Advisories

161

VULNERABLE GEM: ACTIONMAILER@4.2.10

Name:
actionmailer

Version:
4.2.10

ID:
CVE-2024-47889

LINK

Possible ReDoS vulnerability in block_format in Action Mailer

DESCRIPTION:

There is a possible ReDoS vulnerability in the block_format helper in Action Mailer. This vulnerability has been assigned the CVE identifier CVE-2024-47889.

IMPACT

Carefully crafted text can cause the block_format helper to take an unexpected amount of time, possibly resulting in a DoS vulnerability. All users running an affected release should either upgrade or apply the relevant patch immediately.

Ruby 3.2 has mitigations for this problem, so Rails applications using Ruby 3.2 or newer are unaffected. Rails 8.0.0.beta1 requires Ruby 3.2 or greater so is unaffected.

RELEASES

The fixed releases are available at the normal locations.

WORKAROUNDS

Users can avoid calling the `block_format` helper or upgrade to Ruby 3.2.

CREDITS

Thanks to ooooooo_q for the report!

VULNERABLE GEM: ACTIONPACK@4.2.10

Name:
actionpack

Version:
4.2.10

ID:
CVE-2020-8164

LINK

Possible Strong Parameters Bypass in ActionPack

DESCRIPTION:

There is a strong parameters bypass vector in ActionPack.
Versions Affected: rails <= 6.0.3 Not affected: rails < 4.0.0 Fixed Versions: rails >= 5.2.4.3, rails >= 6.0.3.1

IMPACT

In some cases user supplied information can be inadvertently leaked from Strong Parameters. Specifically the return value of `each`, or `each_value`, or `each_pair` will return the underlying "untrusted" hash of data that was read from the parameters. Applications that use this return value may be inadvertently use untrusted user input.
Impacted code will look something like this:

```
 def update
 # Attacker has included the parameter: `{ is_admin: true }`
 User.update(clean_up_params)
end

def clean_up_params
  params.each { |k, v|  SomeModel.check(v) if k == :name }
end
```

Note the mistaken use of `each` in the `clean_up_params` method in the above example.

WORKAROUNDS

Do not use the return values of `each`, `each_value`, or `each_pair` in your application.

VULNERABLE GEM: ACTIONPACK@4.2.10

Name:
actionpack

Version:
4.2.10

ID:
CVE-2020-8166

LINK

Ability to forge per-form CSRF tokens given a global CSRF token

DESCRIPTION:

It is possible to possible to, given a global CSRF token such as the one present in the authenticity_token meta tag, forge a per-form CSRF token for any action for that session.
Versions Affected: rails < 5.2.5, rails < 6.0.4 Not affected: Applications without existing HTML injection vulnerabilities. Fixed Versions: rails >= 5.2.4.3, rails >= 6.0.3.1

IMPACT

Given the ability to extract the global CSRF token, an attacker would be able to construct a per-form CSRF token for that session.

WORKAROUNDS

This is a low-severity security issue. As such, no workaround is necessarily until such time as the application can be upgraded.

VULNERABLE GEM: ACTIONPACK@4.2.10

Name:
actionpack

Version:
4.2.10

ID:
CVE-2021-22885

LINK

Possible Information Disclosure / Unintended Method Execution in Action Pack

DESCRIPTION:

There is a possible information disclosure / unintended method execution vulnerability in Action Pack which has been assigned the CVE identifier CVE-2021-22885.
Versions Affected: >= 2.0.0. Not affected: < 2.0.0. Fixed Versions: 6.1.3.2, 6.0.3.7, 5.2.4.6, 5.2.6

IMPACT

There is a possible information disclosure / unintended method execution vulnerability in Action Pack when using the `redirect_to` or `polymorphic_url` helper with untrusted user input.
Vulnerable code will look like this:

```
redirect_to(params[:some_param])
```

All users running an affected release should either upgrade or use one of the workarounds immediately.

WORKAROUNDS

To work around this problem, it is recommended to use an allow list for valid parameters passed from the user. For example:

```
private def check(param)
case param
when "valid"
  param
else
  "/"
end
end

def index
redirect_to(check(params[:some_param]))
end
```

Or force the user input to be cast to a string like this:

```
def index
redirect_to(params[:some_param].to_s)
end
```

VULNERABLE GEM: ACTIONPACK@4.2.10

Name:
actionpack

Version:
4.2.10

ID:
CVE-2021-22904

LINK

Possible DoS Vulnerability in Action Controller Token Authentication

DESCRIPTION:

There is a possible DoS vulnerability in the Token Authentication logic in Action Controller. This vulnerability has been assigned the CVE identifier CVE-2021-22904.
Versions Affected: >= 4.0.0 Not affected: < 4.0.0 Fixed Versions: 6.1.3.2, 6.0.3.7, 5.2.4.6, 5.2.6

IMPACT

Impacted code uses `authenticate_or_request_with_http_token` or `authenticate_with_http_token` for request authentication. Impacted code will look something like this:

```
class PostsController < ApplicationController
before_action :authenticate

private

def authenticate
  authenticate_or_request_with_http_token do |token, options|
    # ...
  end
end
end
```

All users running an affected release should either upgrade or use one of the workarounds immediately.

RELEASES

The fixed releases are available at the normal locations.

WORKAROUNDS

The following monkey patch placed in an initializer can be used to work around the issue:

```
 module ActionController::HttpAuthentication::Token
 AUTHN_PAIR_DELIMITERS = /(?:,|;|\t)/
end
```

VULNERABLE GEM: ACTIONPACK@4.2.10

Name:
actionpack

Version:
4.2.10

ID:
CVE-2023-22792

LINK

ReDoS based DoS vulnerability in Action Dispatch

DESCRIPTION:

There is a possible regular expression based DoS vulnerability in Action Dispatch. This vulnerability has been assigned the CVE identifier CVE-2023-22792.

Versions Affected: >= 3.0.0 Not affected: < 3.0.0 Fixed Versions: 6.1.7.1, 7.0.4.1

Impact

Specially crafted cookies, in combination with a specially crafted X*FORWARDED*HOST header can cause the regular expression engine to enter a state of catastrophic backtracking. This can cause the process to use large amounts of CPU and memory, leading to a possible DoS vulnerability All users running an affected release should either upgrade or use one of the workarounds immediately.

Workarounds

We recommend that all users upgrade to one of the FIXED versions. In the meantime, users can mitigate this vulnerability by using a load balancer or other device to filter out malicious X*FORWARDED*HOST headers before they reach the application.

VULNERABLE GEM: ACTIONPACK@4.2.10

Name:
actionpack

Version:
4.2.10

ID:
CVE-2023-22795

LINK

ReDoS based DoS vulnerability in Action Dispatch

DESCRIPTION:

There is a possible regular expression based DoS vulnerability in Action Dispatch related to the If-None-Match header. This vulnerability has been assigned the CVE identifier CVE-2023-22795.
Versions Affected: All Not affected: None Fixed Versions: 6.1.7.1, 7.0.4.1

Impact

A specially crafted HTTP If-None-Match header can cause the regular expression engine to enter a state of catastrophic backtracking, when on a version of Ruby below 3.2.0. This can cause the process to use large amounts of CPU and memory, leading to a possible DoS vulnerability All users running an affected release should either upgrade or use one of the workarounds immediately.

Workarounds

We recommend that all users upgrade to one of the FIXED versions. In the meantime, users can mitigate this vulnerability by using a load balancer or other device to filter out malicious If-None-Match headers before they reach the application.
Users on Ruby 3.2.0 or greater are not affected by this vulnerability.

VULNERABLE GEM: ACTIONPACK@4.2.10

Name:
actionpack

Version:
4.2.10

ID:
CVE-2023-28362

LINK

Possible XSS via User Supplied Values to redirect_to

DESCRIPTION:
The redirect_to method in Rails allows provided values to contain characters which are not legal in an HTTP header value. This results in the potential for downstream services which enforce RFC compliance on HTTP response headers to remove the assigned Location header. This vulnerability has been assigned the CVE identifier CVE-2023-28362.
Versions Affected: All. Not affected: None Fixed Versions: 7.0.5.1, 6.1.7.4

Impact

This introduces the potential for a Cross-site-scripting (XSS) payload to be delivered on the now static redirection page. Note that this both requires user interaction and for a Rails app to be configured to allow redirects to external hosts (defaults to false in Rails >= 7.0.x).

Releases

The FIXED releases are available at the normal locations.

Workarounds

Avoid providing user supplied URLs with arbitrary schemes to the redirect_to method.

VULNERABLE GEM: ACTIONPACK@4.2.10

Name:
actionpack

Version:
4.2.10

ID:
CVE-2024-41128

LINK

Possible ReDoS vulnerability in query parameter filtering in Action Dispatch

DESCRIPTION:
There is a possible ReDoS vulnerability in the query parameter filtering routines of Action Dispatch. This vulnerability has been assigned the CVE identifier CVE-2024-41128.

IMPACT
Carefully crafted query parameters can cause query parameter filtering to take an unexpected amount of time, possibly resulting in a DoS vulnerability. All users running an affected release should either upgrade or apply the relevant patch immediately.
Ruby 3.2 has mitigations for this problem, so Rails applications using Ruby 3.2 or newer are unaffected. Rails 8.0.0.beta1 depends on Ruby 3.2 or greater so is unaffected.

RELEASES
The fixed releases are available at the normal locations.

WORKAROUNDS
Users on Ruby 3.2 are unaffected by this issue.

CREDITS
Thanks to scyoon for the report and patches!

VULNERABLE GEM: ACTIONPACK@4.2.10

Name:
actionpack

Version:
4.2.10

ID:
CVE-2024-47887

LINK

Possible ReDoS vulnerability in HTTP Token authentication in Action Controller

DESCRIPTION:
There is a possible ReDoS vulnerability in Action Controller's HTTP Token authentication. This vulnerability has been assigned the CVE identifier CVE-2024-47887.
IMPACT
For applications using HTTP Token authentication via `authenticate_or_request_with_http_token` or similar, a carefully crafted header may cause header parsing to take an unexpected amount of time, possibly resulting in a DoS vulnerability. All users running an affected release should either upgrade or apply the relevant patch immediately. Ruby 3.2 has mitigations for this problem, so Rails applications using Ruby 3.2 or newer are unaffected. Rails 8.0.0.beta1 depends on Ruby 3.2 or greater so is unaffected.
RELEASES
The fixed releases are available at the normal locations.
WORKAROUNDS
Users on Ruby 3.2 are unaffected by this issue.
CREDITS
Thanks to scyoon for reporting

VULNERABLE GEM: ACTIONVIEW@4.2.10

Name:
actionview

Version:
4.2.10

ID:
CVE-2019-5418

LINK

File Content Disclosure in Action View

DESCRIPTION:
There is a possible file content disclosure vulnerability in Action View. This vulnerability has been assigned the CVE identifier CVE-2019-5418. Versions Affected: All. Not affected: None. Fixed Versions: 6.0.0.beta3, 5.2.2.1, 5.1.6.2, 5.0.7.2, 4.2.11.1
IMPACT
There is a possible file content disclosure vulnerability in Action View. Specially crafted accept headers in combination with calls to `render file:`

can cause arbitrary files on the target server to be rendered, disclosing the file contents.
The impact is limited to calls to `render` which render file contents without a specified accept format. Impacted code in a controller looks something like this:

```
class UserController < ApplicationController
 def index
   render file: "#{Rails.root}/some/file"
 end
end
```

Rendering templates as opposed to files is not impacted by this vulnerability.
All users running an affected release should either upgrade or use one of the workarounds immediately.

RELEASES

The 6.0.0.beta3, 5.2.2.1, 5.1.6.2, 5.0.7.2, and 4.2.11.1 releases are available at the normal locations.

WORKAROUNDS

This vulnerability can be mitigated by specifying a format for file rendering, like this:

```
class UserController < ApplicationController
 def index
   render file: "#{Rails.root}/some/file", formats: [:html]
 end
end
```

In summary, impacted calls to `render` look like this:

```
render file: "#{Rails.root}/some/file"
```

The vulnerability can be mitigated by changing to this:

```
render file: "#{Rails.root}/some/file", formats: [:html]
```

Other calls to `render` are not impacted.
Alternatively, the following monkey patch can be applied in an initializer:

```
$ cat config/initializers/formats_filter.rb
# frozen_string_literal: true

ActionDispatch::Request.prepend(Module.new do
 def formats
   super().select do |format|
     format.symbol || format.ref == "*/*"
   end
 end
end)
```

CREDITS

Thanks to John Hawthorn john@hawthorn.email of GitHub

VULNERABLE GEM: ACTIONVIEW@4.2.10

Name:
actionview

Version:
4.2.10

ID:
CVE-2019-5419

LINK

Denial of Service Vulnerability in Action View

DESCRIPTION:

There is a potential denial of service vulnerability in actionview. This vulnerability has been assigned the CVE identifier CVE-2019-5419.

IMPACT

Specially crafted accept headers can cause the Action View template location code to consume 100% CPU, causing the server unable to process requests. This impacts all Rails applications that render views.
All users running an affected release should either upgrade or use one of the workarounds immediately.

WORKAROUNDS

This vulnerability can be mitigated by wrapping `render` calls with `respond_to` blocks. For example, the following example is vulnerable:

```
class UserController < ApplicationController
def index
  render "index"
end
end
```

But the following code is not vulnerable:

```
class UserController < ApplicationController
 def index
   respond_to |format|
     format.html { render "index" }
   end
 end
end
```

Implicit rendering is impacted, so this code is vulnerable:

```
class UserController < ApplicationController
 def index
 end
end
```

But can be changed this this:

```
class UserController < ApplicationController
 def index
   respond_to |format|
     format.html { render "index" }
   end
 end
end
```

Alternatively to specifying the format, the following monkey patch can be applied in an initializer:

```
 $ cat config/initializers/formats_filter.rb
# frozen_string_literal: true

ActionDispatch::Request.prepend(Module.new do
 def formats
   super().select do |format|
     format.symbol || format.ref == "*/*"
   end
 end
end)
```

CREDITS

Thanks to John Hawthorn john@hawthorn.email of GitHub

VULNERABLE GEM: ACTIONVIEW@4.2.10

Name:
actionview

Version:
4.2.10

ID:
CVE-2020-15169

LINK

Potential XSS vulnerability in Action View

DESCRIPTION:

There is a potential Cross-Site Scripting (XSS) vulnerability in Action View's translation helpers. Views that allow the user to control the default (not found) value of the `t` and `translate` helpers could be susceptible to XSS attacks.

IMPACT

When an HTML-unsafe string is passed as the default for a missing translation key named `html` or ending in `_html`, the default string is incorrectly marked as HTML-safe and not escaped. Vulnerable code may look like the following examples:

```
<%# The welcome_html translation is not defined for the current locale: %>
<%= t("welcome_html", default: untrusted_user_controlled_string) %>

<%# Neither the title.html translation nor the missing.html translation is defined for the current locale: %>
<%= t("title.html", default: [:"missing.html", untrusted_user_controlled_string]) %>
```

WORKAROUNDS

Impacted users who canâ€™t upgrade to a patched Rails version can avoid this issue by manually escaping default translations with the `html_escape` helper (aliased as `h`):

```
<%= t("welcome_html", default: h(untrusted_user_controlled_string)) %>
```

VULNERABLE GEM: ACTIONVIEW@4.2.10

Name:
actionview

Version:
4.2.10

ID:
CVE-2020-5267

LINK

Possible XSS vulnerability in ActionView

DESCRIPTION:

There is a possible XSS vulnerability in ActionView's JavaScript literal escape helpers. Views that use the `j` or `escape_javascript` methods may be susceptible to XSS attacks.
Versions Affected: All. Not affected: None. Fixed Versions: 6.0.2.2, 5.2.4.2

IMPACT

There is a possible XSS vulnerability in the `j` and `escape_javascript` methods in ActionView. These methods are used for escaping JavaScript string literals. Impacted code will look something like this:

```
<script>let a = `<%= j unknown_input %>`</script>
```

or

```
<script>let a = `<%= escape_javascript unknown_input %>`</scri
pt>
```

RELEASES

The 6.0.2.2 and 5.2.4.2 releases are available at the normal locations.

WORKAROUNDS

For those that can't upgrade, the following monkey patch may be used:

```
  ActionView::Helpers::JavaScriptHelper::JS_ESCAPE_MAP.merg
e!(
  {
    "`" => "\\`",
    "$" => "\\$"
  }
)

module ActionView::Helpers::JavaScriptHelper
  alias :old_ej :escape_javascript
  alias :old_j :j

  def escape_javascript(javascript)
    javascript = javascript.to_s
    if javascript.empty?
      result = ""
    else
      result = javascript.gsub(/(\\|<\/|\r\n|\342\200\250|\342\200\251|[\
n\r"']|[`]|[$])/u, JS_ESCAPE_MAP)
    end
    javascript.html_safe? ? result.html_safe : result
  end

  alias :j :escape_javascript
end
```

VULNERABLE GEM: ACTIONVIEW@4.2.10

Name:
actionview

Version:
4.2.10

ID:
CVE-2020-8163

LINK

Potential remote code execution of user-provided local names in ActionView

DESCRIPTION:
There was a vulnerability in versions of Rails prior to 5.0.1 that would allow an attacker who controlled the `locals` argument of a `render` call.
Versions Affected: rails < 5.0.1 Not affected: Applications that do not allow users to control the names of locals. Fixed Versions: 4.2.11.2
IMPACT
In the scenario where an attacker might be able to control the name of a local passed into `render`, they can acheive remote code execution.
WORKAROUNDS
Until such time as the patch can be applied, application developers should ensure that all user-provided local names are alphanumeric.

VULNERABLE GEM: ACTIONVIEW@4.2.10

Name:
actionview

Version:
4.2.10

ID:
CVE-2020-8167

LINK

CSRF Vulnerability in rails-ujs

DESCRIPTION:
There is an vulnerability in rails-ujs that allows attackers to send CSRF tokens to wrong domains.
Versions Affected: rails <= 6.0.3 Not affected: Applications which don't use rails-ujs. Fixed Versions: rails >= 5.2.4.3, rails >= 6.0.3.1
IMPACT
This is a regression of CVE-2015-1840.
In the scenario where an attacker might be able to control the href attribute of an anchor tag or the action attribute of a form tag that will trigger a POST action, the attacker can set the href or action to a cross-origin URL, and the CSRF token will be sent.
WORKAROUNDS

To work around this problem, change code that allows users to control the href attribute of an anchor tag or the action attribute of a form tag to filter the user parameters.
For example, code like this:

```
link_to params
```

to code like this:

```
link_to filtered_params

def filtered_params
 # Filter just the parameters that you trust
end
```

VULNERABLE GEM: ACTIONVIEW@4.2.10

Name:
actionview

Version:
4.2.10

ID:
CVE-2022-27777

LINK

Possible XSS Vulnerability in Action View tag helpers

DESCRIPTION:

There is a possible XSS vulnerability in Action View tag helpers. Passing untrusted input as hash keys can lead to a possible XSS vulnerability. This vulnerability has been assigned the CVE identifier CVE-2022-27777.
Versions Affected: ALL Not affected: NONE Fixed Versions: 7.0.2.4, 6.1.5.1, 6.0.4.8, 5.2.7.1

IMPACT

If untrusted data is passed as the hash key for tag attributes, there is a possibility that the untrusted data may not be properly escaped which can lead to an XSS vulnerability.
Impacted code will look something like this:

```
check_box_tag('thename', 'thevalue', false, aria: { malicious_input => 'thevalueofaria' })
```

Where the "malicious_input" variable contains untrusted data.
All users running an affected release should either upgrade or use one of the workarounds immediately.

RELEASES

The FIXED releases are available at the normal locations.

WORKAROUNDS

Escape the untrusted data before using it as a key for tag helper methods.

VULNERABLE GEM: ACTIONVIEW@4.2.10

Name:
actionview

Version:
4.2.10

ID:
CVE-2026-33168

LINK

Rails has a possible XSS vulnerability in its Action View tag helpers

DESCRIPTION:

Impact

When a blank string is used as an HTML attribute name in Action View tag helpers, the attribute escaping is bypassed, producing malformed HTML. A carefully crafted attribute value could then be misinterpreted by the browser as a separate attribute name, possibly leading to XSS. Applications that allow users to specify custom HTML attributes are affected.
Releases

The fixed releases are available at the normal locations.

VULNERABLE GEM: ACTIVEJOB@4.2.10

Name:
activejob

Version:
4.2.10

ID:
CVE-2018-16476

LINK

Broken Access Control vulnerability in Active Job

DESCRIPTION:

There is a vulnerability in Active Job. This vulnerability has been assigned the CVE identifier CVE-2018-16476.
Versions Affected: >= 4.2.0 Not affected: < 4.2.0 Fixed Versions: 4.2.11, 5.0.7.1, 5.1.6.1, 5.2.1.1

IMPACT

Carefully crafted user input can cause Active Job to deserialize it using GlobalId and allow an attacker to have access to information that they should not have.
Vulnerable code will look something like this:

```
MyJob.perform_later(user_input)
```

All users running an affected release should either upgrade or use one of the workarounds immediately.

VULNERABLE GEM: ACTIVERECORD@4.2.10

Name:
activerecord

Version:
4.2.10

ID:
CVE-2021-22880

LINK

Possible DoS Vulnerability in Active Record PostgreSQL adapter

DESCRIPTION:

There is a possible DoS vulnerability in the PostgreSQL adapter in Active Record. This vulnerability has been assigned the CVE identifier CVE-2021-22880.
Versions Affected: >= 4.2.0 Not affected: < 4.2.0 Fixed Versions: 6.1.2.1, 6.0.3.5, 5.2.4.5

IMPACT

Carefully crafted input can cause the input validation in the "money" type of the PostgreSQL adapter in Active Record to spend too much time in a regular expression, resulting in the potential for a DoS attack.
This only impacts Rails applications that are using PostgreSQL along with money type columns that take user input.

WORKAROUNDS

In the case a patch can't be applied, the following monkey patch can be used in an initializer:

```
module ActiveRecord
module ConnectionAdapters
  module PostgreSQL
    module OID # :nodoc:
      class Money < Type::Decimal # :nodoc:
        def cast_value(value)
          return value unless ::String === value

          value = value.sub(/^\((.+)\)$/, '-\1') # (4)
          case value
          when /^-?\D*+[\d,]+\.\d{2}$/  # (1)
            value.gsub!(/[^-\d.]/, "")
          when /^-?\D*+[\d.]+,\d{2}$/  # (2)
            value.gsub!(/[^-\d,]/, "").sub!(/,/, ".")
          end

          super(value)
        end
      end
    end
  end
end
```

VULNERABLE GEM: ACTIVERECORD@4.2.10

Name:
activerecord

Version:
4.2.10

ID:
CVE-2022-32224

LINK

Possible RCE escalation bug with Serialized Columns in Active Record

DESCRIPTION:

There is a possible escalation to RCE when using YAML serialized columns in Active Record. This vulnerability has been assigned the CVE identifier CVE-2022-32224.
Versions Affected: All. Not affected: None Fixed Versions: 7.0.3.1, 6.1.6.1, 6.0.5.1, 5.2.8.1

IMPACT

When serialized columns that use YAML (the default) are deserialized, Rails uses `YAML.unsafe_load` to convert the YAML data in to Ruby objects. If an attacker can manipulate data in the database (via means like SQL injection), then it may be possible for the attacker to escalate to an RCE.
Impacted Active Record models will look something like this:

```
class User < ApplicationRecord
serialize :options      # Vulnerable: Uses YAML for serialization
serialize :values, Array # Vulnerable: Uses YAML for serialization
serialize :values, JSON  # Not vulnerable
end
```

All users running an affected release should either upgrade or use one of the workarounds immediately.

