Digital Security

(Г)розовые облака

Опасности и возможности облачных сервисов

@Ivan_IGC
@rumiljonov

whoarewe

Иван Чалыкин
i.chalykin@dsec.ru

Рустам Комилджонов
r.komildzhonov@dsec.ru

- Twitter: @Ivan_IGC
- Pentester @Digital Security
- Bug hunter (Yandex, Mail.ru, QIWI, Samsung,)
- Sometimes a speaker (ZeroNights)

- Twitter: @rumiljonov
- Pentester @Digital Security
- Lazy bughunter
- Definitely not a speaker

Содержание

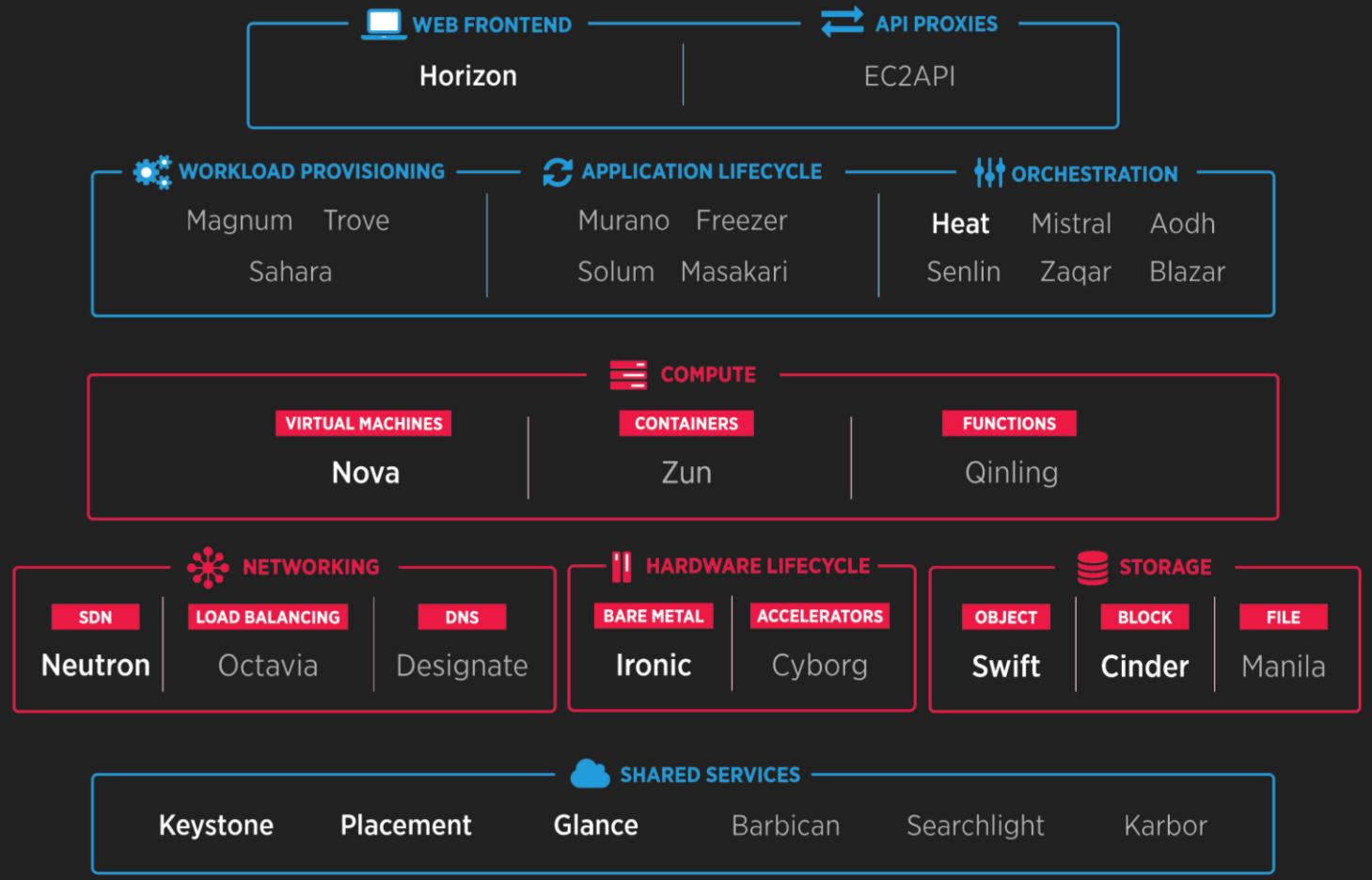
Openstack

Структура и компоненты

Набор программных средств для управления пулом вычислительных ресурсов, хранилищ и сетевых средств.

Особенности:

- Opensource
- Модульная архитектура
- Публичные/приватные облака
- Кроссплатформенность
- Управление через Web UI / API / CLI



Openstack

Vulnerability Management Process

Classes	Outcome	Description
Class A	OSSA	A vulnerability to be fixed in master and all supported releases
Class B1	OSSN	A vulnerability that can only be fixed in master, security note for stable branches, e.g., default config value is insecure
Class B2	OSSN	A vulnerability without a complete fix yet, security note for all versions, e.g., poor architecture / design
Class B3	OSSN	A vulnerability in experimental or debugging features not intended for production use
Class C1	Potential OSSN	Not considered a practical vulnerability (but some people might assign a CVE for it), e.g. one depending on UUID guessing
Class C2	Potential OSSN	A vulnerability, but not in OpenStack supported code, e.g., in a dependency
Class D	Potential OSSN	Not a vulnerability, just a bug with (some) security implications, e.g., strengthening opportunities / misleading documentation

Openstack

Vulnerability Management Process

Classes	Outcome	Description
Class A	OSSA	A vulnerability to be fixed in master and all supported releases
Class B1	OSSN	A vulnerability that can only be fixed in master, security note for stable branches, e.g., default config value is insecure
Class B2	OSSN	A vulnerability without a complete fix yet, security note for all versions, e.g., poor architecture / design
Class B3	OSSN	A vulnerability in experimental or debugging features not intended for production use
Class C1	Potential OSSN	Not considered a practical vulnerability (but some people might assign a CVE for it), e.g. one depending on UUID guessing
Class C2	Potential OSSN	A vulnerability, but not in OpenStack supported code, e.g., in a dependency
Class D	Potential OSSN	Not a vulnerability, just a bug with (some) security implications, e.g., strengthening opportunities / misleading documentation

https://security.openstack.org/ossalist.html

Openstack

Vulnerability Management Process

Classes	Outcome	Description
Class A	OSSA	A vulnerability to be fixed in master and all supported releases
Class B1	OSSN	A vulnerability that can only be fixed in master, security note for stable branches, e.g., default config value is insecure
Class B2	OSSN	A vulnerability without a complete fix yet, security note for all versions, e.g., poor architecture / design
Class B3	OSSN	A vulnerability in experimental or debugging features not intended for production use
Class C1	Potential OSSN	Not considered a practical vulnerability (but some people might assign a CVE for it), e.g. one depending on UUID guessing
Class C2	Potential OSSN	A vulnerability, but not in OpenStack supported code, e.g., in a dependency
Class D	Potential OSSN	Not a vulnerability, just a bug with (some) security implications, e.g., strengthening opportunities / misleading documentation

https://wiki.openstack.org/wiki/Security_Notes

Openstack

Vulnerability Management Process

Classes	Outcome	Description
Class A	OSSA	A vulnerability to be fixed in master and all supported releases
Class B1	OSSN	A vulnerability that can only be fixed in master, security note for stable branches, e.g., default config value is insecure
Class B2	OSSN	A vulnerability without a complete fix yet, security note for all versions, e.g., poor architecture / design
Class B3	OSSN	A vulnerability in experimental or debugging features not intended for production use
Class C1	Potential OSSN	Not considered a practical vulnerability (but some people might assign a CVE for it), e.g. one depending on UUID guessing
Class C2	Potential OSSN	A vulnerability, but not in OpenStack supported code, e.g., in a dependency
Class D	Potential OSSN	Not a vulnerability, just a bug with (some) security implications, e.g., strengthening opportunities / misleading documentation

https://bugs.launchpad.net/openstack/+bugs?field.tag=security&orderby=-id&start=0

Openstack

Автоматическое тестирование

Syntribos – инструмент для автоматического тестирования API и, в частности, Openstack.

https://github.com/openstack/syntribos

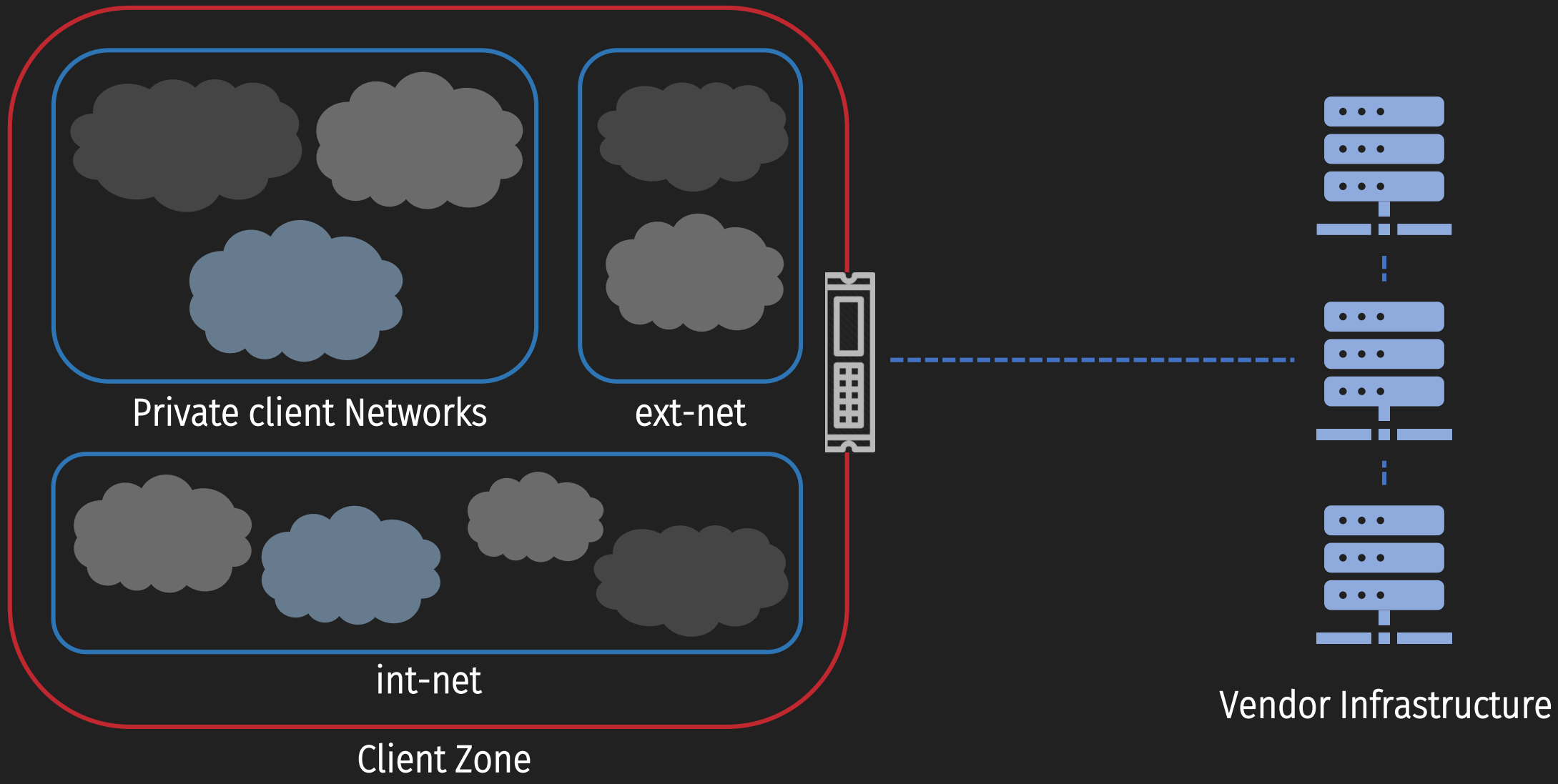
Возможные проверки:

- Buffer Overflow
- Command Injection
- CORS Wildcard
- Integer Overflow
- LDAP Injection
- SQL Injection
- String Validation
- XML External Entity
- Cross Site Scripting (XSS)
- Regex Denial of Service (ReDoS)
- JSON Parser Depth Limit
- User Defined

Доступ к внутренней инфраструктуре

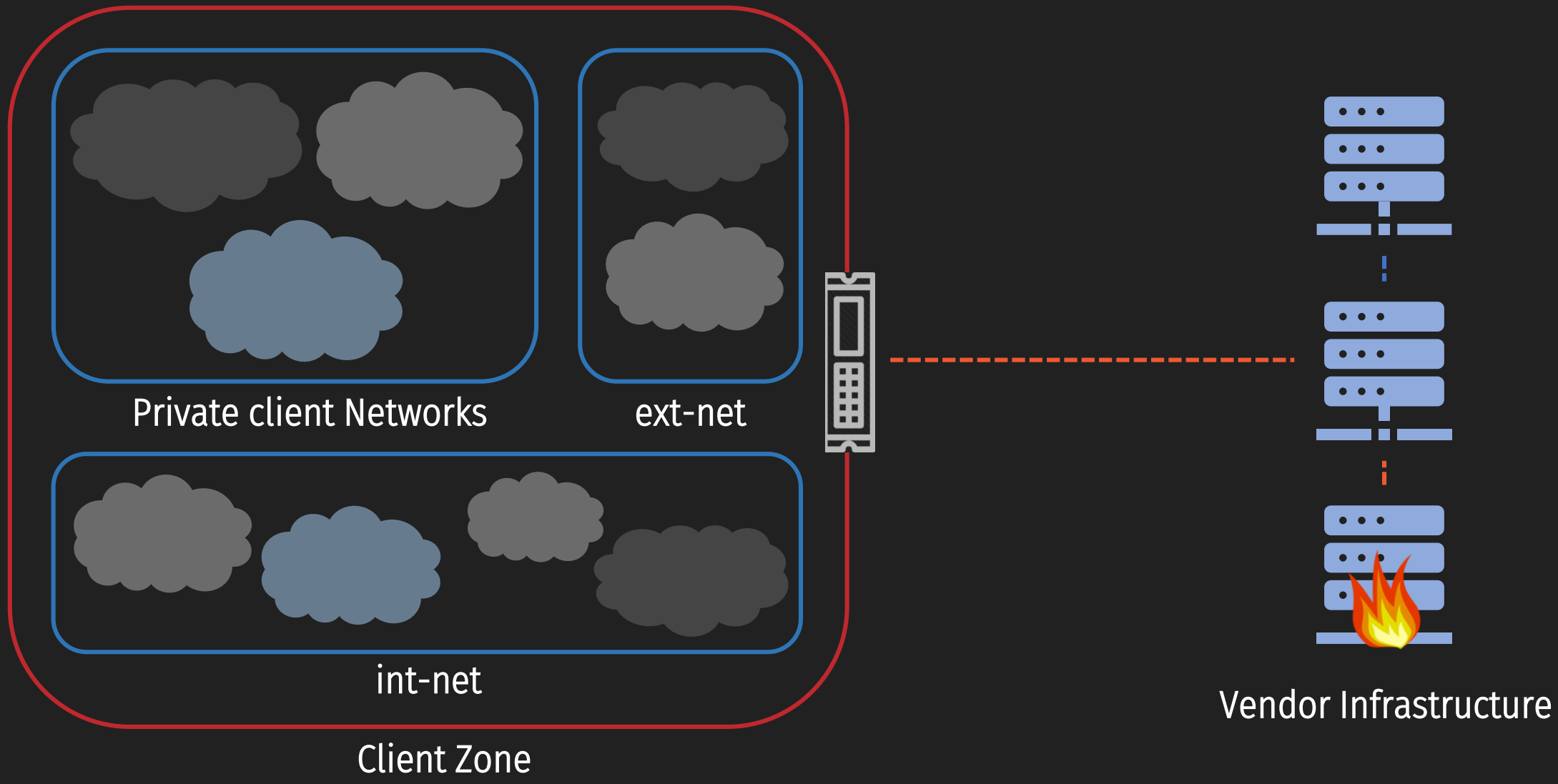
Доступ к внутренней инфраструктуре

Проблемы сегментации



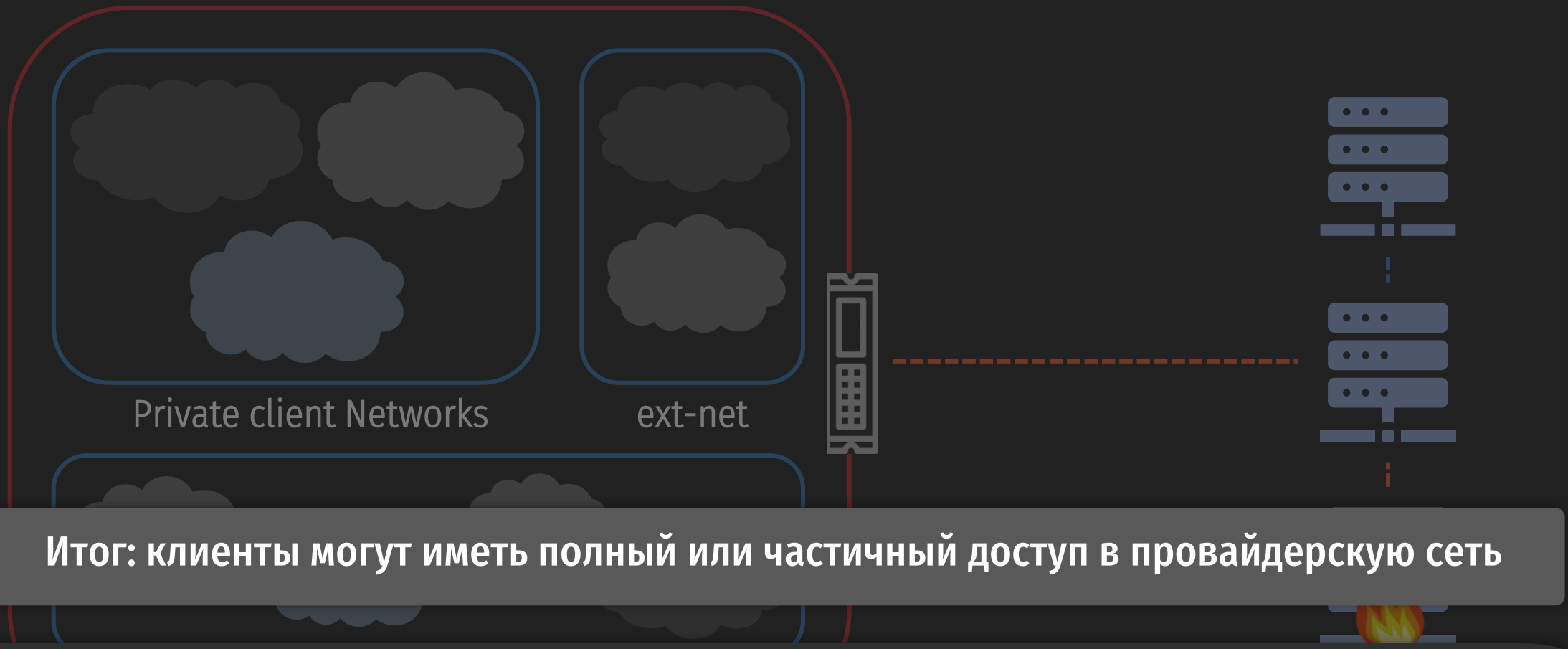
Доступ к внутренней инфраструктуре

Проблемы сегментации



Доступ к внутренней инфраструктуре

Проблемы сегментации

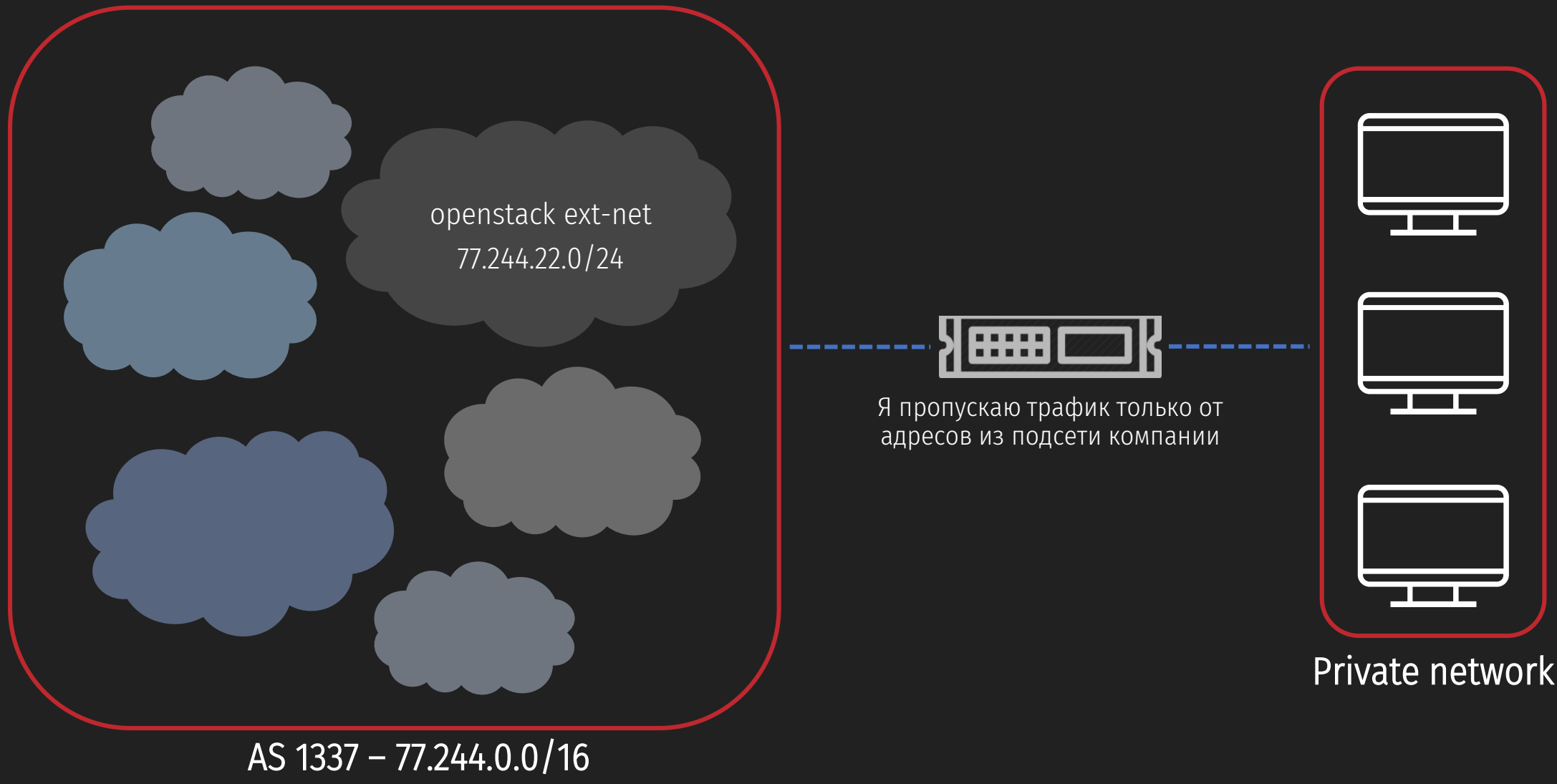


Итог: клиенты могут иметь полный или частичный доступ в провайдерскую сеть

FIX: Проверять сегментацию

