GitLab 任意文件读取漏洞 (CVE-2016-9086) 和任意用户 token 泄露漏洞

笑然 / 2016-11-09 11:58:00 / 浏览数 3306 安全技术 漏洞分析 顶(0) 踩(0)

---

Author:dawu,LG(知道创宇404安全实验室)

Data:2016-10-09

## 0x00 漏洞概述

### 1.漏洞简介

GitLab 是一个利用Ruby on
Rails开发的开源应用程序，实现一个自托管的Git项目仓库，可通过Web界面进行访问公开的或者私人项目。近日研究者发现在其多个版本中存在文件读取漏洞(CVE-2016-9
和 任意用户authentication_token泄漏漏洞，攻击者可以通过这两个漏洞来获取管理员的权限，进而控制所有gitlab项目。

### 2.漏洞影响

任意文件读取漏洞(CVE-2016-9086):

GitLab CE/EEversions 8.9, 8.10, 8.11, 8.12, and 8.13

任意用户authentication_token泄露漏洞：

Gitlab CE/EE versions 8.10.3-8.10.5

## 0x01 漏洞复现

### 1.环境搭建

```
sudo apt-get install curl openssh-server ca-certificates postfix
curl -s https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh | sudo bash
sudo apt-get install gitlab-ce=8.10.3-ce.1  #■■■8.10.3■■■■■■■■■■authentication_token■■■■
■■■■■
sudo gitlab-ctl reconfigure
```

安装完成后访问服务器80端口即可看到GitLab登录页面.
注：8.9.0-8.13.0版本的gitlab的项目导入功能需要管理员开启，8.13.0版本之后所有用户都可以使用导入功能。管理员可以访问http://domain/admin/application
开启，开启之后用任意用户新建项目的时候，可以在import project from一项中看到gitlab export。

## 2.漏洞分析

### 任意文件读取漏洞(CVE-2016-9086)

从8.9.0版本开始，GitLab新增了导入导出项目的功能。
一个空的gitlab项目导出后结构如下：

其中VERSION文件内容为GitLab的导出模块的版本，project.json则包含了项目的配置文件。

当我们导入GitLab的导出文件的时候，GitLab会按照如下步骤处理： 1.服务器根据VERSION文件内容检测导出文件版本，如果版本符合，则导入。
2.服务器根据Project.json文件创建一个新的项目，并将对应的项目文件拷贝到服务器上对应的位置。

检测VERSION文件的代码位于：/lib/gitlab/import_export/version_checker.rb中：

```
...
def check!
 version = File.open(version_file, &:readline)
 verify_version!(version)
rescue => e
 shared.error(e)
 false
end
...
def verify_version!(version)
   if Gem::Version.new(version) != Gem::Version.new(Gitlab::ImportExport.version)
       raise Gitlab::ImportExport::Error.new("Import version mismatch: Required #{Gitlab::ImportExport.version} but was #{vers
   else
       true
```

```
   end
end
...
```

我们可以看到这里的逻辑是读取VERSION文件的第一行赋值给变量version，然后检测verison与当前版本是否相同，相同返回true，不相同则返回错误信息(错误信息中
于是漏洞发现者Jobert Abma巧妙的使用了软链接来达到读取任意文件的目的。首先，我们给VERSION文件加上软链接并重新打包。

```
ln -sf /etc/passwd VERSION
tar zcf change_version.tar.gz ./
```

这样，读取VERSION文件的时候服务器就会根据软链接读取到/etc/passwd的第一行内容并赋值给version。但是由于version与当前版本不相同，所以会输出version

访问之前搭建好的GitLab服务器，创建一个新的项目，填写完项目名称后在Import project from一栏中选择GitLab
export，上传我们修改后的导入包，然后就可以看到/etc/passwd文件第一行

但是，如果只读取任意文件的第一行，能做的事情还是太少了。漏洞发现者显然不满足这一结果，他继续找了下去.
读取Project.json这一配置文件的代码位于：/lib/gitlab/import_export/project_tree_restorer.rb中：

```
...
def restore
 json = IO.read(@path)
 tree_hash = ActiveSupport::JSON.decode(json)
 project_members = tree_hash.delete('project_members')

 ActiveRecord::Base.no_touching do
   create_relations
 end
rescue => e
 shared.error(e)
 false
end
...
```

在这里，我们可以再次使用软链接使变量json获取到任意文件的内容，但是由于获取的文件不是json格式，无法decode，导致异常抛出，最终在前端显示出任意文件的内
添加软链接并打包:

```
ln -sf /etc/passwd project.json
tar zcf change_version.tar.gz ./
```

上传导出包，页面上显示的结果：

### 任意用户authentication_token泄露漏洞

复现步骤为：

1.注册一个普通用户，创建一个新的项目

2.在项目的member选项中，添加管理员到项目中。

3.点击edit project,找到Export project部分，点击Export project，等待几分钟去查看注册邮箱收到的下载地址或者刷新页面，点击Download
export下载导出包。

4.导出包的project.json中已经含有了管理员的authentication_token。

得到authentication_token之后我们就可以通过api做管理员可以做的事情了，比如查看管理员所在的项目：

分析原因：

我们在\app\controllers\projects_controller.rb中找到了export函数，这个函数被用来导出项目文件。

```
def export
   @project.add_export_job(current_user: current_user)

   redirect_to(
     edit_project_path(@project),
     notice: "Project export started. A download link will be sent by email."
   )
 end
```

往下跟add_export_job(),在\app\models\project.rb中：

```
def add_export_job(current_user:)
   job_id = ProjectExportWorker.perform_async(current_user.id, self.id)

   if job_id
     Rails.logger.info "Export job started for project ID #{self.id} with job ID #{job_id}"
   else
     Rails.logger.error "Export job failed to start for project ID #{self.id}"
   end
 end
```