RELEASES

The FIXED releases are available at the normal locations.
The released versions change the default YAML deserializer to use `YAML.safe_load`, which prevents deserialization of possibly dangerous objects. This may introduce backwards compatibility issues with existing data.

In order to cope with that situation, the released version also contains two new Active Record configuration options. The configuration options are as follows:

config.active_record.use_yaml_unsafe_load

When set to true, this configuration option tells Rails to use the old "unsafe" YAML loading strategy, maintaining the existing behavior but leaving the possible escalation vulnerability in place. Setting this option to true is *not* recommended, but can aid in upgrading.

config.active_record.yaml_column_permitted_classes

The "safe YAML" loading method does not allow all classes to be deserialized by default. This option allows you to specify classes deemed "safe" in your application. For example, if your application uses Symbol and Time in serialized data, you can add Symbol and Time to the allowed list as follows:

```
config.active_record.yaml_column_permitted_classes = [Symbol, Date, Time]
```

WORKAROUNDS

There are no feasible workarounds for this issue, but other coders (such as JSON) are not impacted.

VULNERABLE GEM: ACTIVERECORD@4.2.10

Name:
activerecord

Version:
4.2.10

ID:
CVE-2022-44566

LINK

Denial of Service Vulnerability in ActiveRecordâ€™s PostgreSQL adapter

DESCRIPTION:

There is a potential denial of service vulnerability present in

ActiveRecordâ€™s PostgreSQL adapter.
This has been assigned the CVE identifier CVE-2022-44566.
Versions Affected: All. Not affected: None. Fixed Versions: 6.1.7.1, 7.0.4.1

Impact

In ActiveRecord <7.0.4.1 and <6.1.7.1, when a value outside the range for a 64bit signed integer is provided to the PostgreSQL connection adapter, it will treat the target column type as numeric. Comparing integer values against numeric values can result in a slow sequential scan resulting in potential Denial of Service.

Workarounds

Ensure that user supplied input which is provided to ActiveRecord clauses do not contain integers wider than a signed 64bit representation or floats.

VULNERABLE GEM: ACTIVERECORD@4.2.10

Name:
activerecord

Version:
4.2.10

ID:
CVE-2025-55193

LINK

Active Record logging vulnerable to ANSI escape injection

DESCRIPTION:
This vulnerability has been assigned the CVE identifier CVE-2025-55193
Impact

The ID passed to `find` or similar methods may be logged without escaping. If this is directly to the terminal, it may include unescaped ANSI sequences.
Releases

The fixed releases are available at the normal locations.
Credits

Thanks to lio346 for reporting this vulnerability.

VULNERABLE GEM: ACTIVESUPPORT@4.2.10

Name:
activesupport

Version:
4.2.10

ID:
CVE-2020-8165

LINK

Potentially unintended unmarshalling of user-provided objects in MemCacheStore and RedisCacheStore

DESCRIPTION:

There is potentially unexpected behaviour in the MemCacheStore and RedisCacheStore where, when untrusted user input is written to the cache store using the `raw: true` parameter, re-reading the result from the cache can evaluate the user input as a Marshalled object instead of plain text. Vulnerable code looks like:

```
data = cache.fetch("demo", raw: true) { untrusted_string }
```

Versions Affected: rails < 5.2.5, rails < 6.0.4 Not affected: Applications not using MemCacheStore or RedisCacheStore. Applications that do not use the `raw` option when storing untrusted user input. Fixed Versions: rails >= 5.2.4.3, rails >= 6.0.3.1

IMPACT

Unmarshalling of untrusted user input can have impact up to and including RCE. At a minimum, this vulnerability allows an attacker to inject untrusted Ruby objects into a web application.

In addition to upgrading to the latest versions of Rails, developers should ensure that whenever they are calling `Rails.cache.fetch` they are using consistent values of the `raw` parameter for both reading and writing, especially in the case of the RedisCacheStore which does not, prior to these changes, detect if data was serialized using the raw option upon deserialization.

WORKAROUNDS

It is recommended that application developers apply the suggested patch or upgrade to the latest release as soon as possible. If this is not possible, we recommend ensuring that all user-provided strings cached using the `raw`

argument should be double-checked to ensure that they conform to the expected format.

VULNERABLE GEM: ACTIVESUPPORT@4.2.10

Name:
activesupport

Version:
4.2.10

ID:
CVE-2023-22796

LINK

ReDoS based DoS vulnerability in Active Supportâ€™s underscore

DESCRIPTION:
There is a possible regular expression based DoS vulnerability in Active Support. This vulnerability has been assigned the CVE identifier CVE-2023-22796.
Versions Affected: All Not affected: None Fixed Versions: 6.1.7.1, 7.0.4.1

Impact

A specially crafted string passed to the underscore method can cause the regular expression engine to enter a state of catastrophic backtracking. This can cause the process to use large amounts of CPU and memory, leading to a possible DoS vulnerability.
This affects String#underscore, ActiveSupport::Inflector.underscore, String#titleize, and any other methods using these.
All users running an affected release should either upgrade or use one of the workarounds immediately.

Workarounds

There are no feasible workarounds for this issue.
Users on Ruby 3.2.0 or greater may be able to reduce the impact by configuring Regexp.timeout.

VULNERABLE GEM: ACTIVESUPPORT@4.2.10

Name:
activesupport

Version:
4.2.10

ID:
CVE-2023-28120

LINK

Possible XSS Security Vulnerability in SafeBuffer#bytesplice

DESCRIPTION:

There is a vulnerability in ActiveSupport if the new bytesplice method is called on a SafeBuffer with untrusted user input. This vulnerability has been assigned the CVE identifier CVE-2023-28120.

Versions Affected: All. Not affected: None Fixed Versions: 7.0.4.3, 6.1.7.3

Impact

ActiveSupport uses the SafeBuffer string subclass to tag strings as html*safe after they have been sanitized. When these strings are mutated, the tag is should be removed to mark them as no longer being html*safe.

Ruby 3.2 introduced a new bytesplice method which ActiveSupport did not yet understand to be a mutation. Users on older versions of Ruby are likely unaffected.

All users running an affected release and using bytesplice should either upgrade or use one of the workarounds immediately.

Workarounds

Avoid calling bytesplice on a SafeBuffer (html_safe) string with untrusted user input.

VULNERABLE GEM: ACTIVESUPPORT@4.2.10

Name:
activesupport

Version:
4.2.10

ID:
CVE-2026-33169

LINK

Rails Active Support has a possible ReDoS vulnerability in number_to_delimited

DESCRIPTION:
Impact

`NumberToDelimitedConverter` used a regular expression with `gsub!` to insert thousands delimiters. This could produce quadratic time complexity on long digit strings.
Releases

The fixed releases are available at the normal locations.

VULNERABLE GEM: ACTIVESUPPORT@4.2.10

Name:
activesupport

Version:
4.2.10

ID:
CVE-2026-33170

LINK

Rails Active Support has a possible XSS vulnerability in SafeBuffer#%

DESCRIPTION:
Impact

`SafeBuffer#%` does not propagate the `@html_unsafe` flag to the newly created buffer. If a `SafeBuffer` is mutated in place (e.g. via `gsub!`) and then formatted with `%` using untrusted arguments, the result incorrectly

reports `html_safe? == true`, bypassing ERB auto-escaping and possibly leading to XSS.
Releases

The fixed releases are available at the normal locations.

VULNERABLE GEM: ACTIVESUPPORT@4.2.10

Name:
activesupport

Version:
4.2.10

ID:
CVE-2026-33176

LINK

Rails Active Support has a possible DoS vulnerability in its number helpers

DESCRIPTION:

Impact

Active Support number helpers accept strings containing scientific notation (e.g. `1e10000`), which when converted to a string could be expanded into extremely large decimal representations. This can cause excessive memory allocation and CPU consumption when the expanded number is formatted, possibly resulting in a DoS vulnerability.
Releases

The fixed releases are available at the normal locations.

VULNERABLE GEM: ADDRESSABLE@2.5.2

Name:
addressable

Version:
2.5.2

ID:
CVE-2021-32740

LINK

Regular Expression Denial of Service in Addressable templates

DESCRIPTION:

Within the URI template implementation in Addressable, a maliciously crafted template may result in uncontrolled resource consumption, leading to denial of service when matched against a URI. In typical usage, templates would not normally be read from untrusted user input, but nonetheless, no previous security advisory for Addressable has cautioned against doing this. Users of the parsing capabilities in Addressable but not the URI template capabilities are unaffected.

VULNERABLE GEM: ADDRESSABLE@2.5.2

Name:
addressable

Version:
2.5.2

ID:
CVE-2026-35611

LINK

Addressable has a Regular Expression Denial of Service in Addressable templates

DESCRIPTION:

Impact

Within the URI template implementation in Addressable, two classes of URI template generate regular expressions vulnerable to catastrophic backtracking:

- **Templates using the `*` (explode) modifier with any expansion operator (e.g., `{foo*}`, `{+var*}`, `{#var*}`, `{/var*}`, `{.var*}`, `{;var*}`, `{?var*}`, `{&var*}`) generate patterns with nested unbounded quantifiers that are O(2^n) when matched against a maliciously crafted URI.**
- **Templates using multiple variables with the `+` or `#` operators (e.g., `{+v1,v2,v3}`) generate patterns with O(n^k) complexity due to the comma separator being within the matched character class, causing ambiguous backtracking across k variables.**

When matched against a maliciously crafted URI, this can result in catastrophic backtracking and uncontrolled resource consumption, leading to denial of service. The first pattern was partially addressed in 2.8.10 for certain operator combinations. Both patterns are fully remediated in 2.9.0. Users of the URI parsing capabilities in Addressable but not the URI template matching capabilities are unaffected.
Affected Versions

This vulnerability affects Addressable >= 2.3.0 (note: 2.3.0 and 2.3.1 were yanked; the earliest installable release is 2.3.2). It was partially fixed in version 2.8.10 and fully remediated in 2.9.0.
The vulnerability is more exploitable on MRI Ruby < 3.2 and on all versions of JRuby and TruffleRuby. MRI Ruby 3.2 and later ship with Onigmo 6.9, which introduces memoization that prevents catastrophic backtracking for the first class of template. JRuby and TruffleRuby do not implement equivalent memoization and remain vulnerable to all patterns.
This has been confirmed on the following runtimes:
| Runtime | Status | |--------------|--------| | MRI Ruby 2.6 | Vulnerable | | MRI Ruby 2.7 | Vulnerable | | MRI Ruby 3.0 | Vulnerable | | MRI Ruby 3.1 | Vulnerable | | MRI Ruby 3.2 | Partially vulnerable | | MRI Ruby 3.3 | Partially vulnerable | | MRI Ruby 3.4 | Partially vulnerable | | MRI Ruby 4.0 | Partially vulnerable | | JRuby 10.0 | Vulnerable | | TruffleRuby 21.2 | Vulnerable |
Workarounds

- **Upgrade to MRI Ruby 3.2 or later, if your application does not use JRuby or TruffleRuby. The Onigmo memoization introduced in MRI Ruby 3.2 prevents catastrophic backtracking from nested unbounded quantifiers (pattern 1 above â€” templates using the `*` modifier). It does not reliably mitigate the O(n^k) multi-variable case (pattern 2), so upgrading Ruby alone may not be sufficient if your templates use `{+v1,v2,...}` or `{#v1,v2,...}` syntax.**

Avoid using vulnerable template patterns when matching user-supplied input on unpatched versions of the library:

- **Templates using the `*` (explode) modifier: `{foo*}`, `{+var*}`, `{#var*}`, `{.var*}`, `{/var*}`, `{;var*}`, `{?var*}`, `{&var*}`**
- **Templates using multiple variables with the `+` or `#` operators: `{+v1,v2}`, `{#v1,v2,v3}`, etc.**

Apply a short timeout around any call to `Template#match` or `Template#extract` that processes user-supplied data.

Credits

Discovered in collaboration with @jamfish.
For more information

If you have any questions or comments about this advisory: * Open an issue

VULNERABLE GEM: BCRYPT@3.1.11

Name:
bcrypt

Version:
3.1.11

ID:
CVE-2026-33306

LINK

---

bcrypt-ruby has an Integer Overflow that Causes Zero Key-Strengthening Iterations at Cost=31 on JRuby

DESCRIPTION:

Impact

An integer overflow in the Java BCrypt implementation for JRuby can cause zero iterations in the strengthening loop. Impacted applications must be setting the cost to 31 to see this happen.

The JRuby implementation of bcrypt-ruby (`BCrypt.java`) computes the key-strengthening round count as a signed 32-bit integer. When `cost=31` (the maximum allowed by the gem), signed integer overflow causes the round count to become negative, and the strengthening loop executes **zero iterations**. This collapses bcrypt from 2^31 rounds of exponential key-strengthening to effectively constant-time computation â€” only the initial EksBlowfish key setup and final 64x encryption phase remain.
The resulting hash looks valid (`$2a$31$...`) and verifies correctly via `checkpw` , making the weakness invisible to the application. This issue is triggered only when cost=31 is used or when verifying a `$2a$31$` hash.
Patches

This problem has been fixed in version 3.1.22
Workarounds

Set the cost to something less than 31.

VULNERABLE GEM: BOOTSTRAP-SASS@3.3.7

Name:
bootstrap-sass

Version:
3.3.7

ID:
CVE-2016-10735

LINK

XSS vulnerability via data-target in bootstrap-sass

DESCRIPTION:

In Bootstrap 3.x before 3.4.0 and 4.x-beta before 4.0.0-beta.2, XSS is possible in the data-target attribute.

VULNERABLE GEM: BOOTSTRAP-SASS@3.3.7

Name:
bootstrap-sass

Version:
3.3.7

ID:
CVE-2018-14040

LINK

Bootstrap vulnerable to Cross-Site Scripting (XSS)

DESCRIPTION:
In Bootstrap starting in version 2.3.0 and prior to 3.4.0, as well as 4.x before 4.1.2, XSS is possible in the collapse data-parent attribute.

VULNERABLE GEM: BOOTSTRAP-SASS@3.3.7

Name:
bootstrap-sass

Version:
3.3.7

ID:
CVE-2018-14042

LINK

Bootstrap Cross-site Scripting vulnerability

DESCRIPTION:
In Bootstrap starting in version 2.3.0 and prior to versions 3.4.0 and 4.1.2, XSS is possible in the data-container property of tooltip. This is similar to CVE-2018-14041.

VULNERABLE GEM: BOOTSTRAP-SASS@3.3.7

Name:
bootstrap-sass

Version:
3.3.7

ID:
CVE-2018-20676

LINK

XSS vulnerability that affects bootstrap

DESCRIPTION:
In Bootstrap before 3.4.0, XSS is possible in the tooltip data-viewport attribute.

VULNERABLE GEM: BOOTSTRAP-SASS@3.3.7

Name:
bootstrap-sass

Version:
3.3.7

ID:
CVE-2018-20677

LINK

---

bootstrap Cross-site Scripting vulnerability

DESCRIPTION:
In Bootstrap before 3.4.0, XSS is possible in the affix configuration target property.

VULNERABLE GEM: BOOTSTRAP-SASS@3.3.7

Name:
bootstrap-sass

Version:
3.3.7

ID:

CVE-2019-8331

LINK

XSS vulnerability in bootstrap-sass

DESCRIPTION:
In Bootstrap before 3.4.1 and 4.3.x before 4.3.1, XSS is possible in the tooltip or popover data-template attribute.

VULNERABLE GEM: CARRIERWAVE@1.2.2

Name:
carrierwave

Version:
1.2.2

ID:
CVE-2021-21288

LINK

Server-side request forgery in CarrierWave

DESCRIPTION:
Impact

[CarrierWave download feature]
(https://github.com/carrierwaveuploader/carrierwave#uploading-files-from-a-remote-location has an SSRF vulnerability, allowing attacks to provide DNS entries or IP addresses that are intended for internal use and gather information about the Intranet infrastructure of the platform.
Patches

Upgrade to 2.1.1 or 1.3.2.
Workarounds

Using proper network segmentation and applying the principle of least privilege to outbound connections from application servers can reduce the severity of SSRF vulnerabilities. Ideally the vulnerable gem should run on an isolated server without access to any internal network resources or cloud metadata access.

VULNERABLE GEM: CARRIERWAVE@1.2.2

Name:
carrierwave

Version:
1.2.2

ID:
CVE-2021-21305

LINK

Code Injection vulnerability in CarrierWave::RMagick

DESCRIPTION:
Impact

CarrierWave::RMagick has a Code Injection vulnerability. Its `#manipulate!` method inappropriately evals the content of mutation option(`:read` / `:write`), allowing attackers to craft a string that can be executed as a Ruby code. If an application developer supplies untrusted inputs to the option, it will lead to remote code execution(RCE).
(But supplying untrusted input to the option itself is dangerous even in absence of this vulnerability, since is prone to DoS vulnerability - attackers can try to consume massive amounts of memory by resizing to a very large dimension)
Patches

Upgrade to 2.1.1 or 1.3.2.
Workarounds

Stop supplying untrusted input to `#manipulate!` 's mutation option.

VULNERABLE GEM: CARRIERWAVE@1.2.2

Name:
carrierwave

Version:
1.2.2

ID:
CVE-2023-49090

LINK

CarrierWave Content-Type allowlist bypass vulnerability, possibly leading to XSS

DESCRIPTION:

Impact

CarrierWave::Uploader::ContentTypeAllowlist has a Content-Type allowlist bypass vulnerability, possibly leading to XSS.
The validation in `allowlisted_content_type?` determines Content-Type permissions by performing a partial match. If the `content_type` argument of `allowlisted_content_type?` is passed a value crafted by the attacker, Content-Types not included in the `content_type_allowlist` will be allowed. In addition, by setting the Content-Type configured by the attacker at the time of file delivery, it is possible to cause XSS on the user's browser when the uploaded file is opened.
Patches

Upgrade to 3.0.5 or 2.2.5.
Workarounds

When validating with `allowlisted_content_type?` in CarrierWave::Uploader::ContentTypeAllowlist, forward match(`\\A`) the Content-Type set in `content_type_allowlist`, preventing unintentional permission of `text/html;image/png` when you want to allow only `image/png` in `content_type_allowlist`.
References

OWASP - File Upload Cheat Sheet

VULNERABLE GEM: CARRIERWAVE@1.2.2

Name:
carrierwave

Version:
1.2.2

ID:
CVE-2024-29034

LINK

CarrierWave content-Type allowlist bypass vulnerability which possibly leads to XSS remained

DESCRIPTION:

Impact

The vulnerability CVE-2023-49090 wasn't fully addressed.
This vulnerability is caused by the fact that when uploading to object storage, including Amazon S3, it is possible to set a Content-Type value that is interpreted by browsers to be different from what's allowed by `content_type_allowlist`, by providing multiple values separated by commas.
This bypassed value can be used to cause XSS.
Patches

Upgrade to 3.0.7 or 2.2.6.
Workarounds

Use the following monkey patch to let CarrierWave parse the Content-type by using `Marcel::MimeType.for`.

```
 # For CarrierWave 3.x
CarrierWave::SanitizedFile.class_eval do
 def declared_content_type
  @declared_content_type ||
    if @file.respond_to?(:content_type) && @file.content_type
      Marcel::MimeType.for(declared_type: @file.content_type.to_s.
chomp)
    end
 end
end
```

```
 # For CarrierWave 2.x
CarrierWave::SanitizedFile.class_eval do
 def existing_content_type
  if @file.respond_to?(:content_type) && @file.content_type
    Marcel::MimeType.for(declared_type: @file.content_type.to_s.c
homp)
  end
 end
end
```

References

[OWASP - File Upload Cheat Sheet]

VULNERABLE GEM: CARRIERWAVE@1.2.2

Name:
carrierwave

Version:
1.2.2

ID:
CVE-2026-44587

LINK

CarrierWave has a denylisted_content_type bypass via Unescaped

Regex Metacharacters

DESCRIPTION:

Summary

CarrierWave's content*type*denylist check fails to escape regex metacharacters in string entries, causing the denylist to silently not match the content types it is intended to block.
Note: CarrierWave is aware `#content_type_denylist is deprecated for the security reason`, but it still used by developers, and the problem here isn't denylist allows any filetype, and thats not a vulnerability in carrierwave, its an implementation problem in developers using CarrierWave, the problem is its denylist entries are interpolated directly into a regex without `Regexp.quote` or anchoring. The denylist is still useful when developers want to ban specific content types but allow everything else.
Details

In `lib/carrierwave/uploader/content_type_denylist.rb:57`, string denylist entries are interpolated directly into a regex without `Regexp.quote` or anchoring:

```
  def denylisted_content_type?(denylist, content_type)
  Array(denylist).any? { |item| content_type =~ /#{item}/ }
end
```

```
The entry "image/svg+xml" becomes the regex /image\/svg+xml/ w
here +
is a quantifier meaning "one or more g", not a literal +. This
regex never matches the real MIME type "image/svg+xml" which c
ontains
a literal +. This is inconsistent with the allowlist implementation
at lib/carrierwave/uploader/content_type_allowlist.rb:53-57, which
correctly applies both Regexp.quote and a \A anchor:
```

```
rubydef allowlisted_content_type?(allowlist, content_type)
  Array(allowlist).any? do |item|
    item = Regexp.quote(item) if item.class != Regexp
    content_type =~ /\A#{item}/
  end
end
```

Other affected MIME types include `application/xhtml+xml` and any type containing regex metacharacters.
Fix: Apply Regexp.quote for string entries and anchor with \A, matching the existing allowlist implementation:

```
 rubydef denylisted_content_type?(denylist, content_type)
 Array(denylist).any? do |item|
   item = Regexp.quote(item) if item.class != Regexp
   content_type =~ /\A#{item}/
 end
end
```

Impact

Any application that uses content*type*denylist to block image/svg+xml â€” the most common use case, specifically to prevent stored XSS â€” is silently unprotected.

VULNERABLE GEM: CONCURRENT-RUBY@1.0.5

Name:
concurrent-ruby

Version:
1.0.5

ID:
CVE-2026-54904

LINK

Concurrent Ruby - `AtomicReference#update` livelocks when the stored value is `Float::NAN`

DESCRIPTION:
Summary

`Concurrent::AtomicReference#update` can enter a permanent busy retry loop when the current value is `Float::NAN`.
The issue is caused by the interaction between: - `AtomicReference#update`, which retries until `compare_and_set(old_value, new_value)` succeeds. - Numeric `compare_and_set`, which checks `old == old_value` before attempting the underlying atomic swap. - Ruby NaN semantics, where `Float::NAN == Float::NAN` is always `false`.