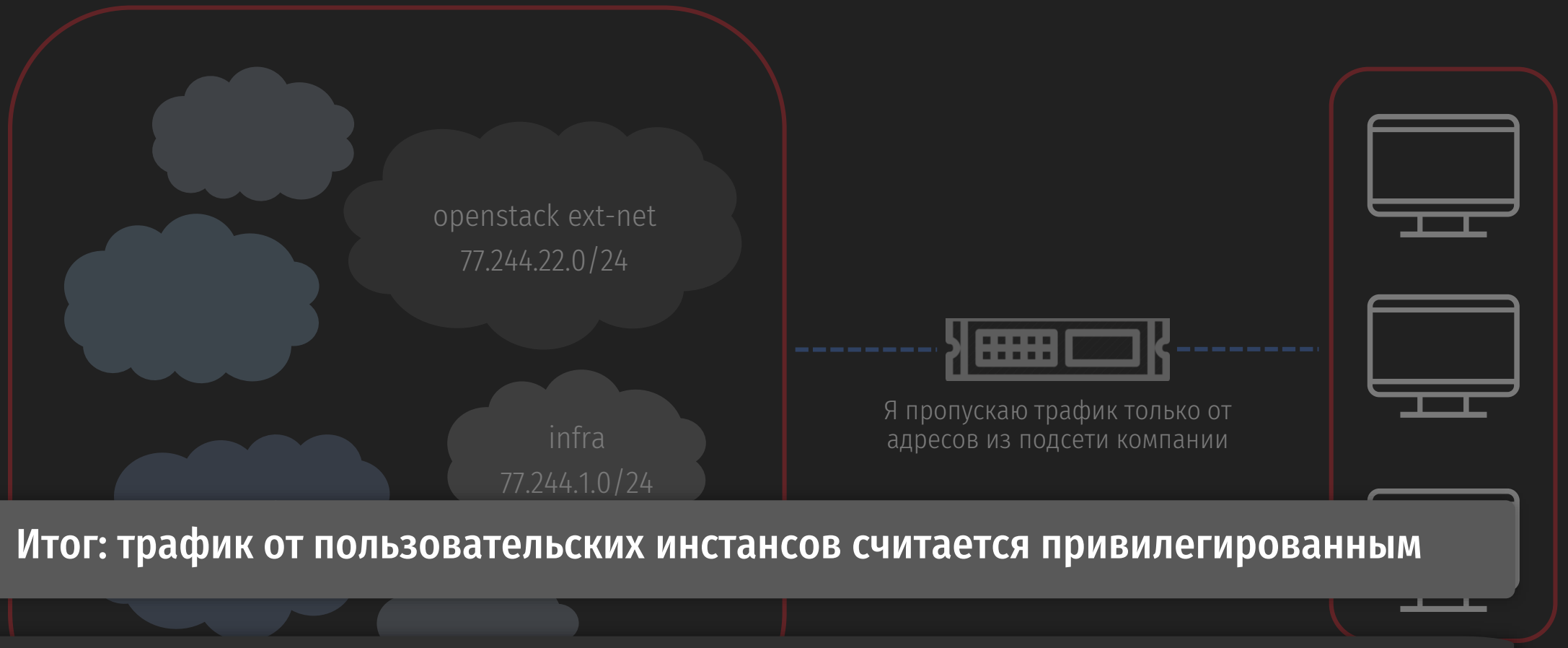
Доступ к внутренней инфраструктуре

«Доверенная» сеть



Доступ к внутренней инфраструктуре

«Доверенная» сеть



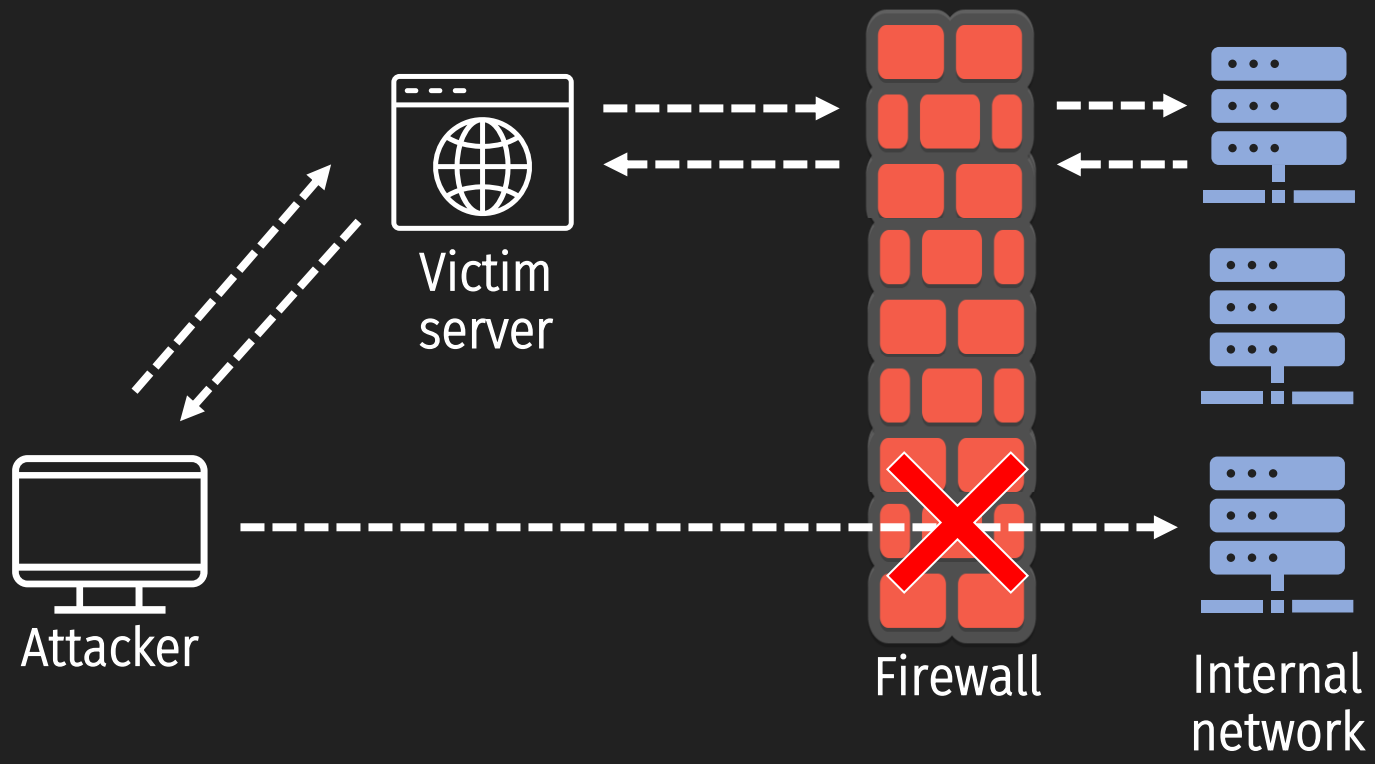
Итог: трафик от пользовательских инстансов считается привилегированным

FIX: Рассматривать ext-net как недоверенную часть сети

Доступ к внутренней инфраструктуре

Server-Side Request Forgery

SSRF (Подделка запросов со стороны сервера) - позволяет злоумышленнику совершать запросы во внутреннюю сеть от имени уязвимого сервера



Возможные последствия:

- Отправка запросов от имени доверенного сервера
- Сканирование внутренней сети
- Чтение файлов на сервере
- Раскрытие информации
- И другие сценарии...

