May 18, 2022

GHSL-2022-031_GHSL-2022-032: Type confusion in Nokogiri leads to memory leak or DoS - CVE-2022-29181


Agustin Gianni

Coordinated Disclosure Timeline

- 2022-05-05: Maintainers contacted via their `h1` program
- 2022-05-07: Maintainers fixed the issues and released an advisory

Summary

Two type confusion issues while processing malicious data can be used to leak the contents of memory or cause a denial-of-service.

Product

Nokogiri

Tested Version

v1.13.4

Details

Issue 1: Type confusion in `xml_sax_parser_context.c` (`GHSL-2022-031`)

An attacker who controls the kind of object passed into the `parse_memory` function can provide an object that is not a ruby string, for example an integer. Since an integer is an immediate value and not a pointer to an object, the process will crash while trying to read from an invalid memory address.

In the snippet bellow, the function `RSTRING_LEN` is used without checking that the type of the object `data` is `T_STRING`. Typically the object type is checked using `Check_Type(data, T_STRING)`, but in this case, no checks are performed.

Snippet from xml_sax_parser_context.c#L68:

```
/*
 * call-seq:
 *  parse_memory(data)
 *
 * Parse the XML stored in memory in +data+
 */
static VALUE
parse_memory(VALUE klass, VALUE data)
{
  xmlParserCtxtPtr ctxt;

  if (NIL_P(data)) {
    rb_raise(rb_eArgError, "data cannot be nil");
  }

  // NOTE: (1)
  if (!(int)RSTRING_LEN(data)) {
    rb_raise(rb_eRuntimeError, "data cannot be empty");
  }

  ctxt = xmlCreateMemoryParserCtxt(StringValuePtr(data),
                                   (int)RSTRING_LEN(data));
  if (ctxt->sax) {
    xmlFree(ctxt->sax);
    ctxt->sax = NULL;
  }

  return Data_Wrap_Struct(klass, NULL, deallocate, ctxt);
}
```

Impact

This issue may lead to DoS or Information Disclosure.

Resources

The following ruby proof of concept will crash the process by dereferencing the address `0xcafecafe`.

```
require "nokogiri"
parser = Nokogiri::XML::SAX::Parser.new
parser.parse 0xcafecafe >> 1
```

Output from an `irb` session:

```
irb(main):003:0> require "nokogiri"
=> true
irb(main):004:0>  parser = Nokogiri::XML::SAX::Parser.new
=> #<Nokogiri::XML::SAX::Parser:0x0000000108d90518 @document=#<Nokogiri::XML::SAX::Document:0x0000000108d90478>, @encoding="UTF-8", @warned=fa
irb(main):005:0> parser.parse 0xcafecafe >> 1
/opt/homebrew/lib/ruby/gems/3.1.0/gems/nokogiri-1.13.4-arm64-darwin/lib/nokogiri/xml/sax/parser.rb:111: [BUG] Segmentation fault at 0x00000000
ruby 3.1.2p20 (2022-04-12 revision 4491bb740a) [arm64-darwin21]
```

In order to leak information about the address space in which nokogiri is running, an attacker can repeatedly crash the server by trying addresses until the remote server does not crash, therefore indicating that the supplied value is a valid memory address.

If nokogiri is being used in a way which reflects the results of the parsing to the attacker, this vulnerability can be used to leak arbitrary ruby strings which may contain secrets that the attacker should not have access to. In order to do this an attacker has to guess the address of the target string in memory.

Issue 2: Type confusion in `html4_sax_parser_context.c` (`GHSL-2022-032`)

An identical vulnerability can be found in the `html4` version of the parser.

html4_sax_parser_context.c#L25

```
static VALUE
parse_memory(VALUE klass, VALUE data, VALUE encoding)
{
  htmlParserCtxtPtr ctxt;

  if (NIL_P(data)) {
    rb_raise(rb_eArgError, "data cannot be nil");
  }
  if (!(int)RSTRING_LEN(data)) {
    rb_raise(rb_eRuntimeError, "data cannot be empty");
  }

  ctxt = htmlCreateMemoryParserCtxt(StringValuePtr(data),
                                    (int)RSTRING_LEN(data));
  if (ctxt->sax) {
    xmlFree(ctxt->sax);
    ctxt->sax = NULL;
  }

  if (RTEST(encoding)) {
    xmlCharEncodingHandlerPtr enc = xmlFindCharEncodingHandler(StringValueCStr(encoding));
    if (enc != NULL) {
      xmlSwitchToEncoding(ctxt, enc);
      if (ctxt->errNo == XML_ERR_UNSUPPORTED_ENCODING) {
        rb_raise(rb_eRuntimeError, "Unsupported encoding %s",
                 StringValueCStr(encoding));
      }
    }
  }

  return Data_Wrap_Struct(klass, NULL, deallocate, ctxt);
}
```

Impact

This issue may lead to DoS or Information Disclosure.

CVE

- CVE-2022-29181

Resources

- GitHub Security Advisory GHSA-xh29-r2w5-wx8m

Credit

These issues were discovered and reported by GHSL team member @agustingianni (Agustin Gianni).

Contact

You can contact the GHSL team at `securitylab@github.com`, please include a reference to `GHSL-2022-031` or `GHSL-2022-032` in any communication regarding these issues.

GitHub

Product

- Features
- Security
- Enterprise
- Customer stories
- Pricing
- Resources

Platform

- Developer API
- Partners
- Atom
- Electron
- GitHub Desktop

Support

- Docs
- Community Forum
- Professional Services
- Status
- Contact GitHub

Company

- About
- Blog
- Careers
- Press
- Shop