[翻译]来自海莲花（OceanLotus）打造的全新基于macOS后门

July 11, 2017 • 带头大哥

介绍

最近在我们WildFire云分析平台上发现了一个新版本的"海莲花（OceanLotus）"后门，这可能是我们在macOS更新日志中比较新的后门。这次迭代的目标是一些在越南的感染用户，同时也修复了低级的AV检测问题，这个问题自从第一次发现，到现在将近有一年了。如果忽略非正式的几次更新，这个系统的使用情况还是很活跃的， 在我们分析的时候也就是2017年6月份上旬，我们可以直接通过发送命令连接和控制服务器。

虽然这个看起来和2015年5月份发现的海莲花（OceanLotus）的一个例子很相似，但从那时起已经有了各种各样的改进。一些改进包括病毒文件的使用，去除使用命令行工具，完善的字符串编码机制，使用加密的自定义二进制协议通信量，还有一个模块化的后门。

感染途径

新的海莲花后门是渗透在一个zip文件中。虽然我们没有直接的原始感染路径证据，但我们推测它很可能通过电子邮件附件。一旦用户提取了ZIP文件，就会看到一个目录，其中包含一个带有微软Word文档图标的文件。该文件实际上是一个应用程序包，其中包含可执行代码。（参见图1）。一旦用户双击据称的Word文档，特洛伊木马程序就会执行，然后启动Word来显示一个诱饵文档。

该恶意软件使用诱饵文件，以帮助掩盖恶意软件的执行。这种技术是基于Windows恶意软件的一种普通方法，但很少在MacOS使用。为了达到这层混淆，恶意软件作者不得不欺骗操作系统相信文件夹是一个应用程序包而忽略扫描这个docx扩展文件。通常，MacOS的恶意软件会模仿合法应用的安装程序如Adobe Flash，这就是如何将OceanLotus的旧版本打包。

