKitFormBoundaryBhenyQZECTNs1wEu--
```

Применение WAF вместе с ФЛК



Дискуссия

1) **Применение методов ФЛК и WAF в совокупности -** будет
2) обеспечивать более полное покрытие проверками ИБ, но несет риски;
2) Не существует общих или известных методов обхода ФЛК
(без знаний о правилах ФЛК или системе);
3) Доработка WAF - долго и дорого. Не гибко.

В кредитно-финансовых или гос. организациях даже ОДНО ложно-положительное срабатывание может быть CRITICAL!

Требуется рассмотреть метод фильтрации/блокировки данных на уровне ПО

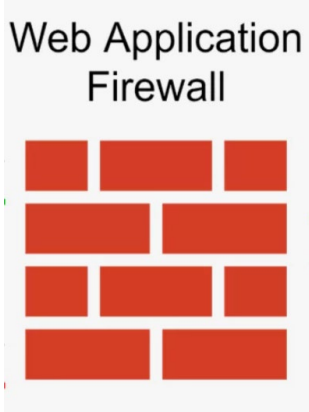
Исследование

Задача:

1) **Исследовать возможность применения белых списков в сочетании с**
2) **форматно-логическим контролем входных данных с учётом WAF;**
3) **Исследовать возможность применения белых списков в сочетании с**
4) **форматно-логическим контролем входных данных без учёта WAF;**
5) **Исследовать возможность применения белых списков и ФЛК,**
6) **разработать методы фильтрации/блокировки на уровне ПО;**

Объекты исследования

1) ModSecurity https://github.com/owasp-modsecurity/ModSecurity
2) NAXSI https://github.com/nbs-system/naxsi
3) WebKnight https://www.aqtronix.com/
4) Coraza https://github.com/corazawaf/coraza
5) Shadow Daemon https://github.com/zecure/shadowd
6) Lua-resty-waf https://github.com/p0pr0ck5/lua-resty-waf
