Backdoor

假死疑云：Wpeeper木马所图为何？

Alex.Turing, Acey9, heziqian

2024年4月29日 • 15 min read

简介

2024年4月18日，`XLab的未知威胁狩猎系统`发现一个VT 0检测的ELF文件正通过2个不同的域名传播，其中一个域名已被3家安全产商标注为恶意，另一个域名为近期注册且无任何检测，这个异常点引起了我们的注意。经过分析，我们确认此ELF是一个针对Android系统的恶意软件，基于它使用被黑的WORD PRESS站点做为中转C2，我们将其命名为Wpeeper。

Wpeeper是一个针对Android系统的典型后门木马，支持收集设备敏感信息，文件&目录管理，上传&下载，执行命令等诸多功能。Wpeeper最大的亮点在于网络层面，以下3点体现了其作者的用心程度：

1. 依托被黑的WORD PRESS站点构建多层级的C2架构，隐藏真正的C2
2. 使用Session字段区分请求，HTTPS协议保护网络流量
3. C2下发的指令使用AES加密并带有椭圆曲线签名，防止被接管

Wpeeper来源于“二次打包”的UPtodown Store应用，攻击者向正常的APK中植入了一小段代码用于下载执行恶意的ELF。由于新增的代码量很少，被修改的APK目前在VT上也是0检测。UPtodown 是一个类似Google Play的第3方应用商店，在全球有许多用户，我们认为这是攻击者选择Uptodown的原因之一。由于视野有限，我们尚不得知攻击者是否有其它的选择。

4月22日，Wpeeper突然停止活动，其C2服务器和下载器均不再提供服务。目前相关的样本仍未被安全厂商检测，可以说它已成功地欺骗了所有安全厂商，没有理由在这个节点选择"退场"，因此我们认为其背后可能有更大的图谋。

戛然而止的Campaign?

关于Wpeeper的此次活动，我们有较为完整的视野。

- 4月17日Wpeeper首次被上传到VT
- 4月18日系统告警，开始分析
- 4月19日开始指令跟踪，收到36个新的C2
- 4月22日8点31分收到最后一条的指令

最后一条指令，功能号为12，它的作用是删除自身。一开始我们以为是指令跟踪暴露了，但在切换过跟踪IP之后依然没有效果；随后Downloader也不再提供样本下载，整个Campaign似乎突然被按下了停止键。

Wpeeper的作者为什么要这样做呢？也许他是放弃了，但我们认为还存在这样的可能性：二次打包的APK是Wpeeper后门的下载器，它们都已成功躲避了杀软的检测，但所谓`草蛇灰线`，只要存在网络活动，那就有可能被发现。当下不如先主动停止网络服务，让APK继续保持杀软眼中的良民身份，先提高APK的安装量，等规模上去之后，就是**Wpeeper露出獠牙之时**。

在野规模

对于Wpeeper的规模，我们没有直接的数据，但从Google以及PDNS的结果来看，Wpeeper的感染量在千级，并没有广泛的传播。

1. APK的安装量

通过Google中查询“二次打包”的UPtodownAPK的MD5可以得到一些结果：一个名为aapks.com的网站提供了下载；而在一个名为Android-apk.org的网站上疑似有统计结果，4月20日的数字是是1743，目前它的下载次数正在增加中。

2. Downloader的PDNS
   APK中内嵌了2个Downloader域名，它们在PDNS系统中的的解析次数可以间接的反应了在野规模。

APK分析

目前我们只捕获了一个“二次打包”的Uptodown应用，它的基本信息如下所示：

```
Family: Wpeeper Downloader
MD5: 3dab5a687ab46dbbd80189d727637542
Type: Android
Package Name: com.uptodown
Version: 5.92
```

攻击者在APK的MainActivity中加入了以下代码，用于创建一个新线程(MainActivity.F3)下载恶意的ELF文件。

MainActivity.F3函数最终会调用MainActivity.h5函数，代码没有被混淆，功能一目了然，即从2个域名下载名为`android`的ELF文件，重命名成com.uptodownload.libs，最后启动执行。