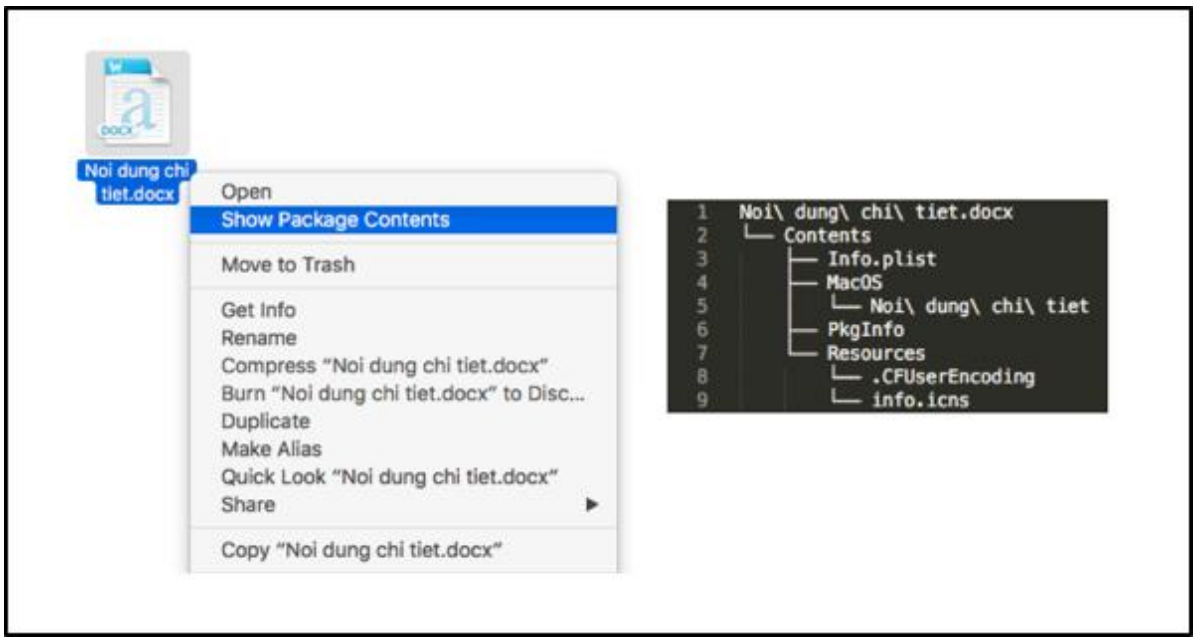

Figure 1

一旦应用程序包被加载，其就会打开包内Resources文件夹中一个名为.CFUserEncoding的隐藏文件。这是一个有密码保护的Word文档（见图2）。它还将该文件复制到可执行路径，并在驻留机制设置之后替换了应用程序包。这将导致受害者相信没有什么不妥，因为他们认为他们正在打开一个word文档，然后这个Word文档就打开了。在这种情况下，该Word文件的名称为“Noi dung chi tiet.docx”，这一表达正是越南语中的“Details(细节信息)”之意。

Figure 2

驻留机制

与这个后门的前一版本相比，这种驻留机制基本保持不变。这个版本创建一个Launch Agent，在受害者主机启动时运行，而早期版本的执行，是当用户登录时。另外，新版本后门会根据运行应用程序之用户的具体UID将自身以不同文件名复制到不同位置。

对于登陆的用户不是超级用户，该后门会采集由 *getpwuid()函数返回的MD5哈希值结构，将这个哈希值拆分成 <前 8 位哈希值>-<接下来16 位哈希值>-<最后8位哈希值>.*。这些分段的MD5哈希值在前面会附上*"0000-"*，然后作为在*~/Library/OpenSSL/*下的一个目录来保存可执行文件（详见图三）。如果登陆的是超级用户，该可执行文件即会被存储在系统系统级的库目录中：/Library/TimeMachine/bin/mtmfs。

值得一提的是，该可执行文件以及plist位置从表面上看与其它合法应用程序无异。

UID	plist Location	Executable Location
0	/Library/LaunchDaemons/com.apple.mtmfsd.plist	/Library/TimeMachine/bin/mtmf
> 0	~/Library/LaunchAgents/com.apple.openssl.plist	~/Library/OpenSSL/0000-<segmented MD hash>/servicessl

Figure 3

一旦这个恶意软件设置成驻留机制，它就会从可执行路径下删除源应用程序包，只留下诱饵文件，然后从新的路径下将自身作为一个服务加载。

去除命令行工具

关于这个后门，我们首先注意到它缺少可疑的字符串，这些字符串通常会表明恶意软件在受害者主机的目的。大多数MacOS的恶意软件，会有些地方用system()或exec()函数来运行其他脚本。在这种情况下，这些没有明显存在调用这些函数，也没有命令行工具字符串，可以轻松地传达应用程序的恶意意图。这说明相比其他攻击者通常只会复制和粘贴来自网络的脚本，这个后门的作者对macOS 平台会有更加深层次的了解。

这种字符串的缺失加大逆向分析技术的难度，会让该恶意软件看起来不那么可疑，尤其是对于基础的静态分析。

字符串解码

由于在明文上几乎没有明显的可疑之处，于是我们转向考虑作者使用编码或者混淆字符串的可能性。

这个后门的字符串解码方式是从前一版本升级来的，其中字符串是以“Variable”作为键进行XOR编码。字符串解码方式现在由bit位移和XOR操作组成，“Variable”键取决于所编码字符串的长度。如果变量XOR的键计算结果是0，则使用“XOR”默认的键0x1B。图4显示了解码函数的Python实现。

```
def revbits(c):
  binary = bin(c)[2:].rjust(8,'0')
  return int(binary[::-1], 2)

def rotbits(c, amt, lr):
  amt = amt&7
  if not lr:
    return ((c << (amt)) | (c >> (8-amt)))&0xff
  else:
    return ((c >> (amt)) | (c << (8-amt)))&0xff

def decode(s):
  s = bytearray(s)
  #reverse bits
  s = bytearray(revbits(c) for c in s)

  #add shift
  shift_amt = 0xfc
  shift_amt = (-(len(s)%8))&0xff or shift_amt
  s = bytearray((c+shift_amt)&0xff for c in s)

  #xor
  xor_amt = 0x1b
  var_key = 0x21 * (  ((0x3E0F83E1*len(s))>>63) + ((0x3E0F83E1*len(s))>>35)  )
  if len(s) != var_key:
    xor_amt = len(s) - var_key
  s = bytearray((c^xor_amt) for c in s)

  #rotate bits in str
  s = bytearray(rotbits(c, len(s)%8 or 4, 0) for c in s)

  return s
```

Figure 4

在解码字符串（参见图5）之后，我们可以观察到恶意软件如何建立永久机制，调查受害者的计算机，并将这些信息发回服务器。在这一点上，这个恶意软件包含后门所实现的功能仍然不明显。

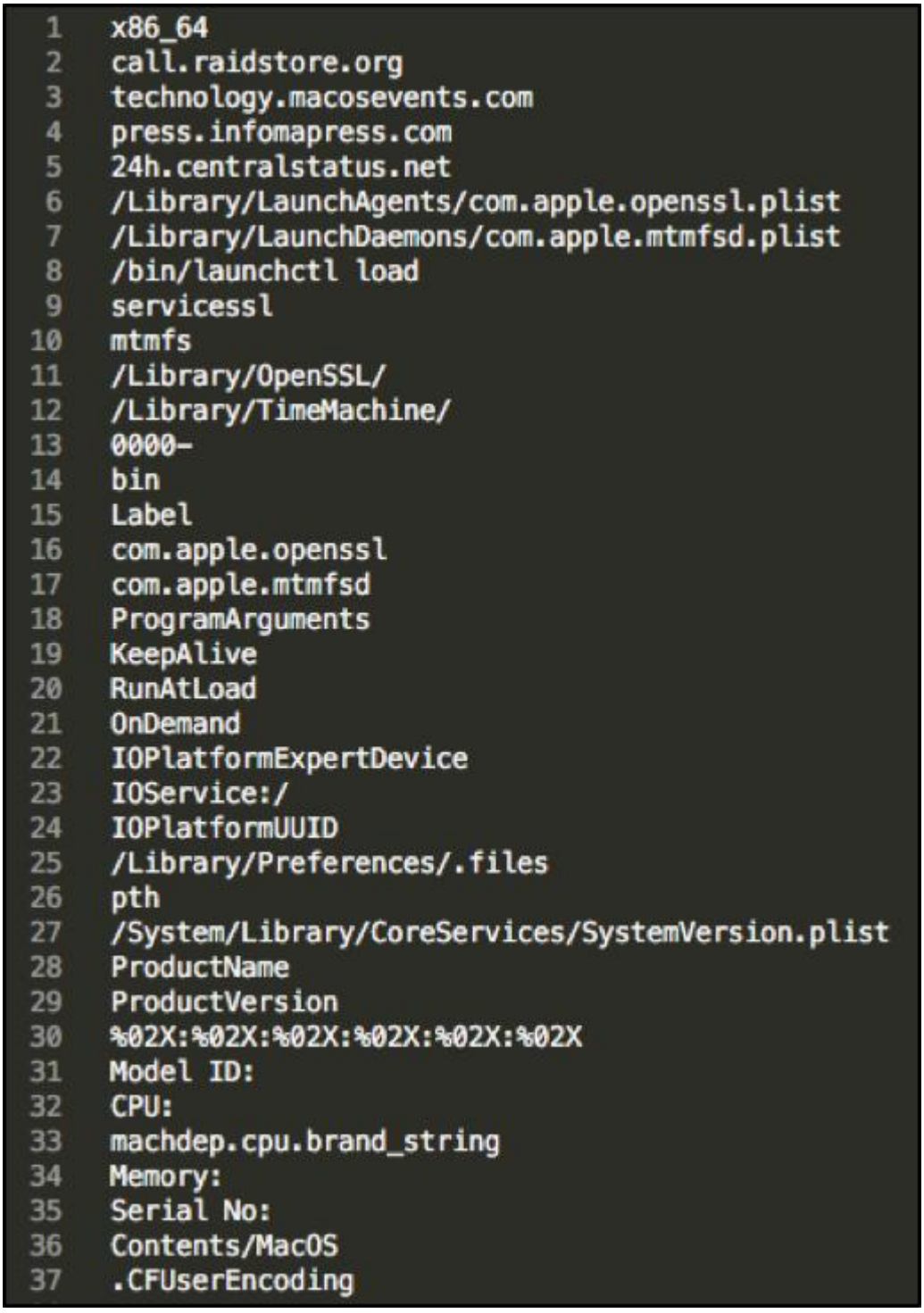

Figure 5

自定义的二进制协议和加密数据包

负责此恶意软件的攻击者似乎花费了大量的精力来开发自己的自定义通信协议。他们并不是简单地使用现成的Web服务器来实现他们的命令和控制服务器，这和常规的手法不同。相反，他们创建了自己的命令和控制机制。

后门使用TCP端口443上的自定义二进制协议，这是一个众所周知的端口，由于它在HTTPS连接中的使用，它不太可能被传统防火墙阻止。如图六所示的数据包利用bit位移（见图七）与XOR配合0x1B键共

同进行编码。在执行XOR运算之前，位总是向左移动3次。这是从以前的版本的包只有XOR配合0x1B键编码的改进。

```
00000000  0b 41 61 54 03 e0 c3 8a  c3 63 63 63 63 63 63 2d   .AaT.... .ccccccc-
00000010  23 81 23 8c 67 ef 67 43  3f 63 63 63 63 8c 63 63   #.#.g.gC ?cccc.cc
00000020  63 63 63 63 63 63 63 63  63 63 63 63 63 63 63 63   cccccccc cccccccc
00000030  63 63 63 63 63 63 63 63  63 63 63 63 63 63 63 63   cccccccc cccccccc
00000040  63 63 5e 63 63 63 63 63  73 43 6f 9c 63 42 63 63   cc^ccccc sCo.cBcc
00000050  77 43                                              wC
```

Figure 6

```
__int64 __fastcall rotbits(unsigned int a1, int amt, char lr)
{
  unsigned int v3; // ecx@3
  char v4; // di@3
  unsigned int v5; // eax@3

  if ( amt > 0 )
  {
    do
    {
      if ( lr )
      {
        v3 = (char)a1;
        v4 = (_BYTE)a1 << 7;
        v5 = v3 >> 1;
        if ( BYTE1(v3) & 1 )
          LOBYTE(v5) = v5 + -128;
        LOBYTE(a1) = v5 | v4;
      }
      else
      {
        a1 = ((unsigned int)(unsigned __int8)a1 >> 7) | (unsigned __int8)(2 * a1);
      }
      --amt;
    }
    while ( amt );
  }
  return (unsigned int)(char)a1;
}
```

Figure 7

解码安装包后，我们可以看到不同字段的断点。图8显示客户端向服务器发送的初始数据包。除了“magic”字节、数据长度和通信类型之外，数据包相对比较空。

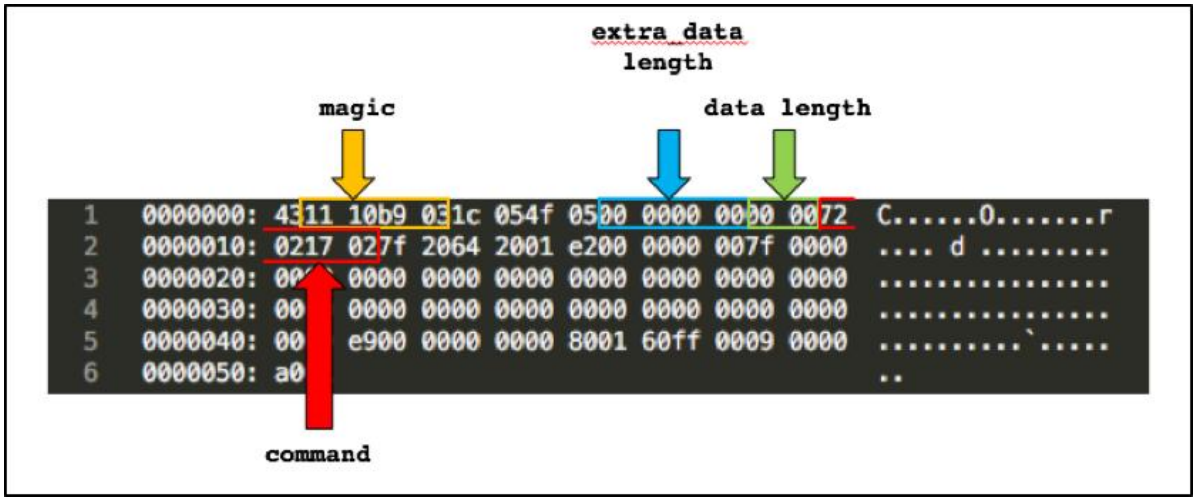

Figure 8

根据从服务器发回来的命令，数据包可能会大于0x52个字节。超过0x52个字节部分的数据会用zlib进行压缩，然后，在CBC模式下用AES加密，同时带有空的初始化向量（IV）和密钥填充的32个字节从服务器发送出来。

我们捕捉到从服务器发出来的实时流量，我们观察到从服务器发出来的秘钥是一次性的。这就意味着每一次同服务器建立新的会话都会得到不同的秘钥来加密发送回来的数据。相比于旧版本，这是一次标志性的改进，旧版本只用单字节密钥对XOR编码进行加密。

在对从服务器接收到的数据包解码之后，后门会去验证某些字段，如“magic”字节，并确保接收到的数据的长度不超过一定值。在整个程序执行过程中，它还检查和处理可能生成的任何错误。

C&C通信

C&C服务器通信的顺序如下“

1. 客户机要向服务器发起一次会话，会发送一个命令段为0x2170272的数据包
2. 服务器会响应一个带有一次性秘钥和一个命令的数据包。
3. 客户端检查接收到的数据包是否有效。
4. 客户端执行服务器发送的命令，响应一个zlib压缩和AES加密的乱序结果然后发送回服务器。

不像以前的版本，海莲花（oceanlotus）的命令可以很容易地从它的字符串收集到，作者函数与常数打乱。我们解码了以下可用命令，如图9所示。

命令	**命令描述**
0x2170272	初始化
0x5CCA727	???
0x2E25992	从服务器接收文件
0x2CD9070	在文件/路径下获取相关信息

0x12B3629	删除文件/路径
0x138E3E6	???
0x25D5082	从一个动态库内执行一个函数
0x25360EA	发送文件到服务器
0x17B1CC4	???
0x18320E0	发送附有水印的受害者和电脑信息
0x1B25503	从一个动态库内执行一个函数
0x1532E65	从一个动态库内执行一个函数

Figure 9

Command 0x2170272

当这个后门开始执行时，它会根据受害者的用户ID在 */Library/Preferences/.files* or *~/Library/Preferences/.files路径下创建一个文件。这个文件（参见图10）包含一个时间戳和与机器的序列号相关联的受害目标名称，然后用MD5。进行两次的希哈化。然后复制到缓冲区，并由AES以CBC模式利用一条空IV与一个"pth"键进行加密。加密后的结果将被保存在该文件当中。。然后将其保存到文件中。*

时间戳+MD5(MD5<受害目标名称+设备序列号>)

当这个文件创建之后，这个客户机胡发送他的命令段带有0x2170272的第一个数据包。服务器使用相同的命令确认并回应，客户机随后会确认这个文件是否已经建好。

```
\xa7\xf1\xd9*\x82\xc8\xd8\xfe4137674062B3226FE630C24F7DE1021E\x00\x00\x00\x00\x00\x0
```

```
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
00\x00\x00\x00\x00\x00\x00\x00\x00\x00
```

Figure 10

命令0x18320E0

客户机发出0x2170272命令后，服务器会发送一个带有密钥的命令。客户机收集图11所示的所有数据，用服务器提供的密钥加密这些数据并将其发送回服务器。有一点要注意的是， Base64字符串是在这个数据包发送的。这个字符串在二进制中是静态的，不会改变，这可能是用于某种活动或版本标识的标记。

```
\x00\x00\x004137674062B3226FE630C24F7DE1021E\xe9\x0f\x00\x00\x00Mac   OS X 10.X.X\xb

\x00username\t\x00\x00\x00localhost\x18\x00\x00\x00Ze0pXcpfbqbS4wD0eS/LVQ==\xb6\xbc\
ABCDEF01-23456789ABCDEF01-23456789/

servicessl\x8b\xbc\x1cY\x00\x00\x00\x00\x17\x00\x00\x00en0     :            AA:BB:CC:DD:EE:FF[\
fe80::aaaa:bbbb:cccc:111\nen0 : 192.168.1.254

\x05\x01\x00\x00f\x00\x00\x00Model   ID:iMac8,1\nCPU:Intel(R) Core(TM)2 Duo CPU      T7700
```