7) Vulture (OS) https://vultureproject.github.io/
8) IronBee https://github.com/ironbee/ironbee/ (не поддерживается)
9) open-appsec WAF https://github.com/openappsec/openappsec
10) bunkerweb https://github.com/bunkerity/bunkerweb
11) janusec https://github.com/Janusec/janusec
12) OpenWAF https://github.com/titansec/OpenWAF



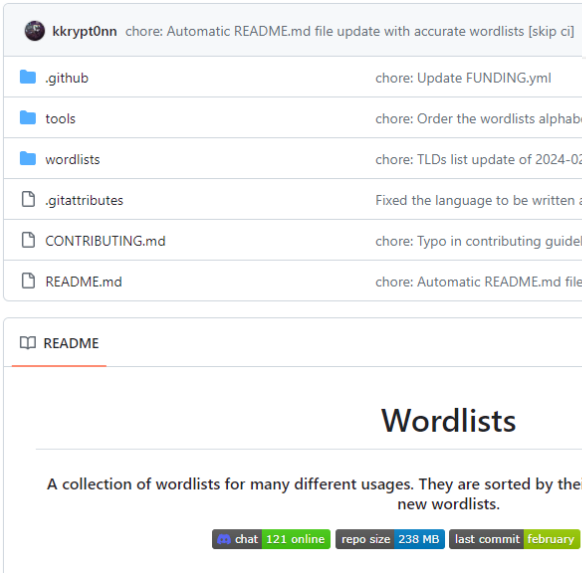
ФЛК **+** **Функция в составе ПО проверки по белому списку + правила Применительно к микросервису Validator()**

1) **Регулярные выражения**
2) **Правила ФЛК**

```
# Api Security
class FLC():

    def validator(self):
        validate_data = ["delete", "save", "put", "close"] # WHITE LIST
        data = "save"
```

Выбор тестовых данных

1) Список (wordlist) с типовыми полезными нагрузками (payloads);



Выбор тестовых данных

2) Открытое (OpenSource) ПО для тестирования WAF

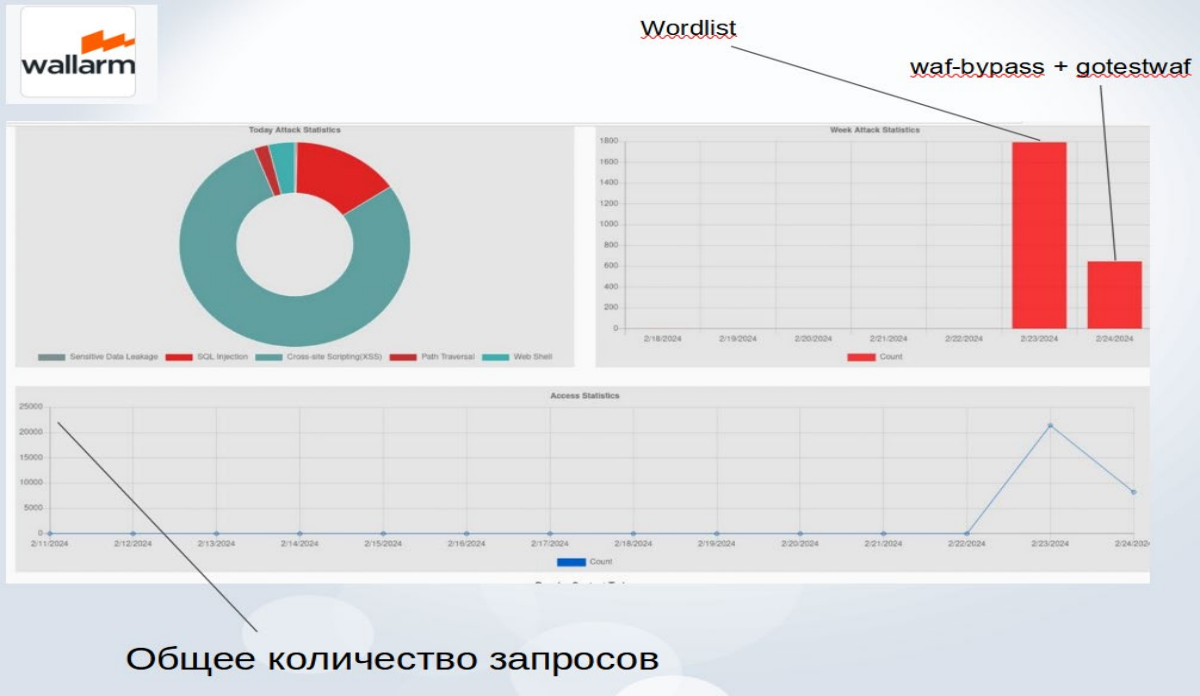
- https://github.com/nemesida-waf/waf-bypass
- https://github.com/wallarm/gotestwaf

3) WebServer (микросервис)

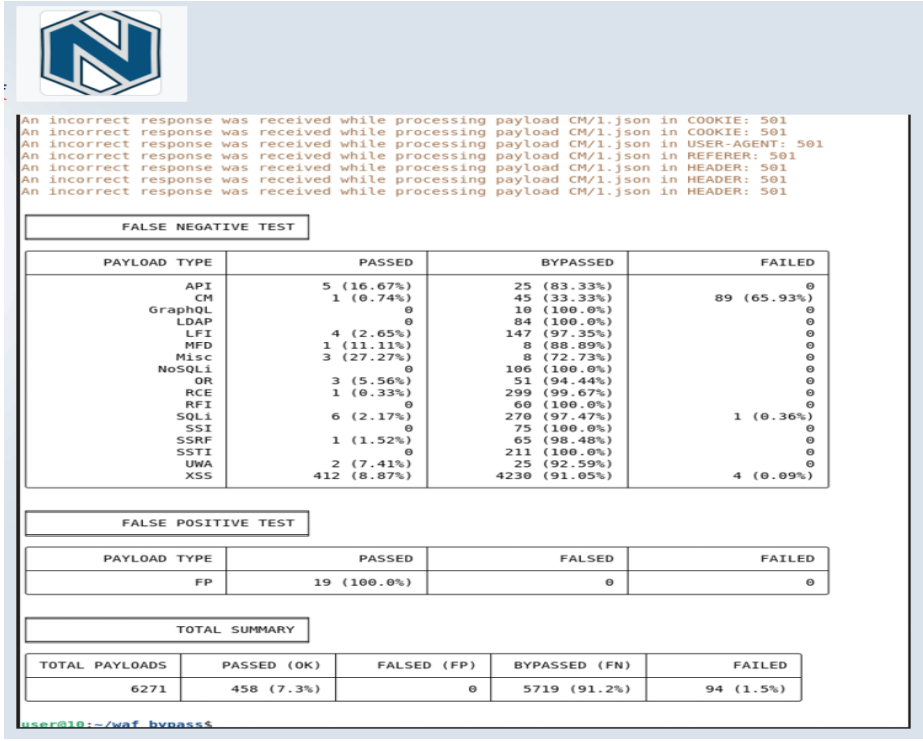
```
from http.server import BaseHTTPRequestHandler, HTTPServer
import logging
from sys import argv

class S(BaseHTTPRequestHandler):

    def _set_response(self):
        self.send_response(200)
        #self.protocol_version = "321"
