



POSITIVE DEVELOPMENT USER GROUP

# Server Side Request Forgery

Аудитор  
Digital Security  
Денис Рыбин

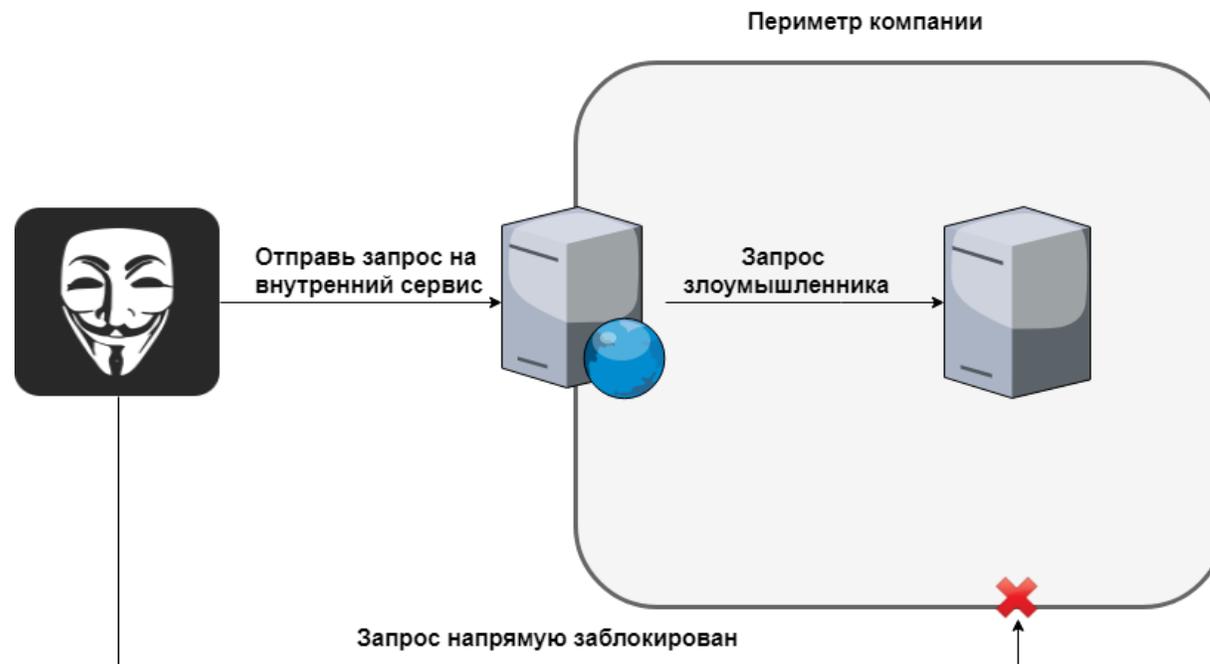
# Обо мне



- Денис Рыбин
- Телеграмм @ttffdd
- Аудитор в Digital Security
- Не разработчик

# Что же такое Server Side Request Forgery?

Атака SSRF возможно в случае наличия уязвимости ПО, позволяющей злоумышленнику спровоцировать сервер отправить запрос на произвольный адрес.



# Пример: Корпоративный чат

Техническое задание:

- Собственный чат
- Карточки для ссылок «как в телеграмме»
- Карточки должны формироваться и храниться на бекенде

