OWASP Top 10 for Rails Developers

OWASP Top 10

Top 10 most common vulnerabilities found in web applications
Last updated in 2021
Next update comes out next year

Server-Side Request Forgery

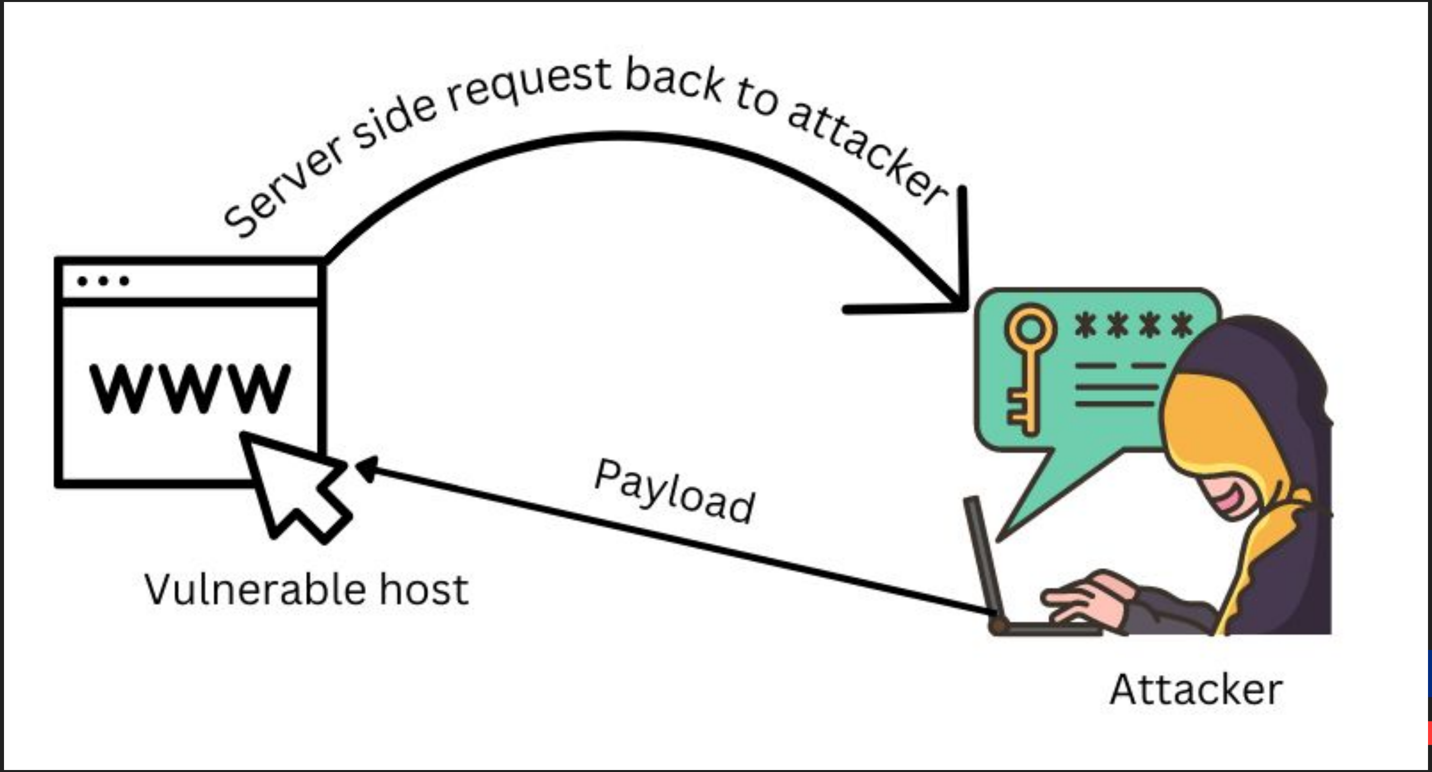
Malicious requests from a vulnerable server to **internal** or external **resources**

Can be used for:

