Доклад: Ошибки проверки вводимых данных: инъекция E-mail

Воробьев Данил Павлович НБИбд-02-23

Содержание

Список иллюстраций

Список таблиц

0.1 Введение

0.2 Актуальность проблемы

В современном цифровом мире электронная почта остается ключевым идентификатором пользователя. Согласно исследованию Verizon (2023), 80% кибератак начинаются с компрометации электронной почты. При этом около 40% уязвимостей связаны с недостаточной валидацией вводимых данных. Определение понятия

Инъекция E-mail - это класс уязвимостей, возникающих при некорректной обработке адресов электронной почты, позволяющий злоумышленнику:

```
Обходить бизнес-логику приложения

Получать несанкционированный доступ

Выполнять произвольные команды
```

0.3 Основные виды атак

1. SQL-инъекция через E-mail

Механизм атаки: sql
SELECT * FROM users WHERE email = ‘user@example.com’ OR 1=1 –’
Реальные последствия:

```
Получение полного доступа к БД
```

```
Обход аутентификации
```

```
Массовая утечка данных
```

Кейс: В 2021 году через подобную уязвимость в CMS WordPress было скомпрометировано более 50,000 сайтов (Wordfence, 2021). 2. XSS через E-mail

Пример инъекции: html

user@example.com”>

Последствия:

```
Кража сессий пользователей
```

```
Перенаправление на фишинговые страницы
```

```
Подмена контента
```

3. Подделка заголовков (CRLF Injection)

Пример:

test@example.com%0D%0ACc:%20victim@bank.com%0D%0ABcc:%20attacker@evil.com

Последствия:

```
Несанкционированная рассылка
```

```
Фишинг от имени доверенного домена
```

```
Компрометация SMTP-сервера
```

0.4 Методы защиты

1. Валидация по стандартам

Рекомендации:

```
RFC 5322 (основной стандарт)
```

```
RFC 6531 (поддержка Unicode)
```

Пример кода на Python: python

import re from email.utils import parseaddr

def validate_email(email): if not re.match(r'[1]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,}$', email): return False return'@' in parseaddr(email)[1]

2. Защита от SQL-инъекций

Лучшие практики:

```
Подготовленные выражения (Prepared Statements)
```

```
ORM с параметризованными запросами
```

```
Строгая типизация
```

Пример на PHP/PDO: php

$stmt = $db->prepare("SELECT * FROM users WHERE email = :email");stmt->execute(['email' => $email]);

3. Защита от XSS

Многоуровневая защита:

[1] a-zA-Z0-9._%+-

```
HTML-экранирование

CSP (Content Security Policy)

HttpOnly флаги для кук
```

4. Защита SMTP-серверов

Меры:

```
Фильтрация CRLF (%0D%0A)

Санкционирование исходящих писем

DKIM/SPF/DMARC аутентификация
```

0.5 Реальные кейсы

Кейс 1: Уязвимость в Roundcube (2022)

Проблема: Инъекция через поле “From” Последствия: Рассылка спама с серверов жертв Решение: Патч CVE-2022-30307 Кейс 2: Массовая утечка данных через форму восстановления (2023)

Проблема: SQL-инъекция в валидации email Масштаб: 250,000 пользователей Решение: Переход на параметризованные запросы

0.6 Заключение

Ключевые выводы

```
Инъекция email остается серьезной угрозой
```

```
Требуется комплексный подход к защите


Регулярный аудит кода обязателен
```

0.7 Список литературы

https://owasp.org/www-community/attacks/Email_Injection https://tools.ietf.org/html/rfc5322 https://www.verizon.com/business/resources/reports/dbir/ https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-30307}