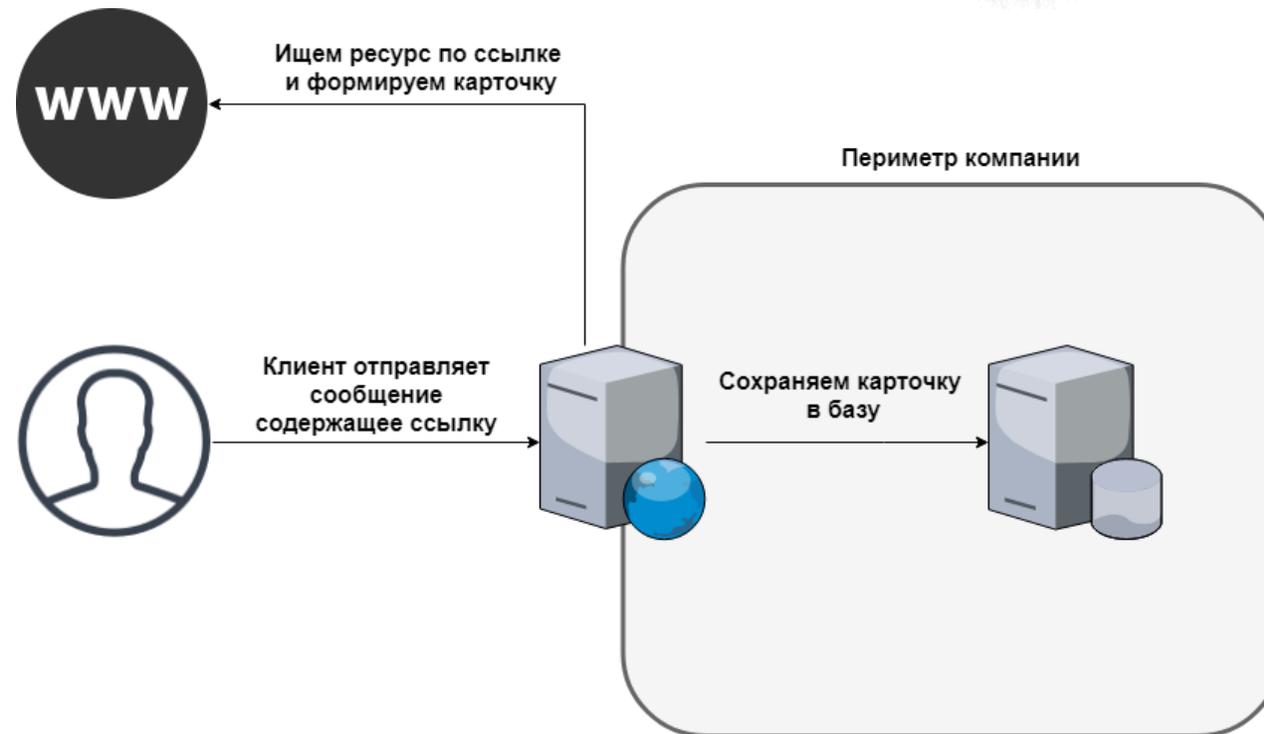
Catoftheday

Cat of the Day - Every day a new cat photo and story since 1998.