        #self.sys_version = "4444"
        self.send_header('Content-type', 'text/html')
        #self.send_header('Server', '222')
        self.end_headers()
```

```
def run(server_class=HTTPServer, handler_class=S, port=8082):

    logging.basicConfig(filename="log.txt",level=logging.INFO)
    server_address = ('', port)
    httpd = server_class(server_address, handler_class)
    logging.info('Starting httpd...\n')
```

Примеры работы с janusec при тестировании открытыми решениями



Примеры работы с janusec при тестировании открытыми решениями



```
An incorrect response was received while processing payload CM/1.json in COOKIE: 501
An incorrect response was received while processing payload CM/1.json in COOKIE: 501
An incorrect response was received while processing payload CM/1.json in USER-AGENT: 501
An incorrect response was received while processing payload CM/1.json in REFERER: 501
An incorrect response was received while processing payload CM/1.json in HEADER: 501
An incorrect response was received while processing payload CM/1.json in HEADER: 501
An incorrect response was received while processing payload CM/1.json in HEADER: 501
```

FALSE NEGATIVE TEST

PAYLOAD TYPE	PASSED	BYPASSED	FAILED
API	5 (16.67%)	25 (83.33%)	0
CM	1 (0.74%)	45 (33.33%)	89 (65.93%)