Figure 11

在图11中没有明显的标记，但在这个包中也包含了内核引导时间，C&C服务器可以使用它来帮助确定后门是否在沙箱环境中运行。

命令0x25D5082, 0x1B25503, 0x1532E65

这些命令使用dlopen()加载动态库，同时包含一项函数指针以利用dlsym()在该共享库内实现执行。遗憾的是,我们不知道哪一个动态库或函数用于每个命令。因为这些是由服务器提供的，而且我们没有能够捕获任何使用这些命令的会话。

然而,我们可以推测,因为函数具有相同数量的参数且各命令常量的开头部分非常相似（详见图十二），而此后门又有接收文件的功能,这些函数可能对应于一个务器上传到受害目标的一个共享库。这意味着通过C&C服务器直接上载一些模块到这个后门一实现添加额外的功能。

```
v12 = ((__int64 (__fastcall *)(signed __int64, _QWORD, _QWORD, void **, int *, int *))func_ptr)(
        0x116DE9CLL,
        0LL,
        0LL,
        v20,
        &v23,
        &v22);

func_ptr(0x179896CLL, v6, v24, v18, &v21, &v20);


if ( func_ptr(0x5E57538LL, v8, v21, v17, &v20, &v19) )
```

Figure 12

结论

大多数基于macOS的恶意软件在如今都不不是很复杂，但是攻击者已经在快速提高他们的间谍情报技术。恶意软件的老练和复杂程度的提高可能在暗示着在未来以macOS设备为目标的攻击行为会越来越多。根据这次的海莲花（OceanLotus）和Sofacy组织对近来macOS版本推出的工具包，我们可以观察到许多针对mcOS的入侵方案。机构有必要将用于防御Windows设备同样等级的安全措施和策略也用于macOS。

我们已经观察到多种专门针对Mac OS的入侵威胁方案。正因为如此，各类组织机构需要甚至必须以等同于Windows设备防御水平的安全实践与实施策略对自有Mac OS设备加以保护。

苹果公司已经更新了macOS保护系统来防御这个变种的海莲花（OceanLotus）病毒的变种

Hashes

b3337016785333070494568 4c50ce0af6eb27838e1e3f88ea457d2c88a223d8b Noi dung chi tiet.zip

b3cf3e3b52b4b899cd0814fc75698ea24f08ce18642665adcd3555a068b5c16d Info.plist

07154b7a45937f2f5a2cda5b701504b179d0304fc653edb2d0672f54796c35f7 Noi dung chi tiet

82502191c9484b04d685374f9879a0066069c49b8acae7a04b01d38d07e8eca0 PkgInfo

f0c1b360c0b24b5450a79138650e6ee254afae6ce8f6c68da7d1f32f91582680 .CFUserEncoding

e84b5c5152d8edf1e814cc4b4975bfe4dc0063ef90294cc96b383f523042f783 info.icns

C&C 服务器

call[.]raidstore[.]org

technology[.]macosevents[.]com

press[.]infomapress[.]com

24h[.]centralstatus[.]net

93.115.38.178

Dropped Files

UID == 0	UID > 0
/Library/LaunchDaemons/com.apple.mtmfsd.plist	~/Library/LaunchAgents/com.apple.opens

/Library/TimeMachine/bin/mtmfs	~/Library/OpenSSL/0000-<segmented hash>/servicessl
/Library/Preferences/.files	~/Library/Preferences/.files

Tags: 服务器 , 文件 , 命令 , 数据包 , 后门 , 字符串 , 发送 , 加密 , 用户 , 解码 ,

为您推荐了相关的技术文章:

1. 漏洞检测的那些事儿 - 从理论到实战
2. Struts2 历史 RCE 漏洞回顾不完全系列
3. 从反序列化到命令执行 - Java 中的 POP 执行链
4. "安全线"大型目标渗透 - 01信息搜集|漏洞研究 - 安全技术社区
5. 记一次ThinkPHP源码审计

原文链接： bbs.pediy.com