Доступ к внутренней инфраструктуре

Server-Side Request Forgery

Heat SSRF [OSSA 2016-013]

- Blind
- Error and Time based
- HTTP GET Only

```
POST /v1/{pid}/validate HTTP/1.1
Host: my.cloud
X-Auth: {token}
Content-Type: application/json

{"template_url": "http://localhost:22"}
```

Glance SSRF [OSSN-0078]

- non-Blind
- HTTP GET Only

Доступ к внутренней инфраструктуре

Server-Side Request Forgery

Heat SSRF [OSSA 2016-013]

- Blind
- Error and Time based
- HTTP GET Only

```
POST /v1/{pid}/validate HTTP/1.1
Host: my.cloud
X-Auth: {token}
Content-Type: application/json

{"template_url": "http://localhost:22"}
```

Glance SSRF [OSSN-0078]

Non-Blind

HTTP GET Only

DEPRECATED

Доступ к внутренней инфраструктуре

Server-Side Request Forgery

Heat SSRF [OSSA 2016-013]

- Blind
- Error and Time based
- HTTP GET Only

```
POST /v1/{pid}/validate HTTP/1.1
Host: my.cloud
X-Auth: {token}
Content-Type: application/json

{"template_url": "http://localhost:22"}
```

Glance Image Import

- non-Blind
- HTTP GET Only
- Async
- Whitelist and Blacklist

```
POST /v2/images/{image_id}/import
HTTP/1.1
Host: my.cloud
X-Auth: {token}
Content-Type: application/json

{"method": {"name": "web-
download","uri": "http://localhost:22"}}
```

Доступ к внутренней инфраструктуре

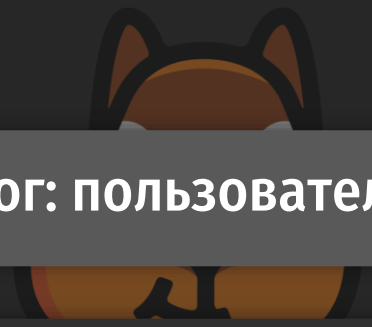
Server-Side Request Forgery

Heat SSRF [OSSA 2016-013]

- Blind
- Error and Time based
- HTTP GET Only

```
POST /v1/{pid}/validate HTTP/1.1
Host: my.cloud
X-Auth: {token}
Content-Type: application/json

{"template_url": "http://localhost:22"}
```



Glance Image Import

```
POST /v2/images/{image_id}/import
HTTP/1.1
Host: my.cloud
```

Итог: пользователь имеет ограниченный доступ во внутреннюю сеть

HOTFIX: Запретить доступ ролевыми политикам

FIX: Обновить модули

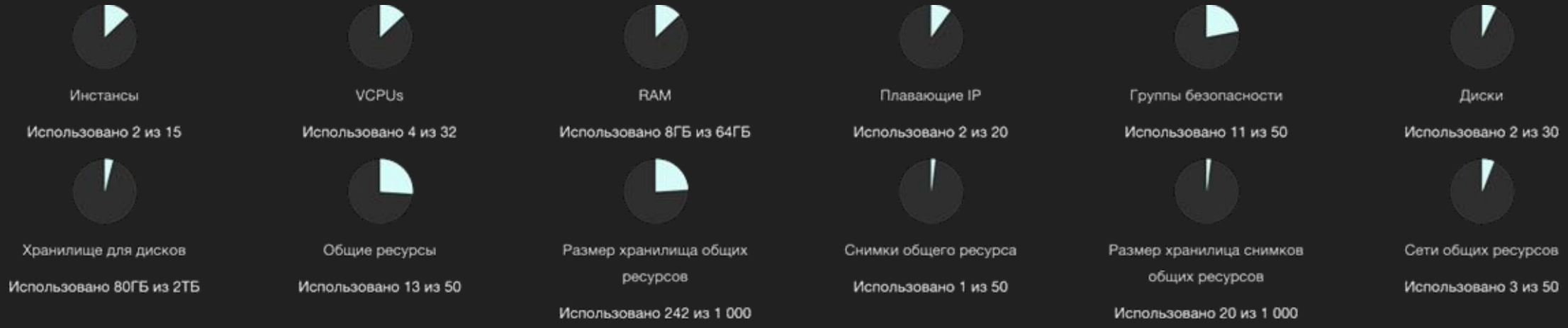
Обход лимитов

Обход лимитов

Лимиты на ресурсы в Openstack

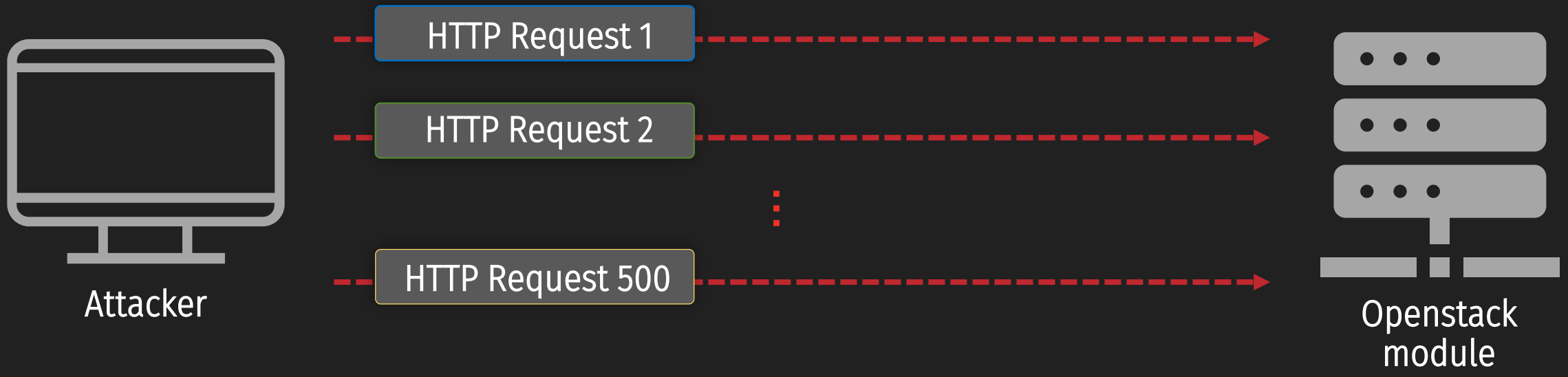
Openstack поддерживает механизм квот, что позволяет ограничивать максимальное количество ресурсов, выделяемых на отдельные проекты.

```
> openstack quota show -f yaml
backup-gigabytes: -1
backups: 200
cores: 32
fixed-ips: -1
floating-ips: 20
gigabytes: 2000
...
```



Обход лимитов

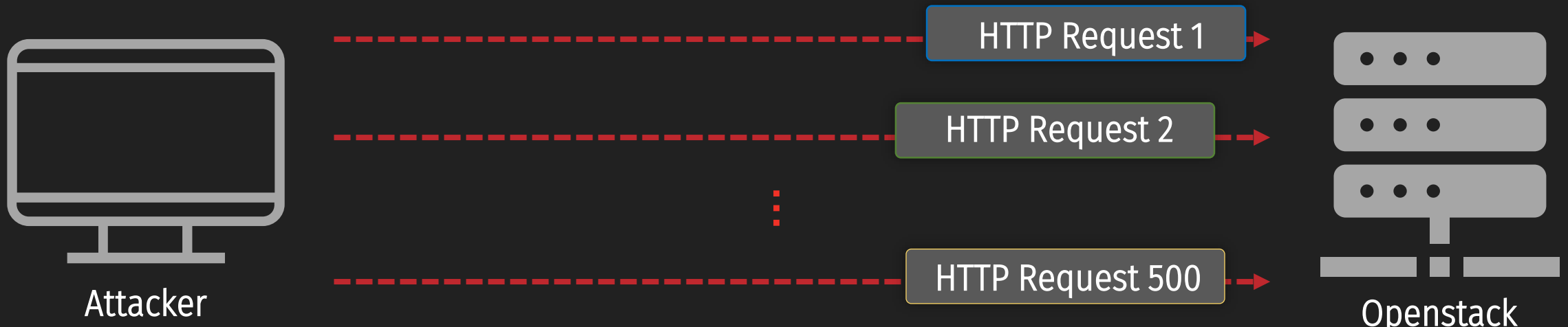
Race condition



Race condition (состояние гонки) возникает при ошибке в исходном коде многопоточного приложения и позволяет нарушить логику его работу.

Обход лимитов

Race condition



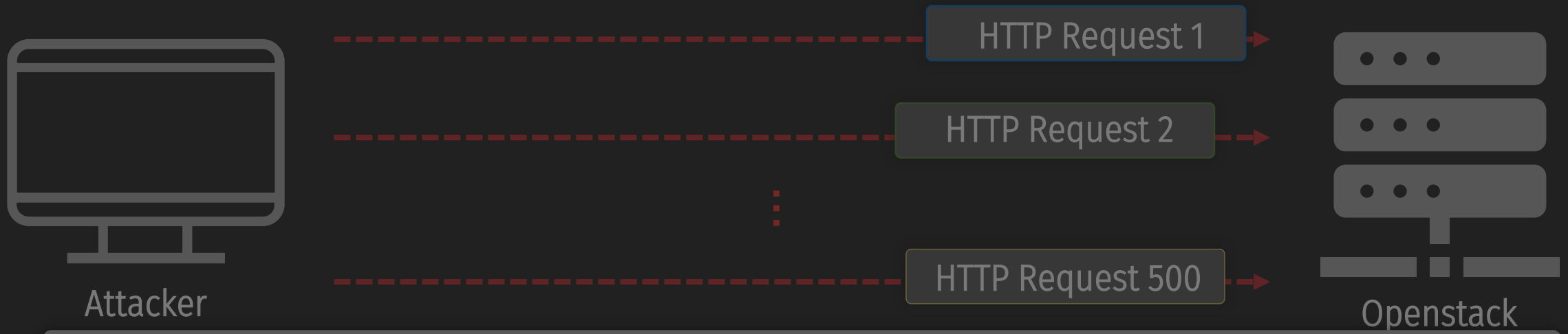
Увеличивая количество параллельных запросов можно повысить вероятность срабатывания состояния гонки.

Race condition в веб-приложениях

https://habr.com/ru/post/460339/

Обход лимитов

Race condition

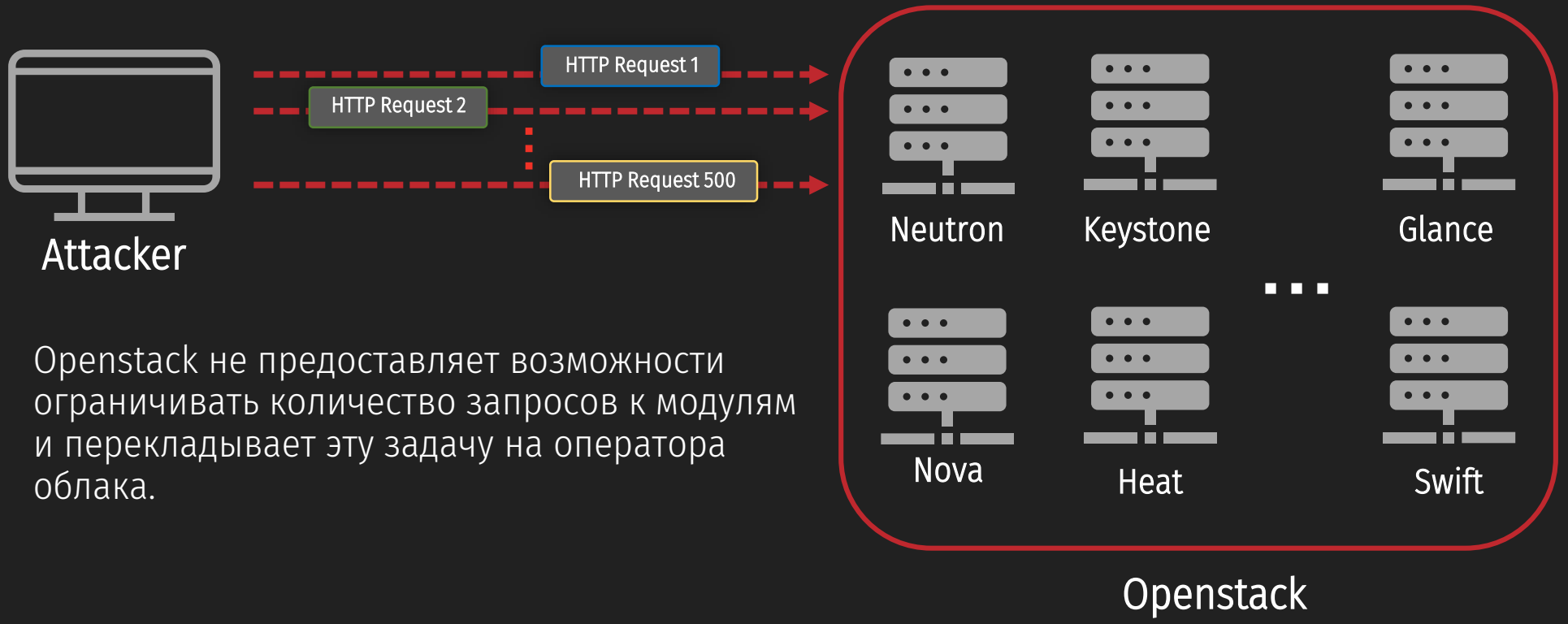


Итог: Пользователь может превышать назначенные ему квоты

FIX: Отсутствует. Операторам необходимо самим реализовывать решения для борьбы с Race Condition уязвимостями

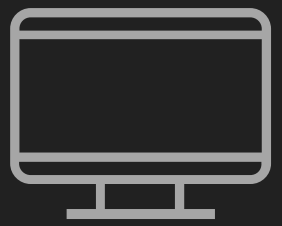
Отказ в обслуживании

Отказ в обслуживании



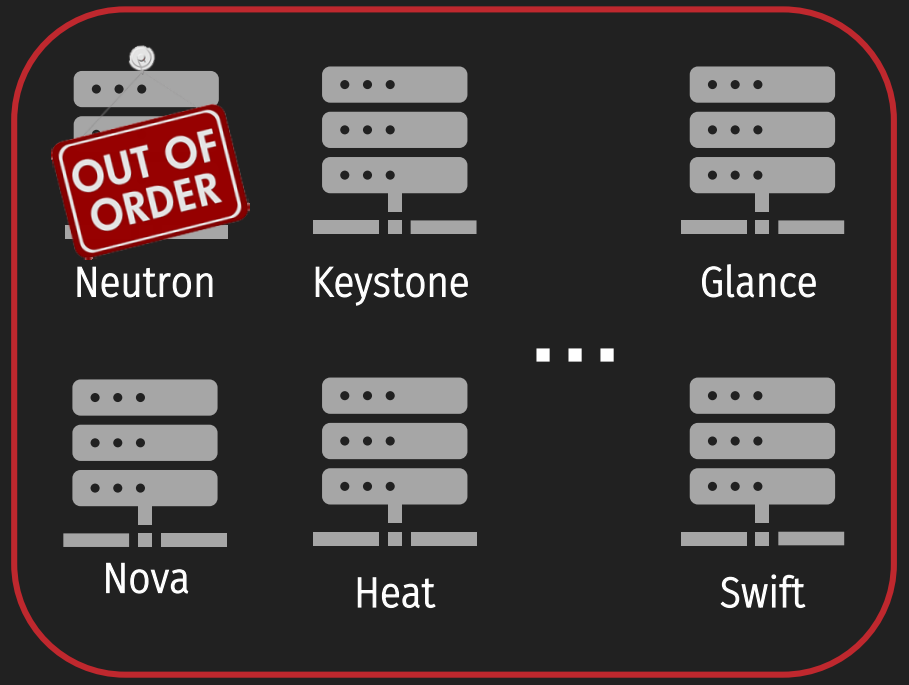
Openstack не предоставляет возможности ограничивать количество запросов к модулям и перекладывает эту задачу на оператора облака.

Отказ в обслуживании

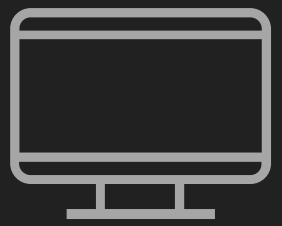


Attacker

При частом обращении к ресурсозатратной функциональности конкретного модуля возможен вызов его отказа в обслуживании.

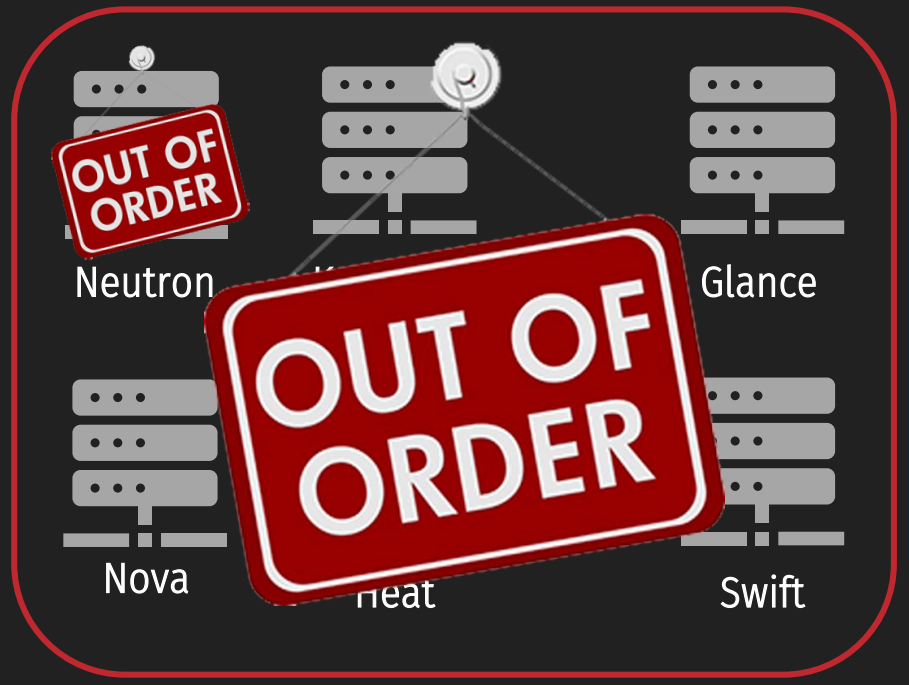


Отказ в обслуживании



Attacker

При выходе из строя одного из core модулей Опенстака, выходит из строя вся система.

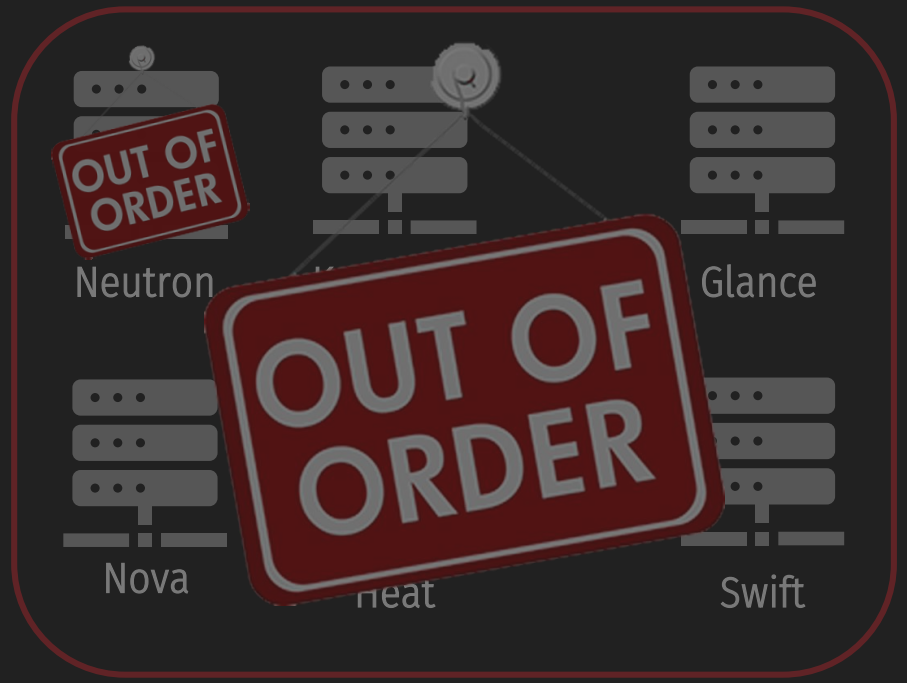


Openstack

Отказ в обслуживании

Attacker

При выходе из строя одного из core модулей Опенстака, выходит из строя вся система.



Openstack

”Operators are recommended to place rate-limiting solutions in front of API endpoints to reduce the impact a user can cause (either intentionally or accidentally) by making rapid-fire requests.“

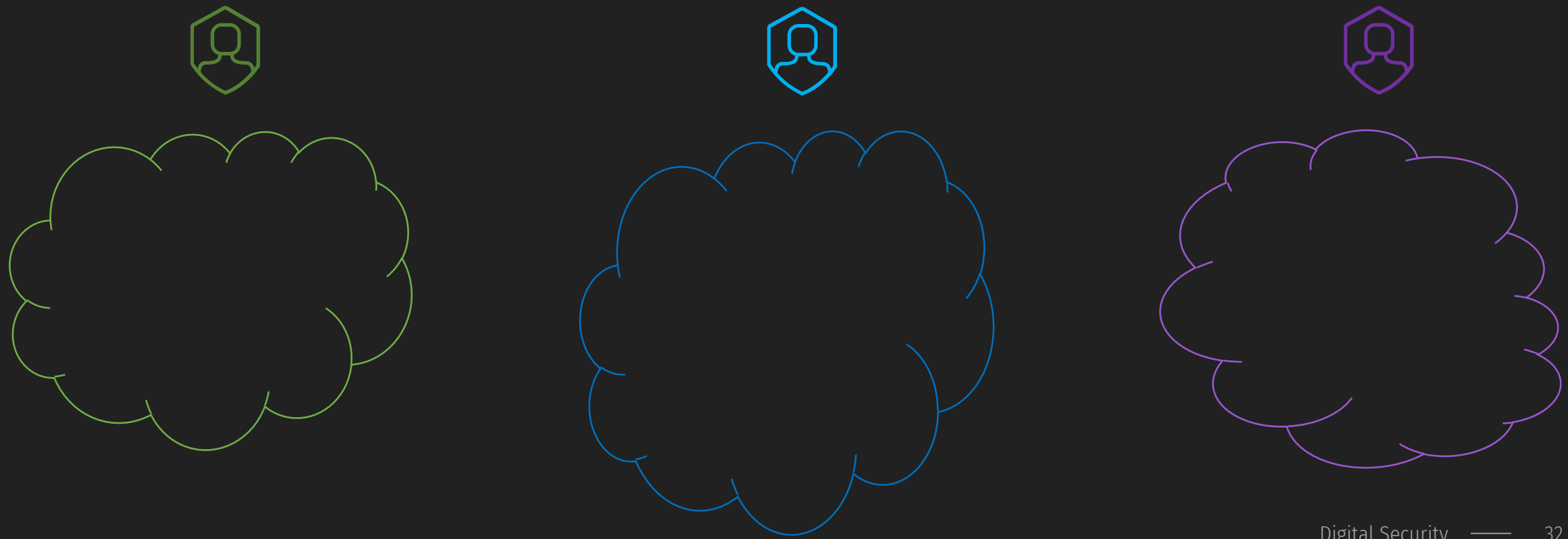
Сетевые проблемы

Сетевые проблемы

Отсутствие фильтрации между сегментами сети

При создании сервера не указали собственную приватную сеть?

Тогда сервер попадает в общую. В данной сети к нему будут иметь доступ другие пользователи .

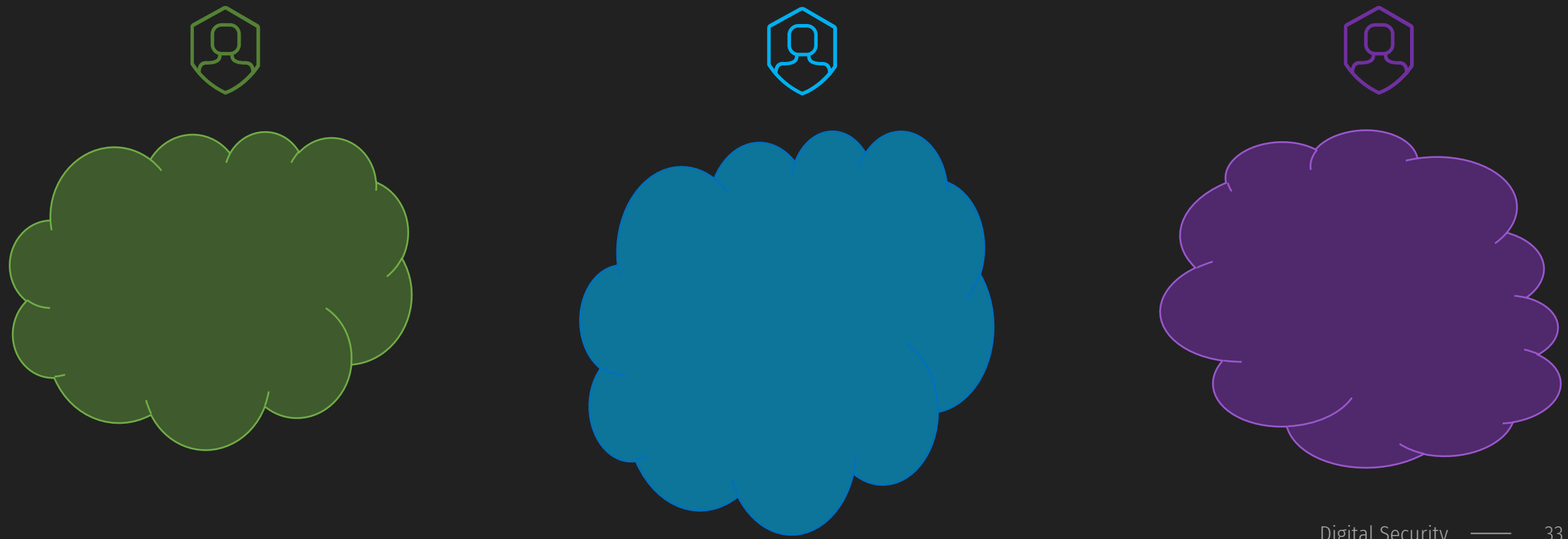


Сетевые проблемы

Отсутствие фильтрации между сегментами сети

При создании сервера не указали собственную приватную сеть?

Тогда сервер попадает в общую. В данной сети к нему будут иметь доступ другие пользователи .

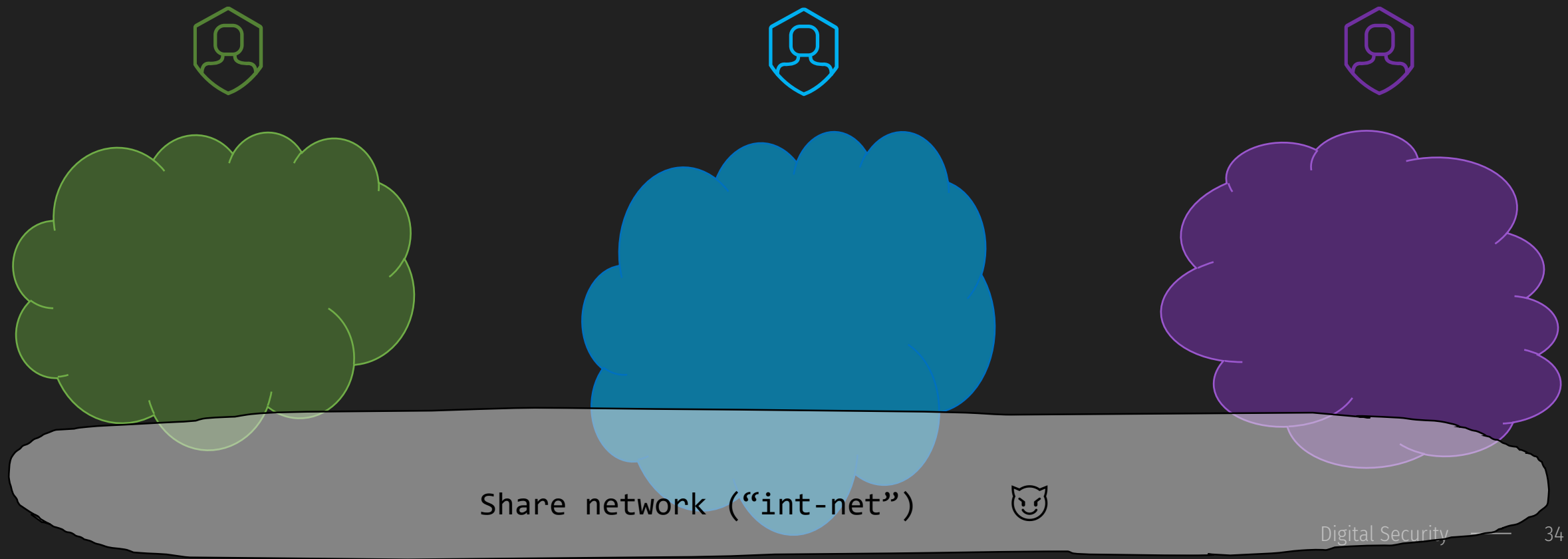


Сетевые проблемы

Отсутствие фильтрации между сегментами сети

При создании сервера не указали собственную приватную сеть?

Тогда сервер попадает в общую. В данной сети к нему будут иметь доступ другие пользователи .

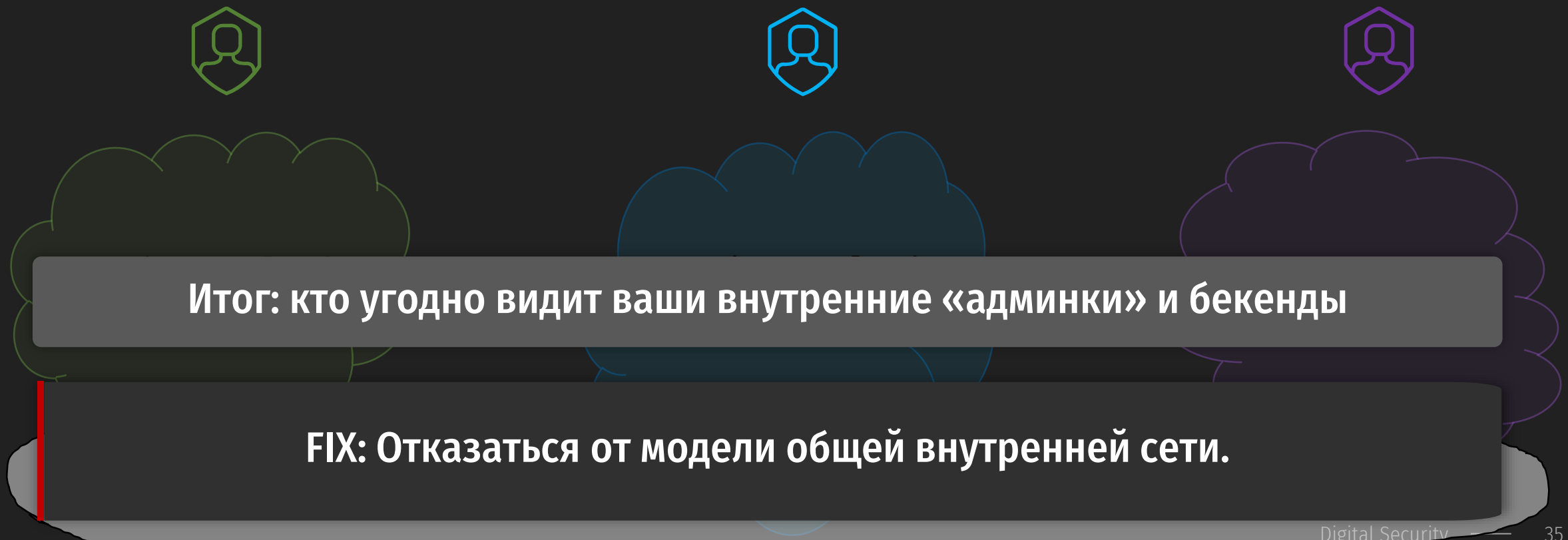


Сетевые проблемы

Отсутствие фильтрации между сегментами сети

При создании сервера не указали собственную приватную сеть?

Тогда сервер попадает в общую. В данной сети к нему будут иметь доступ другие пользователи .



Итог: кто угодно видит ваши внутренние «админки» и бекенды

FIX: Отказаться от модели общей внутренней сети.

Сетевые проблемы

Подверженность сетевым атакам в стандартной конфигурации

Виртуальные сети также могут быть подвержены «обычным» сетевым атакам — MITM/SPOOFING/etc

```
~# ./network_security_check.py

[-] ARP protection enabled
[-] ICMPv4 Redirect protection enabled
[-] DHCPv4 protection enabled
[+] ICMPv6 Router Advertisement protection disabled
[-] ICMPv6 Neighbor Advertisement protection enabled
[+] DHCPv6 protection disabled
```

github.com/raw-packet

Сетевые проблемы

Подверженность сетевым атакам в стандартной конфигурации

Виртуальные сети также могут быть подвержены «обычным» сетевым атакам — MITM/SPOOFING/etc

```
~# ./network_security_check.py

[-] ARP protection enabled
[-] ICMPv4 Redirect protection enabled
[-] DHCPv4 protection enabled
[+] ICMPv6 Router Advertisement protection disabled
[-] ICMPv6 Neighbor Advertisement protection enabled
[+] DHCPv6 protection disabled
```

github.com/raw-packet

Сетевые проблемы

Подверженность сетевым атакам в стандартной конфигурации

Виртуальные сети также могут быть подвержены «обычным» сетевым атакам — MITM/SPOOFING/etc

```
~# ./network_security_check.py

[-] ARP protection enabled
[-] ICMPv4 Redirect protection enabled
[-] DHCPv4 protection enabled
[+] ICMPv6 Router Advertisement protection disabled
[-] ICMPv6 Neighbor Advertisement protection enabled
[+] DHCPv6 protection disabled
```

github.com/raw-packet

DHCPv6 / ICMPv6_Router_Advertisement спуфинг позволяет нарушителю управлять выдачей сетевых настроек компьютерам сегмента сети

«Теперь я тут шлюз и DNS!»

Сетевые проблемы

Подверженность сетевым атакам в стандартной конфигурации

Виртуальные сети также могут быть подвержены «обычным» сетевым атакам — MITM/SPOOFING/etc

```
~# ./network_security_check.py

[-] ARP protection enabled
[-] ICMPv4 Redirect protection enabled
[-] DHCPv4 protection enabled
[+] ICMPv6 Router Advertisement protection disabled
[-] ICMPv6 Neighbor Advertisement protection enabled
```

Итог: один скомпрометированный сервер может нарушить работу всего сегмента

FIX: Настроить изоляцию. Использовать TLS

Ролевая модель и связанные с ней сложности

Ролевая модель Openstack

- Каждый модуль имеет собственную политики (Identity | Compute | Networking | Storage | Databases)
- Внутри политики описывается вся функциональность модуля и требуемая роль для вызова ("<target>" : "<rule>")
- Политики задаются в JSON/YAML формате

```
"admin_required": "role:admin",
"cloud_admin": "rule:admin_required and domain_id:admin_domain_id",
"service_role": "role:service",
"service_or_admin": "rule:admin_required or rule:service_role",
"default": "rule:admin_required",

"identity:get_service": "rule:admin_or_cloud_admin",
"identity:list_services": "rule:admin_or_cloud_admin",
"identity:create_service": "rule:cloud_admin",
"identity:update_service": "rule:cloud_admin",
"identity:delete_service": "rule:cloud_admin",

"identity:get_endpoint": "rule:admin_or_cloud_admin",
"identity:list_endpoints": "rule:admin_or_cloud_admin",
"identity:create_endpoint": "rule:cloud_admin",
"identity:update_endpoint": "rule:cloud_admin",
"identity:delete_endpoint": "rule:cloud_admin",
```

Ролевая модель Openstack

1. Учесть все модули
2. Учесть все API-методы модуля, в том числе устаревшие, но доступные
3. Учесть все собственные роли
4. Учесть, что пользователь может входить в проект
5. Учесть, что пользователь может входить в домен
6. Учесть, что пользователь может быть владельцем, а может и не быть (но входить в проект)
7. Учесть, что пользователь может аутентифицироваться совершенно разными способами (websso openid/federation/tokens/cert/log:pass/saml/….)
8. Учесть xxx, yyy, zzz…

Ролевая модель Openstack

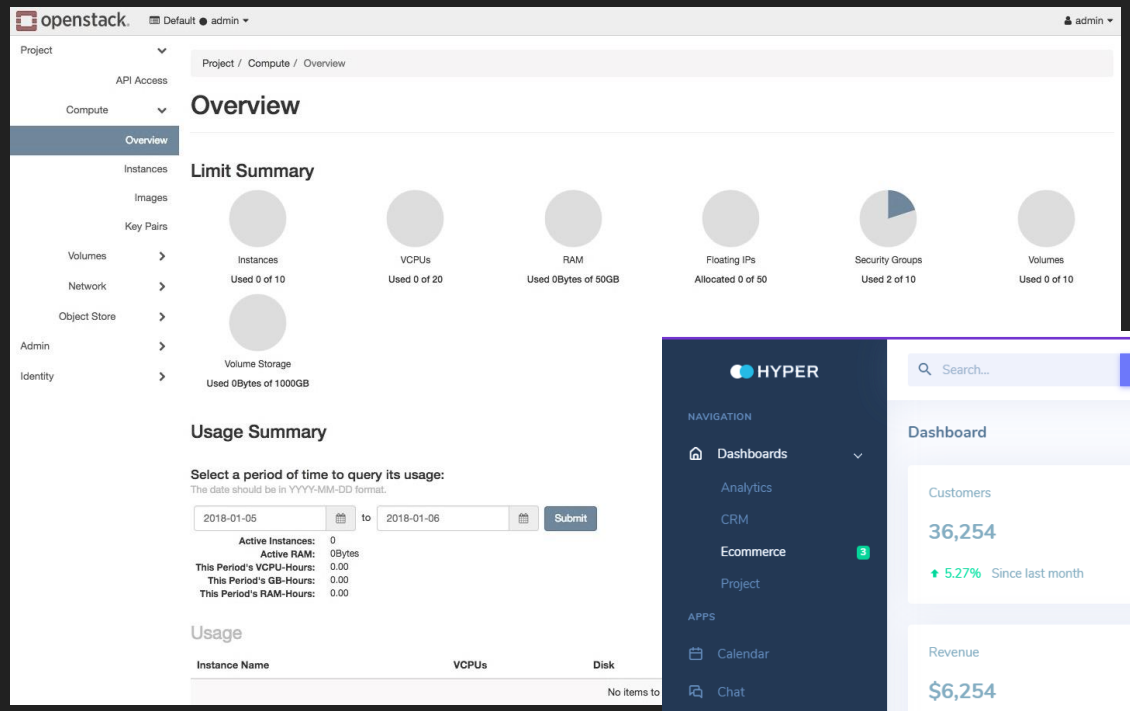
1. Учесть все модули
2. Учесть все API-методы модуля, в том числе устаревшие, но доступные
3. Учесть все собственные роли
4. Учесть, что пользователь может входить в проект
5. Учесть, что пользователь может входить в домен
6. Учесть, что пользователь может быть владельцем, а может и не быть (но входить в проект)
7. Учесть, что пользователь может аутентифицироваться совершенно разными способами (websso openid/federation/tokens/cert/log:pass/saml/….)
8. Учесть xxx, yyy, zzz…

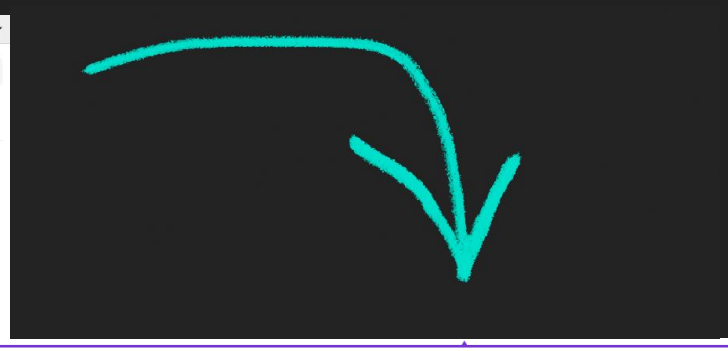
Допустить ошибку крайне легко, а функциональность важная

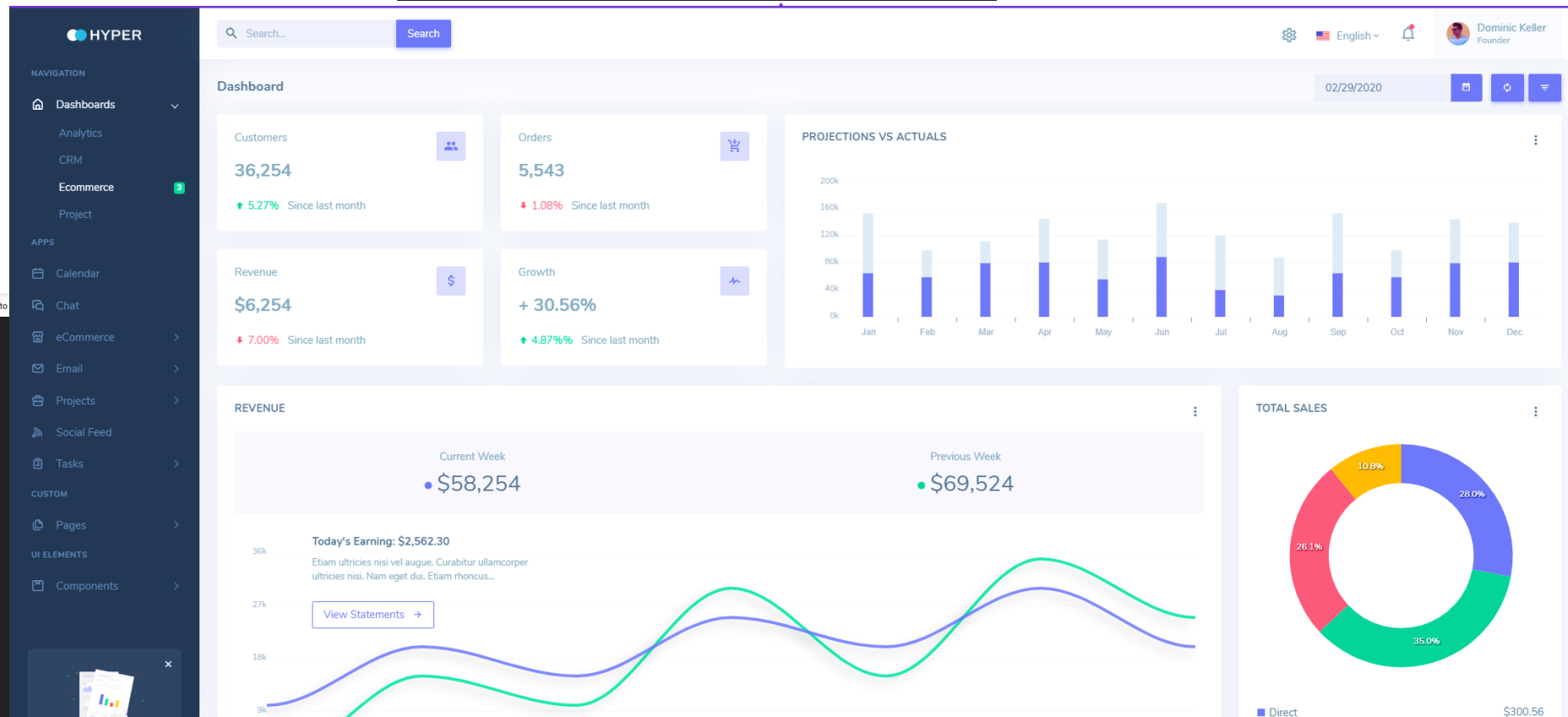
FIX: тщательно тестировать матрицу ролей

Сложности с панелями управления

Сложности с панелями управления







Сложности с панелями управления

Разрабатывать собственные админки — непросто [1]

IDOR — Insecure Direct Object Reference

Возникает, если к объекту можно получить доступ, зная лишь его идентификатор (без проверки прав доступа)

```
GET/dashboard/project/instances/c3ca8ff3-bf1f-4aac-a80f-e864d40c257c/ HTTP/1.1
Host: openstack-dev.test
User-Agent: Mozilla/5.0
Cookie: sessionid=c0e3df4c1e804d90929e0d15da50ffe2
```

```
DELETE /api/v1/users/3542244423 HTTP/1.1
Host: openstack-dev.test
User-Agent: Mozilla/5.0
Cookie: sessionid=c0e3df4c1e804d90929e0d15da50ffe2
```

Сложности с панелями управления

Разрабатывать собственные админки — непросто [1]

IDOR — Insecure Direct Object Reference

Возникает, если к объекту можно получить доступ, зная лишь его идентификатор (без проверки прав доступа)

Итог: смогли удалять «админов» в чужих облаках

FIX: методичная проверка всех API методов

Сложности с панелями управления

Разрабатывать собственные админки — непросто [2]

2FA — многофакторная аутентификация

Отлично защищает от различных атак на «клиента»

ПЛАН

1. OTP действительно проверяется?
2. OTP вырабатывается безопасно с точки зрения криптографии?
3. OTP уникален для каждого пользователя?
4. Нельзя ли подбирать OTP методом грубой силы 0000-9999?

Не забыть предусмотреть во всех местах с 2fa

Сложности с панелями управления

Разрабатывать собственные админки — непросто [2]

2FA — многофакторная аутентификация

Отлично защищает от различных атак на «клиента»

ПЛАН

1. OTP действительно проверяется?
2. OTP вырабатывается безопасно с точки зрения криптографии?
3. OTP уникален для каждого пользователя?
4. Нельзя ли подбирать OTP методом грубой силы 0000-9999?

Не забыть предусмотреть во всех местах с 2fa

Сложности с панелями управления

Разрабатывать собственные админки — непросто [2]

2FA — многофакторная аутентификация

Отлично защищает от различных атак на «клиента»

ПЛАН

1. OTP действительно проверяется?

Итог: можем отключить 2FA, заманив админа к себе на страницу

FIX: С осторожностью использовать собственные реализации

Сложности с панелями управления

Разрабатывать собственные админки — непросто [3]

Давайте сделаем собственную реализацию защиты от CSRF атак!

1. Пользователь приходит на нашу API с сессионными куками
   Cookie: user_id=i.petrov; SessionID=7815696ecbf1c96e6894b779456d330e
2. Извлечем из Cookies user_id
3. Пусть token = sha512(user_id+app_secret) + timestamp
4. Для проверки повторим эту операцию, проверим, что timestamp отличается не более, чем на 24 часа

Сложности с панелями управления

Разрабатывать собственные админки — непросто [3]

Давайте сделаем собственную реализацию защиты от CSRF атак!

1. Пользователь приходит на нашу API с сессионными куками
   Cookie: user_id=i.petrov; SessionID=7815696ecbf1c96e6894b779456d330e
2. Извлечем из Cookies user_id
3. Пусть token = sha512(user_id+app_secret) + timestamp
4. Для проверки повторим эту операцию, проверим, что timestamp отличается не более, чем на 24 часа

Итог: можно выписать токен на любого пользователя

FIX: С осторожностью использовать собственные реализации

«Рассинхронизация» панелей управления

Рассинхронизация систем

Создали новую админку со «своей» логикой?

–Не забудьте про уже имеющиеся в Openstack!

–Внедрили крутой «антифрод» в основной админке сервиса?

-А что если фродер просто пойдет и авторизуется в Keystone+Horizon, минуя основную панель?

Рассинхронизация систем

Создали новую админку со «своей» логикой?

–Не забудьте про уже имеющиеся в Openstack!

–Внедрили крутой «антифрод» в основной админке сервиса?

Итог: смогли обходить внутренние блокировки и лимиты сервиса

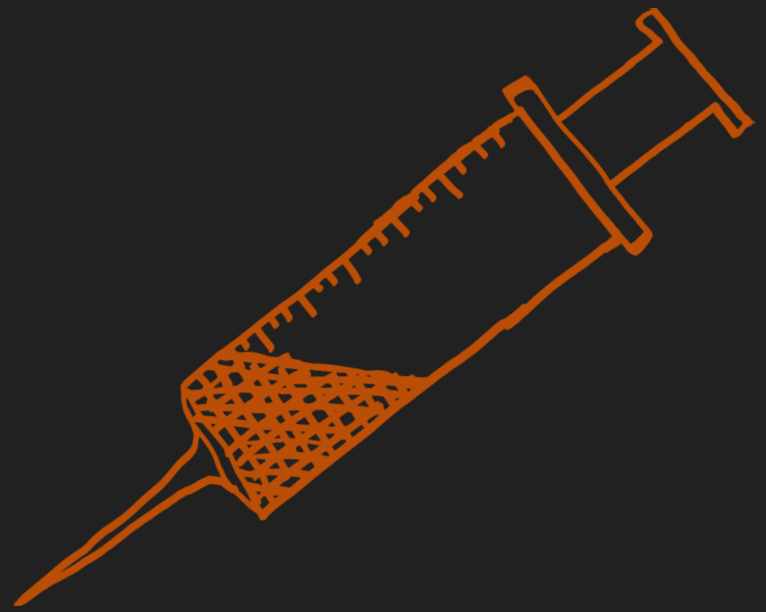
FIX: Ограничивать прямой доступ к модулям Openstack

Валидация всех данных

Валидация всех данных

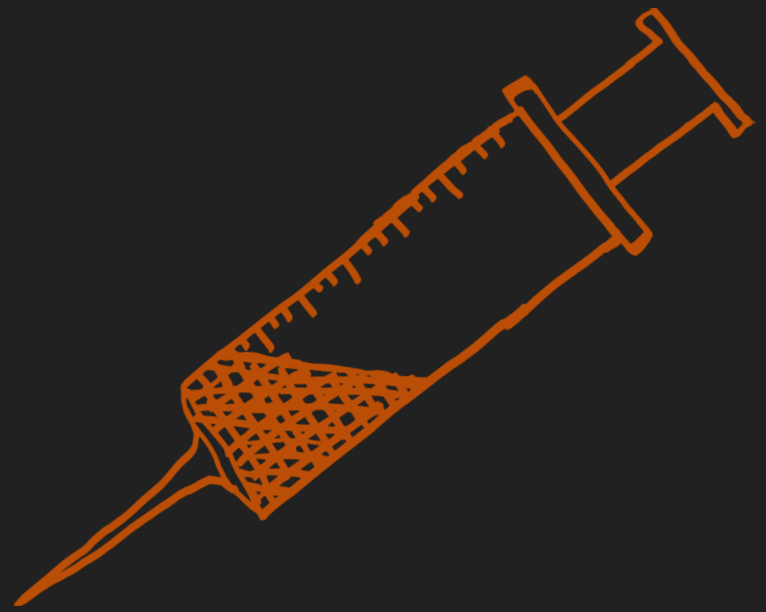
Куда злоумышленник может внедрить вредоносный код?

- OS Command injection
- SQL statements injection
- HTML injection
- Javascript injection
- XML injections
- XPATH injections
- LDAP injections
- CSS injections



Валидация всех данных

— В курсе про «инъекции» и фильтруете все приходящие от пользователя данные?



Валидация всех данных

— **В курсе про «инъекции» и фильтруете все приходящие от пользователя данные?**

— А если данные пришли от модуля “Openstack” ? Нет, потому что они «доверенные»?

Валидация всех данных

— **В курсе про «инъекции» и фильтруете все приходящие от пользователя данные?**

— А если данные пришли от модуля “Openstack” ? Нет, потому что они «доверенные»?

Cloud Panel

Openstack Module

```
POST /cloud/create?type=server
Host: my.cloud

name=“><script>evil()></script>
```

Валидация всех данных

— **В курсе про «инъекции» и фильтруете все приходящие от пользователя данные?**

— А если данные пришли от модуля “Openstack” ? Нет, потому что они «доверенные»?

Cloud Panel

Openstack Module

Валидация всех данных

— **В курсе про «инъекции» и фильтруете все приходящие от пользователя данные?**

— А если данные пришли от модуля “Openstack” ? Нет, потому что они «доверенные»?

Cloud Panel

Openstack Module

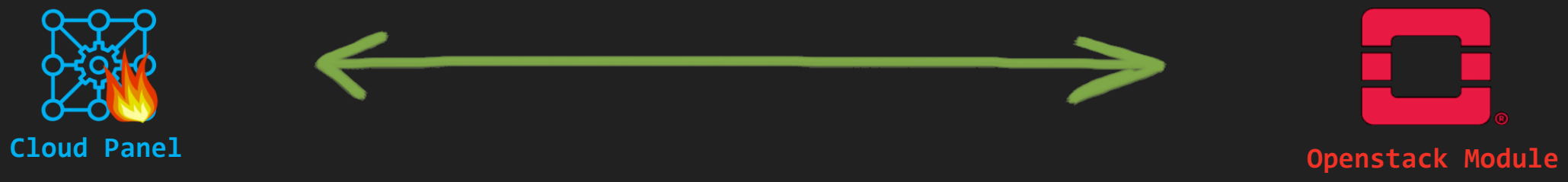
```
POST /openstack/int-api
Host: nova:8086

{
  “action”:”create”,
  ”name”:”’><script>evil()</script>”
}
```

Валидация всех данных

— **В курсе про «инъекции» и фильтруете все приходящие от пользователя данные?**

— А если данные пришли от модуля “Openstack” ? Нет, потому что они «доверенные»?



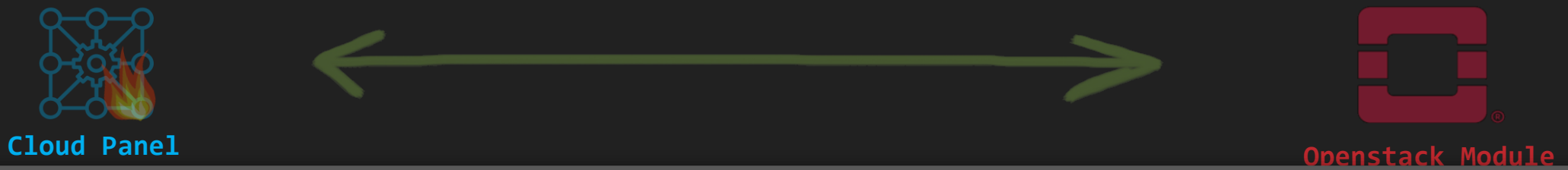
```
POST /openstack/int-api
Host: nova:8086

{
  “action”:”create”,
  ”name”:”’><script>evil()</script>”
}
```

Валидация всех данных

— **В курсе про «инъекции» и фильтруете все приходящие от пользователя данные?**

— А если данные пришли от модуля “Openstack” ? Нет, потому что они «доверенные»?



Итог: доставили «инъекцию» через «доверенный» модуль

FIX: Стараться валидировать данные из всех источников

Рекомендации и выводы

- Следить за **актуальными уязвимостями** в технологии в официальных источниках
- Следить за безопасностью не только во внешнем, но и во **внутреннем периметре**
- Ряд решений по обеспечению безопасности остается **на совести провайдера** облака, а не разработчика
- **Разработка админки — сложная задача.**
  Тщательно тестируем, а потом еще раз. Делаем регулярные ретесты
- **Дефолтные** конфигурации ПО могут быть **далеки от безопасности**