GraphQL	0	10 (100.0%)	0
LDAP	0	84 (100.0%)	0
LFI	4 (2.65%)	147 (97.35%)	0
MFD	1 (11.11%)	8 (88.89%)	0
Misc	3 (27.27%)	8 (72.73%)	0
NoSQLi	0	106 (100.0%)	0
OR	3 (5.56%)	51 (94.44%)	0
RCE	1 (0.33%)	299 (99.67%)	0
RFI	0	60 (100.0%)	0
SQLi	6 (2.17%)	270 (97.47%)	1 (0.36%)
SSI	0	75 (100.0%)	0
SSRF	1 (1.52%)	65 (98.48%)	0
SSTI	0	211 (100.0%)	0
UWA	2 (7.41%)	25 (92.59%)	0
XSS	412 (8.87%)	4230 (91.05%)	4 (0.09%)

FALSE POSITIVE TEST

PAYLOAD TYPE	PASSED	FALSED	FAILED
FP	19 (100.0%)	0	0

TOTAL SUMMARY

TOTAL PAYLOADS	PASSED (OK)	FALSED (FP)	BYPASSED (FN)	FAILED
6271	458 (7.3%)	0	5719 (91.2%)	94 (1.5%)

```
user@10:~/waf_bypass$
```

Тестирование WAF. Результаты.

Nemesida				
	Total payloads	Passed (OK) / Блок	Bypassed (FN) / Обход	Failed / Ошибка
Janusec	6271	458 (7.3%)	5719 (91.2%)	94 (1.5%)
OpenAppSec	6271	4718 (75.24%)	1547 (24.67%)	6 (0.1%)
NAXSI	6271	3253 (51.87%)	2941 (41.9%)	69 (1.1%)
NAXSI_lib	6271	3691 (58.7%)	2503 (39.91%)	69 (1.1%)
ModSecCRS	6271	4419 (70.47%)	1831 (29.2%)	12 (0.19%)
Coraza	6271	4410 (70.32%)	1842 (29.37%)	10 (0.16%)
WebKnight	6271	6233 (99.39%)	0	19 (0.3%)
Shadow Daemon	6271	317 (5.07 %)	5844 (93.2%)	110 (1.75%)

Wallarm				
	API Security: true positive blocked, payloads #1, блок	Application Security: true positive blocked, payloads #2, блок	Application Security: true negative passed, прошло	Application Security: Grade, среднее