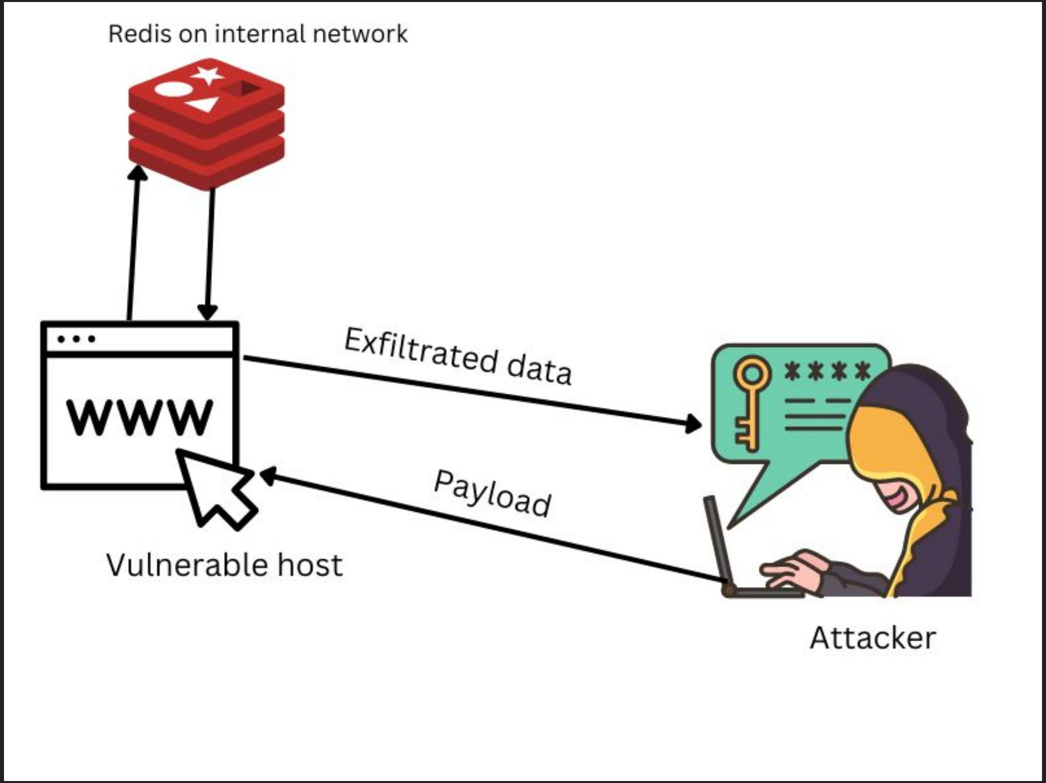
accessing restricted resources

bypass firewalls and security controls

External SSRF



Internal SSRF



SSRF

Webhooks

SSRF Examples

Webhooks
Unfurling URLs

Prevention and Mitigation

- Validate user supplied data before using it for for requests
- Network segmentation
- Don’t send raw responses to clients(this makes data exfiltration more difficult)
- Monitor logs for suspicious activities

Security Logging and Monitoring Failures

Security Logging and Monitoring Failures

Log suspicious activities

Security Logging and Monitoring Failures

Log suspicious activities
Monitor your logs

Security Logging and Monitoring Failures

Log suspicious activities
Monitor your logs
Don’t log sensitive information

Security Logging and Monitoring Failures

Log suspicious activities
Monitor your logs
Don’t log sensitive information

```
# config/initializers/filter_parameter_logging.rb
Rails.application.config.filter_parameters += [
  :passw, :email, :secret, :token, :_key, :crypt, :salt, :certificate, :otp, :ssn, :cvv, :cvc
]
```

Identification and Authentication Failures

Identification and Authentication Failures

Minimal viable authentication

Identification and Authentication Failures

Strong password validations. Complexity, and leaked passwords.

Identification and Authentication Failures

Minimal viable authentication:

Strong password validations. Complexity, and leaked passwords.
MFA for at least the password reset if you have email based one.