继续到\app\workers\project_export_worker.rb文件的ProjectExportWorker.perform_async():

```
class ProjectExportWorker
 include Sidekiq::Worker

 sidekiq_options queue: :gitlab_shell, retry: 3

 def perform(current_user_id, project_id)
   current_user = User.find(current_user_id)
   project = Project.find(project_id)

   ::Projects::ImportExport::ExportService.new(project, current_user).execute
 end
end
```

这里我们可以看到current获取的是User.find(current_user_id)的内容，然后调用::Projects::ImportExport::ExportService.new(project, current_user).execute 由于笔者之前没有接触过ruby，这里只好采用gitlab-rails console来找到User.find()的值。可以看到，在User.find()中,存在authentication_token的值。

跟到\app\services\project\import_export\export_service.rb，这里执行version_saver, avatar_saver, project_tree_saver, uploads_saver, repo_saver, wiki_repo_saver这五个函数来写各种导出文件，其中project_tree_saver()负责导出project.json

```
module Projects
 module ImportExport
   class ExportService < BaseService
     def execute(_options = {})
       @shared = Gitlab::ImportExport::Shared.new(relative_path: File.join(project.path_with_namespace, 'work'))
       save_all
     end

     private

     def save_all
       if [version_saver, avatar_saver, project_tree_saver, uploads_saver, repo_saver, wiki_repo_saver].all?(&:save)
         Gitlab::ImportExport::Saver.save(project: project, shared: @shared)
         notify_success
       else
         cleanup_and_notify
       end
     end

     def version_saver
     ...
   end
 end
end
```

跳过之后的几个繁琐的调用之后，执行了lib/gitlab/import_export/json_hash_builder.rb中的create_model_value函数。

```
# Constructs a new hash that will hold the configuration for that particular object
 # It may include exceptions or other attribute detail configuration, parsed by +@attributes_finder+
 #
 # +current_key+ main model that will be a key in the hash
 # +value+ existing model to be included in the hash
 # +json_config_hash+ the original hash containing the root model
 def create_model_value(current_key, value, json_config_hash)
   parsed_hash = { include: value }
   parse_hash(value, parsed_hash)

   json_config_hash[current_key] = parsed_hash
```

```
end

# Calls attributes finder to parse the hash and add any attributes to it
#
# +value+ existing model to be included in the hash
# +parsed_hash+ the original hash
def parse_hash(value, parsed_hash)
  @attributes_finder.parse(value) do |hash|
    parsed_hash = { include: hash_or_merge(value, hash) }
  end
end
```

这里出现了逻辑问题，由于parsed_hash这个变量不是全局变量，所以create_model_value()中执行parse_hash()时，parse_hash()中的parsed_hash被改变，

我们在gitlab-rails
console里展示了这两者的区别。当value=user的时候，parsed_hash={:include=>:user}，输出的结果如同图中的user.as_json()，会将所有内容输出，包括a
:email, :username]}}}时，输出结果与user.as_json(only: [:id, :email, :username])相同。

后续RCE方式的探讨

在hackone的两个报告中，漏洞发现者都提到了leads to
RCE，笔者尝试去实现这一点。由于GitLab源码在gitlab.com上，所以当获取了GitLab的管理员权限后，我们可以通过authentication_token修改GitLab项目的源
为了重现这种情况，我们在本地新建一个新的项目去通过authentication_token和GitLab api来修改项目文件。

用root账户创建一个项目:test_rce，其中README.md的内容为created by
root接下来，我们要用gitlab的api来修改它。首先，根据projects■api找到test_rce项目对应的id，这里是18

```
curl -H "PRIVATE-TOKEN: wTPMMapDwpfkKfNws7xp" "http://domain/api/v3/projects"
```

我们再根据api读取一下文件

```
curl -H "PRIVATE-TOKEN: wTPMMapDwpfkKfNws7xp" "http://domain/api/v3/projects/18/repository/files?file_path=README.md&ref=maste
```

这里，content为Y3JlYXRlZCBieSByb290，这是文件内容被base64加密后的结果，解密一下就可以看到created by root

根据api的要求，我们通过PUT数据来修改文件，将README.md修改为change by notroot。
当我们再读一次，content内容为：Y2hhbmdlIGJ5IG5vdHJvb3Q=，解码之后就是change by notroot

不得不说，笔者所实现的这种方式攻击时间跨度很长，能否执行命令取决于开发者下一次更新的时间，这也是这种方法的缺点之一。

## 0x02 官方修复分析

### 任意文件读取漏洞(CVE-2016-9086)修复分析

我们可以看到，官方先移除了导入包里的软连接，其次，读取VERSION的内容和project.json的内容出错后将内容输出到日志里而非返回到前端。

### 任意用户authentication_token泄露漏洞修复分析

官方让json_config_hash[current_key]获取到parse_hash()处理后的值。

## 0x03 参考


- https://www.seebug.org/vuldb/ssvid-92529
- https://www.seebug.org/vuldb/ssvid-92516
- https://hackerone.com/reports/178152
- https://hackerone.com/reports/158330
- https://github.com/gitlabhq/gitlabhq/commit/912e1ff4284eb39fe020b8e823085a2cb7f244fb
- https://github.com/gitlabhq/gitlabhq/commit/4389f09e668c043c8a347c4c63f06795110dfbb3#diff-b10a896b29121489e3b2fb396bc53d8a
- https://gitlab.com/gitlab-org/gitlab-ce/issues/20802
- https://gitlab.com/help/user/project/settings/import_export.md
- https://docs.gitlab.com/ce/api/





上一篇：【讨论】Linux应急响应 下一篇：Python代码审计连载之一：CSRF


1. 0 条回复
   - 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

---

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板