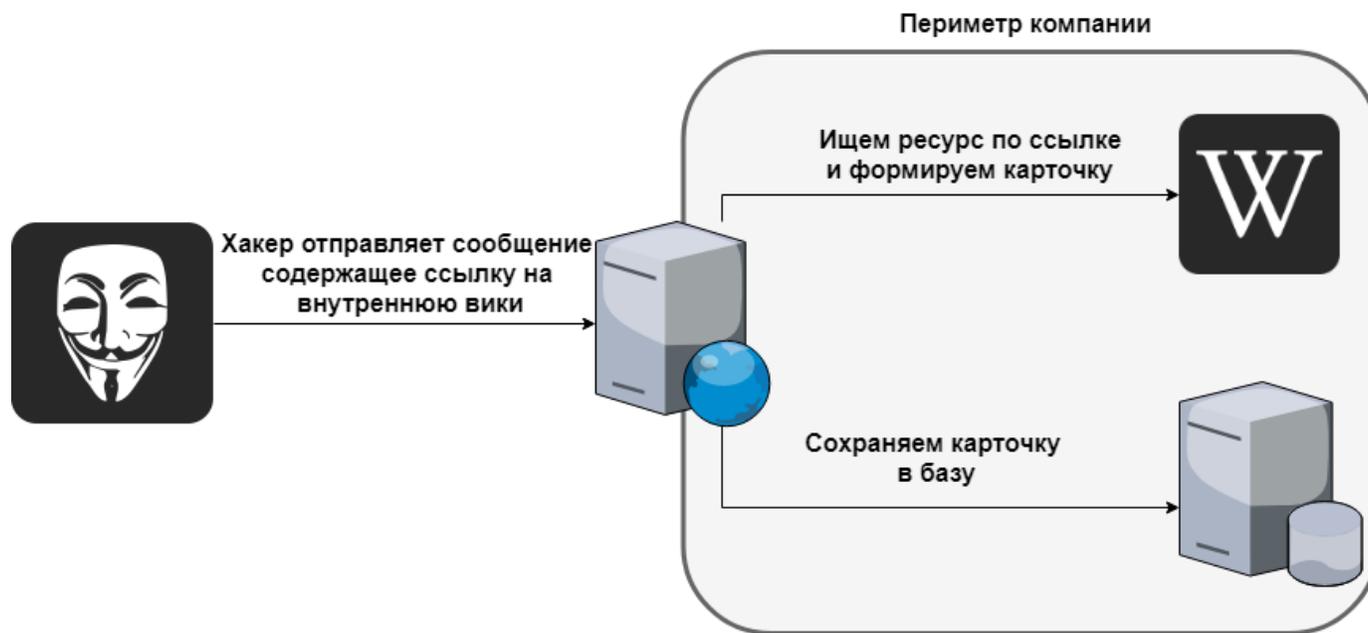
Cat of the Day features a new story and photo of what makes your cat wonderful every day since 1998. Cat of the Day is a simple award-winning, family-friendly, free, and fun ...

13:47 ✓

# Первая реализация



# Первые проблемы



# Масштаб трагедии

Злоумышленнику теперь потенциально доступно:

- Сканирование внутренней сети компании;
- Обращение к внутренним ресурсам компании, которые могут содержать чувствительную информацию;
- Обращение к метаданной облачного провайдера;
- API Kubernetes;
- Исполнение произвольных HTTP GET-запросов;
- А может даже и произвольных TCP пакетов.

# Облака

## Облачные API:

- AWS, например <http://169.254.169.254/latest/user-data>;
- Google Cloud, например <http://169.254.169.254/computeMetadata/v1/> с дополнительным заголовком;
- Digital Ocean, например <http://169.254.169.254/metadata/v1.json> ;
- Packetcloud, например <https://metadata.packet.net/userdata>;
- Azure, например <http://169.254.169.254/metadata/instance> с дополнительным заголовком;
- Oracle Cloud, например <http://169.254.169.254/opc/v1/instance/>.

# Знакомьтесь, gopher://

```
gopher://  
smtp.internal:25/xHELO%20localhost%25d%250a  
MAIL%20FROM%3A%3Chacker@site.com%3E%25  
0d%250aRCPT%20TO%3A%3Cvictim@site.com%3E  
%250d%250aDATA%250d%250aFrom%3A%20%5B  
Hacker%5D%20%3Chacker@site.com%3E%250d%  
250aTo%3A%20%3Cvictime@site.com%3E%250d%  
250aDate%3A%20Tue%2C%2015%20Sep%202017  
%2017%3A20%3A26%20400%250d%250aSubject  
%3A%20AH%20AH%20AH%250d%250a%250d%25  
0aYou%20didn%27t%20say%20the%20magic%20w  
ord%20%21%250d%250a%250d%250a%250d%25  
0a.%250d%250aQUIT%250d%250a
```



```
HELO localhost  
MAIL FROM:<hacker@site.com>  
RCPT TO:<victim@site.com>  
DATA  
From: [Hacker] <hacker@site.com>  
To: <victime@site.com>  
Date: Tue, 15 Sep 2017 17:20:26 -0400  
Subject: Ah Ah AH  
  
You didn't say the magic word !  
  
.   
QUIT
```

# Про другие схемы тоже не стоит забывать

Потенциально проблемные схемы:

- Схема HTTP/HTTPS, произвольный GET-запрос
- Схема GOPHER, произвольные TCP пакеты (очень часто всплывают ограничения)
- Схема TFTP, произвольные UDP датаграммы (тоже есть ограничения)
- Схема FILE, потенциальное обращение к диску
- Схема FTP, SFTP, LDAP и т.д.

# Хорошо, теперь мы напряглись

Первый интуитивные механизмы защиты:

- Проверять схему в ссылке и работать только с **HTTP/HTTPS**
- Проверять адрес в ссылке и блокировать запросы на внутренние подсети (**192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8, 127.0.0.0/8** и т.д.)

# А вот злоумышленник пока нет

Основные техники обхода фильтрации:

- Собственные DNS-записи указывающие на локальные адреса;
- Использование DNS-rebinding;
- Перенаправление, в том числе со сменой схемы;
- Альтернативные представления, про которые все забывают;
- Внезапная нормализация;
- IPv6, просто IPv6;
- Запутанные URL, которые разные парсеры понимают по разному;

Собственные  
DNS-записи  
указывающие  
на локальные  
адреса

```
→ ~ nslookup localtest.me
Server:          8.8.8.8
Address:         8.8.8.8#53

Non-authoritative answer:
Name:   localtest.me
Address: 127.0.0.1

→ ~
[cli] 0:~* 1:~/dirsearch 2:~-
```

# Использование DNS-rebinding

```
→ ~ nslookup ssrf-race-169.254.169.254.localdomain.pw
Server:      8.8.8.8
Address:     8.8.8.8#53

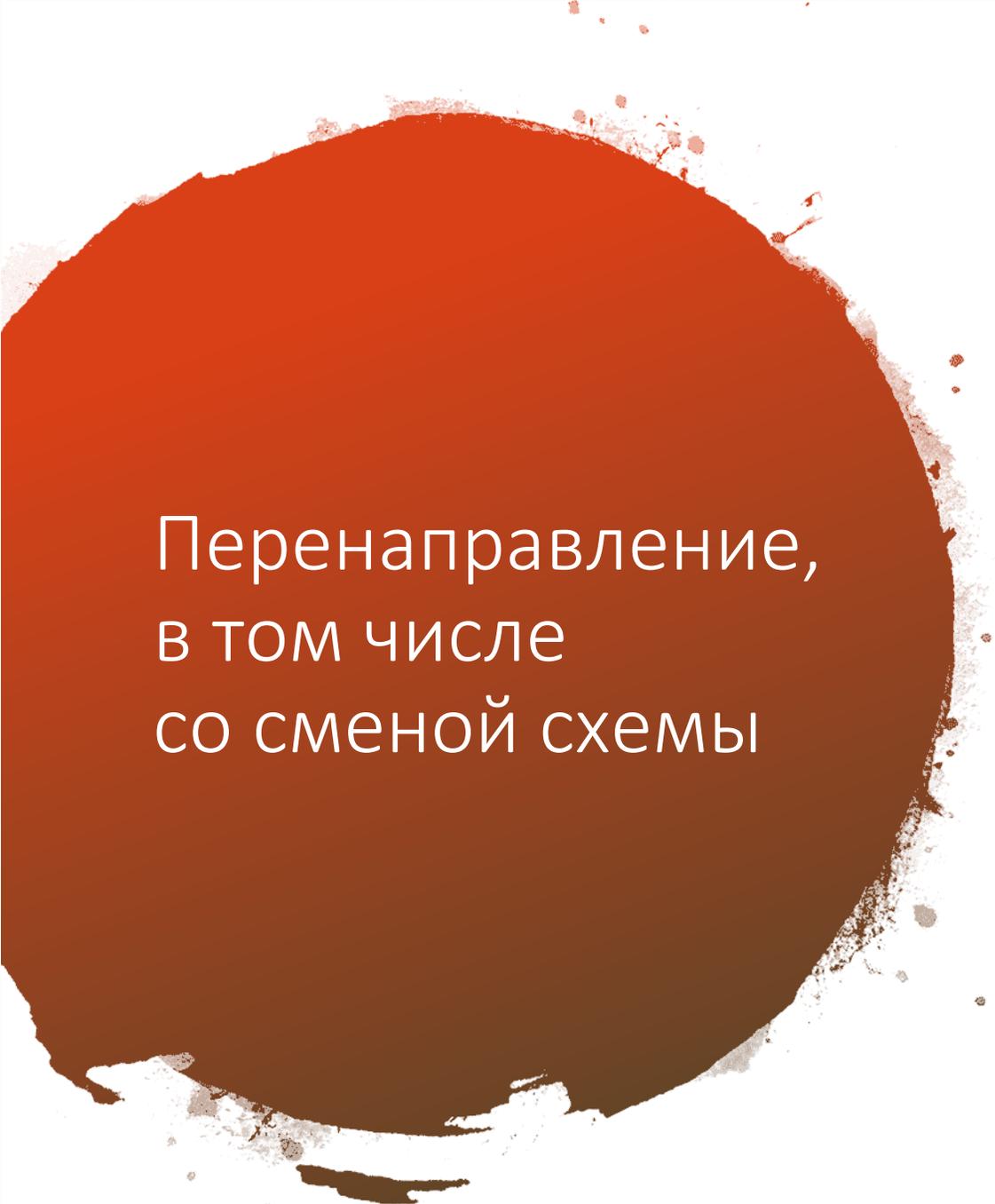
Non-authoritative answer:
Name:   ssrf-race-169.254.169.254.localdomain.pw
Address: 216.58.214.206
Name:   ssrf-race-169.254.169.254.localdomain.pw
Address: 169.254.169.254

→ ~ nslookup ssrf-race-169.254.169.254.localdomain.pw
Server:      8.8.8.8
Address:     8.8.8.8#53

Non-authoritative answer:
Name:   ssrf-race-169.254.169.254.localdomain.pw
Address: 169.254.169.254
Name:   ssrf-race-169.254.169.254.localdomain.pw
Address: 216.58.214.206

→ ~
[cli] 0:~* 1:~/dirsearch 2:~-
```

```
1 # Наша бизнес-логика с защитой от обращения к метаданным
2 def action(url):
3     if get_ip_by_domain(url) != "169.254.169.254":
4         send_data_to(url)
5     else:
6         print "incorect domain"
7
8 # Реализация функции отправки данных
9 def send_data_to(url):
10    ip = get_ip_by_domain(url)
11    send_data(ip, "data")
```



Перенаправление,  
в том числе  
со сменой схемы

Запрос:

<https://hackers-normal-site.com>

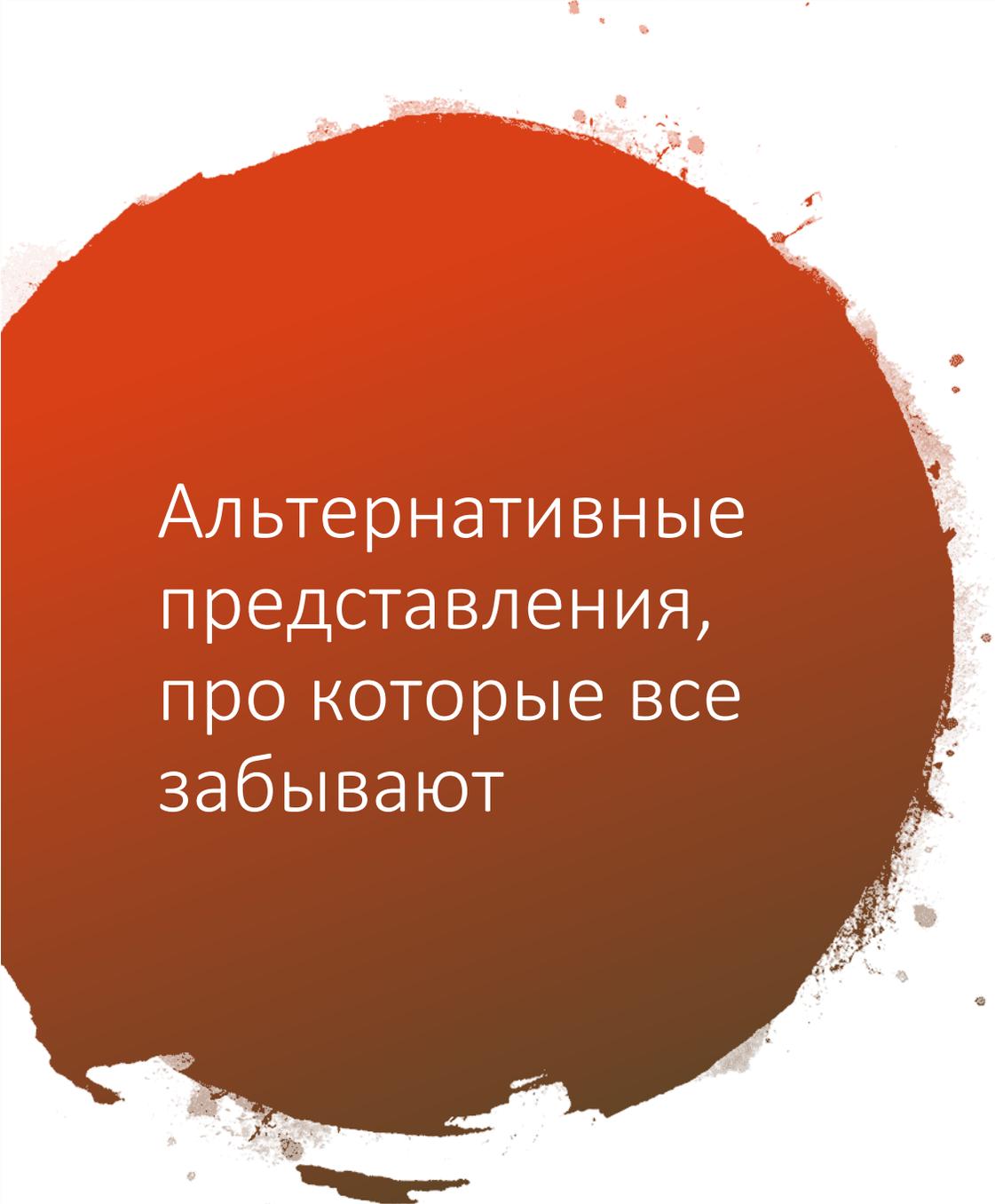
Ответ:

HTTP/1.1 302 Found

Date: Fri, 24 Aug 2018 12:15:36 GMT

Location: <http://169.254.169.254/>

Может быть несколько последовательных  
редиректов



Альтернативные  
представления,  
про которые все  
забывают

Многие библиотеки поддерживают сокращения. **127.0.0.1** и **127.1** для них одно и то же.

Многие библиотеки поддерживают отличные от десятичного представления октетов. **127.0.0.1** и **0177.1** для них одно и то же.

Многие библиотеки поддерживают смешанные представления октетов. **127.127.0.1** и **0x7f.0177.1** для них одно и то же.

Выходит, например, для cURL нету разницы между **127.0.0.1** и **127.1**, и **0177.1**, и **0x7f.1**, и **2130706433**. Ааааа!

# Внезапная нормализация

Да, всё это может оказаться равно  
`http://169.254.169.254/`:

- `http://169。254。169。254/`
- `http://169。254。169。254/`
- `http://(16)(9)。 (2)(5)(4)。 (16)(9)。 (2)(5)(4)/`
- `http://(2)(8)(5)(2)(0)(3)(9)(1)(6)(6):80/`
- `http://(0)(0)(2)(5)(1)。 (0)(x)(f)(e)。 (4)(3)(5)(1)(8):80/`

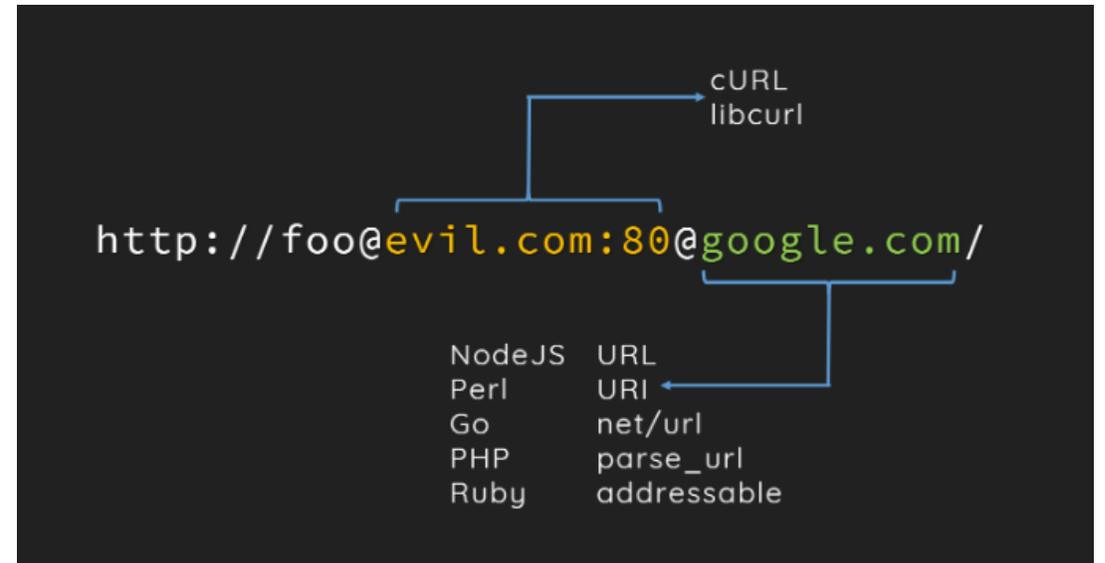


IPv6, просто IPv6

Не стоит забывать про IPv6, возможно  
вы поддерживаете его и даже не знаете:  
127.0.0.1 == [::1]

Запутанные URL,  
которые разные  
парсеры  
понимают по  
разному

Спасибо ребятам из ORANGE TSAI:



Особенно актуально, если вы используете микросервисы на разных технологических стеках

# Итак, как же бороться с SSRF?

В первую очередь:

- Ограничить поддерживаемые схемы;
- Ограничить допустимые порты;
- Отключить поддержку перенаправлений или проверять каждый шаг;
- Валидировать доменные имена;
- Разобраться как используемая библиотека обрабатывает адреса;

Лишним не будет:

- Ограничить доступ к внутренней инфраструктуре для потенциально подверженных SSRF серверов.
- Вынести подобный функционал в отдельный изолированный сервис общий для всех команд (решения для крупных компаний)
- Metadata concealment

# Наконец-то, наш чатик в безопасности, но в безопасности ли мы?

Когда стоит задуматься о потенциальной SSRF?

- При реализации механизма webhooks;
- При работе обработке форматов основанных на XML (уязвимость XXE);
- При рендере скриншотов (для успешного рендера зачастую необходимо исполнить произвольный javascript, который может содержать функционал отправки запросов);
- При работе с видео (не обновлённые версии библиотеки Ffmpeg подвержены SSRF);
- Всегда при отправке запросов на предоставляемый пользователем адрес.

# Материалы для заинтересовавшихся

Наиболее актуальные:

- <https://github.com/cujanovic/SSRF-Testing>
- <https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SSRF%20injection>
- SSRF bible. <https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/>
- <https://hackerone.com/reports/341876> [ 25 000\$ за захват k8s через SSRF ]

Спасибо!



POSITIVE DEVELOPMENT USER GROUP

POSITIVE  
DEVELOPMENT  
USER  
GROUP