As a result, once an `AtomicReference` contains `Float::NAN`, calling `#update` repeatedly evaluates the caller's block and never returns. In services that store externally derived numeric values in an `AtomicReference`, this can cause CPU exhaustion or permanent request/job hangs.
Impact

This is an application-level denial of service issue. If an application stores externally derived numeric data in a `Concurrent::AtomicReference`, an attacker or faulty upstream data source may be able to cause the stored value to become `Float::NAN`. Any later call to `AtomicReference#update` on that reference will spin indefinitely, repeatedly executing the update block and consuming CPU.
Credit

Pranjali Thakur - depthfirst (depthfirst.com)

VULNERABLE GEM: CONCURRENT-RUBY@1.0.5

Name:
concurrent-ruby

Version:
1.0.5

ID:
CVE-2026-54905

LINK

Concurrent Ruby - `ReentrantReadWriteLock` read-count overflow grants a write lock without exclusivity

DESCRIPTION:
Summary

`Concurrent::ReentrantReadWriteLock` can incorrectly grant a write lock after one thread acquires the read lock 32,768 times.
The lock stores a thread's local read and write hold counts in one integer. The low 15 bits are used for the read hold count, and bit 15 is used as `WRITE_LOCK_HELD`. After 32,768 reentrant read acquisitions, the local

read count crosses into the write-lock bit. `try_write_lock` then treats the thread as already holding a write lock and returns `true` without setting the global `RUNNING_WRITER` bit.
This breaks the core mutual-exclusion guarantee: the caller is told it has a write lock, but other threads can still hold or acquire read locks at the same time.
Impact

This breaks the write-lock exclusivity guarantee. After the overflow, a thread can be told it has acquired the write lock while other threads can still hold or acquire read locks, allowing races and inconsistent reads of protected mutable state.
Credit

Pranjali Thakur - depthfirst (depthfirst.com)

VULNERABLE GEM: CONCURRENT-RUBY@1.0.5

Name:
concurrent-ruby

Version:
1.0.5

ID:
CVE-2026-54906

LINK

Concurrent Ruby - ReadWriteLock allows wrong-thread write release and stray read-release counter corruption

DESCRIPTION:
Summary

`Concurrent::ReadWriteLock#release_write_lock` does not verify that the calling thread acquired the write lock. Any thread with access to the lock object can release an active write lock held by another thread. A second writer can then enter its critical section while the first writer is still running. `Concurrent::ReadWriteLock#release_read_lock` also decrements the shared counter even when no read lock is held. Calling it on a fresh lock

changes the counter from `0` to `-1`, after which normal read acquisition raises `Concurrent::ResourceLimitError`.
This is a synchronization correctness issue in the public `Concurrent::ReadWriteLock` API. It should not be framed as an authorization bypass; the lock is an in-process concurrency primitive, not an access-control boundary.
Impact

This can break the write-lock mutual exclusion guarantee and can also leave a lock unusable after a stray read release. The impact is local to applications that expose or misuse the manual `acquire_*` / `release_*` APIs. If the lock protects integrity-sensitive mutable state, wrong-thread write release can allow concurrent writers and data races. The stray read-release path can cause denial of service by corrupting the lock counter.
Credit

Pranjali Thakur - depthfirst (depthfirst.com)

VULNERABLE GEM: CRASS@1.0.3

Name:
crass

Version:
1.0.3

ID:
GHSA-6jxj-px6v-747w

LINK

Deeply nested CSS blocks and functions can trigger a SystemStackError or excessive memory usage

DESCRIPTION:
IMPACT
Crass recursively parses CSS simple blocks and functions without a depth guard. An attacker-controlled value containing many deeply nested blocks can recurse until Ruby raises SystemStackError: stack level too deep, or can cause excessive memory usage.

VULNERABLE GEM: CRASS@1.0.3

Name:
crass

Version:
1.0.3

ID:
GHSA-6wmf-3r64-vcwv

LINK

Large numeric exponents cause CPU and memory denial of service

DESCRIPTION:
IMPACT
Crass converts CSS scientific notation number values with unbounded exponentiation before it clamps the result to Float::MAX. Applications that use Crass to parse attacker-controlled CSS strings can be forced to spend disproportionate CPU and memory parsing a tiny input, possibly resulting in a crash.
Exponents are now bounded before 10**exponent is computed.

VULNERABLE GEM: CRASS@1.0.3

Name:
crass

Version:
1.0.3

ID:
GHSA-8vfg-2r28-hvhj

LINK

Non-ASCII characters cause superlinear CPU consumption

DESCRIPTION:
IMPACT
When parsing an input containing non-ASCII characters, inefficiencies in how Crass tracks the positions of multi-byte characters result in superlinear parsing time. An attacker-controlled input consisting of many non-ASCII characters could cause excessive CPU consumption and potentially denial of service.

VULNERABLE GEM: CRASS@1.0.3

Name:
crass

Version:
1.0.3

ID:
GHSA-wwpr-jff3-395c

LINK

A large number of adjacent CSS comments can trigger a SystemStackError

DESCRIPTION:
IMPACT
When the :preserve_comments option is not enabled (which is the default behavior), Crass discards CSS comments by recursively consuming the next token. An attacker who provides a stylesheet containing a very large number of adjacent comments can cause excessive recursion and trigger a SystemStackError.

VULNERABLE GEM: DEVISE@3.5.10

Name:
devise

Version:
3.5.10

ID:
CVE-2019-16109

LINK

Devise Gem for Ruby confirmation token validation with a blank string

DESCRIPTION:
Devise before 4.7.1 confirms accounts upon receiving a request with a blank confirmation*token, if a database record has a blank value in the confirmation*token column. However, there is no scenario within Devise itself in which such database records would exist.

VULNERABLE GEM: DEVISE@3.5.10

Name:
devise

Version:
3.5.10

ID:
CVE-2019-5421

LINK

Devise Gem for Ruby Time-of-check Time-of-use race condition with lockable module

DESCRIPTION:

Devise ruby gem before 4.6.0 when the `lockable` module is used is vulnerable to a time-of-check time-of-use (TOCTOU) race condition due to `increment_failed_attempts` within the `Devise::Models::Lockable` class not being concurrency safe.

VULNERABLE GEM: DEVISE@3.5.10

Name:
devise

Version:
3.5.10

ID:
CVE-2026-32700

LINK

Confirmable "change email" race condition permits user to confirm email they have no access to

DESCRIPTION:

IMPACT

A race condition in Devise's Confirmable module allows an attacker to confirm an email address they do not own. This affects any Devise application using the reconfirmable option (the default when using Confirmable with email changes).

By sending two concurrent email change requests, an attacker can desynchronize the confirmation*token and unconfirmed*email fields. The confirmation token is sent to an email the attacker controls, but the unconfirmed_email in the database points to a victim's email address. When the attacker uses the token, the victim's email is confirmed on the attacker's account.

PATCH

This is patched in Devise v5.0.3. Users should upgrade as soon as possible.

WORKAROUND

Applications can override this specific method from Devise models to force unconfirmed_email to be persisted when unchanged: (assuming your model is User)

```
 class User < ApplicationRecord
 protected

 def postpone_email_change_until_confirmation_and_regenerate_
confirmation_token
   unconfirmed_email_will_change!
   super
 end
end
```

Note: Mongoid does not seem to respect that will*change! should force the attribute to be persisted, even if it did not really change, so you might have to implement a workaround similar to Devise by setting changed*attributes["unconfirmed_email"] = nil as well.

VULNERABLE GEM: DEVISE@3.5.10

Name:
devise

Version:
3.5.10

ID:
CVE-2026-40295

LINK

Devise has an Open Redirect via Unvalidated `request.referrer` in

Timeoutable Session Timeout Handler

DESCRIPTION:

SUMMARY

When the `Timeoutable` module is enabled in Devise, the `FailureApp#redirect_url` method returns `request.referrer` â€" the HTTP `Referer` header, which is attacker-controllable â€" without validation for any non-GET request that results in a session timeout. An attacker who hosts a page with an auto-submitting cross-origin form can cause a victim with an expired Devise session to be redirected to an arbitrary external URL. This contrasts with the GET timeout path (which uses server-side `attempted_path`) and Devise's own `store_location_for` mechanism (which strips external hosts via `extract_path_from_location`), both of which are protected; only the non-GET timeout redirect path is unprotected.

DETAILS

The vulnerable code is in `lib/devise/failure_app.rb`:

```
def redirect_url
if warden_message == :timeout
  flash[:timedout] = true if is_flashing_format?

  path = if request.get?
    attempted_path          # safe: server-side value from warden opt
ons
  else
    request.referrer        # UNSAFE: HTTP Referer header, attacker
-controlled
  end

  path || scope_url
else
  scope_url
end
end
```

This is passed directly to `redirect_to`:

```
def redirect
store_location!
# ...
redirect_to redirect_url   # redirect_url may be an external attacker
URL
end
```

The GET timeout path uses `attempted_path`, which is set server-side by Warden and cannot be influenced by the client. The `store_location!` method also only runs for GET requests, so no session-based protection is applied on POST timeouts.

By contrast, Devise's `store_location_for` method (used elsewhere) correctly sanitizes URLs via `extract_path_from_location`, which strips the scheme and host.

IMPACT

Victims with expired sessions who click any attacker-crafted link or visit an attacker page with an auto-submitting form are redirected to an arbitrary external URL.

The redirect happens transparently via a trusted domain (the target app's domain), bypassing browser phishing warnings.

An attacker can redirect victims to a fake login page to harvest credentials (phishing), or to malicious download sites.

Note: Rails' built-in open-redirect protection does not mitigate this issue. `Devise::FailureApp` is an `ActionController::Metal` app with its own isolated copy of the relevant redirect configuration, so `config.action_controller.action_on_open_redirect = :raise` (and the older `raise_on_open_redirects` setting) do not reach it.

PATCHES

This is patched in Devise v5.0.4. Users should upgrade as soon as possible.

WORKAROUND

None beyond upgrading. If an upgrade is not immediately possible, the same changes from the patch commit can be applied as a monkey-patch in a Rails initializer (`Devise::FailureApp#redirect_url` and `Devise::Controllers::StoreLocation#extract_path_from_location`). Remove the monkey-patch after upgrading.

VULNERABLE GEM: FARADAY@0.12.2

Name:
faraday

Version:
0.12.2

ID:
CVE-2026-25765

LINK

Faraday affected by SSRF via protocol-relative URL host override in build_exclusive_url

DESCRIPTION:
Impact

Faraday's `build_exclusive_url` method (in `lib/faraday/connection.rb`) uses Ruby's `URI#merge` to combine the connection's base URL with a user-supplied path. Per RFC 3986, protocol-relative URLs (e.g. `//evil.com/path`) are treated as network-path references that override the base URL's host/authority component.
This means that if any application passes user-controlled input to Faraday's `get()`, `post()`, `build_url()`, or other request methods, an attacker can supply a protocol-relative URL like `//attacker.com/endpoint` to redirect the request to an arbitrary host, enabling Server-Side Request Forgery (SSRF).
The `./` prefix guard added in v2.9.2 (PR #1569) explicitly exempts URLs starting with `/`, so protocol-relative URLs bypass it entirely.
Example ```ruby conn = Faraday.new(url: 'https://api.internal.com') conn.get('//evil.com/steal')

Request is sent to https://evil.com/steal instead of api.internal.com

Patches

Faraday v2.14.1 is patched against this security issue. All versions of Faraday up to 2.14.0 are affected.

Workarounds

**NOTE: Upgrading to Faraday v2.14.1+ is the recommended action
to mitigate this issue, however should that not be an option please continue reading.**

Applications should validate and sanitize any user-controlled input before passing it to Faraday request methods. Specifically:

- Reject or strip input that starts with // followed by a non-/ character.
- Use an allowlist of permitted path prefixes.
- Alternatively, prepend ./ to all user-supplied paths before passing them to Faraday.

Example validation:
```ruby
def safe_path(user_input)
  raise ArgumentError, "Invalid path" if user_input.match?(r{\A//[^/]})
  user_input
end
```

VULNERABLE GEM: FARADAY@0.12.2

Name:
faraday

Version:
0.12.2

ID:

CVE-2026-54297

LINK

Faraday - Uncontrolled recursion in NestedParamsEncoder allows stack exhaustion DoS via deeply nested query parameters

DESCRIPTION:

Uncontrolled Recursion in NestedParamsEncoder Allows Stack

Exhaustion DoS via Deeply Nested Query Parameters

SUMMARY

`Faraday::NestedParamsEncoder`, the default nested query parameter encoder/decoder in Faraday, decodes nested query strings without enforcing a maximum nesting depth.

A crafted query string such as:

```
a[x][x][x][x]...[x]=1
```

causes Faraday to build a deeply nested Ruby `Hash` structure. The internal `dehash` routine then recursively walks this attacker-controlled structure without a depth limit. At sufficient depth, Ruby raises an uncaught `SystemStackError` (`stack level too deep`), crashing the calling thread or worker.

This can lead to denial of service in applications that pass attacker-controlled query strings to Faraday's nested query parsing or URL-building paths.

IMPACT

A relatively small query string can trigger a `SystemStackError` and crash the calling Ruby thread or worker.

In my local test environment, a payload of approximately 9.4 KB was sufficient:

```
depth=3119
bytes=9360
result=SystemStackError
message="stack level too deep"
```

Repeated requests with such payloads may cause a denial of service against applications whose request path forwards, parses, or rebuilds attacker-controlled query strings through Faraday.

This issue does not provide remote code execution, authentication bypass, or data disclosure. The confirmed impact is availability loss.

PATCHED VERSIONS

The fix was released in Faraday 2.14.3 and backported to the 1.x branch in Faraday 1.10.6, which adds a `param_depth_limit` to `NestedParamsEncoder`.

REPORTER

Reported by: Emre Koca

VULNERABLE GEM: FFI@1.9.21

Name:
ffi

Version:
1.9.21

ID:
CVE-2018-1000201

LINK

---

ruby-ffi DDL loading issue on Windows OS

DESCRIPTION:
ruby-ffi version 1.9.23 and earlier has a DLL loading issue which can be hijacked on Windows OS, when a Symbol is used as DLL name instead of a String This vulnerability appears to have been fixed in v1.9.24 and later.

VULNERABLE GEM: GLOBALID@0.4.1

Name:
globalid

Version:
0.4.1

ID:
CVE-2023-22799

LINK

ReDoS based DoS vulnerability in GlobalID

DESCRIPTION:
There is a ReDoS based DoS vulnerability in the GlobalID gem. This vulnerability has been assigned the CVE identifier CVE-2023-22799. Versions Affected: >= 0.2.1 Not affected: < 0.2.1 Fixed Versions: 1.0.1

Impact

There is a possible DoS vulnerability in the model name parsing section of the GlobalID gem. Carefully crafted input can cause the regular expression engine to take an unexpected amount of time. All users running an affected release should either upgrade or use one of the workarounds immediately.

Workarounds

There are no feasible workarounds for this issue.

VULNERABLE GEM: JQUERY-RAILS@4.3.1

Name:
jquery-rails

Version:
4.3.1

ID:
CVE-2019-11358

LINK

Prototype pollution attack through jQuery $.extend

DESCRIPTION:
jQuery before 3.4.0 mishandles jQuery.extend(true, {}, ...) because of bject.prototype pollution. If an unsanitized source object contained an enumerable **proto** property, it could extend the native Object.prototype.

VULNERABLE GEM: JQUERY-RAILS@4.3.1

Name:
jquery-rails

Version:
4.3.1

ID:
CVE-2020-11023

LINK

Potential XSS vulnerability in jQuery

DESCRIPTION:
IMPACT
Passing HTML containing `<option>` elements from untrusted sources - even after sanitizing them - to one of jQuery's DOM manipulation methods (i.e. `.html()`, `.append()`, and others) may execute untrusted code.
WORKAROUNDS
To workaround this issue without upgrading, use DOMPurify with its `SAFE_FOR_JQUERY` option to sanitize the HTML string before passing it to a jQuery method.

VULNERABLE GEM: JQUERY-RAILS@4.3.1

Name:
jquery-rails

Version:
4.3.1

ID:
CVE-2020-23064

LINK

jQuery Cross Site Scripting vulnerability

DESCRIPTION:
Cross Site Scripting vulnerability in jQuery v.2.2.0 until v.3.5.0 allows a remote attacker to execute arbitrary code via the `<options>` element.

VULNERABLE GEM: JSON@2.1.0

Name:
json

Version:
2.1.0

ID:
CVE-2020-10663

LINK

json Gem for Ruby Unsafe Object Creation Vulnerability (additional fix)

DESCRIPTION:
There is an unsafe object creation vulnerability in the json gem bundled with Ruby. This vulnerability has been assigned the CVE identifier CVE-2020-10663. We strongly recommend upgrading the json gem.

DETAILS
When parsing certain JSON documents, the json gem (including the one bundled with Ruby) can be coerced into creating arbitrary objects in the

target system.
This is the same issue as CVE-2013-0269. The previous fix was incomplete, which addressed JSON.parse(user*input), but didnâ€™t address some other styles of JSON parsing including JSON(user*input) and JSON.parse(user_input, nil).
See CVE-2013-0269 in detail. Note that the issue was exploitable to cause a Denial of Service by creating many garbage-uncollectable Symbol objects, but this kind of attack is no longer valid because Symbol objects are now garbage-collectable. However, creating arbitrary objects may cause severe security consequences depending upon the application code.

VULNERABLE GEM: JWT@1.5.6

Name:
jwt

Version:
1.5.6

ID:
CVE-2026-45363

LINK

---

ruby-jwt: Empty-key HMAC bypass; cross-language sibling of CVE-2026-44351

DESCRIPTION:

`JWT.decode(token, '', true, algorithm: 'HS256')` accepts an attacker-forged token. `OpenSSL::HMAC.digest('SHA256', '', payload)` returns a valid digest under an empty key, and no `raise InvalidKeyError if key.empty?` precondition exists in the HMAC algorithm.

```
JWT.decode(token, "", true, algorithm: 'HS256')
 -> JWA::Hmac.verify(verification_key: "", ...)
 -> OpenSSL::HMAC.digest('SHA256', "", signing_input) == signature
```

The same path is reached when a keyfinder block or key*finder: argument returns "", nil, or an array containing nil for an unknown key.
JWT::Decode#find*key only rejects literal nil and empty arrays, and JWT::JWA::Hmac silently coerces nil to "" (signing_key ||= '') before signing.

```
JWT.decode(token, nil, true, algorithms: ['HS256']) { |_h| "" }
-> find_key returns ""              # "" && !Array("").empty? == true
-> JWA::Hmac.verify(verification_key: "", ...)
-> verifies
```

Common application patterns that produce the unsafe value: `redis.get("kid:#{kid}").to_s`, ORM string columns with `default: ''`, `ENV['SECRET'] || ''`, `Hash.new('')` lookups, `[primary, fallback]` where fallback may be nil. Applications passing a non-empty static `key:`, or whose keyfinder returns nil / raises on miss, are not affected.
The existing `enforce_hmac_key_length` option would block this but defaults to false. On OpenSSL â‰¥ 3.5 the empty-key HMAC.digest call no longer raises, so the OpenSSL-3.0 rescue in JWA::Hmac#sign does not fire. Affects HS256/HS384/HS512 via both JWT.decode (positional key and block keyfinder) and `JWT::EncodedToken#verify_signature!(key_finder:)`.

VULNERABLE GEM: KAMINARI@1.1.1

Name:
kaminari

Version:
1.1.1

ID:
CVE-2020-11082

LINK

Cross-Site Scripting in Kaminari via `original_script_name` parameter

DESCRIPTION:
Impact

There was a vulnerability in versions of Kaminari that would allow an attacker to inject arbitrary code into pages with pagination links.
For example, an attacker could craft pagination links that link to other domain or host: https://example.com/posts?page=4&original*script*name=https://another-host.example.com
In addition, an attacker could also craft pagination links that include JavaScript code that runs when a user clicks the link:

https://example.com/posts?
page=4&original*script*name=javascript:alert(42)%3b//
Releases

The 1.2.1 gem including the patch has already been released. All past released versions are affected by this vulnerability.
Workarounds

Application developers who can't update the gem can workaround by overriding the `PARAM_KEY_EXCEPT_LIST` constant.

```
module Kaminari::Helpers
PARAM_KEY_EXCEPT_LIST = [:authenticity_token, :commit, :utf8, :_method, :script_name, :original_script_name].freeze
end
```

VULNERABLE GEM: LOOFAH@2.2.2

Name:
loofah

Version:
2.2.2

ID:
CVE-2018-16468

LINK

Loofah XSS Vulnerability

DESCRIPTION:
In the Loofah gem, through v2.2.2, unsanitized JavaScript may occur in sanitized output when a crafted SVG element is republished.

VULNERABLE GEM: LOOFAH@2.2.2

Name:
loofah

Version:
2.2.2

ID:
CVE-2019-15587

LINK

Loofah XSS Vulnerability

DESCRIPTION:
In the Loofah gem, through v2.3.0, unsanitized JavaScript may occur in sanitized output when a crafted SVG element is republished.

VULNERABLE GEM: LOOFAH@2.2.2

Name:
loofah

Version:
2.2.2

ID:
CVE-2022-23514

LINK

Inefficient Regular Expression Complexity in Loofah

DESCRIPTION:
SUMMARY
Loofah `< 2.19.1` contains an inefficient regular expression that is susceptible to excessive backtracking when attempting to sanitize certain SVG attributes. This may lead to a denial of service through CPU resource consumption.
MITIGATION
Upgrade to Loofah `>= 2.19.1`.

VULNERABLE GEM: LOOFAH@2.2.2

Name:
loofah

Version:
2.2.2

ID:
CVE-2022-23515

LINK

Improper neutralization of data URIs may allow XSS in Loofah

DESCRIPTION:
SUMMARY
Loofah `>= 2.1.0, < 2.19.1` is vulnerable to cross-site scripting via the `image/svg+xml` media type in data URIs.
MITIGATION
Upgrade to Loofah `>= 2.19.1`.

VULNERABLE GEM: LOOFAH@2.2.2

Name:
loofah

Version:
2.2.2

ID:
CVE-2022-23516

LINK

Uncontrolled Recursion in Loofah

DESCRIPTION:

SUMMARY

Loofah `>= 2.2.0, < 2.19.1` uses recursion for sanitizing `CDATA` sections, making it susceptible to stack exhaustion and raising a `SystemStackError` exception. This may lead to a denial of service through CPU resource consumption.

MITIGATION

Upgrade to Loofah `>= 2.19.1`.
Users who are unable to upgrade may be able to mitigate this vulnerability by limiting the length of the strings that are sanitized.

VULNERABLE GEM: LOOFAH@2.2.2

Name:
loofah

Version:
2.2.2

ID:
GHSA-9wjq-cp2p-hrgf

LINK

SVG `href` attribute bypasses local-reference restriction in Loofah

DESCRIPTION:

SUMMARY

Loofah's HTML5 sanitizer restricted only the xlink:href attribute on certain SVG elements to local, same-document references. Browsers also accept a plain href attribute as an alternative to the deprecated xlink:href per the SVG 2 spec, but Loofah did not apply the same restriction to it, allowing those elements to reference arbitrary external documents.

IMPACT

SVG can load and render external SVG content by reference. If the referenced external SVG is same-origin and contains scripts or other dangerous content, it could execute in the context of the sanitized document. can load external images, which can be used for tracking. Modern browsers restrict cross-origin fetches, which limits but does not eliminate the risk. Applications that sanitize user-supplied SVG (directly, or as part of HTML) with Loofah's default allowlist are affected.

CREDIT

Found by the maintainer, Mike Dalessio, during a security audit.

VULNERABLE GEM: MINI_MAGICK@4.8.0

Name:
mini_magick

Version:
4.8.0

ID:
CVE-2019-13574

LINK

Remote command execution via filename

DESCRIPTION:

A remote shell execution vulnerability when using MiniMagick::Image.open with URL coming from unsanitized user input. e.g.
`MiniMagick::Image.open("| touch.txt")`

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2018-14404

LINK

Nokogiri gem, via libxml2, is affected by multiple vulnerabilities

DESCRIPTION:

Nokogiri 1.8.5 has been released.
This is a security and bugfix release. It addresses two CVEs in upstream libxml2 rated as "medium" by Red Hat, for which details are below.
If you're using your distro's system libraries, rather than Nokogiri's vendored libraries, there's no security need to upgrade at this time, though you may want to check with your distro whether they've patched this (Canonical has patched Ubuntu packages). Note that these patches are not yet (as of 2018-10-04) in an upstream release of libxml2.
Full details about the security update are available in Github Issue #1785.

[MRI] Pulled in upstream patches from libxml2 that address CVE-2018-14404 and CVE-2018-14567. Full details are available in #1785. Note that these patches are not yet (as of 2018-10-04) in an upstream release of libxml2.