Janusec	8.30%	11.30%	100.00%	55.60%
OpenAppSec	73.68%	68.72%	90.28%	79.50%
NAXSI	88.90%	67.80%	65.30%	66.60%
NAXSI_lib	100.00%	100.00%	0.00%	50.00%
ModSecCRS	60.50%	54.70%	93.50%	74.10%
Coraza	57.90%	54.40%	86.10%	70.20%
WebKnight	100.00%	98.70%	0.00%	49.40%
Shadow Daemon	98.65%	87.70%	74.90%	64.70%

Функция проверки данных в составе микросервиса

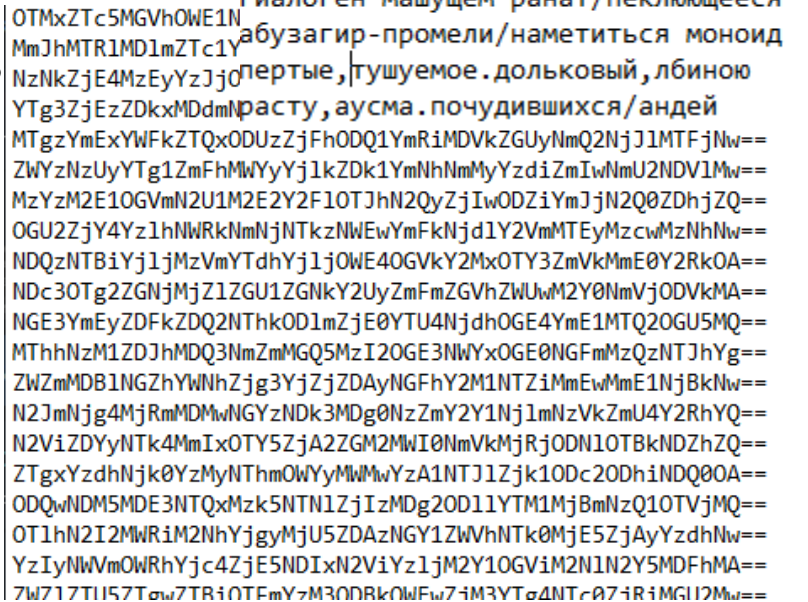
```python
# Api Security
class FLC():

    def validator(self):
        validate_data = ["delete","save", "put", "close"]# W
        data = "save"
        passport = "433411212312"
        address = 'ул. Пушкина, д. Заглушкина, кв. 11'
        inn = "444455556666"
        if (len(data) and len(passport) and len(address) and len(inn)) != 0:
            print("Len OK")
        else:
            return
        if (len(data) < 6) and (len(passport) == 12) and (len(address) < 50) and (len(inn) == 10 or len(inn) == 12):
            print("Overflow OK")
        else:
            return
        if data in validate_data:
            print(f'{data} is present in the list')
            data_match = "OK"
        else:
            return

        passport_match = re.match(r"^([0-9]{12})?$", passport)  # RULE или справочник