ELF分析

从俩个域名下载的恶意ELF文件，没有差异，它的基本信息如下：

```
Family: Wpeeper
MD5: 8e28f482dab8c52864b0a73c3c5c7337
Magic: ELF 64-bit LSB pie executable, ARM aarch64, version 1 (SYSV), dynamically li

Packer: None
```

Wpeeper没有使用对抗技术，对它的逆向分析比较容易。它的的功能比较简单，概括来说，当它在受害者设备运行时，检测同路径是否有store.lock文件，此文件存有配置信息，包括C2，Cookie，Interval等。若存在store.lock文件，则使用AES CBC模式解密得到C2，通过libcurl构造POST请求，与C2建立通信，等待执行C2下发的指令；若不存在，则使用Base64解码样本内嵌的C2，将它们加密保存到store.lock文件中，然后同样的进入和建立通信，执行指令的流程。下文将围绕这些方面，剖析Wpeeper的功能实现。

获取C2

Wpeeper使用一个我们称之为c2_list的数组来保存当前的C2，最多支持30个，它的初始值为空。当它运行时，首要任务就是填充c2_list，填充内容的来源有以下2种。

来源1：硬编码

Bot使用以下代码解码样本内嵌的C2，填充c2_list。内嵌的C2是base64编码的，解码后总计9个（详情见IOC C2 Hardcoded)。

来源2：store.lock

Bot使用以下代码读取解密store.lock文件，填充c2_list。

store.lock的前32字节是AES秘钥，剩余部分是密文。以Wpeeper在测试设备运行一段时间之后生成的store.lock为例，解密前后如下所示：

PlainText中除了C2之外，还有id，cookie，interval等信息，其中flag是定值0x000004，interval用于控制休眠时长，cookie和id则参与http header Cookie字段的构造。

```
struct c2info
{
uint32 lenOfC2;
char[lenOfC2] C2s;
uint32 lenOfCookie;
char[lenOfCookie] cookie;
uint32 flag;
uint32 id;
uint32 flag;
uint32 interval;
}
```

除了上述2种填充方式，c2_list也支持运行时更新，当Wpeeper收到3号指令时，使用C2下发的payload更新c2_list。

网络通信

Wpeepber使用libcurl库构造POST请求与C2进行交互。其中Header中的Cookie，Session2个字段尤为重要，它们用来标识不同类型的请求。

以Bot与http[s]://snipsnack.com/T8Q2BN/产生的上线通信流量为例，

BOT ---> C2

Cookie字段的值使用base64解码后为“nrjzekrxohxw0”，其中“nrjzekrxohxw”为随机生成的小写字串长度为12，“0”是初始id；而Session解码后为“{ "auth": 1, "type": "search" }”，id与session的组合，表示这是Bot首次加入了网络，此次请求为beacon。

Wpeerer的服务端正是通过Session来区分不同类型的请求。

ID	SESSION	BODY	PURPOSE
From Bot (fixed 0)	{ "auth": 1, "type": "search" }	None	beacon
From C2	{ "auth": 0, "type": "search" }	None	request cmd
From C2	{ ["auth":] "type": "login" }	cmd result	upload result
From C2	{"o": xx, "l": xx, "type": "avatar" }	file context	upload specific length co
From C2	{ "l":1, "type": "query" }	None	request payload length
From C2	{"o": xx, "l": xx, "type": "query" }	None	request specific length c

C2 --->BOT

C2返回一个JSON对应，有id,data俩个字段，id用于和随机字串生成Cookie，data则是C2下发的指令。

当Bot收到data数据后，首先使用base64进行解码得到raw_data，raw_data的前64字节为数字签名，接着的32字节为AES秘钥，剩余部分为密文。

```
struct raw_data
{
uint8[64] signature;
uint8[32] key;
uint8[..] ciphertext;
}
```

只有当数字签名检验成功后，Bot才会继续AES解密，执行相应的指令。Wpeeper使用被黑的WORD PRESS站点做为中继C2，这种做法固然能很好的隐藏真实C2，但同时也引入了可靠性问题，毕竟中继C2不是真正掌握到攻击者手中。假设某个被黑站点（中继C2）的管理员发现了Wpeeper的痕迹，他是可以做到下发指令的，如果Bot对指令的“合法性”没有校验，那么整个网络是极易被摧毁的。Wpeeper所使用的**验签机制**消除了这种可能性，很好的保证了网络不被投毒，接管，或摧毁。

ciphertext使用AES CBC模式解密后就得到cmd_context，它的结构如下所示：

```
struct cmd_context
{
uint32 id;
uint32 cmd;
uint32 payload_len;
uint8[cmd_len] payload;
uint32 randstr_len;
uint8[randstr_len] cookie;
}
```

最后Bot根据cmd_context中cmd的值，执行不同的命令，并将结果回传给C2。

目前Wpeeper一共支持13个指令，以下是它们的编号与对应的功能。

CMD	FUNCTION
1	collect device info
2	collect pkg list
3	update c2
4	set interval
5	update pubkey
6	download
7	collect arbitrary file info
8	collect arbitrary dir info
9	exec arbitrary cmd via shell
10	download from C2 , then exec
11	update and exec
12	delete
13	download from arbitrary URL, then exec

我们实现了Appendix中的python，cyberchef脚本用于解密raw_data（对于解密感兴趣的读者，可以直接复制Appendix中的cyberchef到浏览器中，它很直观的展示了解密的过程），以上文C2发送的报文为例，解密后得到的plaintext如下所示，可

以看出cmd的值为3，当Bot收到这个报文时，就会执行3号指令，以payload中的域名对c2_list进行更新。

指令跟踪

前文对Wpeeper的逆向分析只回答了**它能做什么**这个问题，我们对于**它在做什么**依然是一片空白。为了回答做了什么这个问题，我们在`XLab指令跟踪系统`中实现了Wpeeper的网络协议，指令跟踪很快就给我们一些答案。

目前Wpeeper的指令数据量不大，但是稍加挖掘之后，我们有了一些有趣的发现。从统计数据来看Wpeeper的指令高度集中在1,2,3,4号指令中，即Wpeeper正在收集受害者设备的各种敏感信息，更新C2等，除此之外还有13号指令，通过它我们捕获了一个新的ELF文件**AppInstallerEx**，它的功能也是收集设备信息。

馈赠1: 新的C2

通过给Wpeeper的样本中硬编码的9个C2发送beacon请求，我们收到了3500个3号指令，我们捕获了3500组新C2，去重后共记36个（详情见IOC C2 NEW）。

馈赠2: AppInstallerEx

4月21日，我们收到了13号指令，下载执行一个名为AppInstallerEx的程序。

AppInstallerEx的功能是收集设备信息，上报到`eamdomai.com`。

C2角色分析

此次活动中Wpeeper使用了45个C2服务器。从这些服务器背后展示出的设计思想来看，我们认为攻击者非常有经验，并且明显遵循了攻击的“最佳实践”。

45个C2中大部分是各式各样的被黑Word Press站点，主题涵盖美食，医药，运动，色情等。C2服务端肯定不可能直接部署在被黑的站点，因为被暴露的风险太高。其实从严格意义上来说，与其说它们是C2，不如说是C2 Redirector，即它们的作用是将Bot的请求，转发给真实的C2，旨在保护真实的C2免遭识别（为了行文方便，下文依然使用C2称呼这45个域名）。

一开始我们将硬编码C2以及新的C2标定为同一层级Tier3，一段时间之后，我们发现Bot执行3号指令更新C2后，新的C2会响应Bot的"request cmd"请求，接着发下1，2，4号指令；而硬编码的C2只响应上线报文，所有回包都是3号指令。很明显它们的模式完全不同，硬编码C2只是用于引出新增C2，新增C2才符合Tier3的定义，即`Used for general commands, enumeration, scanning, data exfiltration, etc.`

硬编码的C2已固定在恶意样本中，无法更改。若这些C2完全由被黑的网站组成，将存在控制失去的风险。因此，在这9个硬编码的C2中，必然包含一些攻击者自己的服务器。我们通过分析注册时间和是否托管正常页面，认为tartarcusp.com很可能是攻击者的资产。

总结

Wpeeper所使用的加密，验签，C2 Redirector等各种机制无不体现了其作者良好的“专业素养”，甚至它当前诡异的“静默”状态也很有可能是攻击策略的一种，目的是以可信任的身份进入杀软的AI学习样本集。一旦Wpeeper的各种特征被当成正常行为模式被AI学习到，那么此类威胁将会隐藏更久。

本文所述为我们当前掌握的关于Wpeeper的基本情报。我们诚邀具有独特视角的同行企业及受此后门木马影响的网站管理员提供进一步的线索。同时，对我们研究感兴趣的读者也可通过Twitter联系我们获取更多详细信息。

IOC

MD5

```
APK
3dab5a687ab46dbbd80189d727637542
ELF
003577a70748ab4ed18af5aecbd0b529
32e92509bc4a5e3eb2146fe119c45f55
```

Reporter

```
https://eamdomai.com/e?token=Tp5D1nRiu3rFOaCbT4PVcewqIhqbQspd8/3550AI/b1MMJttn+xr4o
```

Downloader

```
https://appflyer.co/downloads/latest/device/android/
https://dn.jnipatch.com/downloads/latest/device/android
```

C2 Redirectors

```
Hardcoded
    https://tartarcusp.com/BZRAWE/
    https://www.chasinglydie.com/7V5QT0/
    https://www.civitize.com/0SA67H/
    https://wyattotero.com/AQVLLY/
    https://web.rtekno.com/5XPOS2/
    https://dermocuidado.com/8QSCZP/
    https://ocalacommercialconstruction.com/WXFHF6/
    https://scatsexo.com/NVZ4L0/
    https://snipsnack.com/T8Q2BN/

New

    https://4devsolutions.com/4NUAK1/
    https://atba3li.com/Z99QQ6/
```

```
    https://avsecretarial.com/PYWDEL/
    https://barbeariadomarfim.com/BN2TTO/
    https://beanblisscafe.com/MX1OAS/
    https://carloadspry.com/SJI4C1/
    https://carshringaraligarh.com/TBHH4O/
    https://coexisthedge.com/ZF57OA/
    https://dibplumber.com/LCN9UJ/
    https://dodgeagonize.com/KJSLOT/
    https://essentialelearning.com/EVSKOT/
    https://focusframephoto.com/1J10V9/
    https://fontshown.com/4D69BN/
    https://gadeonclub.com/Q9DVGH/
    https://hhfus.com/CUGCCO/
    https://kiwisnowman.com/DC4O03/
    https://masterlogisticsfzco.com/5CBSYC/
    https://mrscanology.com/8GVHT3/
    https://naroyaldiamonds.com/WZJ236/
    https://nt-riccotech.com/Q4LQKN/
    https://nutrivital-in.com/7DB9BC/
    https://petintrip.com/QPNQSM/
    https://qualitygoodsforconfectioners.com/3QLS47/
    https://rastellimeeting.com/9Q4GOM/
    https://schatzrestaurant.com/J2WMA6/
    https://socktopiashop.com/4WYZ7I/
    https://speedyrent-sa.com/AIOFB2/
    https://stilesmcgraw.com/1WN2BH/
    https://toubainfo.com/G1ACF0/
    https://trashspringield.com/GYNH3A/
    https://vaticanojoyas.com/R5Q7G4/
    https://wendyllc.com/QD8490/
    https://www.cureoscitystaging.com/YKUCU8/
    https://www.elcomparadorseguros.com/A5FDX7/
    https://www.francescocutrupi.com/WJYP89/
    https://www.yitaichi.com/K7ODU6/
```

Appendix

Python Script

```
from Crypto.Cipher import AES
import json

import hexdump
import base64
rspdata='''
[{"id":34204,
"data":"xdzIhP7WDEGlhUq6OZGGBCmsftzOQqyt7zB2hJe4Dbcxlx0bwTX8jk+YW1R5RtuspkN29hYjaUM
'''
```

```
rsplist=json.loads(rspdata)
for itmp in rsplist:
    raw_data=base64.b64decode(itmp['data'])
    ciphertext=raw_data[96:]
    aeskey=raw_data[64:96]
    aesiv=b"\x00"*16

    aes=AES.new(key=aeskey,iv=aesiv,mode=AES.MODE_CBC)
    plaintext=aes.decrypt(ciphertext)
    hexdump.hexdump(plaintext)
```

CyberChef Recipe

```
https://gchq.github.io/CyberChef/#recipe=JSON_Beautify('%20%20%20%20',false,true)Re
```