Identification and Authentication Failures

Minimal viable authentication:

Strong password validations. Complexity, and leaked passwords.
MFA for at least the password reset if you have email based one.
Generic message at authentication failures.

Identification and Authentication Failures

Minimal viable authentication:

Strong password validations. Complexity, and leaked passwords.
MFA for at least the password reset if you have email based one.
Generic message at authentication failures.
Rate-limiting against credential stuffing and brute force attacks

Identification and Authentication Failures

Minimal viable authentication:

Strong password validations. Complexity, and leaked passwords.
MFA for at least the password reset if you have email based one.
Generic message at authentication failures.
Rate-limiting against credential stuffing and brute force attacks
Strong cryptography, prevent timing attacks

Identification and Authentication Failures

How can you achieve these in Rails?

Identification and Authentication Failures

ActiveModel::Validations

```
validate :password_complexity

def password_complexity
  if password.present? and !password.match(/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).{12,}$/)
    errors.add :password, "must include at least one lowercase letter, one uppercase letter, one
digit, and needs to be minimum 12 characters."
  end
end
```

Identification and Authentication Failures

pwned gem

```ruby
class User < ApplicationRecord
  validates :password, not_pwned: true
end
```

Identification and Authentication Failures

devise-two-factor or rotp gem
don’t forget password resets

Identification and Authentication Failures

has_secure_password, has_secure_token, etc

Identification and Authentication Failures

authenticate_by and find_by_token prevents timing attacks

Identification and Authentication Failures

Rails 7.2 has a built-in rate-limiter

```
rate_limit to: 10, within: 3.minutes, only: :create
```

Identification and Authentication Failures

Groups requests by the IP by default

```
rate_limit to: 10, within: 3.minutes, only: :create, by: -> { request.remote_ip }
```

Identification and Authentication Failures

Response can be changed

```
rate_limit to: 10, within: 3.minutes, only: :create, with: -> { redirect_to
root_url, alert: 'Slow your horses!'}
```

ActionController::RateLimiting

Storage

```
rate_limit to: 10, within: 3.minutes, only: :create, store:
ActiveSupport::Cache::RedisCacheStore.new(url: ENV["REDIS_URL"])
```

Identification and Authentication Failures

rack-attack

Vulnerable and Outdated Components

You should periodically check for vulnerable dependencies

Vulnerable and Outdated Components

You should periodically check for vulnerable dependencies
Dependabot

Vulnerable and Outdated Components

You should periodically check for vulnerable dependencies
Dependabot
Bundler Audit

Security Misconfiguration

Don't enable development features in production

Security Misconfiguration

Don't enable development features in production
Lack of authentication on Sidekiq UI, Mission Control

Insecure Design

Think about every feature from a security perspective too
Make sure you cover the unhappy paths

Injection

Injection

calculate

Injection

calculate

average
count
maximum
minimum
sum

Injection

```
LineItem.sum(params[:total_by])
```

Injection

delete_by, destroy_by
exists?
find_by, find_by!
from
group, having, joins
lock, not
select, reselect
where, rewhere
update_all

Injection

https://rails-sqli.org/

Injection

Second order SQL Injection

Injection

```
class ReportsController < ApplicationController
  def create
    @report = Report.new(report_params)
    if @report.save
      redirect_to reports_path
    else
      render :new
    end
  end

  private
    def report_params
      params.require(:report).permit(:group, :columns)
    end
end
```

Injection

```
def show
  @report = Report.find(params[:id])
  @result = Order.select(@report.columns).group(@report.group)
end
```

Cryptographic Failures

Use Active Record Encryption

Cryptographic Failures

Use Active Record Encryption
has_secure_password, has_secure_token

Broken Access Control

Broken Access Control

Strong authorization

Broken Access Control

Strong authorization
Whitelist approach

Broken Access Control

Strong authorization
Whitelist approach
UUIDs are not equal to authorization

Broken Access Control

Strong authorization
Whitelist approach
UUIDs are not equal to authorization
Foreign keys

Thank you

You can follow me on Twitter @gregmolnar
And subscribe to This Week in Rails!