        address_match = re.match(r"/(?:^|[^а-яА-яёЁ])д\.\s*(\d+(?:[а-яА-яёЁ]|\/\d+)?)$/;", address)  # RULE или справочник
        inn_match = re.match(r"^(([0-9]{12})|([0-9]{10}))?$", inn)      # RULE или справочник
```

OpenAPI Validator

The IBM OpenAPI Validator lets you validate OpenAPI 3.0.x and OpenAPI 3.1.x documents for compliance with the OpenAPI specifications, as well as IBM-defined best practices.

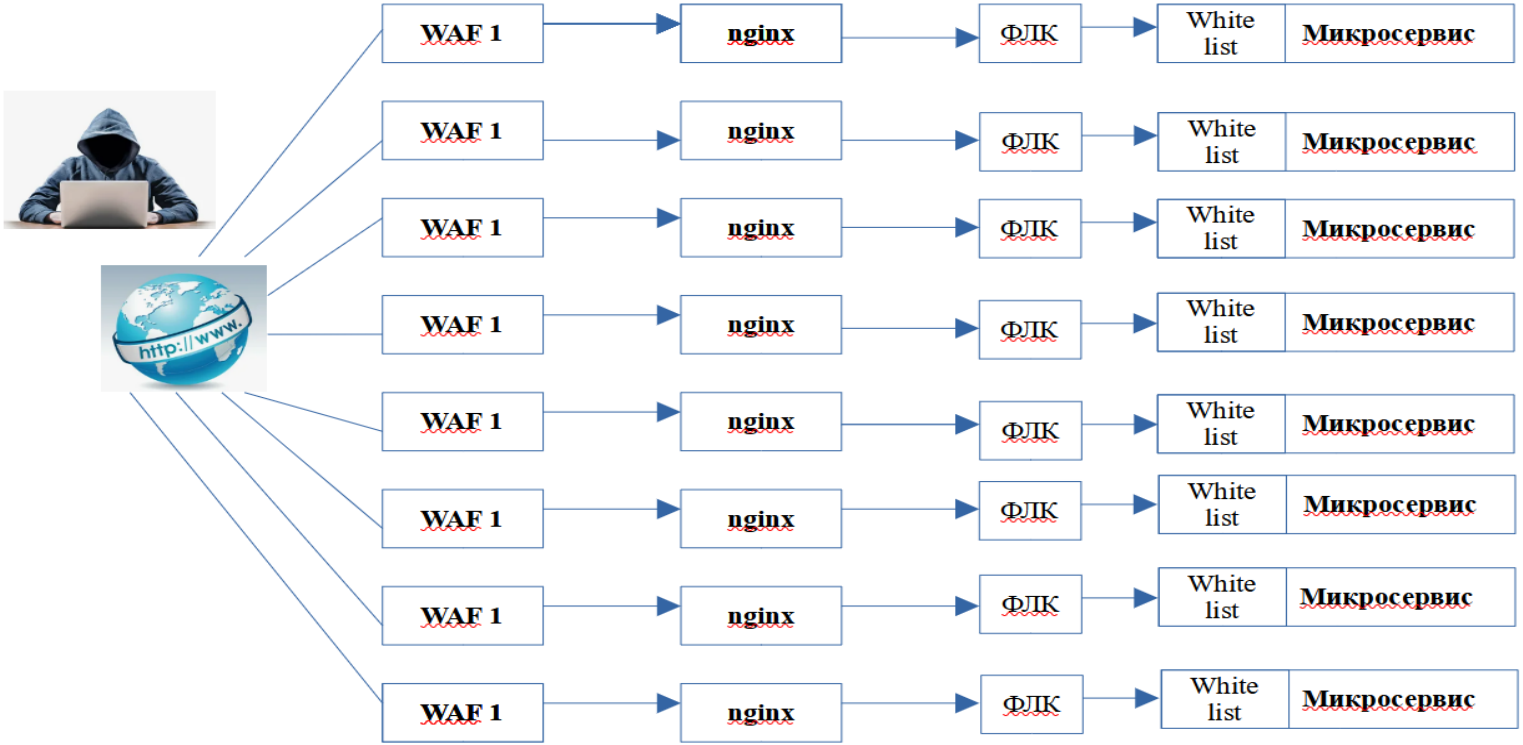
Для оценки срабатываний FP

- Сгенерировано множество base64 encode значений
- Сгенерировано множество urlenc значений
- Сгенерировано множество слов из символов Кириллицы и знаков

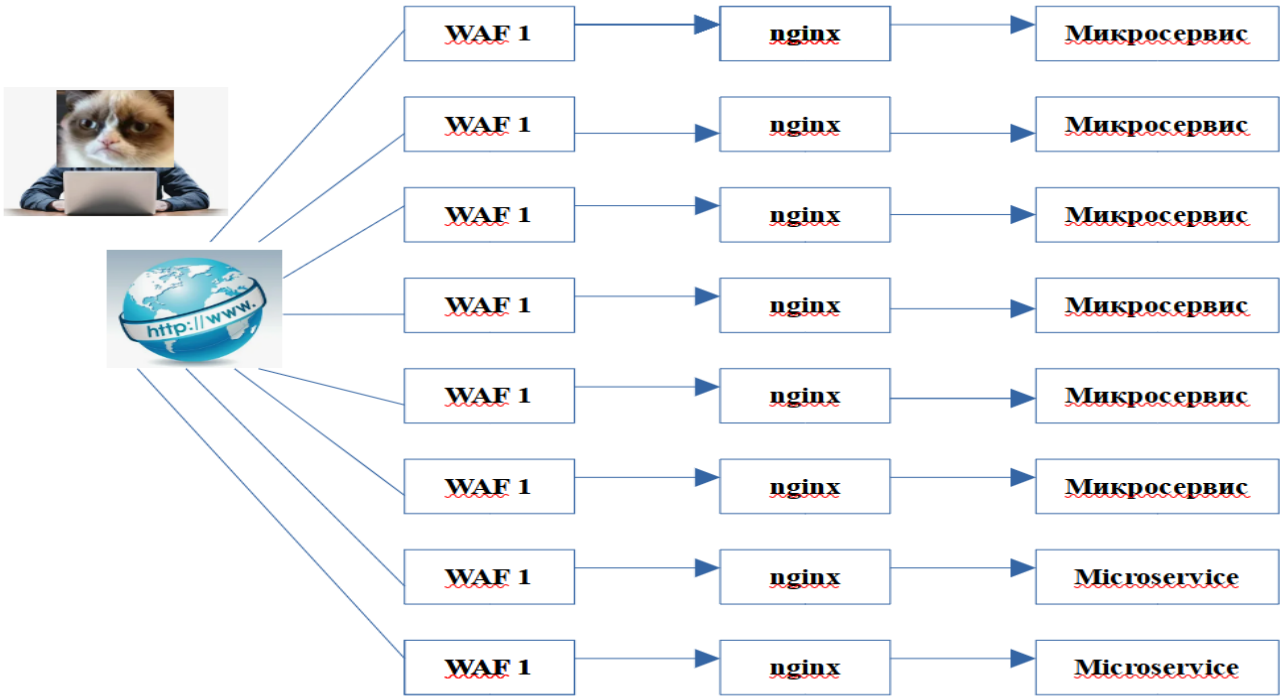


солидничать;гришин/оплыв;утряситесь
утопчетесь/кошеву-утолченное мздою
гиалоген машущем ранат/пеклющееся
абузагир-промели/наметиться моноид
пертые,тушуемое.дольковый,лбиною
расту,аусма.почудившихся/андей

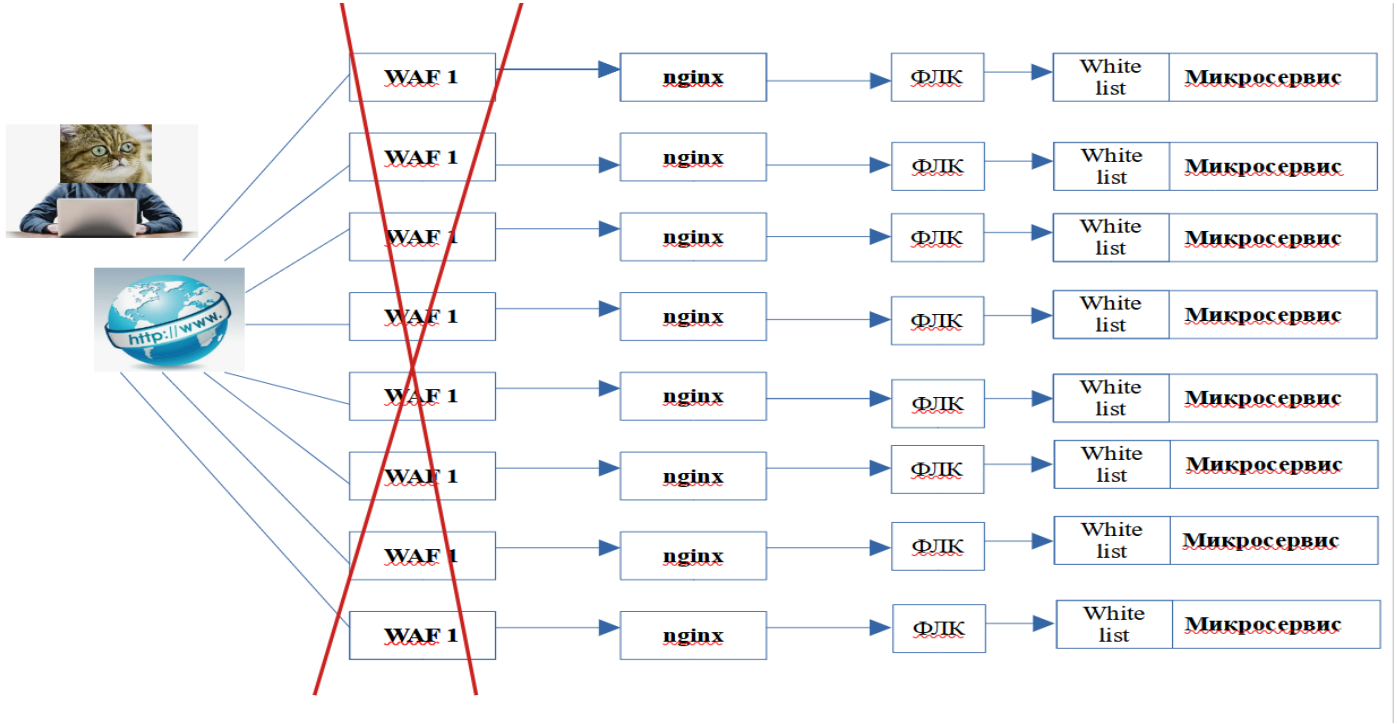
Стенд / объект исследования



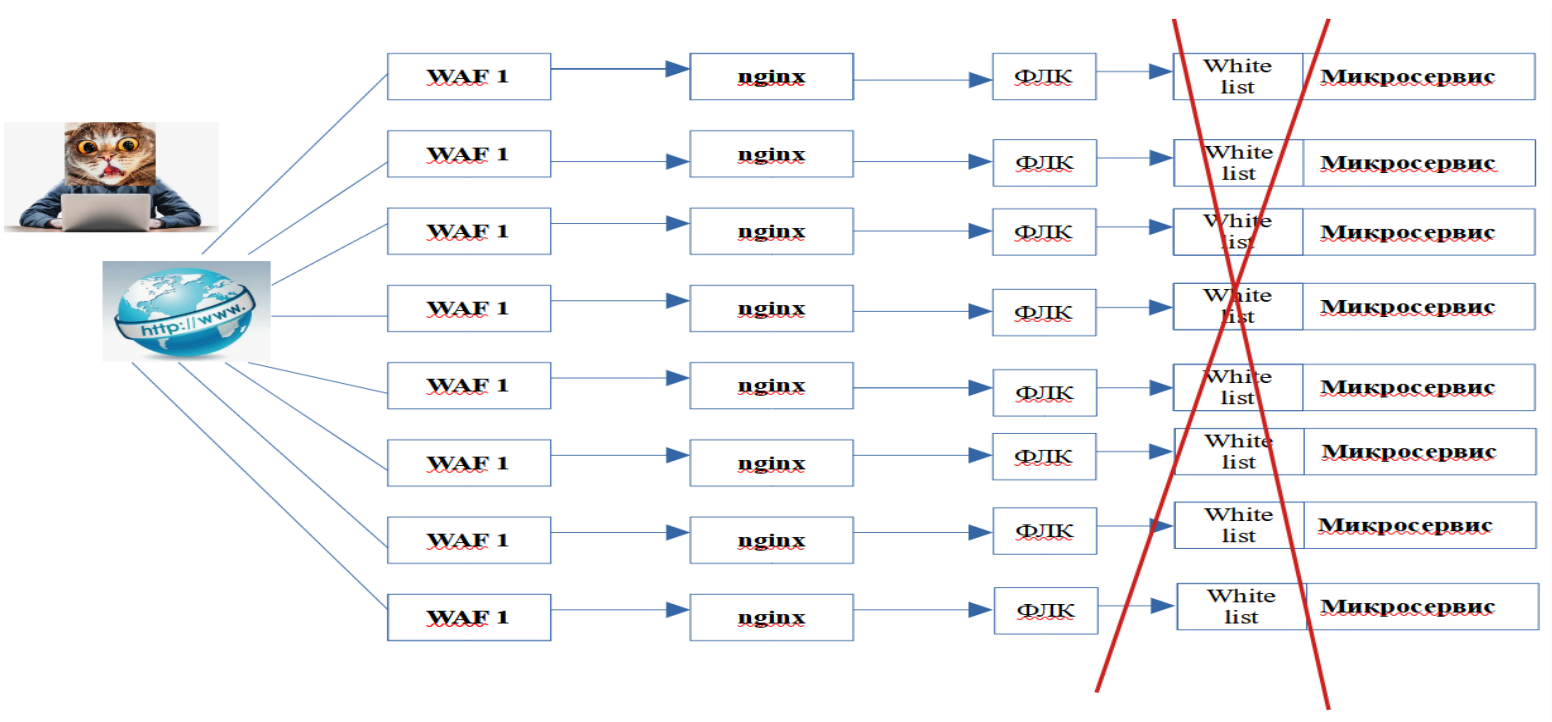
Стенд # 2. Только WAF



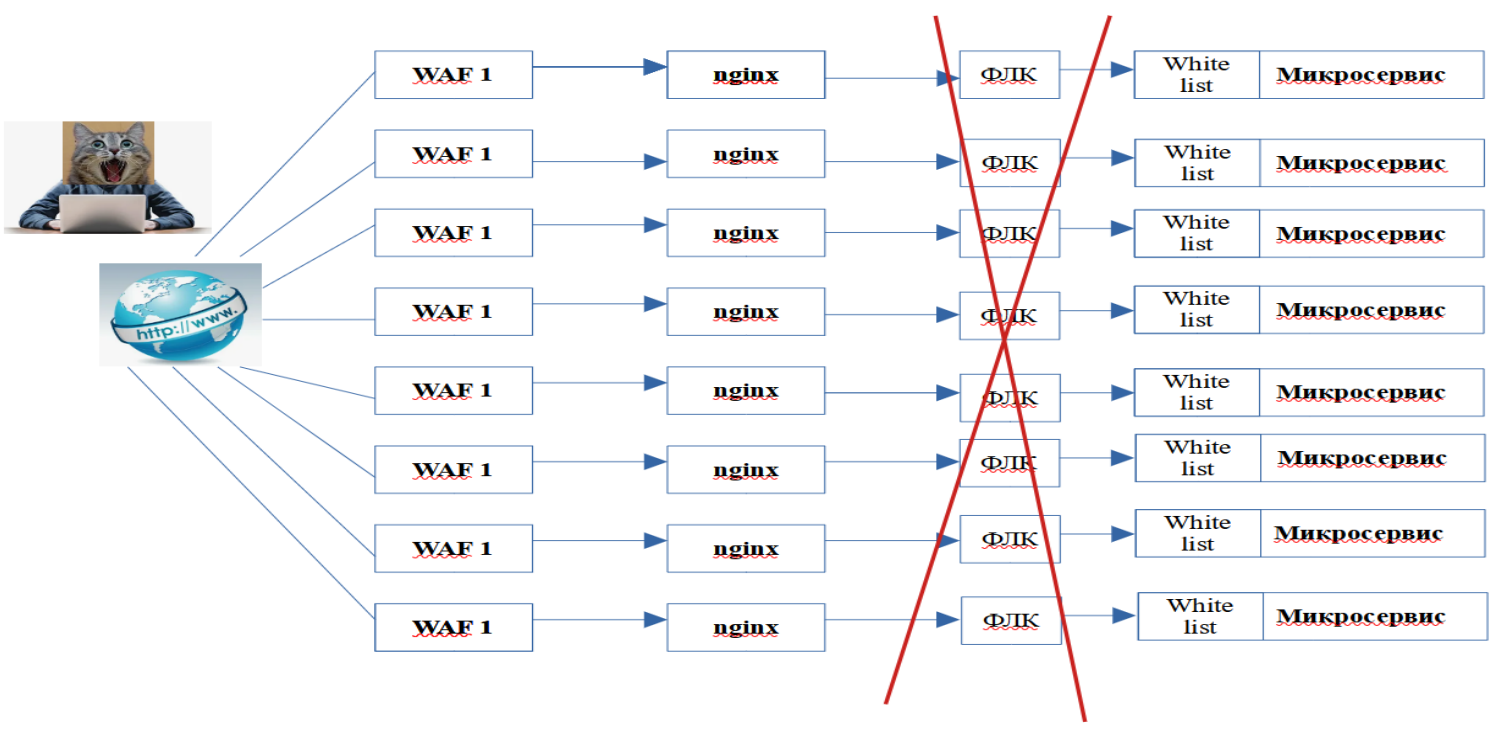
Стенд #3 ФЛК + White list



Стенд #4 WAF + ФЛК



Стенд #5 WAF + White list



Результаты # 1

	Случайные данные	Случайные данные	Wordlist	False-Positive
	Base64	**URL Encoding**	**Russians+symbols**	
	blocked	blocked	blocked	