CVE-2018-14404
Permalink:

https://people.canonical.com/~ubuntu-security/cve/2018/CVE-2018-14404.html
Description:
A NULL pointer dereference vulnerability exists in the xpath.c:xmlXPathCompOpEval() function of libxml2 through 2.9.8 when parsing an invalid XPath expression in the XPATH*OP*AND or XPATH*OP*OR case. Applications processing untrusted XSL format inputs with the use of the libxml2 library may be vulnerable to a denial of service attack due to a crash of the application
Canonical rates this vulnerability as "Priority: Medium"

CVE-2018-14567
Permalink:
https://people.canonical.com/~ubuntu-security/cve/2018/CVE-2018-14567.html
Description:
infinite loop in LZMA decompression
Canonical rates this vulnerability as "Priority: Medium"

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2018-25032

LINK

Out-of-bounds Write in zlib affects Nokogiri

DESCRIPTION:
SUMMARY
Nokogiri v1.13.4 updates the vendored zlib from 1.2.11 to 1.2.12, which addresses CVE-2018-25032. That CVE is scored as CVSS 7.4 "High" on the NVD record as of 2022-04-05.
Please note that this advisory only applies to the CRuby implementation of Nokogiri `< 1.13.4`, and only if the packaged version of `zlib` is being used. Please see this document for a complete description of which platform

gems vendor `zlib`. If you've overridden defaults at installation time to use system libraries instead of packaged libraries, you should instead pay attention to your distro's `zlib` release announcements.

MITIGATION

Upgrade to Nokogiri `>= v1.13.4`.

IMPACT

CVE-2018-25032 in zlib

Type: CWE-787 Out of bounds write

Severity: High

Description: zlib before 1.2.12 allows memory corruption when deflating (i.e., when compressing) if the input has many distant matches.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2018-8048

LINK

Revert libxml2 behavior in Nokogiri gem that could cause XSS

DESCRIPTION:

[MRI] Behavior in libxml2 has been reverted which caused CVE-2018-8048 (loofah gem), CVE-2018-3740 (sanitize gem), and CVE-2018-3741 (rails-html-sanitizer gem). The commit in question is here: https://github.com/GNOME/libxml2/commit/960f0e2
and more information is available about this commit and its impact here: https://github.com/flavorjones/loofah/issues/144
This release simply reverts the libxml2 commit in question to protect users of Nokogiri's vendored libraries from similar vulnerabilities.
If you're offended by what happened here, I'd kindly ask that you comment on the upstream bug report here: https://bugzilla.gnome.org/show_bug.cgi?id=769760

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2019-11068

LINK

Nokogiri gem, via libxslt, is affected by improper access control vulnerability

DESCRIPTION:

Nokogiri v1.10.3 has been released.
This is a security release. It addresses a CVE in upstream libxslt rated as "Priority: medium" by Canonical, and "NVD Severity: high" by Debian. More details are available below.
If you're using your distro's system libraries, rather than Nokogiri's vendored libraries, there's no security need to upgrade at this time, though you may want to check with your distro whether they've patched this (Canonical has patched Ubuntu packages). Note that this patch is not yet (as of 2019-04-22) in an upstream release of libxslt.
Full details about the security update are available in Github Issue [#1892] https://github.com/sparklemotion/nokogiri/issues/1892.

CVE-2019-11068
Permalinks are: - Canonical: https://people.canonical.com/~ubuntu-security/cve/CVE-2019-11068 - Debian: https://security-tracker.debian.org/tracker/CVE-2019-11068
Description:

> *libxslt through 1.1.33 allows bypass of a protection mechanism because callers of xsltCheckRead and xsltCheckWrite permit access even upon receiving a -1 error code. xsltCheckRead can return -1 for a crafted URL that is not actually invalid and is subsequently loaded.*

Canonical rates this as "Priority: Medium".

Debian rates this as "NVD Severity: High (attack range: remote)".

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2019-13117

LINK

Nokogiri gem, via libxslt, is affected by multiple vulnerabilities

DESCRIPTION:

Nokogiri v1.10.5 has been released.
This is a security release. It addresses three CVEs in upstream libxml2, for which details are below.
If you're using your distro's system libraries, rather than Nokogiri's vendored libraries, there's no security need to upgrade at this time, though you may want to check with your distro whether they've patched this (Canonical has patched Ubuntu packages). Note that libxslt 1.1.34 addresses these vulnerabilities.
Full details about the security update are available in Github Issue [#1943] https://github.com/sparklemotion/nokogiri/issues/1943.

CVE-2019-13117
https://people.canonical.com/~ubuntu-security/cve/2019/CVE-2019-13117.html
Priority: Low
Description: In numbers.c in libxslt 1.1.33, an xsl:number with certain format strings could lead to a uninitialized read in xsltNumberFormatInsertNumbers. This could allow an attacker to discern whether a byte on the stack contains the characters A, a, I, i, or 0, or any other character.
Patched with commit
https://gitlab.gnome.org/GNOME/libxslt/commit/c5eb6cf3aba0af048596106ed839b4ae17ecbcb1

CVE-2019-13118
https://people.canonical.com/~ubuntu-security/cve/2019/CVE-2019-13118.html
Priority: Low
Description: In numbers.c in libxslt 1.1.33, a type holding grouping characters of an xsl:number instruction was too narrow and an invalid character/length combination could be passed to xsltNumberFormatDecimal, leading to a read of uninitialized stack data
Patched with commit
https://gitlab.gnome.org/GNOME/libxslt/commit/6ce8de69330783977dd14f6569419489875fb71b

---

CVE-2019-18197
https://people.canonical.com/~ubuntu-security/cve/2019/CVE-2019-18197.html
Priority: Medium
Description: In xsltCopyText in transform.c in libxslt 1.1.33, a pointer variable isn't reset under certain circumstances. If the relevant memory area happened to be freed and reused in a certain way, a bounds check could fail and memory outside a buffer could be written to, or uninitialized data could be disclosed.
Patched with commit
https://gitlab.gnome.org/GNOME/libxslt/commit/2232473733b7313d67de8836ea3b29eec6e8e285

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2019-13118

LINK

---

libxslt Type Confusion vulnerability that affects Nokogiri

DESCRIPTION:
In `numbers.c` in libxslt 1.1.33, a type holding grouping characters of an

`xsl:number` instruction was too narrow and an invalid character/length combination could be passed to `xsltNumberFormatDecimal`, leading to a read of uninitialized stack data.
Nokogiri prior to version 1.10.5 used a vulnerable version of libxslt. Nokogiri 1.10.5 updated libxslt to version 1.1.34 to address this and other vulnerabilities in libxslt.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2019-18197

LINK

Nokogiri affected by libxslt Use of Uninitialized Resource/ Use After Free vulnerability

DESCRIPTION:

In xsltCopyText in transform.c in libxslt 1.1.33, a pointer variable isn't reset under certain circumstances. If the relevant memory area happened to be freed and reused in a certain way, a bounds check could fail and memory outside a buffer could be written to, or uninitialized data could be disclosed.
Nokogiri prior to version 1.10.5 contains a vulnerable version of libxslt. Nokogiri version 1.10.5 upgrades the dependency to libxslt 1.1.34, which contains a patch for this issue.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2019-5477

LINK

Nokogiri Command Injection Vulnerability via Nokogiri::CSS::Tokenizer#load_file

DESCRIPTION:

A command injection vulnerability in Nokogiri v1.10.3 and earlier allows commands to be executed in a subprocess by Ruby's `Kernel.open` method. Processes are vulnerable only if the undocumented method `Nokogiri::CSS::Tokenizer#load_file` is being passed untrusted user input. This vulnerability appears in code generated by the Rexical gem versions v1.0.6 and earlier. Rexical is used by Nokogiri to generate lexical scanner code for parsing CSS queries. The underlying vulnerability was addressed in Rexical v1.0.7 and Nokogiri upgraded to this version of Rexical in Nokogiri v1.10.4.
Upgrade to Nokogiri v1.10.4, or avoid calling the undocumented method `Nokogiri::CSS::Tokenizer#load_file` with untrusted user input.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2019-5815

LINK

Nokogiri implementation of libxslt vulnerable to heap corruption

DESCRIPTION:
Type confusion in `xsltNumberFormatGetMultipleLevel` prior to libxslt 1.1.33 could allow attackers to potentially exploit heap corruption via crafted XML data.
Nokogiri prior to version 1.10.5 contains a vulnerable version of libxslt. Nokogiri version 1.10.5 upgrades the dependency to libxslt 1.1.34, which contains a patch for this issue.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2020-26247

LINK

Nokogiri::XML::Schema trusts input by default, exposing risk of an XXE vulnerability

DESCRIPTION:
Description

In Nokogiri versions <= 1.11.0.rc3, XML Schemas parsed by `Nokogiri::XML::Schema` are **trusted** by default, allowing external resources to be accessed over the network, potentially enabling XXE or SSRF attacks.
This behavior is counter to the security policy followed by Nokogiri maintainers, which is to treat all input as **untrusted** by default whenever possible.
Please note that this security fix was pushed into a new minor version, 1.11.x, rather than a patch release to the 1.10.x branch, because it is a

breaking change for some schemas and the risk was assessed to be "Low Severity".
Affected Versions

Nokogiri `<= 1.10.10` as well as prereleases `1.11.0.rc1`, `1.11.0.rc2`, and `1.11.0.rc3`
Mitigation

There are no known workarounds for affected versions. Upgrade to Nokogiri `1.11.0.rc4` or later.
If, after upgrading to `1.11.0.rc4` or later, you wish to re-enable network access for resolution of external resources (i.e., return to the previous behavior):

> **Ensure the input is trusted. Do not enable this option for untrusted input.**
> **When invoking the `Nokogiri::XML::Schema` constructor, pass as the second parameter an instance of `Nokogiri::XML::ParseOptions` with the `NONET` flag turned off.**

So if your previous code was:

```
 # in v1.11.0.rc3 and earlier, this call allows resources to be acces
sed over the network
# but in v1.11.0.rc4 and later, this call will disallow network access f
or external resources
schema = Nokogiri::XML::Schema.new(schema)

# in v1.11.0.rc4 and later, the following is equivalent to the code ab
ove
# (the second parameter is optional, and this demonstrates its defa
ult value)
schema = Nokogiri::XML::Schema.new(schema, Nokogiri::XML::Pa
rseOptions::DEFAULT_SCHEMA)
```

Then you can add the second parameter to indicate that the input is trusted by changing it to:

```
 # in v1.11.0.rc3 and earlier, this would raise an ArgumentError
# but in v1.11.0.rc4 and later, this allows resources to be accessed
over the network
schema = Nokogiri::XML::Schema.new(trusted_schema, Nokogiri::
XML::ParseOptions.new.nononet)
```

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2020-7595

LINK

---

libxml2 2.9.10 has an infinite loop in a certain end-of-file situation

DESCRIPTION:

Nokogiri has backported the patch for CVE-2020-7595 into its vendored version of libxml2, and released this as v1.10.8
CVE-2020-7595 has not yet been addressed in an upstream libxml2 release, and so Nokogiri versions <= v1.10.7 are vulnerable.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2021-30560

LINK

Update packaged libxml2 (2.9.12 â†’ 2.9.13) and libxslt (1.1.34 â†’ 1.1.35)

DESCRIPTION:

SUMMARY

Nokogiri v1.13.2 upgrades two of its packaged dependencies:

- **vendored libxml2 from v2.9.12 to v2.9.13**
- **vendored libxslt from v1.1.34 to v1.1.35**

Those library versions address the following upstream CVEs:

- **libxslt: CVE-2021-30560 (CVSS 8.8, High severity)**
- **libxml2: CVE-2022-23308 (Unspecified severity, see more information below)**

Those library versions also address numerous other issues including performance improvements, regression fixes, and bug fixes, as well as memory leaks and other use-after-free issues that were not assigned CVEs. Please note that this advisory only applies to the CRuby implementation of Nokogiri < 1.13.2, and only if the packaged libraries are being used. If you've overridden defaults at installation time to use system libraries instead of packaged libraries, you should instead pay attention to your distro's `libxml2` and `libxslt` release announcements.

MITIGATION

Upgrade to Nokogiri >= 1.13.2.
Users who are unable to upgrade Nokogiri may also choose a more complicated mitigation: compile and link an older version Nokogiri against external libraries libxml2 >= 2.9.13 and libxslt >= 1.1.35, which will also address these same CVEs.

IMPACT

libxslt CVE-2021-30560	**CVSS3 score: 8.8 (High)**

Fixed by https://gitlab.gnome.org/GNOME/libxslt/-/commit/50f9c9c
All versions of libxslt prior to v1.1.35 are affected.
Applications using untrusted XSL stylesheets to transform XML are vulnerable to a denial-of-service attack and should be upgraded immediately. libxml2 CVE-2022-23308 * As of the time this security advisory was published, there is no officially published information available about this CVE's severity. The above NIST link does not yet have a published record, and the libxml2 maintainer has declined to provide a severity score. * Fixed by https://gitlab.gnome.org/GNOME/libxml2/-/commit/652dd12 * Further explanation is at https://mail.gnome.org/archives/xml/2022-February/msg00015.html
The upstream commit and the explanation linked above indicate that an application may be vulnerable to a denial of service, memory disclosure, or code execution if it parses an untrusted document with parse options `DTDVALID` set to true, and `NOENT` set to false.
An analysis of these parse options:

While `NOENT` is off by default for Document, DocumentFragment, Reader, and Schema parsing, it is on by default for XSLT (stylesheet) parsing in Nokogiri v1.12.0 and later.
`DTDVALID` is an option that Nokogiri does not set for any operations, and so this CVE applies only to applications setting this option explicitly.

It seems reasonable to assume that any application explicitly setting the parse option `DTDVALID` when parsing untrusted documents is vulnerable and should be upgraded immediately.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2021-3517

LINK

Nokogiri contains libxml Out-of-bounds Write vulnerability

DESCRIPTION:

There is a flaw in the xml entity encoding functionality of libxml2 in versions before 2.9.11. An attacker who is able to supply a crafted file to be processed by an application linked with the affected functionality of libxml2 could trigger an out-of-bounds read. The most likely impact of this flaw is to application availability, with some potential impact to confidentiality and integrity if an attacker is able to use memory information to further exploit the application.
Nokogiri prior to version 1.11.4 used a vulnerable version of libxml2. Nokogiri 1.11.4 updated libxml2 to version 2.9.11 to address this and other vulnerabilities in libxml2.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2021-3518

LINK

Nokogiri Implements libxml2 version vulnerable to use-after-free

DESCRIPTION:

There's a flaw in libxml2 in versions before 2.9.11. An attacker who is able to submit a crafted file to be processed by an application linked with libxml2 could trigger a use-after-free. The greatest impact from this flaw is to confidentiality, integrity, and availability.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2021-3537

LINK

Nokogiri Implements libxml2 version vulnerable to null pointer dereferencing

DESCRIPTION:
A vulnerability found in libxml2 in versions before 2.9.11 shows that it did not propagate errors while parsing XML mixed content, causing a NULL dereference. If an untrusted XML document was parsed in recovery mode and post-validated, the flaw could be used to crash the application. The highest threat from this vulnerability is to system availability.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2021-41098

LINK

Improper Restriction of XML External Entity Reference (XXE) in Nokogiri on JRuby

DESCRIPTION:
Severity

The Nokogiri maintainers have evaluated this as **High Severity** 7.5 (CVSS3.0) for JRuby users. (This security advisory does not apply to CRuby users.)
Impact

In Nokogiri v1.12.4 and earlier, **on JRuby only**, the SAX parser resolves external entities by default.
Users of Nokogiri on JRuby who parse untrusted documents using any of these classes are affected:

Nokogiri::XML::SAX::Parser
Nokogiri::HTML4::SAX::Parser or its alias
Nokogiri::HTML::SAX::Parser
Nokogiri::XML::SAX::PushParser
Nokogiri::HTML4::SAX::PushParser or its alias
Nokogiri::HTML::SAX::PushParser

Mitigation

JRuby users should upgrade to Nokogiri v1.12.5 or later. There are no workarounds available for v1.12.4 or earlier.
CRuby users are not affected.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2022-23437

LINK

XML Injection in Xerces Java affects Nokogiri

DESCRIPTION:
SUMMARY
Nokogiri v1.13.4 updates the vendored `xerces:xercesImpl` from 2.12.0 to 2.12.2, which addresses CVE-2022-23437. That CVE is scored as CVSS 6.5 "Medium" on the NVD record.
Please note that this advisory only applies to the **JRuby** implementation of Nokogiri `< 1.13.4`.
MITIGATION
Upgrade to Nokogiri `>= v1.13.4`.
IMPACT
CVE-2022-23437 in xerces-J

Severity:
Medium
Type: CWE-91 XML Injection (aka Blind XPath Injection)
Description: There's a vulnerability within the Apache Xerces Java (XercesJ) XML parser when handling specially crafted XML document payloads. This causes, the XercesJ XML parser to wait in an infinite loop, which may sometimes consume system resources for prolonged duration. This vulnerability is present within XercesJ version 2.12.1 and the previous versions.
See also: https://github.com/advisories/GHSA-h65f-jvqw-m9fj

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2022-24836

LINK

Inefficient Regular Expression Complexity in Nokogiri

DESCRIPTION:
SUMMARY
Nokogiri `< v1.13.4` contains an inefficient regular expression that is susceptible to excessive backtracking when attempting to detect encoding in HTML documents.
MITIGATION
Upgrade to Nokogiri `>= 1.13.4`.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2022-24839

LINK

Denial of Service (DoS) in Nokogiri on JRuby

DESCRIPTION:

SUMMARY

Nokogiri `v1.13.4` updates the vendored `org.cyberneko.html` library to `1.9.22.noko2` which addresses CVE-2022-24839. That CVE is rated 7.5 (High Severity).
See GHSA-9849-p7jc-9rmv for more information.
Please note that this advisory only applies to the **JRuby** implementation of Nokogiri `< 1.13.4`.

MITIGATION

Upgrade to Nokogiri `>= 1.13.4`.

IMPACT

CVE-2022-24839 in nekohtml

Severity: High
7.5
Type: CWE-400 Uncontrolled Resource Consumption
Description: The fork of `org.cyberneko.html` used by Nokogiri (Rubygem) raises a `java.lang.OutOfMemoryError` exception when parsing ill-formed HTML markup.
See also: GHSA-9849-p7jc-9rmv

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
CVE-2022-29181

LINK

Improper Handling of Unexpected Data Type in Nokogiri

DESCRIPTION:

Summary

Nokogiri `< v1.13.6` does not type-check all inputs into the XML and HTML4 SAX parsers. For CRuby users, this may allow specially crafted untrusted inputs to cause illegal memory access errors (segfault) or reads from unrelated memory.

Severity

The Nokogiri maintainers have evaluated this as **High 8.2** (CVSS3.1).

Mitigation

CRuby users should upgrade to Nokogiri `>= 1.13.6`.
JRuby users are not affected.

Workarounds

To avoid this vulnerability in affected applications, ensure the untrusted input is a `String` by calling `#to_s` or equivalent.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-2qc6-mcvw-92cw

LINK

Update bundled libxml2 to v2.10.3 to resolve multiple CVEs

DESCRIPTION:

Summary

Nokogiri v1.13.9 upgrades the packaged version of its dependency libxml2 to v2.10.3 from v2.9.14.
libxml2 v2.10.3 addresses the following known vulnerabilities:

CVE-2022-2309 **CVE-2022-40304** **CVE-2022-40303**

Please note that this advisory only applies to the CRuby implementation of Nokogiri `< 1.13.9`, and only if the *packaged* libraries are being used. If you've overridden defaults at installation time to use *system* libraries instead of packaged libraries, you should instead pay attention to your distro's `libxml2` release announcements.
Mitigation

Upgrade to Nokogiri `>= 1.13.9`.
Users who are unable to upgrade Nokogiri may also choose a more complicated mitigation: compile and link Nokogiri against external libraries libxml2 `>= 2.10.3` which will also address these same issues.
Impact

libxml2 CVE-2022-2309

CVSS3 score: Under evaluation
Type: Denial of service

Description: NULL Pointer Dereference allows attackers to cause a denial of service (or application crash). This only applies when lxml is used together with libxml2 2.9.10 through 2.9.14. libxml2 2.9.9 and earlier are not affected. It allows triggering crashes through forged input data, given a vulnerable code sequence in the application. The vulnerability is caused by the iterwalk function (also used by the canonicalize function). Such code shouldn't be in wide-spread use, given that parsing + iterwalk would usually be replaced with the more efficient iterparse function. However, an XML converter that serialises to C14N would also be vulnerable, for example, and there are legitimate use cases for this code sequence. If untrusted input is received (also remotely) and processed via iterwalk function, a crash can be triggered.

Nokogiri maintainers investigated at #2620 and determined this CVE does not affect Nokogiri users.

libxml2 CVE-2022-40304

CVSS3 score: Unspecified upstream
Type: Data corruption, denial of service
Description: When an entity reference cycle is detected, the entity content is cleared by setting its first byte to zero. But the entity content might be allocated from a dict. In this case, the dict entry becomes corrupted leading to all kinds of logic errors, including memory errors like double-frees.

See https://gitlab.gnome.org/GNOME/libxml2/-/commit/644a89e080bced793295f61f18aac8cfad6bece2

libxml2 CVE-2022-40303

CVSS3 score: Unspecified upstream
Type: Integer overflow
Description: Integer overflows with XML*PARSE*HUGE

See https://gitlab.gnome.org/GNOME/libxml2/-/commit/c846986356fc149915a74972bf198abc266bc2c0

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-353f-x4gh-cqq8

LINK

Nokogiri patches vendored libxml2 to resolve multiple CVEs

DESCRIPTION:

SUMMARY

Nokogiri v1.18.9 patches the vendored libxml2 to address CVE-2025-6021, CVE-2025-6170, CVE-2025-49794, CVE-2025-49795, and CVE-2025-49796.

IMPACT AND SEVERITY

CVE-2025-6021

A flaw was found in libxml2's xmlBuildQName function, where integer overflows in buffer size calculations can lead to a stack-based buffer overflow. This issue can result in memory corruption or a denial of service when processing crafted input.
NVD claims a severity of 7.5 High
(CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H)
Fixed by applying https://gitlab.gnome.org/GNOME/libxml2/-/commit/17d950ae
CVE-2025-6170

A flaw was found in the interactive shell of the xmllint command-line tool, used for parsing XML files. When a user inputs an overly long command, the program does not check the input size properly, which can cause it to crash. This issue might allow attackers to run harmful code in rare configurations without modern protections.
NVD claims a severity of 2.5 Low
(CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:N/I:N/A:L)
Fixed by applying https://gitlab.gnome.org/GNOME/libxml2/-/commit/5e9ec5c1
CVE-2025-49794

A use-after-free vulnerability was found in libxml2. This issue occurs when parsing XPath elements under certain circumstances when the XML schematron has the schema elements. This flaw allows a malicious actor to craft a malicious XML document used as input for libxml, resulting in the

program's crash using libxml or other possible undefined behaviors.
NVD claims a severity of 9.1 Critical
(CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:H)
Fixed by applying https://gitlab.gnome.org/GNOME/libxml2/-/commit/81cef8c5
CVE-2025-49795

A NULL pointer dereference vulnerability was found in libxml2 when processing XPath XML expressions. This flaw allows an attacker to craft a malicious XML input to libxml2, leading to a denial of service.
NVD claims a severity of 7.5 High
(CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H)
Fixed by applying https://gitlab.gnome.org/GNOME/libxml2/-/commit/62048278
CVE-2025-49796

A vulnerability was found in libxml2. Processing certain sch:name elements from the input XML file can trigger a memory corruption issue. This flaw allows an attacker to craft a malicious XML input file that can lead libxml to crash, resulting in a denial of service or other possible undefined behavior due to sensitive data being corrupted in memory.
NVD claims a severity of 9.1 Critical
(CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:H)
Fixed by applying https://gitlab.gnome.org/GNOME/libxml2/-/commit/81cef8c5

AFFECTED VERSIONS

Nokogiri < 1.18.9 when using CRuby (MRI) with vendored libxml2

PATCHED VERSIONS

Nokogiri >= 1.18.9

MITIGATION

Upgrade to Nokogiri v1.18.9 or later.
Users who are unable to upgrade Nokogiri may also choose a more complicated mitigation: compile and link Nokogiri against patched external libxml2 libraries which will also address these same issues.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-5prr-v3j2-97mh

LINK

Nokogiri: Possible Out-of-Bounds Read in `Nokogiri::XML::NodeSet#[]`

DESCRIPTION:

Summary

`Nokogiri::XML::NodeSet#[]` (and its alias `#slice`) checked the requested index against the node set's bounds using a 32-bit-truncated copy of the index. A large negative index could pass the check and then be used at full width, reading outside the node set's storage. On CRuby this is an out-of-bounds read that typically crashes the process; on JRuby it is not memory-unsafe but returns an incorrect node.
Nokogiri 1.19.4 performs the bounds check against the full-width index.
Severity

The Nokogiri maintainers have evaluated this as medium severity.
Exploitation requires an application to pass an attacker-controlled integer to `NodeSet#[]`. The primary impact is a controlled crash (denial of service), with potential for memory disclosure on CRuby.
On JRuby, Nokogiri is not affected by this vulnerability.
Mitigation

Upgrade to Nokogiri 1.19.4 or later.
As a workaround, applications that index a `NodeSet` with externally-supplied integers can validate the index against `node_set.length` before use, or avoid passing untrusted values as an index.
Credit

This issue was responsibly reported by Zheng Yu from depthfirst.com.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-5v8h-3h3q-446p

LINK

Nokogiri: Possible Use-After-Free when `Nokogiri::XML::Document#encoding=` raises an exception

DESCRIPTION:

Summary

Calling `Document#encoding=` with an invalid encoding (e.g., a non-string, or a string containing a null byte) raises an exception, but only after freeing the document's current encoding string without replacing it. The document is left referencing freed memory, so the next call to `Document#encoding` reads invalid memory, which can cause a segfault or leak freed bytes into a Ruby `String`.
Affects the CRuby (libxml2) implementation only; JRuby is not affected.
Severity

The Nokogiri maintainers have evaluated this as low severity. Reaching it requires an unusual API-usage pattern that does not arise during normal use. The application must pass an invalid encoding to `Document#encoding=`, rescue the resulting exception, and then continue using the same document. Nokogiri 1.19.4 makes this pattern safe with no change to the public API. The document no longer references freed memory after the exception is raised.
Mitigation

Upgrade to Nokogiri 1.19.4 or later.
If users are unable to upgrade, avoid passing attacker-controlled values to `Document#encoding=`. Applications that only assign developer-authored encodings are not directly exposed.
Credit

This issue was responsibly reported by Zheng Yu from depthfirst.com.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-5w6v-399v-w3cc

LINK

Nokogiri updates packaged libxml2 to v2.13.8 to resolve CVE-2025-32414 and CVE-2025-32415

DESCRIPTION:

SUMMARY

Nokogiri v1.18.8 upgrades its dependency libxml2 to v2.13.8.
libxml2 v2.13.8 addresses:

CVE-2025-32414
described at
https://gitlab.gnome.org/GNOME/libxml2/-/issues/889

CVE-2025-32415
described at
https://gitlab.gnome.org/GNOME/libxml2/-/issues/890

IMPACT

CVE-2025-32414: No impact

In libxml2 before 2.13.8 and 2.14.x before 2.14.2, out-of-bounds memory access can occur in the Python API (Python bindings) because of an incorrect return value. This occurs in xmlPythonFileRead and xmlPythonFileReadRaw because of a difference between bytes and characters.
There is no impact from this CVE for Nokogiri users.
CVE-2025-32415: Low impact

In libxml2 before 2.13.8 and 2.14.x before 2.14.2, xmlSchemaIDCFillNodeTables in xmlschemas.c has a heap-based buffer under-read. To exploit this, a crafted XML document must be validated against an XML schema with certain identity constraints, or a crafted XML schema must be used.

In the upstream issue, further context is provided by the maintainer:

> *The bug affects validation against untrusted XML Schemas (.xsd) and validation of untrusted documents against trusted Schemas if they make use of xsd:keyref in combination with recursively defined types that have additional identity constraints.*

MITRE has published a severity score of 2.9 LOW (CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L) for this CVE.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-7rrm-v45f-jp64

LINK

Update packaged dependency libxml2 from 2.9.10 to 2.9.12

DESCRIPTION:
Summary

Nokogiri v1.11.4 updates the vendored libxml2 from v2.9.10 to v2.9.12 which addresses:

- **CVE-2019-20388 (Medium severity)**
- **CVE-2020-24977 (Medium severity)**
- **CVE-2021-3517 (Medium severity)**
- **CVE-2021-3518 (Medium severity)**
- **CVE-2021-3537 (Low severity)**
- **CVE-2021-3541 (Low severity)**

Note that two additional CVEs were addressed upstream but are not relevant to this release. CVE-2021-3516 via `xmllint` is not present in Nokogiri, and

CVE-2020-7595 has been patched in Nokogiri since v1.10.8 (see #1992). Please note that this advisory only applies to the CRuby implementation of Nokogiri `< 1.11.4`, and only if the packaged version of libxml2 is being used. If you've overridden defaults at installation time to use system libraries instead of packaged libraries, you should instead pay attention to your distro's `libxml2` release announcements.
Mitigation

Upgrade to Nokogiri `>= 1.11.4`.
Impact

I've done a brief analysis of the published CVEs that are addressed in this upstream release. The libxml2 maintainers have not released a canonical set of CVEs, and so this list is pieced together from secondary sources and may be incomplete.
All information below is sourced from security.archlinux.org, which appears to have the most up-to-date information as of this analysis.
CVE-2019-20388

Severity: Medium **Type: Denial of service**
Description: A memory leak was found in the xmlSchemaValidateStream function of libxml2. Applications that use this library may be vulnerable to memory not being freed leading to a denial of service.
Fixed: https://gitlab.gnome.org/GNOME/libxml2/commit/7ffcd44d7e6c46704f8af0321d9314cd26e0e18a

Verified that the fix commit first appears in v2.9.11. It seems possible that this issue would be present in programs using Nokogiri < v1.11.4.
CVE-2020-7595

Severity: Medium **Type: Denial of service**
Description: xmlStringLenDecodeEntities in parser.c in libxml2 2.9.10 has an infinite loop in a certain end-of-file situation.
Fixed: https://gitlab.gnome.org/GNOME/libxml2/commit/0e1a49c8907645d2e155f0d89d4d9895ac5112b5

This has been patched in Nokogiri since v1.10.8 (see #1992).
CVE-2020-24977

Severity: Medium **Type: Information disclosure**
Description: GNOME project libxml2 <= 2.9.10 has a global buffer over-read vulnerability in xmlEncodeEntitiesInternal at libxml2/entities.c.

Fixed:
https://gitlab.gnome.org/GNOME/libxml2/commit/50f06b3efb638efb0abd95dc62dca05ae67882c2

Verified that the fix commit first appears in v2.9.11. It seems possible that this issue would be present in programs using Nokogiri < v1.11.4.

CVE-2021-3516

Severity:
Medium
Type: Arbitrary code execution (no remote vector)
Description: A use-after-free security issue was found libxml2 before version 2.9.11 when "xmllint --html --push" is used to process crafted files.
Issue: https://gitlab.gnome.org/GNOME/libxml2/-/issues/230
Fixed: https://gitlab.gnome.org/GNOME/libxml2/-/commit/1358d157d0bd83be1dfe356a69213df9fac0b539

Verified that the fix commit first appears in v2.9.11. This vector does not exist within Nokogiri, which does not ship `xmllint`.

CVE-2021-3517

Severity: **Medium**
Type: Arbitrary code execution
Description: A heap-based buffer overflow was found in libxml2 before version 2.9.11 when processing truncated UTF-8 input.
Issue: https://gitlab.gnome.org/GNOME/libxml2/-/issues/235
Fixed: https://gitlab.gnome.org/GNOME/libxml2/-/commit/bf22713507fe1fc3a2c4b525cf0a88c2dc87a3a2

Verified that the fix commit first appears in v2.9.11. It seems possible that this issue would be present in programs using Nokogiri < v1.11.4.

CVE-2021-3518

Severity: **Medium**
Type: Arbitrary code execution
Description: A use-after-free security issue was found in libxml2 before version 2.9.11 in xmlXIncludeDoProcess() in xinclude.c when processing crafted files.
Issue: https://gitlab.gnome.org/GNOME/libxml2/-/issues/237
Fixed: https://gitlab.gnome.org/GNOME/libxml2/-/commit/1098c30a040e72a4654968547f415be4e4c40fe7

Verified that the fix commit first appears in v2.9.11. It seems possible that this issue would be present in programs using Nokogiri < v1.11.4.
CVE-2021-3537

Severity: Low
Type: Denial of service
Description: It was found that libxml2 before version 2.9.11 did not propagate errors while parsing XML mixed content, causing a NULL dereference. If an untrusted XML document was parsed in recovery mode and post-validated, the flaw could be used to crash the application.
Issue: https://gitlab.gnome.org/GNOME/libxml2/-/issues/243
Fixed: https://gitlab.gnome.org/GNOME/libxml2/-/commit/babe75030c7f64a37826bb3342317134568bef61

Verified that the fix commit first appears in v2.9.11. It seems possible that this issue would be present in programs using Nokogiri < v1.11.4.
CVE-2021-3541

Severity: Low
Type: Denial of service
Description: A security issue was found in libxml2 before version 2.9.11. Exponential entity expansion attack its possible bypassing all existing protection mechanisms and leading to denial of service.
Fixed: https://gitlab.gnome.org/GNOME/libxml2/-/commit/8598060bacada41a0eb09d95c97744ff4e428f8e

Verified that the fix commit first appears in v2.9.11. It seems possible that this issue would be present in programs using Nokogiri < v1.11.4, however Nokogiri's default parse options prevent the attack from succeeding (it is necessary to opt into `DTDLOAD` which is off by default).
For more details supporting this analysis of this CVE, please visit #2233.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:	Version:
nokogiri	1.8.2

ID:
GHSA-8678-w3jw-xfc2

LINK

Nokogiri: XML::Schema on JRuby allows network requests when NONET is set, bypassing CVE-2020-26247

DESCRIPTION:

Summary

The `NONET` parse option, which Nokogiri turns on by default for `Nokogiri::XML::Schema` (see CVE-2020-26247), was not correctly enforced on the JRuby implementation. As a result, a schema parsed with default options could still cause external resources to be fetched over the network, potentially enabling SSRF or XXE attacks.

Nokogiri 1.19.4 replaces the scheme denylist with an allowlist. When `NONET` is enabled, only local resources (a `file:` scheme, or a relative or absolute path with no scheme) are resolved, and every network scheme is blocked, case-insensitively. This brings the JRuby behavior in line with CRuby.

Only the JRuby implementation is affected. CRuby is not affected, because libxml2's `xmlNoNetExternalEntityLoader` blocks all network schemes at the I/O layer regardless of scheme or case.

Severity

The Nokogiri maintainers have evaluated this as low severity (CVSS 2.6, `CVSS:3.0/AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:N/A:N`). It is a bypass of CVE-2020-26247, which was scored the same way.

Mitigation

Upgrade to Nokogiri 1.19.4 or later.

There are no known workarounds for affected versions.

This change properly enforces `NONET` on JRuby, which is a breaking change for any code that (perhaps unknowingly) relied on the previous behavior to load network resources with default parse options. If you trust your input and want to allow external resources to be accessed over the network, you can explicitly disable `NONET`, exactly as documented for CVE-2020-26247:

> **Ensure the input is trusted. Do not enable this option for untrusted input.**
>
> **Pass a `Nokogiri::XML::ParseOptions` with the `NONET` flag turned off:**

```
# allows resources to be accessed over the network for trusted input
schema = Nokogiri::XML::Schema.new(trusted_schema, Nokogiri::XML::ParseOptions.new.nononet)
```

References

Bypass of: https://github.com/sparklemotion/nokogiri/security/advisories/GHSA-vr8q-g5c7-m54m

Credit

This issue was responsibly reported by @bilerden.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-9cv2-cfxc-v4v2

LINK

Nokogiri: Null Pointer Dereference calling methods on uninitialized wrapper classes

DESCRIPTION:

Summary

Nokogiri contains a bug when calling certain methods on allocated-but-uninitialized native wrapper classes that inherit from `Nokogiri::XML::Node`. This caused a NULL pointer dereference that could crash the process. Nokogiri 1.19.4 checks for missing native data pointers and raises a `RuntimeError`.
JRuby is not affected.
Severity

The Nokogiri maintainers have evaluated this as low severity. This is only triggered by a programming error. It requires application code to call `.allocate` directly on a native-backed class and then invoke methods on the resulting uninitialized object. It cannot be triggered by untrusted input or

through normal use of the public API.
Mitigation

Upgrade to Nokogiri 1.19.4 or later.
Avoid calling `.allocate` directly on Nokogiri native-backed classes. Use the documented constructors and factory methods instead.
Credit

This issue was responsibly reported by Zheng Yu from depthfirst.com.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-c4rq-3m3g-8wgx

LINK

Nokogiri CSS selector tokenizer has regular expression backtracking

DESCRIPTION:
SUMMARY
Nokogiri's CSS selector tokenizer contains regular expressions whose construction may result in exponential regex backtracking on adversarial selectors. Three ReDoS vectors are addressed in this release:

- **String-literal tokenization on certain unterminated quoted-string input.**
- **String-literal tokenization on a separate class of hex-escape-rich input.**
- **Identifier tokenization on hex-escape-rich input.**

The public CSS selector methods that funnel through the affected tokenizer are `Nokogiri::CSS.xpath_for`, `Node#css`, `Node#at_css`, `Searchable#search`, and `CSS::Parser#parse`.
MITIGATION
Upgrade to Nokogiri `>= 1.19.3`.

If users are unable to upgrade, two options are available:

Avoid the use of attacker-controlled text in CSS selectors. Applications that only pass developer-authored selectors to Nokogiri are not directly exposed.

Set global `Regexp.timeout` (Ruby 3.2+, JRuby 9.4+) to bound parse time.

SEVERITY

The Nokogiri maintainers have evaluated this as **High Severity** (CVSS 7.5, `AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H`).

An attacker able to inject user-supplied text into a CSS selector parse method can cause exponential backtracking, resulting in a potential denial of service.

RESOURCES

[CWE-1333: Inefficient Regular Expression Complexity]

CREDIT

Vector 1 was responsibly reported by @colby-swandale. Vectors 2 and 3 were discovered by @flavorjones during the response to the original report.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-cgx6-hpwq-fhv5

LINK

Integer Overflow or Wraparound in libxml2 affects Nokogiri

DESCRIPTION:

Summary

Nokogiri v1.13.5 upgrades the packaged version of its dependency libxml2

from v2.9.13 to v2.9.14.
libxml2 v2.9.14 addresses CVE-2022-29824. This version also includes several security-related bug fixes for which CVEs were not created, including a potential double-free, potential memory leaks, and integer-overflow.
Please note that this advisory only applies to the CRuby implementation of Nokogiri `< 1.13.5`, and only if the *packaged* libraries are being used. If you've overridden defaults at installation time to use *system* libraries instead of packaged libraries, you should instead pay attention to your distro's `libxml2` and `libxslt` release announcements.
Mitigation

Upgrade to Nokogiri `>= 1.13.5`.
Users who are unable to upgrade Nokogiri may also choose a more complicated mitigation: compile and link Nokogiri against external libraries libxml2 `>= 2.9.14` which will also address these same issues.
Impact

libxml2 CVE-2022-29824

CVSS3 score:

Unspecified upstream
Nokogiri maintainers evaluate at 8.6 (High) (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:H).
Note that this is different from the CVSS assessed by NVD.

Type: Denial of service, information disclosure
Description: In libxml2 before 2.9.14, several buffer handling functions in buf.c (xmlBuf*) and tree.c (xmlBuffer*) don't check for integer overflows. This can result in out-of-bounds memory writes. Exploitation requires a victim to open a crafted, multi-gigabyte XML file. Other software using libxml2's buffer functions, for example libxslt through 1.1.35, is affected as well.
Fixed: https://gitlab.gnome.org/GNOME/libxml2/-/commit/2554a24

All versions of libml2 prior to v2.9.14 are affected.
Applications parsing or serializing multi-gigabyte documents (in excess of INT_MAX bytes) may be vulnerable to an integer overflow bug in buffer handling that could lead to exposure of confidential data, modification of unrelated data, or a segmentation fault resulting in a denial-of-service.
References

libxml2 v2.9.14 release notes

CVE-2022-29824

[CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer]

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-g9g8-vgvw-g3vf

LINK

Possible invalid memory read when calling `Nokogiri::XML::Node#initialize_copy_with_args` with incorrect argument type

DESCRIPTION:

SUMMARY

The protected copy helper behind Node#dup and #clone unwrapped its source argument as an xmlNode without a type check. Supplying a non-Node (e.g. a Namespace) made it read an xmlNs out of bounds, crashing the process.
Nokogiri 1.19.4 performs a type check and raises TypeError when an argument of invalid type is passed.
Only CRuby is affected. JRuby is not affected.

SEVERITY

The Nokogiri maintainers have evaluated this as low severity. This is only triggered by a programming error. It requires application code to call the protected internal initialize*copy*with_args method with an argument that is not a Nokogiri::XML::Node. Nokogiri 1.19.4 now raises TypeError instead of reading out of bounds. It cannot be triggered by untrusted input or through normal use of the public API.

MITIGATION

Upgrade to Nokogiri 1.19.4 or later. There is no workaround.

CREDIT

This issue was responsibly reported by Zheng Yu from depthfirst.com.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-mrxw-mxhj-p664

LINK

Nokogiri updates packaged libxslt to v1.1.43 to resolve multiple CVEs

DESCRIPTION:
SUMMARY
Nokogiri v1.18.4 upgrades its dependency libxslt to v1.1.43.
libxslt v1.1.43 resolves:

CVE-2025-24855: Fix use-after-free of XPath context node
CVE-2024-55549: Fix UAF related to excluded namespaces

IMPACT
CVE-2025-24855

"Use-after-free due to xsltEvalXPathStringNs leaking xpathCtxt->node"
MITRE has rated this 7.8 High
CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:C/C:N/I:H/A:H
Upstream report: https://gitlab.gnome.org/GNOME/libxslt/-/issues/128
NVD entry: https://nvd.nist.gov/vuln/detail/CVE-2025-24855

CVE-2024-55549

"Use-after-free related to excluded result prefixes"
MITRE has rated this 7.8 High
CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:C/C:N/I:H/A:H
Upstream report: https://gitlab.gnome.org/GNOME/libxslt/-/issues/127
NVD entry: https://nvd.nist.gov/vuln/detail/CVE-2024-55549

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-p67v-3w7g-wjg7

LINK

Nokogiri: Possible Use-After-Free when directly using `NokogirI::XML::XPathContext` beyond document lifetime

DESCRIPTION:

Summary

`Nokogiri::XML::XPathContext` did not keep its source document alive for garbage collection. If an `XPathContext` outlived its document and the document was collected, evaluating an XPath expression could read invalid memory and potentially segfault.
This is only reachable when application code constructs an `XPathContext` directly and lets the document become unreachable while continuing to use the context. The normal `Document#xpath`, `#css`, and related search methods are not affected, and it is not triggerable by malicious document input.
Nokogiri 1.19.4 makes `XPathContext` keep its source document alive for as long as the context exists.
Only the CRuby implementation is affected. JRuby is not affected.

Severity

The Nokogiri maintainers have evaluated this as low severity. Reaching it requires an unusual API-usage pattern that does not arise during normal use. The application must construct an `XML::XPathContext` directly and continue using it after allowing its source document to be garbage-collected. Nokogiri 1.19.4 makes this pattern safe with no change to the public API. The context now keeps its source document alive for as long as it exists.
Mitigation

Upgrade to Nokogiri 1.19.4 or later.
As a workaround, ensure the source document remains referenced for as long as any `XPathContext` created from it is in use. The standard `Document#xpath`, `#css`, and related search methods already do this and are unaffected.
Credit

This issue was responsibly reported by Zheng Yu from depthfirst.com.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-phwj-rprq-35pp

LINK

Nokogiri: Possible Use-After-Free when setting an attribute value via `Nokogiri::XML::Attr#value=` or `#content=`

DESCRIPTION:
Summary

Nokogiri's CRuby native extension could leave a Ruby wrapper pointing to freed memory when replacing the value of an XML attribute. If Ruby code had already accessed an attribute child node, `Nokogiri::XML::Attr#value=` could free the underlying native child node while the wrapper remained

reachable through the document node cache. A later use of the freed child node or a Ruby GC mark could dereference an invalid pointer, causing an invalid read and a possible segfault.
Nokogiri 1.19.4 preserves any already-wrapped attribute child nodes before replacing the attribute value.
JRuby is not affected.
Severity

The Nokogiri maintainers have evaluated this as low severity. Reaching it requires an unusual API-usage pattern that does not arise during normal use. The application must directly access an attribute's child node and then replace that same attribute's value via `Attr#value=` or `#content=`.
Nokogiri 1.19.4 makes this pattern safe with no change to the public API. Already-wrapped attribute child nodes are preserved before the value is replaced.
Mitigation

Upgrade to Nokogiri 1.19.4 or later.
As a workaround, avoid accessing attribute child nodes directly via `Attr#child` or similar before mutating the same attributeâ€™s value.
Credit

This issue was responsibly reported by Zheng Yu from depthfirst.com.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-pxvg-2qj5-37jq

LINK

Update packaged libxml2 to v2.10.4 to resolve multiple CVEs

DESCRIPTION:
Summary

Nokogiri v1.14.3 upgrades the packaged version of its dependency libxml2 to v2.10.4 from v2.10.3.
libxml2 v2.10.4 addresses the following known vulnerabilities:

- **CVE-2023-29469: Hashing of empty dict strings isn't deterministic**
- **CVE-2023-28484: Fix null deref in xmlSchemaFixupComplexType**
- **Schemas: Fix null-pointer-deref in xmlSchemaCheckCOSSTDerivedOK**

Please note that this advisory only applies to the CRuby implementation of Nokogiri `< 1.14.3`, and only if the *packaged* libraries are being used. If you've overridden defaults at installation time to use *system* libraries instead of packaged libraries, you should instead pay attention to your distro's `libxml2` release announcements.
Mitigation

Upgrade to Nokogiri `>= 1.14.3`.
Users who are unable to upgrade Nokogiri may also choose a more complicated mitigation: compile and link Nokogiri against external libraries libxml2 `>= 2.10.4` which will also address these same issues.
Impact

No public information has yet been published about the security-related issues other than the upstream commits. Examination of those changesets indicate that the more serious issues relate to libxml2 dereferencing NULL pointers and potentially segfaulting while parsing untrusted inputs.
The commits can be examined at:

- **[CVE-2023-29469] Hashing of empty dict strings isn't deterministic (09a2dd45)**
- **[CVE-2023-28484] Fix null deref in xmlSchemaFixupComplexType (647e072e)**
- **schemas: Fix null-pointer-deref in xmlSchemaCheckCOSSTDerivedOK (4c6922f7)**

VULNERABLE GEM: NOKOGIRI@1.8.2

Name: **Version:**

nokogiri **1.8.2**

ID:
GHSA-r95h-9x8f-r3f7

LINK

Nokogiri updates packaged libxml2 to v2.12.7 to resolve CVE-2024-34459

DESCRIPTION:
SUMMARY
Nokogiri v1.16.5 upgrades its dependency libxml2 to 2.12.7 from 2.12.6. libxml2 v2.12.7 addresses CVE-2024-34459:

- **described at https://gitlab.gnome.org/GNOME/libxml2/-/issues/720**
- **patched by https://gitlab.gnome.org/GNOME/libxml2/-/commit/2876ac53**

IMPACT
There is no impact to Nokogiri users because the issue is present only in libxml2's `xmllint` tool which Nokogiri does not provide or expose.
TIMELINE

- **2024-05-13 05:57 EDT, libxml2 2.12.7 release is announced**
- **2024-05-13 08:30 EDT, nokogiri maintainers begin triage**
- **2024-05-13 10:05 EDT, nokogiri v1.16.5 is released and this GHSA made public**

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-v2fc-qm4h-8hqv

LINK

Nokogiri XSLT transform has a memory leak

DESCRIPTION:

SUMMARY

Nokogiri's `Nokogiri::XSLT::Stylesheet#transform` leaks a small heap allocation when passed a Ruby string parameter containing a null byte.
For applications that pass attacker-controlled input through `XSLT.transform` parameters, this may be a vector for a denial of service attack against long-running processes.

MITIGATION

Upgrade to Nokogiri `>= 1.19.3`.
Users may also be able to mitigate this issue without upgrading by validating untrusted transform parameters before passing them to `Nokogiri::XSLT::Stylesheet#transform`.

SEVERITY

The Nokogiri maintainers have evaluated this as **Moderate Severity**, CVSS 5.3.
Each leaked allocation is approximately 24â€“32 bytes, so meaningful memory growth requires sustained attacker-controlled traffic at high call rates. The bug does not cause memory corruption, information disclosure, or any change in the behavior of the transform itself, and the string-handling exception is raised as expected.
Applications that do not pass raw attacker-controlled bytes to XSLT parameters are unlikely to be affected in practice.

RESOURCES

[CWE-401: Missing Release of Memory after Effective Lifetime]

CREDIT

This vulnerability was responsibly reported by @Captainjack-kor.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-vvfq-8hwr-qm4m

LINK

Nokogiri updates packaged libxml2 to 2.13.6 to resolve CVE-2025-24928 and CVE-2024-56171

DESCRIPTION:
SUMMARY
Nokogiri v1.18.3 upgrades its dependency libxml2 to v2.13.6.
libxml2 v2.13.6 addresses:

CVE-2025-24928
described at
https://gitlab.gnome.org/GNOME/libxml2/-/issues/847

CVE-2024-56171
described at
https://gitlab.gnome.org/GNOME/libxml2/-/issues/828

IMPACT
CVE-2025-24928

Stack-buffer overflow is possible when reporting DTD validation errors if the input contains a long (~3kb) QName prefix.
CVE-2024-56171

Use-after-free is possible during validation against untrusted XML Schemas (.xsd) and, potentially, validation of untrusted documents against trusted Schemas if they make use of `xsd:keyref` in combination with recursively defined types that have additional identity constraints.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name: **Version:**

nokogiri **1.8.2**

ID:
GHSA-wfpw-mmfh-qq69

LINK

Nokogiri: Possible Use-After-Free in XInclude Processing

DESCRIPTION:

Summary

XInclude substitution performed by `Nokogiri::XML::Node#do_xinclude` replaced each `<xi:include>` in place, freeing the include node along with its children (such as `<xi:fallback>` and its descendants) and any namespaces declared on them. If an application had already exposed one of those nodes or namespaces to Ruby, the corresponding Ruby object was left pointing at freed memory. Using the object could result in invalid reads or writes to memory.
Nokogiri 1.19.4 substitutes each `<xi:include>` on a defensive copy by default, so the structures libxml2 frees are never the ones bound to live Ruby objects.
Only the CRuby implementation is affected; JRuby is not affected.
Severity

The Nokogiri maintainers have evaluated this as low severity. Reaching it requires an unusual API-usage pattern that does not arise during normal use. The application must parse a document without XInclude, traverse into an `<xi:include>` subtree to expose its nodes or namespaces to Ruby, and only then invoke XInclude processing. The common case, requesting XInclude at parse time, operates on a freshly parsed document whose nodes are not yet exposed to Ruby and is not affected. Nokogiri 1.19.4 makes this pattern safe by default and requires no change to application code.
Mitigation

Upgrade to Nokogiri 1.19.4 or later.
As a workaround for earlier versions, perform XInclude substitution at parse time (with the `xinclude` parse option) rather than calling `#do_xinclude` on a document that has already been traversed. A freshly parsed document has no nodes exposed to Ruby, so the substitution is safe.
Credit

This issue was responsibly reported by Zheng Yu from depthfirst.com.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-wjv4-x9w8-wm3h

LINK

Nokogiri: Possible Use-After-Free when setting `Document#root=` to an invalid node type

DESCRIPTION:

Summary

`Nokogiri::XML::Document#root=` validated only that the new root was a `Nokogiri::XML::Node`, allowing a DTD node to be set as the document root. The result is a heap use-after-free during garbage collection or finalization, leading to an invalid memory read or potentially a segfault.
Nokogiri 1.19.4 restricts `Document#root=` to element nodes, raising `TypeError` for any other node type.
This memory-safety issue affects only the CRuby implementation (libxml2). The JRuby implementation was not affected; the same input validation was added there for behavioral parity.

Severity

The Nokogiri maintainers have evaluated this as low severity. This is only triggered by a programming error. It requires application code to assign a non-element node such as a DTD as the document root via `Document#root=`. Nokogiri 1.19.4 now raises `TypeError` instead of allowing a use-after-free. It cannot be triggered by untrusted input or through normal use of the public API.

Mitigation

Upgrade to Nokogiri 1.19.4 or later.
As a workaround, applications that cannot upgrade should avoid assigning a DTD (or any non-element node) via `Document#root=`.

Credit

This issue was responsibly reported by Zheng Yu from depthfirst.com.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-wx95-c6cv-8532

LINK

Nokogiri does not check the return value from xmlC14NExecute

DESCRIPTION:

SUMMARY

Nokogiri's CRuby extension fails to check the return value from `xmlC14NExecute` in the method `Nokogiri::XML::Document#canonicalize` and `Nokogiri::XML::Node#canonicalize`. When canonicalization fails, an empty string is returned instead of raising an exception. This incorrect return value may allow downstream libraries to accept invalid or incomplete canonicalized XML, which has been demonstrated to enable signature validation bypass in SAML libraries.
JRuby is not affected, as the Java implementation correctly raises `RuntimeError` on canonicalization failure.

MITIGATION

Upgrade to Nokogiri `>= 1.19.1`.

SEVERITY

The maintainers have assessed this as **Medium** severity. Nokogiri itself is a parsing library without a clear security boundary related to canonicalization, so the direct impact is that a method returns incorrect data on invalid input. However, this behavior was exploited in practice to bypass SAML signature validation in downstream libraries (see References).

CREDIT

This vulnerability was responsibly reported by HackerOne researcher `d4d`.

VULNERABLE GEM: NOKOGIRI@1.8.2

Name:
nokogiri

Version:
1.8.2

ID:
GHSA-xc9x-jj77-9p9j

LINK

Use-after-free in libxml2 via Nokogiri::XML::Reader

DESCRIPTION:

Summary

Nokogiri upgrades its dependency libxml2 as follows: - v1.15.6 upgrades libxml2 to 2.11.7 from 2.11.6 - v1.16.2 upgrades libxml2 to 2.12.5 from 2.12.4
libxml2 v2.11.7 and v2.12.5 address the following vulnerability: CVE-2024-25062 / https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2024-25062 - described at https://gitlab.gnome.org/GNOME/libxml2/-/issues/604 - patched by https://gitlab.gnome.org/GNOME/libxml2/-/commit/92721970
Please note that this advisory only applies to the CRuby implementation of Nokogiri, and only if the packaged libraries are being used. If you've overridden defaults at installation time to use system libraries instead of packaged libraries, you should instead pay attention to your distro's libxml2 release announcements.
JRuby users are not affected.
Severity

The Nokogiri maintainers have evaluated this as **Moderate**.
Impact

From the CVE description, this issue applies to the `xmlTextReader` module (which underlies `Nokogiri::XML::Reader`):

> *When using the XML Reader interface with DTD validation and XInclude expansion enabled, processing crafted XML documents can lead to an xmlValidatePopElement use-after-free.*

Mitigation

Upgrade to Nokogiri `~> 1.15.6` or `>= 1.16.2`.
Users who are unable to upgrade Nokogiri may also choose a more complicated mitigation: compile and link Nokogiri against patched external

libxml2 libraries which will also address these same issues.

VULNERABLE GEM: OAUTH2@1.4.0

Name:
oauth2

Version:
1.4.0

ID:
CVE-2026-54603

LINK

OAuth2::Client#request: Protocol-relative redirect Location overrides authority, leaking bearer Authorization to attacker host

DESCRIPTION:

SUMMARY

When an application uses OAuth2::Client (typically via an OAuth2::AccessToken) and the configured authorization server returns a redirect whose Location header is a protocol-relative URI of the form //attacker.example/leak, OAuth2::Client#request resolves the redirect with response.response.env.url.merge(location). Per RFC 3986 Â§5.2, an input that starts with // is a network-path reference and replaces the authority of the base URL: URI("http://idp.trusted/userinfo").merge("//attacker.example/leak") returns http://attacker.example/leak. The recursive request(verb, full*location, req*opts) call then re-sends the request to the attacker host while preserving the Authorization: Bearer header that OAuth2::AccessToken#configure*authentication! installed on req*opts[:headers] for the original request.
The result is a one-shot cross-origin credential disclosure: any 30x response from the IdP that an attacker can influence (a compromised endpoint, a tenant-controlled IdP in a multi-tenant deployment, or an open-redirect handler that does not normalise the Location it emits) can extract the bearer access token of the calling user.

AFFECTED

oauth2 v2.0.21 and all prior versions back to and including v0.4.0.

The underlying unsafe redirect-following behavior that reuses headers starts in v0.4.0 via b944da5. The vulnerability was retained when the code was refactored to a redirect helper, and in this form has been present in OAuth2::Client#request since v2.0.0 via b944da5.

Ruby 4.0.5 on arm64-darwin25. The behaviour of URI#merge for protocol-relative inputs is RFC-conformant and the same on every supported Ruby (â‰¥ 2.x). Adapter independence: confirmed against the default faraday 2.14.2 + faraday-net_http 3.4.3 stack. The URI#merge call is in oauth2 itself, not in Faraday, so the issue is not adapter-specific.

IMPACT

A consumer that uses OAuth2::AccessToken#get / #post / #request against an IdP whose redirect target an attacker can influence (open redirect, malicious tenant, or in-path adversary) loses two things at once:

Cross-origin credential disclosure. The connection-scoped Authorization: Bearer header attached by OAuth2::AccessToken#configure_authentication! is sent to the attacker host on the very next request, with no second user interaction.

SSRF from the application server. The OAuth2 client follows the redirect on behalf of the application, so the host that ultimately receives the request is one the attacker chooses â€” useful for hitting internal addresses (//169.254.169.254/..., //127.0.0.1:.../...) that the application server can reach but the attacker cannot.

The combined primitive is stronger than the usual cross-origin-redirect leak because no application-level cooperation is required and no Location: http://attacker/... is needed â€” the protocol-relative //attacker/x form slips past naive scheme-based Location filters that allow same-scheme-implicit redirects.

CREDIT

Reported by tonghuaroot.

VULNERABLE GEM: OMNIAUTH@1.8.1

Name:
omniauth

Version:
1.8.1

ID:
CVE-2015-9284

LINK

CSRF vulnerability in OmniAuth's request phase

DESCRIPTION:

The request phase of the OmniAuth Ruby gem is vulnerable to Cross-Site Request Forgery (CSRF) when used as part of the Ruby on Rails framework, allowing accounts to be connected without user intent, user interaction, or feedback to the user. This permits a secondary account to be able to sign into the web application as the primary account.
In order to mitigate this vulnerability, Rails users should consider using the `omniauth-rails_csrf_protection` gem.
More info is available here:
https://github.com/omniauth/omniauth/wiki/Resolving-CVE-2015-9284

VULNERABLE GEM: OMNIAUTH@1.8.1

Name:
omniauth

Version:
1.8.1

ID:
CVE-2020-36599

LINK

OmniAuth's `lib/omniauth/failure_endpoint.rb` does not escape

`message_key` value

DESCRIPTION:
lib/omniauth/failure*endpoint.rb in OmniAuth before 1.9.2 (and before 2.0) does not escape the message*key value.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2018-16471

LINK

Possible XSS vulnerability in Rack

DESCRIPTION:
There is a possible vulnerability in Rack. This vulnerability has been assigned the CVE identifier CVE-2018-16471.
Versions Affected: All. Not affected: None. Fixed Versions: 2.0.6, 1.6.11

IMPACT
There is a possible XSS vulnerability in Rack. Carefully crafted requests can impact the data returned by the `scheme` method on `Rack::Request`. Applications that expect the scheme to be limited to "http" or "https" and do not escape the return value could be vulnerable to an XSS attack.
Vulnerable code looks something like this:

```
<%= request.scheme.html_safe %>
```

Note that applications using the normal escaping mechanisms provided by Rails may not impacted, but applications that bypass the escaping mechanisms, or do not use them may be vulnerable.

All users running an affected release should either upgrade or use one of the workarounds immediately.

RELEASES

The 2.0.6 and 1.6.11 releases are available at the normal locations.

WORKAROUNDS

The following monkey patch can be applied to work around this issue:

```
  require "rack"
require "rack/request"

class Rack::Request
SCHEME_WHITELIST = %w(https http).freeze

def scheme
  if get_header(Rack::HTTPS) == 'on'
    'https'
  elsif get_header(HTTP_X_FORWARDED_SSL) == 'on'
    'https'
  elsif forwarded_scheme
    forwarded_scheme
  else
    get_header(Rack::RACK_URL_SCHEME)
  end
end

def forwarded_scheme
  scheme_headers = [
    get_header(HTTP_X_FORWARDED_SCHEME),
    get_header(HTTP_X_FORWARDED_PROTO).to_s.split(',')[0]
  ]

  scheme_headers.each do |header|
    return header if SCHEME_WHITELIST.include?(header)
  end

  nil
end
end
```

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2019-16782

LINK

Possible information leak / session hijack vulnerability

DESCRIPTION:

There's a possible information leak / session hijack vulnerability in Rack. Attackers may be able to find and hijack sessions by using timing attacks targeting the session id. Session ids are usually stored and indexed in a database that uses some kind of scheme for speeding up lookups of that session id. By carefully measuring the amount of time it takes to look up a session, an attacker may be able to find a valid session id and hijack the session.
The session id itself may be generated randomly, but the way the session is indexed by the backing store does not use a secure comparison.
Impact:
The session id stored in a cookie is the same id that is used when querying the backing session storage engine. Most storage mechanisms (for example a database) use some sort of indexing in order to speed up the lookup of that id. By carefully timing requests and session lookup failures, an attacker may be able to perform a timing attack to determine an existing session id and hijack that session.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2020-8161

LINK

Directory traversal in Rack::Directory app bundled with Rack

DESCRIPTION:

There was a possible directory traversal vulnerability in the Rack::Directory app that is bundled with Rack.
Versions Affected: rack < 2.2.0 Not affected: Applications that do not use Rack::Directory. Fixed Versions: 2.1.3, >= 2.2.0

IMPACT

If certain directories exist in a director that is managed by `Rack::Directory`, an attacker could, using this vulnerability, read the contents of files on the server that were outside of the root specified in the Rack::Directory initializer.

WORKAROUNDS

Until such time as the patch is applied or their Rack version is upgraded, we recommend that developers do not use Rack::Directory in their applications.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2020-8184

LINK

Percent-encoded cookies can be used to overwrite existing prefixed cookie names

DESCRIPTION:

It is possible to forge a secure or host-only cookie prefix in Rack using an arbitrary cookie write by using URL encoding (percent-encoding) on the name of the cookie. This could result in an application that is dependent on this prefix to determine if a cookie is safe to process being manipulated into processing an insecure or cross-origin request. This vulnerability has been assigned the CVE identifier CVE-2020-8184.
Versions Affected: rack < 2.2.3, rack < 2.1.4 Not affected: Applications which

do not rely on _*Host- and* _Secure- prefixes to determine if a cookie is safe to process Fixed Versions: rack >= 2.2.3, rack >= 2.1.4

IMPACT

An attacker may be able to trick a vulnerable application into processing an insecure (non-SSL) or cross-origin request if they can gain the ability to write arbitrary cookies that are sent to the application.

WORKAROUNDS

If your application is impacted but you cannot upgrade to the released versions or apply the provided patch, this issue can be temporarily addressed by adding the following workaround:

```
 module Rack
 module Utils
   module_function def parse_cookies_header(header)
     return {} unless header
     header.split(/[;] */n).each_with_object({}) do |cookie, cookies|
       next if cookie.empty?
       key, value = cookie.split('=', 2)
       cookies[key] = (unescape(value) rescue value) unless cookies
key?(key)
     end
   end
 end
end
```

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2022-30122

LINK

Denial of Service Vulnerability in Rack Multipart Parsing

DESCRIPTION:
There is a possible denial of service vulnerability in the multipart parsing component of Rack. This vulnerability has been assigned the CVE identifier CVE-2022-30122.
Versions Affected: >= 1.2 Not affected: < 1.2 Fixed Versions: 2.0.9.1, 2.1.4.1, 2.2.3.1

IMPACT
Carefully crafted multipart POST requests can cause Rack's multipart parser to take much longer than expected, leading to a possible denial of service vulnerability.
Impacted code will use Rack's multipart parser to parse multipart posts. This includes directly using the multipart parser like this:

```
params = Rack::Multipart.parse_multipart(env)
```

But it also includes reading POST data from a Rack request object like this:

```
p request.POST # read POST data
p request.params # reads both query params and POST data
```

All users running an affected release should either upgrade or use one of the workarounds immediately.

WORKAROUNDS
There are no feasible workarounds for this issue.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2022-30123

LINK

Possible shell escape sequence injection vulnerability in Rack

DESCRIPTION:
There is a possible shell escape sequence injection vulnerability in the Lint and CommonLogger components of Rack. This vulnerability has been

assigned the CVE identifier CVE-2022-30123.
Versions Affected: All. Not affected: None Fixed Versions: 2.0.9.1, 2.1.4.1, 2.2.3.1

IMPACT

Carefully crafted requests can cause shell escape sequences to be written to the terminal via Rack's Lint middleware and CommonLogger middleware. These escape sequences can be leveraged to possibly execute commands in the victim's terminal.
Impacted applications will have either of these middleware installed, and vulnerable apps may have something like this:

```
use Rack::Lint
```

Or

```
use Rack::CommonLogger
```

All users running an affected release should either upgrade or use one of the workarounds immediately.

WORKAROUNDS

Remove these middleware from your application

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2022-44570

LINK

Denial of service via header parsing in Rack

DESCRIPTION:

There is a possible denial of service vulnerability in the Range header parsing component of Rack. This vulnerability has been assigned the CVE identifier CVE-2022-44570.
Versions Affected: >= 1.5.0 Not affected: None. Fixed Versions: 2.0.9.2, 2.1.4.2, 2.2.6.2, 3.0.4.1

Impact

Carefully crafted input can cause the Range header parsing component in Rack to take an unexpected amount of time, possibly resulting in a denial of service attack vector. Any applications that deal with Range requests (such as streaming applications, or applications that serve files) may be impacted.

Workarounds

There are no feasible workarounds for this issue.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2022-44571

LINK

Denial of Service Vulnerability in Rack Content-Disposition parsing

DESCRIPTION:

There is a denial of service vulnerability in the Content-Disposition parsing component of Rack. This vulnerability has been assigned the CVE identifier CVE-2022-44571.
Versions Affected: >= 2.0.0 Not affected: None. Fixed Versions: 2.0.9.2, 2.1.4.2, 2.2.6.1, 3.0.4.1

Impact

Carefully crafted input can cause Content-Disposition header parsing in Rack to take an unexpected amount of time, possibly resulting in a denial of service attack vector. This header is used typically used in multipart parsing. Any applications that parse multipart posts using Rack (virtually all Rails applications) are impacted.

Workarounds

There are no feasible workarounds for this issue.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2022-44572

LINK

Denial of service via multipart parsing in Rack

DESCRIPTION:
There is a denial of service vulnerability in the multipart parsing component of Rack. This vulnerability has been assigned the CVE identifier CVE-2022-44572.
Versions Affected: >= 2.0.0 Not affected: None. Fixed Versions: 2.0.9.2, 2.1.4.2, 2.2.6.1, 3.0.4.1

Impact

Carefully crafted input can cause RFC2183 multipart boundary parsing in Rack to take an unexpected amount of time, possibly resulting in a denial of service attack vector. Any applications that parse multipart posts using Rack (virtually all Rails applications) are impacted.

Workarounds

There are no feasible workarounds for this issue.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:

CVE-2023-27530

LINK

Possible DoS Vulnerability in Multipart MIME parsing

DESCRIPTION:
There is a possible DoS vulnerability in the Multipart MIME parsing code in Rack. This vulnerability has been assigned the CVE identifier CVE-2023-27530.
Versions Affected: All. Not affected: None Fixed Versions: 3.0.4.2, 2.2.6.3, 2.1.4.3, 2.0.9.3

Impact

The Multipart MIME parsing code in Rack limits the number of file parts, but does not limit the total number of parts that can be uploaded. Carefully crafted requests can abuse this and cause multipart parsing to take longer than expected.
All users running an affected release should either upgrade or use one of the workarounds immediately.

Workarounds

A proxy can be configured to limit the POST body size which will mitigate this issue.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2023-27539

LINK

Possible Denial of Service Vulnerability in Rackâ€™s header parsing

DESCRIPTION:

There is a denial of service vulnerability in the header parsing component of Rack. This vulnerability has been assigned the CVE identifier CVE-2023-27539.
Versions Affected: >= 2.0.0 Not affected: None. Fixed Versions: 2.2.6.4, 3.0.6.1

Impact

Carefully crafted input can cause header parsing in Rack to take an unexpected amount of time, possibly resulting in a denial of service attack vector. Any applications that parse headers using Rack (virtually all Rails applications) are impacted.

Workarounds

Setting Regexp.timeout in Ruby 3.2 is a possible workaround.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2024-25126

LINK

Denial of Service Vulnerability in Rack Content-Type Parsing

DESCRIPTION:
There is a possible denial of service vulnerability in the content type parsing component of Rack. This vulnerability has been assigned the CVE identifier CVE-2024-25126.
Versions Affected: >= 0.4 Not affected: < 0.4 Fixed Versions: 3.0.9.1, 2.2.8.1

Impact

Carefully crafted content type headers can cause Rackâ€™s media type parser to take much longer than expected, leading to a possible denial of service vulnerability.
Impacted code will use Rackâ€™s media type parser to parse content type headers. This code will look like below:

```
request.media_type

## OR
request.media_type_params

## OR
Rack::MediaType.type(content_type)
```

Some frameworks (including Rails) call this code internally, so upgrading is recommended!
All users running an affected release should either upgrade or use one of the workarounds immediately.

Releases

The fixed releases are available at the normal locations.

Workarounds

There are no feasible workarounds for this issue.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2024-26141

LINK

Possible DoS Vulnerability with Range Header in Rack

DESCRIPTION:
There is a possible DoS vulnerability relating to the Range request header in Rack. This vulnerability has been assigned the CVE identifier CVE-2024-26141.
Versions Affected: >= 1.3.0. Not affected: < 1.3.0 Fixed Versions: 3.0.9.1, 2.2.8.1

Impact

Carefully crafted Range headers can cause a server to respond with an

unexpectedly large response. Responding with such large responses could lead to a denial of service issue.
Vulnerable applications will use the `Rack::File` middleware or the `Rack::Utils.byte_ranges` methods (this includes Rails applications).

Releases

The fixed releases are available at the normal locations.

Workarounds

There are no feasible workarounds for this issue.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2024-26146

LINK

Possible Denial of Service Vulnerability in Rack Header Parsing

DESCRIPTION:
There is a possible denial of service vulnerability in the header parsing routines in Rack. This vulnerability has been assigned the CVE identifier CVE-2024-26146.
Versions Affected: All. Not affected: None Fixed Versions: 2.0.9.4, 2.1.4.4, 2.2.8.1, 3.0.9.1

Impact

Carefully crafted headers can cause header parsing in Rack to take longer than expected resulting in a possible denial of service issue. `Accept` and `Forwarded` headers are impacted.
Ruby 3.2 has mitigations for this problem, so Rack applications using Ruby 3.2 or newer are unaffected.

Releases

The fixed releases are available at the normal locations.

Workarounds

There are no feasible workarounds for this issue.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2025-25184

LINK

Possible Log Injection in Rack::CommonLogger

DESCRIPTION:

SUMMARY

`Rack::CommonLogger` can be exploited by crafting input that includes newline characters to manipulate log entries. The supplied proof-of-concept demonstrates injecting malicious content into logs.

DETAILS

When a user provides the authorization credentials via `Rack::Auth::Basic`, if success, the username will be put in `env['REMOTE_USER']` and later be used by `Rack::CommonLogger` for logging purposes.
The issue occurs when a server intentionally or unintentionally allows a user creation with the username contain CRLF and white space characters, or the server just want to log every login attempts. If an attacker enters a username with CRLF character, the logger will log the malicious username with CRLF characters into the logfile.

IMPACT

Attackers can break log formats or insert fraudulent entries, potentially obscuring real activity or injecting malicious data into log files.

MITIGATION

Update to the latest version of Rack.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2025-27111

LINK

Escape Sequence Injection vulnerability in Rack lead to Possible Log Injection

DESCRIPTION:

SUMMARY

`Rack::Sendfile` can be exploited by crafting input that includes newline characters to manipulate log entries.

DETAILS

The `Rack::Sendfile` middleware logs unsanitized header values from the `X-Sendfile-Type` header. An attacker can exploit this by injecting escape sequences (such as newline characters) into the header, resulting in log injection.

IMPACT

This vulnerability can distort log files, obscure attack traces, and complicate security auditing.

MITIGATION

Update to the latest version of Rack,
or
Remove usage of
`Rack::Sendfile`.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2025-27610

LINK

Local File Inclusion in Rack::Static

DESCRIPTION:

SUMMARY

`Rack::Static` can serve files under the specified `root:` even if `urls:` are provided, which may expose other files under the specified `root:` unexpectedly.

DETAILS

The vulnerability occurs because `Rack::Static` does not properly sanitize user-supplied paths before serving files. Specifically, encoded path traversal sequences are not correctly validated, allowing attackers to access files outside the designated static file directory.

IMPACT

By exploiting this vulnerability, an attacker can gain access to all files under the specified `root:` directory, provided they are able to determine then path of the file.

MITIGATION

Update to the latest version of Rack,
or
Remove usage of `Rack::Static`,
or
Ensure that `root:` points at a directory path which only contains files which should be accessed publicly.

It is likely that a CDN or similar static file server would also mitigate the issue.

VULNERABLE GEM: RACK@1.6.10

Name:

Version:

rack **1.6.10**

ID:
CVE-2025-32441

LINK

Rack session gets restored after deletion

DESCRIPTION:

Summary

When using the `Rack::Session::Pool` middleware, simultaneous rack requests can restore a deleted rack session, which allows the unauthenticated user to occupy that session.

Details

Rack session middleware prepares the session at the beginning of request, then saves is back to the store with possible changes applied by host rack application. This way the session becomes to be a subject of race conditions in general sense over concurrent rack requests.

Impact

When using the `Rack::Session::Pool` middleware, and provided the attacker can acquire a session cookie (already a major issue), the session may be restored if the attacker can trigger a long running request (within that same session) adjacent to the user logging out, in order to retain illicit access even after a user has attempted to logout.

MITIGATION

Update to the latest version of `rack`,
or
Ensure your application invalidates sessions atomically by marking them as logged out e.g., using a `logged_out` flag, instead of deleting them, and check this flag on every request to prevent reuse, or
Implement a custom session store that tracks session invalidation timestamps and refuses to accept session data if the session was invalidated after the request began.

Related

As this code was moved to `rack-session` in Rack 3+, see https://github.com/rack/rack-session/security/advisories/GHSA-9j94-67jr-4cqj for the equivalent advisory in `rack-session` (affecting Rack 3+ only).

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2025-46727

LINK

Rack has an Unbounded-Parameter DoS in Rack::QueryParser

DESCRIPTION:

SUMMARY

`Rack::QueryParser` parses query strings and `application/x-www-form-urlencoded` bodies into Ruby data structures without imposing any limit on the number of parameters, allowing attackers to send requests with extremely large numbers of parameters.

DETAILS

The vulnerability arises because `Rack::QueryParser` iterates over each `&`-separated key-value pair and adds it to a Hash without enforcing an upper bound on the total number of parameters. This allows an attacker to send a single request containing hundreds of thousands (or more) of parameters, which consumes excessive memory and CPU during parsing.

IMPACT

An attacker can trigger denial of service by sending specifically crafted HTTP requests, which can cause memory exhaustion or pin CPU resources, stalling or crashing the Rack server. This results in full service disruption until the affected worker is restarted.

MITIGATION

Update to a version of Rack that limits the number of parameters parsed, or
Use middleware to enforce a maximum query string size or parameter count, or
Employ a reverse proxy (such as Nginx) to limit request sizes and reject oversized query strings or bodies.

Limiting request body sizes and query string lengths at the web server or CDN level is an effective mitigation.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2025-59830

LINK

Rack has an unsafe default in Rack::QueryParser allows params_limit bypass via semicolon-separated parameters

DESCRIPTION:

SUMMARY

`Rack::QueryParser` in version `< 2.2.18` enforces its `params_limit` only for parameters separated by `&`, while still splitting on both `&` and `;`. As a result, attackers could use `;` separators to bypass the parameter count limit and submit more parameters than intended.

DETAILS

The issue arises because `Rack::QueryParser#check_query_string` counts only `&` characters when determining the number of parameters, but the default separator regex `DEFAULT_SEP = /[&;] */n` splits on both `&` and `;`. This mismatch means that queries using `;` separators were not included in the parameter count, allowing `params_limit` to be bypassed. Other safeguards (`bytesize_limit` and `key_space_limit`) still applied, but did not prevent this particular bypass.

IMPACT

Applications or middleware that directly invoke `Rack::QueryParser` with its default configuration (no explicit delimiter) could be exposed to increased CPU and memory consumption. This can be abused as a limited denial-of-service vector.

`Rack::Request`, the primary entry point for typical Rack applications, uses `QueryParser` in a safe way and does not appear vulnerable by default. As such, the severity is considered **low**, with the impact limited to edge cases where `QueryParser` is used directly.

MITIGATION

Upgrade to a patched version of Rack where both `&` and `;` are counted consistently toward `params_limit`.

If upgrading is not immediately possible, configure `QueryParser` with an explicit delimiter (e.g., `&`) to avoid the mismatch.
As a general precaution, enforce query string and request size limits at the web server or proxy layer (e.g., Nginx, Apache, or a CDN) to mitigate excessive parsing overhead.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2025-61770

LINK

Rack's unbounded multipart preamble buffering enables DoS (memory exhaustion)

DESCRIPTION:

SUMMARY

`Rack::Multipart::Parser` buffers the entire multipart **preamble** (bytes before the first boundary) in memory without any size limit. A client can send a large preamble followed by a valid boundary, causing significant memory use and potential process termination due to out-of-memory (OOM) conditions.

DETAILS

While searching for the first boundary, the parser appends incoming data into a shared buffer (`@sbuf.concat(content)`) and scans for the boundary pattern:

```
@sbuf.scan_until(@body_regex)
```

If the boundary is not yet found, the parser continues buffering data indefinitely. There is no trimming or size cap on the preamble, allowing attackers to send arbitrary amounts of data before the first boundary.

IMPACT

Remote attackers can trigger large transient memory spikes by including a long preamble in multipart/form-data requests. The impact scales with allowed request sizes and concurrency, potentially causing worker crashes or severe slowdown due to garbage collection.

MITIGATION

Upgrade: Use a patched version of Rack that enforces a preamble size limit (e.g., 16 KiB) or discards preamble data entirely per RFC 2046 Â§ 5.1.1.

Workarounds:

Limit total request body size at the proxy or web server level.

Monitor memory and set per-process limits to prevent OOM conditions.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2025-61771

LINK

Multipart parser buffers large nonâ€‘file fields entirely in memory, enabling DoS (memory exhaustion)

DESCRIPTION:

SUMMARY

`Rack::Multipart::Parser` stores non-file form fields (parts without a `filename`) entirely in memory as Ruby `String` objects. A single large text field in a multipart/form-data request (hundreds of megabytes or more) can consume equivalent process memory, potentially leading to out-of-memory (OOM) conditions and denial of service (DoS).

DETAILS

During multipart parsing, file parts are streamed to temporary files, but non-file parts are buffered into memory:

```
 body = String.new  # non-file â†’ in-RAM buffer
@mime_parts[mime_index].body << content
```

There is no size limit on these in-memory buffers. As a result, any large text fieldâ€”while technically validâ€”will be loaded fully into process memory before being added to `params`.

IMPACT

Attackers can send large non-file fields to trigger excessive memory usage. Impact scales with request size and concurrency, potentially leading to worker crashes or severe garbage-collection overhead. All Rack applications processing multipart form submissions are affected.

MITIGATION

Upgrade: Use a patched version of Rack that enforces a reasonable size cap for non-file fields (e.g., 2 MiB).

Workarounds:

Restrict maximum request body size at the web-server or proxy layer (e.g., Nginx `client_max_body_size`).

Validate and reject unusually large form fields at the application level.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2025-61772

LINK

Rack's multipart parser buffers unbounded per-part headers, enabling DoS (memory exhaustion)

DESCRIPTION:

SUMMARY

`Rack::Multipart::Parser` can accumulate unbounded data when a multipart partâ€™s header block never terminates with the required blank line (`CRLFCRLF`). The parser keeps appending incoming bytes to memory without a size cap, allowing a remote attacker to exhaust memory and cause a denial of service (DoS).

DETAILS

While reading multipart headers, the parser waits for `CRLFCRLF` using:

```
@sbuf.scan_until(/(.*?\r
)\r
/m)
```

If the terminator never appears, it continues appending data (`@sbuf.concat(content)`) indefinitely. There is no limit on accumulated header bytes, so a single malformed part can consume memory proportional to the request body size.

IMPACT

Attackers can send incomplete multipart headers to trigger high memory use, leading to process termination (OOM) or severe slowdown. The effect scales with request size limits and concurrency. All applications handling multipart uploads may be affected.

MITIGATION

Upgrade to a patched Rack version that caps per-part header size (e.g., 64 KiB).

Until then, restrict maximum request sizes at the proxy or web server layer (e.g., Nginx `client_max_body_size`).

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2025-61780

LINK

Rack has a Possible Information Disclosure Vulnerability

DESCRIPTION:

SUMMARY

A possible information disclosure vulnerability existed in `Rack::Sendfile` when running behind a proxy that supports `x-sendfile` headers (such as Nginx). Specially crafted headers could cause `Rack::Sendfile` to miscommunicate with the proxy and trigger unintended internal requests, potentially bypassing proxy-level access restrictions.

DETAILS

When `Rack::Sendfile` received untrusted `x-sendfile-type` or `x-accel-mapping` headers from a client, it would interpret them as proxy configuration directives. This could cause the middleware to send a "redirect" response to the proxy, prompting it to reissue a new internal request that was **not subject to the proxy's access controls**.

An attacker could exploit this by: 1. Setting a crafted `x-sendfile-type: x-accel-redirect` header. 2. Setting a crafted `x-accel-mapping` header. 3. Requesting a path that qualifies for proxy-based acceleration.

IMPACT

Attackers could bypass proxy-enforced restrictions and access internal endpoints intended to be protected (such as administrative pages). The vulnerability did not allow arbitrary file reads but could expose sensitive application routes.

This issue only affected systems meeting all of the following conditions:

- **The application used `Rack::Sendfile` with a proxy that supports `x-accel-redirect` (e.g., Nginx).**
- **The proxy did not always set or remove the `x-sendfile-type` and `x-accel-mapping` headers.**
- **The application exposed an endpoint that returned a body responding to `.to_path`.**

MITIGATION

Upgrade to a fixed version of Rack which requires explicit configuration to enable `x-accel-redirect`:

```
use Rack::Sendfile, "x-accel-redirect"
```

Alternatively, configure the proxy to always set or strip the headers (you should be doing this!):

```
proxy_set_header x-sendfile-type x-accel-redirect;
proxy_set_header x-accel-mapping /var/www/=/files/;
```

Or in Rails applications, disable sendfile completely:

```
config.action_dispatch.x_sendfile_header = nil
```

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2025-61919

LINK

Rack is vulnerable to a memory-exhaustion DoS through unbounded URL-encoded body parsing

DESCRIPTION:

SUMMARY

`Rack::Request#POST` reads the entire request body into memory for `Content-Type: application/x-www-form-urlencoded`, calling `rack.input.read(nil)` without enforcing a length or cap. Large request bodies can therefore be buffered completely into process memory before parsing, leading to denial of service (DoS) through memory exhaustion.

DETAILS

When handling non-multipart form submissions, Rackâ€™s request parser performs:

```
form_vars = get_header(RACK_INPUT).read
```

Since `read` is called with no argument, the entire request body is loaded into a Ruby `String`. This occurs before query parameter parsing or enforcement of any `params_limit`. As a result, Rack applications without an upstream body-size limit can experience unbounded memory allocation proportional to request size.

IMPACT

Attackers can send large `application/x-www-form-urlencoded` bodies to consume process memory, causing slowdowns or termination by the operating system (OOM). The effect scales linearly with request size and concurrency. Even with parsing limits configured, the issue occurs *before*

those limits are enforced.

MITIGATION

Update to a patched version of Rack that enforces form parameter limits using `query_parser.bytesize_limit`, preventing unbounded reads of `application/x-www-form-urlencoded` bodies.

Enforce strict maximum body size at the proxy or web server layer (e.g., Nginx `client_max_body_size`, Apache `LimitRequestBody`).

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2026-22860

LINK

Rack has a Directory Traversal via Rack:Directory

DESCRIPTION:

SUMMARY

`Rack::Directory` â€™s path check used a string prefix match on the expanded path. A request like `/../root_example/` can escape the configured root if the target path starts with the root string, allowing directory listing outside the intended root.

DETAILS

In `directory.rb`, `File.expand_path(File.join(root, path_info)).start_with?(root)` does not enforce a path boundary. If the server root is `/var/www/root`, a path like `/var/www/root_backup` passes the check because it shares the same prefix, so `Rack::Directory` will list that directory also.

IMPACT

Information disclosure via directory listing outside the configured root when `Rack::Directory` is exposed to untrusted clients and a directory shares the

root prefix (e.g., `public2`, `www_backup`).

MITIGATION

Update to a patched version of Rack that correctly checks the root prefix.\n* Don't name directories with the same prefix as one which is exposed via `Rack::Directory`."

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2026-25500

LINK

Stored XSS in Rack::Directory via javascript: filenames rendered into anchor href

DESCRIPTION:

SUMMARY

`Rack::Directory` generates an HTML directory index where each file entry is rendered as a clickable link. If a file exists on disk whose basename begins with the `javascript:` scheme (e.g. `javascript:alert(1)`), the generated index includes an anchor whose `href` attribute is exactly `javascript:alert(1)`. Clicking this entry executes arbitrary JavaScript in the context of the hosting application.

This results in a client-side XSS condition in directory listings generated by `Rack::Directory`.

DETAILS

`Rack::Directory` renders directory entries using an HTML row template similar to:

```
<a href='%s'>%s</a>
```

The `%s` placeholder is populated directly with the fileâ€™s basename. If the basename begins with `javascript:`, the resulting HTML contains an executable JavaScript URL:

```
<a href='javascript:alert(1)'>javascript:alert(1)</a>
```

Because the value is inserted directly into the `href` attribute without scheme validation or normalization, browsers interpret it as a JavaScript URI. When a user clicks the link, the JavaScript executes in the origin of the Rack application.

IMPACT

If `Rack::Directory` is used to expose filesystem contents over HTTP, an attacker who can create or upload files within that directory may introduce a malicious filename beginning with `javascript:`.

When a user visits the directory listing and clicks the entry, arbitrary JavaScript executes in the application's origin. Exploitation requires user interaction (clicking the malicious entry).

MITIGATION

Update to a patched version of Rack in which `Rack::Directory` prefixes generated anchors with a relative path indicator (e.g. `./filename`).

Avoid exposing user-controlled directories via `Rack::Directory`.

Apply a strict Content Security Policy (CSP) to reduce impact of potential client-side execution issues.

Where feasible, restrict or sanitize uploaded filenames to disallow dangerous URI scheme prefixes.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2026-26961

LINK

Rack's greedy multipart boundary parsing can cause parser differentials and WAF bypass.

DESCRIPTION:

SUMMARY

`Rack::Multipart::Parser` extracts the `boundary` parameter from `multipart/form-data` using a greedy regular expression. When a `Content-Type` header contains multiple `boundary` parameters, Rack selects the last one rather than the first.

In deployments where an upstream proxy, WAF, or intermediary interprets the first `boundary` parameter, this mismatch can allow an attacker to smuggle multipart content past upstream inspection and have Rack parse a different body structure than the intermediary validated.

DETAILS

Rack identifies the multipart boundary using logic equivalent to:

```
MULTIPART = %r|\Amultipart/.*boundary=\"?([^\";,]+)\"?|ni
```

Because the expression is greedy, it matches the last `boundary=` parameter in a header such as:

```
Content-Type: multipart/form-data; boundary=safe; boundary=mal
icious
```

As a result, Rack parses the request body using `malicious`, while another component may interpret the same header using `safe`.

This creates an interpretation conflict. If an upstream WAF or proxy inspects multipart parts using the first boundary and Rack later parses the body using the last boundary, a client may be able to place malicious form fields or uploaded content in parts that Rack accepts but the upstream component did not inspect as intended.

This issue is most relevant in layered deployments where security decisions are made before the request reaches Rack.

IMPACT

Applications that accept `multipart/form-data` uploads behind an inspecting proxy or WAF may be affected.

In such deployments, an attacker may be able to bypass upstream filtering of uploaded files or form fields by sending a request with multiple `boundary` parameters and relying on the intermediary and Rack to parse the request differently.

The practical impact depends on deployment architecture. If no upstream component relies on a different multipart interpretation, this behavior may not provide meaningful additional attacker capability.

MITIGATION

Update to a patched version of Rack that rejects ambiguous multipart `Content-Type` headers or parses duplicate `boundary` parameters consistently.

Reject requests containing multiple `boundary` parameters.

Normalize or regenerate multipart metadata at the trusted edge before forwarding requests to Rack.

Avoid relying on upstream inspection of malformed multipart requests unless duplicate parameter handling is explicitly consistent across components.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2026-34230

LINK

Rack has quadratic complexity in Rack::Utils.select_best_encoding via wildcard Accept-Encoding header

DESCRIPTION:

SUMMARY

`Rack::Utils.select_best_encoding` processes `Accept-Encoding` values with quadratic time complexity when the header contains many wildcard (`*`) entries. Because this method is used by `Rack::Deflater` to choose a response encoding, an unauthenticated attacker can send a single request with a crafted `Accept-Encoding` header and cause disproportionate CPU consumption on the compression middleware path.
This results in a denial of service condition for applications using `Rack::Deflater`.

DETAILS

`Rack::Utils.select_best_encoding` expands parsed `Accept-Encoding` values into a list of candidate encodings. When an entry is `*`, the method computes the set of concrete encodings by subtracting the encodings already present in the request:

```
 if m == "*"
 (available_encodings - accept_encoding.map(&:first)).each do |m
2|
   expanded_accept_encoding << [m2, q, preference]
 end
else
 expanded_accept_encoding << [m, q, preference]
end
```

Because `accept_encoding.map(&:first)` is evaluated inside the loop, it is recomputed for each wildcard entry. If the request contains `N` wildcard entries, this produces repeated scans over the full parsed header and causes quadratic behavior.
After expansion, the method also performs additional work over `expanded_accept_encoding`, including per-entry deletion, which further increases the cost for large inputs.
`Rack::Deflater` invokes this method for each request when the middleware is enabled:
`Utils.select_best_encoding(ENCODINGS, Utils.parse_encodings(accept_encoding))`

As a result, a client can trigger this expensive code path simply by sending a large `Accept-Encoding` header containing many repeated wildcard values. For example, a request with an approximately 8 KB `Accept-Encoding` header containing about 1,000 `*;q=0.5` entries can cause roughly 170 ms of CPU time in a single request on the `Rack::Deflater` path, compared to a negligible baseline for a normal header.
This issue is distinct from CVE-2024-26146. That issue concerned regular expression denial of service during `Accept` header parsing, whereas this issue arises later during encoding selection after the header has already been parsed.

IMPACT

Any Rack application using `Rack::Deflater` may be affected.
An unauthenticated attacker can send requests with crafted `Accept-Encoding` headers to trigger excessive CPU usage in the encoding selection logic. Repeated requests can consume worker time disproportionately and reduce application availability.
The attack does not require invalid HTTP syntax or large payload bodies. A single header-sized request is sufficient to reach the vulnerable code path.

MITIGATION

Update to a patched version of Rack in which encoding selection does not repeatedly rescan the parsed header for wildcard entries.
Avoid enabling `Rack::Deflater` on untrusted traffic.
Apply request filtering or header size / format restrictions at the reverse proxy or application boundary to limit abusive `Accept-Encoding` values.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2026-34763

LINK

Rack has a root directory disclosure via unescaped regex interpolation in Rack::Directory

DESCRIPTION:

SUMMARY

`Rack::Directory` interpolates the configured `root` path directly into a regular expression when deriving the displayed directory path. If `root` contains regex metacharacters such as `+`, `*`, or `.`, the prefix stripping can fail and the generated directory listing may expose the full filesystem path in the HTML output.

DETAILS

`Rack::Directory::DirectoryBody#each` computes the visible path using code equivalent to:

```
show_path = Utils.escape_html(path.sub(/\A#{root}/, ''))
```

Here, `root` is a developer-configured filesystem path. It is normalized earlier with `File.expand_path(root)` and then inserted directly into a regular expression without escaping.
Because the value is treated as regex syntax rather than as a literal string, metacharacters in the configured path can change how the prefix match behaves. When that happens, the expected root prefix is not removed from `path`, and the absolute filesystem path is rendered into the HTML directory listing.

IMPACT

If `Rack::Directory` is configured to serve a directory whose absolute path contains regex metacharacters, the generated directory listing may disclose the full server filesystem path instead of only the request-relative path.
This can expose internal deployment details such as directory layout, usernames, mount points, or naming conventions that would otherwise not be visible to clients.

MITIGATION

Update to a patched version of Rack in which the root prefix is removed using an escaped regular expression. Avoid using `Rack::Directory` with a root path that contains regular expression metacharacters.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2026-34785

LINK

Rack::Static prefix matching can expose unintended files under the static root

DESCRIPTION:

SUMMARY

`Rack::Static` determines whether a request should be served as a static file using a simple string prefix check. When configured with URL prefixes such as `"/css"`, it matches any request path that begins with that string, including unrelated paths such as `"/css-config.env"` or `"/css-backup.sql"`.

As a result, files under the static root whose names merely share the configured prefix may be served unintentionally, leading to information disclosure.

DETAILS

`Rack::Static#route_file` performs static-route matching using logic equivalent to:

```
@urls.any? { |url| path.index(url) == 0 }
```

This checks only whether the request path starts with the configured prefix string. It does not require a path segment boundary after the prefix.

For example, with:

```
use Rack::Static, urls: ["/css", "/js"], root: "public"
```

the following path is matched as intended:

```
/css/style.css
```

but these paths are also matched:

```
/css-config.env
/css-backup.sql
/csssecrets.yml
```

If such files exist under the configured static root, Rack forwards the request to the file server and serves them as static content.
This means a configuration intended to expose only directory trees such as `/css/...` and `/js/...` may also expose sibling files whose names begin with those same strings.

IMPACT

An attacker can request files under the configured static root whose names share a configured URL prefix and obtain their contents.
In affected deployments, this may expose configuration files, secrets, backups, environment files, or other unintended static content located under the same root directory.

MITIGATION

Update to a patched version of Rack that enforces a path boundary when matching configured static URL prefixes.
Match only paths that are either exactly equal to the configured prefix or begin with `prefix + "/"`.
Avoid placing sensitive files under the `Rack::Static` root directory.
Prefer static URL mappings that cannot overlap with sensitive filenames.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2026-34786

LINK

Rack:: Static header_rules bypass via URL-encoded paths

DESCRIPTION:

SUMMARY

`Rack::Static#applicable_rules` evaluates several `header_rules` types against the raw URL-encoded `PATH_INFO`, while the underlying file-

serving path is decoded before the file is served. As a result, a request for a URL-encoded variant of a static path can serve the same file without the headers that `header_rules` were intended to apply.
In deployments that rely on `Rack::Static` to attach security-relevant response headers to static content, this can allow an attacker to bypass those headers by requesting an encoded form of the path.

DETAILS

`Rack::Static#applicable_rules` matches rule types such as `:fonts`, `Array`, and `Regexp` directly against the incoming `PATH_INFO`. For example:

```
when :fonts
  /\.(?:ttf|otf|eot|woff2|woff|svg)\z/.match?(path)
when Array
  /\.(#{rule.join('|')})\z/.match?(path)
when Regexp
  rule.match?(path)
```

These checks operate on the raw request path. If the request contains encoded characters such as `%2E` in place of `.`, the rule may fail to match even though the file path is later decoded and served successfully by the static file server.
For example, both of the following requests may resolve to the same file on disk:

```
/fonts/test.woff
/fonts/test%2Ewoff
```

but only the unencoded form may receive the headers configured through `header_rules`.
This creates a canonicalization mismatch between the path used for header policy decisions and the path ultimately used for file serving.

IMPACT

Applications that rely on `Rack::Static` `header_rules` to apply security-relevant headers to static files may be affected.
In affected deployments, an attacker can request an encoded variant of a static file path and receive the same file without the intended headers. Depending on how `header_rules` are used, this may bypass protections such as clickjacking defenses, content restrictions, or other response policies applied to static content.
The practical impact depends on the configured rules and the types of files being served. If `header_rules` are only used for non-security purposes such as caching, the issue may have limited security significance.

MITIGATION

Update to a patched version of Rack that applies `header_rules` to a decoded path consistently with static file resolution.

Do not rely solely on `Rack::Static` `header_rules` for security-critical headers where encoded path variants may reach the application.

Prefer setting security headers at the reverse proxy or web server layer so they apply consistently to both encoded and unencoded path forms.
Normalize or reject encoded path variants for static content at the edge, where feasible.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2026-34826

LINK

Rack's multipart byte range processing allows denial of service via excessive overlapping ranges

DESCRIPTION:

SUMMARY

`Rack::Utils.get_byte_ranges` parses the HTTP `Range` header without limiting the number of individual byte ranges. Although the existing fix for CVE-2024-26141 rejects ranges whose total byte coverage exceeds the file size, it does not restrict the count of ranges. An attacker can supply many small overlapping ranges such as `0-0,0-0,0-0,...` to trigger disproportionate CPU, memory, I/O, and bandwidth consumption per request.
This results in a denial of service condition in Rack file-serving paths that process multipart byte range responses.

DETAILS

`Rack::Utils.get_byte_ranges` accepts a comma-separated list of byte ranges and validates them based on their aggregate size, but does not impose a limit on how many individual ranges may be supplied.
As a result, a request such as:

```
Range: bytes=0-0,0-0,0-0,0-0,...
```

can contain thousands of overlapping one-byte ranges while still satisfying the total-size check added for CVE-2024-26141.
When such a header is processed by Rackâ€™s file-serving code, each range causes additional work, including multipart response generation, per-range iteration, file seek and read operations, and temporary string allocation for response size calculation and output. This allows a relatively small request header to trigger disproportionately expensive processing and a much larger multipart response.
The issue is distinct from CVE-2024-26141. That fix prevents range sets whose total byte coverage exceeds the file size, but does not prevent a large number of overlapping ranges whose summed size remains within that limit.

IMPACT

Applications that expose file-serving paths with byte range support may be vulnerable to denial of service.
An unauthenticated attacker can send crafted `Range` headers containing many small overlapping ranges to consume excessive CPU time, memory, file I/O, and bandwidth. Repeated requests may reduce application availability and increase pressure on workers and garbage collection.

MITIGATION

Update to a patched version of Rack that limits the number of accepted byte ranges.
Reject or normalize multipart byte range requests containing excessive range counts.
Consider disabling multipart range support where it is not required.
Apply request filtering or header restrictions at the reverse proxy or application boundary to limit abusive `Range` headers.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2026-34829

LINK

Rack's multipart parsing without Content-Length header allows unbounded chunked file uploads

DESCRIPTION:

SUMMARY

`Rack::Multipart::Parser` only wraps the request body in a `BoundedIO` when `CONTENT_LENGTH` is present. When a `multipart/form-data` request is sent without a `Content-Length` header, such as with HTTP chunked transfer encoding, multipart parsing continues until end-of-stream with no total size limit.

For file parts, the uploaded body is written directly to a temporary file on disk rather than being constrained by the buffered in-memory upload limit. An unauthenticated attacker can therefore stream an arbitrarily large multipart file upload and consume unbounded disk space.

This results in a denial of service condition for Rack applications that accept multipart form data.

DETAILS

`Rack::Multipart::Parser.parse` applies `BoundedIO` only when `content_length` is not `nil`:

```
io = BoundedIO.new(io, content_length) if content_length
```

When `CONTENT_LENGTH` is absent, the parser reads the multipart body until EOF without a global byte limit.

Although Rack enforces `BUFFERED_UPLOAD_BYTESIZE_LIMIT` for retained non-file parts, file uploads are handled differently. When a multipart part includes a filename, the body is streamed to a `Tempfile`, and the retained-size accounting is not applied to that file content. As a result, file parts are not subject to the same upload size bound.

An attacker can exploit this by sending a chunked `multipart/form-data` request containing a file part and continuously streaming data without declaring a `Content-Length`. Rack will continue writing the uploaded data to disk until the client stops or the server exhausts available storage.

IMPACT

Any Rack application that accepts `multipart/form-data` uploads may be affected if no upstream component enforces a request body size limit.

An unauthenticated attacker can send a large chunked file upload to consume disk space on the application host. This may cause request failures, application instability, or broader service disruption if the host runs out of available storage.

The practical impact depends on deployment architecture. Reverse proxies or application servers that enforce upload limits may reduce or eliminate exploitability, but Rack itself does not impose a total multipart upload limit in this code path when `CONTENT_LENGTH` is absent.

MITIGATION

Update to a patched version of Rack that enforces a total multipart upload size limit even when `CONTENT_LENGTH` is absent.

Enforce request body size limits at the reverse proxy or application server.

Isolate temporary upload storage and monitor disk consumption for multipart endpoints.

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2026-34830

LINK

Rack::Sendfile header-based X-Accel-Mapping regex injection enables unauthorized X-Accel-Redirect

DESCRIPTION:

SUMMARY

`Rack::Sendfile#map_accel_path` interpolates the value of the `X-Accel-Mapping` request header directly into a regular expression when rewriting file paths for `X-Accel-Redirect`. Because the header value is not escaped, an attacker who can supply `X-Accel-Mapping` to the backend can inject regex metacharacters and control the generated `X-Accel-Redirect` response header.

In deployments using `Rack::Sendfile` with `x-accel-redirect`, this can allow an attacker to cause nginx to serve unintended files from configured internal locations.

DETAILS

`Rack::Sendfile#map_accel_path` processes header-supplied mappings using logic equivalent to:

```
mapping.split(',').map(&:strip).each do |m|
internal, external = m.split('=', 2).map(&:strip)
new_path = path.sub(/\A#{internal}/i, external)
return new_path unless path == new_path
end
```

Here, `internal` comes from the `HTTP_X_ACCEL_MAPPING` request

header and is inserted directly into a regular expression without escaping. This gives the header value regex semantics rather than treating it as a literal prefix.
As a result, an attacker can supply metacharacters such as `.*` or capture groups to alter how the path substitution is performed. For example, a mapping such as:

```
X-Accel-Mapping: .*=/protected/secret.txt
```

causes the entire source path to match and rewrites the redirect target to a clean attacker-chosen internal path.
This differs from the documented behavior of the header-based mapping path, which is described as a simple substitution. While application-supplied mappings may intentionally support regular expressions, header-supplied mappings should be treated as literal path prefixes.
The issue is only exploitable when untrusted `X-Accel-Mapping` headers can reach Rack. One realistic case is a reverse proxy configuration that intends to set `X-Accel-Mapping` itself, but fails to do so on some routes, allowing a client-supplied header to pass through unchanged.

IMPACT

Applications using `Rack::Sendfile` with `x-accel-redirect` may be affected if the backend accepts attacker-controlled `X-Accel-Mapping` headers.
In affected deployments, an attacker may be able to control the `X-Accel-Redirect` response header and cause nginx to serve files from internal locations that were not intended to be reachable through the application. This can lead to unauthorized file disclosure.
The practical impact depends on deployment architecture. If the proxy always strips or overwrites `X-Accel-Mapping`, or if the application uses explicit configured mappings instead of the request header, exploitability may be eliminated.

MITIGATION

- **Update to a patched version of Rack that treats header-supplied `X-Accel-Mapping` values as literal strings rather than regular expressions.**
- **Strip or overwrite inbound `X-Accel-Mapping` headers at the reverse proxy so client-supplied values never reach Rack.**
- **Prefer explicit application-configured sendfile mappings instead of relying on request-header mappings.**
- **Review proxy sub-locations and inherited header settings to ensure `X-Accel-Mapping` is consistently set on all backend routes.**

VULNERABLE GEM: RACK@1.6.10

Name:
rack

Version:
1.6.10

ID:
CVE-2026-34831

LINK

Rack has Content-Length mismatch in Rack::Files error responses

DESCRIPTION:

SUMMARY

`Rack::Files#fail` sets the `Content-Length` response header using `String#size` instead of `String#bytesize`. When the response body contains multibyte UTF-8 characters, the declared `Content-Length` is smaller than the number of bytes actually sent on the wire.

Because `Rack::Files` reflects the requested path in 404 responses, an attacker can trigger this mismatch by requesting a non-existent path containing percent-encoded UTF-8 characters.

This results in incorrect HTTP response framing and may cause response desynchronization in deployments that rely on the incorrect `Content-Length` value.

DETAILS

`Rack::Files#fail` constructs error responses using logic equivalent to:

```
 def fail(status, body, headers = {})
 body += "
"
 [
  status,
  {
   "content-type" => "text/plain",
   "content-length" => body.size.to_s,
   "x-cascade" => "pass"
  }.merge!(headers),
  [body]
 ]
end
```

Here, `body.size` returns the number of characters, not the number of bytes. For multibyte UTF-8 strings, this produces an incorrect `Content-Length` value.

`Rack::Files` includes the decoded request path in 404 responses. A

request containing percent-encoded UTF-8 path components therefore causes the response body to contain multibyte characters, while the `Content-Length` header still reflects character count rather than byte count.
As a result, the server can send more bytes than declared in the response headers.
This violates HTTP message framing requirements, which define `Content-Length` as the number of octets in the message body.

IMPACT

Applications using `Rack::Files` may emit incorrectly framed error responses when handling requests for non-existent paths containing multibyte characters.
In some deployment topologies, particularly with keep-alive connections and intermediaries that rely on `Content-Length`, this mismatch may lead to response parsing inconsistencies or response desynchronization. The practical exploitability depends on the behavior of downstream proxies, clients, and connection reuse.
Even where no secondary exploitation is possible, the response is malformed and may trigger protocol errors in strict components.

MITIGATION

Update to a patched version of Rack that computes `Content-Length` using `String#bytesize`.
Avoid exposing `Rack::Files` directly to untrusted traffic until a fix is available, if operationally feasible.
Where possible, place Rack behind a proxy or server that normalizes or rejects malformed backend responses.
Prefer closing backend connections on error paths if response framing anomalies are a concern.

VULNERABLE GEM: RAILS-HTML-SANITIZER@1.0.4

Name:
rails-html-sanitizer

Version:
1.0.4

ID:
CVE-2022-23517

LINK

Inefficient Regular Expression Complexity in rails-html-sanitizer

DESCRIPTION:
SUMMARY
Certain configurations of rails-html-sanitizer `< 1.4.4` use an inefficient regular expression that is susceptible to excessive backtracking when attempting to sanitize certain SVG attributes. This may lead to a denial of service through CPU resource consumption.
MITIGATION
Upgrade to rails-html-sanitizer `>= 1.4.4`.

VULNERABLE GEM: RAILS-HTML-SANITIZER@1.0.4

Name:
rails-html-sanitizer

Version:
1.0.4

ID:
CVE-2022-23518

LINK

Improper neutralization of data URIs may allow XSS in rails-html-sanitizer

DESCRIPTION:
SUMMARY
rails-html-sanitizer `>= 1.0.3, < 1.4.4` is vulnerable to cross-site scripting via data URIs when used in combination with Loofah `>= 2.1.0`.
MITIGATION
Upgrade to rails-html-sanitizer `>= 1.4.4`.

VULNERABLE GEM: RAILS-HTML-SANITIZER@1.0.4

Name:
rails-html-sanitizer

Version:
1.0.4

ID:
CVE-2022-23519

LINK

Possible XSS vulnerability with certain configurations of rails-html-sanitizer

DESCRIPTION:

SUMMARY

There is a possible XSS vulnerability with certain configurations of Rails::Html::Sanitizer.

Versions affected:
ALL

Not affected:
NONE

Fixed versions:
1.4.4

IMPACT

A possible XSS vulnerability with certain configurations of Rails::Html::Sanitizer may allow an attacker to inject content if the application developer has overridden the sanitizer's allowed tags in either of the following ways:

allow both "math" and "style" elements,
or allow both "svg" and "style" elements

Code is only impacted if allowed tags are being overridden. Applications may be doing this in four different ways:

using application configuration:

```
# In config/application.rb
config.action_view.sanitized_allowed_tags = ["math", "style"]
# or
config.action_view.sanitized_allowed_tags = ["svg", "style"]
```

see https://guides.rubyonrails.org/configuring.html#configuring-action-view

using a `:tags` option to the Action View helper `sanitize`:

```
<%= sanitize @comment.body, tags: ["math", "style"] %>
<%# or %>
<%= sanitize @comment.body, tags: ["svg", "style"] %>
```

see https://api.rubyonrails.org/classes/ActionView/Helpers/SanitizeHelper.html#method-i-sanitize

using Rails::Html::SafeListSanitizer class method `allowed_tags=`:

```
# class-level option
Rails::Html::SafeListSanitizer.allowed_tags = ["math", "style"]
# or
Rails::Html::SafeListSanitizer.allowed_tags = ["svg", "style"]
```

using a `:tags` options to the Rails::Html::SafeListSanitizer instance method `sanitize`:

```
# instance-level option
Rails::Html::SafeListSanitizer.new.sanitize(@article.body, tags: ["math", "style"])
# or
Rails::Html::SafeListSanitizer.new.sanitize(@article.body, tags: ["svg", "style"])
```

All users overriding the allowed tags by any of the above mechanisms to include (("math" or "svg") and "style") should either upgrade or use one of the workarounds immediately.

WORKAROUNDS

Remove "style" from the overridden allowed tags, or remove "math" and "svg" from the overridden allowed tags.

VULNERABLE GEM: RAILS-HTML-SANITIZER@1.0.4

Name:
rails-html-sanitizer

Version:
1.0.4

ID:
CVE-2022-23520

LINK

Possible XSS vulnerability with certain configurations of rails-html-sanitizer

DESCRIPTION:
SUMMARY
There is a possible XSS vulnerability with certain configurations of Rails::Html::Sanitizer. This is due to an incomplete fix of CVE-2022-32209.

Versions affected:
ALL

Not affected:
NONE

Fixed versions:
1.4.4

IMPACT
A possible XSS vulnerability with certain configurations of Rails::Html::Sanitizer may allow an attacker to inject content if the application developer has overridden the sanitizer's allowed tags to allow both "select" and "style" elements.
Code is only impacted if allowed tags are being overridden using either of the following two mechanisms:

Using the Rails configuration `config.action_view.sanitized_allow_tags=`:

```
# In config/application.rb
config.action_view.sanitized_allowed_tags = ["select", "style"]
```

(see https://guides.rubyonrails.org/configuring.html#configuring-action-view)

Using the class method `Rails::Html::SafeListSanitizer.allowed_tags=`:

```
# class-level option
Rails::Html::SafeListSanitizer.allowed_tags = ["select", "style"]
```

All users overriding the allowed tags by either of the above mechanisms to include both "select" and "style" should either upgrade or use one of the workarounds immediately.
NOTE: Code is *not* impacted if allowed tags are overridden using either of

the following mechanisms:

- the `:tags` option to the Action View helper method `sanitize`.
- the `:tags` option to the instance method `SafeListSanitizer#sanitize`.

WORKAROUNDS

Remove either "select" or "style" from the overridden allowed tags.

VULNERABLE GEM: RAILS-HTML-SANITIZER@1.0.4

Name:
rails-html-sanitizer

Version:
1.0.4

ID:
CVE-2022-32209

LINK

Possible XSS vulnerability with certain configurations of Rails::Html::Sanitizer

DESCRIPTION:

There is a possible XSS vulnerability with certain configurations of Rails::Html::Sanitizer. This vulnerability has been assigned the CVE identifier CVE-2022-32209.

Versions Affected: ALL Not affected: NONE Fixed Versions: v1.4.3

IMPACT

A possible XSS vulnerability with certain configurations of Rails::Html::Sanitizer may allow an attacker to inject content if the application developer has overridden the sanitizer's allowed tags to allow both `select` and `style` elements.

Code is only impacted if allowed tags are being overridden. This may be done via application configuration:

```
# In config/application.rb
config.action_view.sanitized_allowed_tags = ["select", "style"]
```

see https://guides.rubyonrails.org/configuring.html#configuring-action-view
Or it may be done with a `:tags` option to the Action View helper `sanitize`:

```
<%= sanitize @comment.body, tags: ["select", "style"] %>
```

see https://api.rubyonrails.org/classes/ActionView/Helpers/SanitizeHelper.html#method-i-sanitize
Or it may be done with Rails::Html::SafeListSanitizer directly:

```
# class-level option
Rails::Html::SafeListSanitizer.allowed_tags = ["select", "style"]
```

or

```
# instance-level option
Rails::Html::SafeListSanitizer.new.sanitize(@article.body, tags: ["select", "style"])
```

All users overriding the allowed tags by any of the above mechanisms to include both "select" and "style" should either upgrade or use one of the workarounds immediately.

WORKAROUNDS

Remove either `select` or `style` from the overridden allowed tags.

VULNERABLE GEM: RAILS-HTML-SANITIZER@1.0.4

Name:
rails-html-sanitizer

Version:
1.0.4

ID:
GHSA-cj75-f6xr-r4g7

LINK

Possible XSS vulnerability with certain configurations of rails-html-sanitizer

DESCRIPTION:

SUMMARY

There is a possible cross-site scripting vulnerability in rails-html-sanitizer when the sanitizer is configured to allow an SVG reference element such as . See related GHSA-9wjq-cp2p-hrgf in Loofah, whose SVG local-reference logic rails-html-sanitizer mirrors.

VULNERABLE GEM: RAKE@12.3.0

Name:
rake

Version:
12.3.0

ID:
CVE-2020-8130

LINK

OS Command Injection in Rake

DESCRIPTION:

There is an OS command injection vulnerability in Ruby Rake < 12.3.3 in Rake::FileList when supplying a filename that begins with the pipe character `|`.

VULNERABLE GEM: REDCARPET@3.4.0

Name:
redcarpet

Version:
3.4.0

ID:
CVE-2020-26298

LINK

Injection/XSS in Redcarpet

DESCRIPTION:

Redcarpet is a Ruby library for Markdown processing. In Redcarpet before version 3.5.1, there is an injection vulnerability which can enable a cross-site scripting attack. In affected versions no HTML escaping was being performed when processing quotes. This applies even when the `:escape_html` option was being used.

VULNERABLE GEM: TZINFO@1.2.5

Name:
tzinfo

Version:
1.2.5

ID:
CVE-2022-31163

LINK

TZInfo relative path traversal vulnerability allows loading of arbitrary files

DESCRIPTION:

Impact

AFFECTED VERSIONS

0.3.60 and earlier.

1.0.0 to 1.2.9 when used with the Ruby data source (tzinfo-data).

VULNERABILITY

With the Ruby data source (the tzinfo-data gem for tzinfo version 1.0.0 and later and built-in to earlier versions), time zones are defined in Ruby files. There is one file per time zone. Time zone files are loaded with `require` on demand. In the affected versions, `TZInfo::Timezone.get` fails to validate time zone identifiers correctly, allowing a new line character within the identifier. With Ruby version 1.9.3 and later, `TZInfo::Timezone.get` can be made to load unintended files with `require`, executing them within the Ruby process.

For example, with version 1.2.9, you can run the following to load a file with path `/tmp/payload.rb`:

```
TZInfo::Timezone.get(\"foo\
/../../../../../../../../../../../../../../../../../tmp/payload\")
```

The exact number of parent directory traversals needed will vary depending on the location of the tzinfo-data gem.

TZInfo versions 1.2.6 to 1.2.9 can be made to load files from outside of the Ruby load path. Versions up to and including 1.2.5 can only be made to load files from directories within the load path.

This could be exploited in, for example, a Ruby on Rails application using tzinfo version 1.2.9, that allows file uploads and has a time zone selector that accepts arbitrary time zone identifiers. The CVSS score and severity have been set on this basis.

Versions 2.0.0 and later are not vulnerable.

Patches

Versions 0.3.61 and 1.2.10 include fixes to correctly validate time zone identifiers.

Note that version 0.3.61 can still load arbitrary files from the Ruby load path if their name follows the rules for a valid time zone identifier and the file has a prefix of `tzinfo/definition` within a directory in the load path. For example if `/tmp/upload` was in the load path, then `TZInfo::Timezone.get('foo')` could load a file with path `/tmp/upload/tzinfo/definition/foo.rb`. Applications should ensure that untrusted files are not placed in a directory on the load path.

Workarounds

As a workaround, the time zone identifier can be validated before passing to `TZInfo::Timezone.get` by ensuring it matches the regular expression `\\A[A-Za-z0-9+\\-_]+(?:\\/[A-Za-z0-9+\\-_]+)*\\z`.