FLK	5000	5138	0	Not
WL	5000	7724	8000	Not
ModSecCRS	0	3373	207	yes
ModSecCRS+FLC	5000	5828	207	yes
Naxsi	0	3684	3900	yes
Naxsi_lib	0	3696	3900	yes
Naxsi_lib_FLC	5000	6198	3900	yes
Naxsi_FLC	5000	6187	3900	yes
Coraza	0	3381	207	yes
Coraza_FLC	5000	5443	207	yes
WebKnight	0	5227	0	Not
WebKnight_FLC	5000	5647	0	Not
OpenAppsec	0	4621	0	Not
OpenAppsec + FLC	5000	1809	0	Not
Janusec	0	7413	0	Not
Janusec + FLC	5000	2433	0	Not
Всего запросов	**5000**	**7724**	**8000**	

Результаты # 2

	Total evil payloads #1	Passed	Block
FLK	7726	931	6795
WL	7726	0	7726
ModSecCRS	7726	4339	3387
ModSecCRS+FLC	7726	2930	4796
Naxsi	7726	4039	3687
Naxsi_lib	7726	4025	3701
Naxsi_lib_FLC	7726	2963	4763
Naxsi_FLC	7726	2974	4752
Coraza	7726	4352	3374
Coraza_FLC	7726	3090	4636
WebKnight	7726	2227	5499
WebKnight_FLC	7726	1515	6211
OpenAppsec	7726	4631	3095
OpenAppsec + FLC	7726	2865	4861
Janusec	7726	7415	311
Janusec + FLC	7726	4215	3511

Заключение

- При выборе решений и правил ФЛК следует учитывать бизнес-потребности, как основной критерий;
- Ложно-положительные срабатывания явно влияют на прибыль организации и стабильность;
- Следует проводить аналитику совместно с командами разработки и заказчиками;
- Требуется обязательное тестирование после каждого изменения политик;
- Необходимо предусмотреть ресурс и вакансии для работы с командами,
прямо участвовать в SDL продуктов;
- Учитывать информационную архитектуру при встраивании решений;
- Осуществлять постоянный мониторинг срабатываний WAF/ФЛК;
- WL в составе ФЛК - не подходит для User Input;
- У злоумышленника нет методов обойти ФЛК с WL без знаний о системе;
- ФЛК + WAF - обеспечивает наиболее полное покрытие, но несет риски;
- Отсутствие WAF (но, нежелательно) возможно при соблюдений правил безопасной разработки с использованием ФЛК;
- При соблюдении принципов безопасной разработки (SSDL) становится возможным сокращать количество СЗИ;
- Соблюдение принципа «золотая середина» обеспечивает продуктивную работу сотрудников ИБ и разработчиков.

Дальнейшие направления работ/исследований

- Создание оптимального/гибкого решения для форматно-логического контроля/фильтра, как библиотека/SDK (OpenSource);
- Создание автоматического парсера для правил на основании Swagger/XSD/OpenAPI разных схем/спецификаций;
- Создание универсальных тестовых данных для проведения анализа эффективности работы WAF и тестирования решений в совокупности с современными бизнес-потребностями, с учётом RU сегмента;
- Анализ современных AI решений для WAF;
- Публикации исследований;

Спасибо за внимание

Контактная информация

- **Telegramm:**

  @ ArtemiyUeax

Группа tg: