Backdoor

Playing Possum: What's the Wpeeper Backdoor Up To?

Alex.Turing, Acey9, heziqian

2024年4月29日 • 11 min read

Summary

On April 18, 2024, `XLab's threat hunting system` detected an ELF file with zero detections on VirusTotal being distributed through two different domains. One of the domains was marked as malicious by three security firms, while the other was recently registered and had no detections, drawing our attention. Upon analysis, we confirmed that this ELF was malware targeting Android systems, utilizing compromised WordPress sites as relay C2 servers, and we named it **Wpeeper**.

Wpeeper is a typical backdoor Trojan for Android systems, supporting functions such as collecting sensitive device information, managing files and directories, uploading and downloading, and executing commands. The most notable feature of Wpeeper lies in its network operations, which reflect the meticulous efforts of its creators in three aspects:

1. Building a multi-level C2 architecture relying on compromised WordPress sites to hide the true C2 server.
2. Using the Session field to differentiate requests, with HTTPS protocol to protect network traffic.
3. Commands issued by C2 are encrypted with AES and accompanied by an elliptic curve signature to prevent takeover.

Wpeeper originated from repackaged applications in the UPtodown Store, where attackers embedded a small code snippet into regular APKs to download and execute the malicious ELF. Due to the minimal amount of added code, the modified APKs currently also show zero detections on VirusTotal. UPtodown is a third-party app store similar to Google Play, with a vast global user base, which is likely why attackers chose it. Due to limited visibility, we do not know if the attackers had other options.

On April 22, Wpeeper suddenly ceased activity; its C2 servers and downloaders stopped providing services. Currently, the relevant samples remain undetected by security firms, effectively deceiving them all. There appears to be no reason for this abrupt halt at this juncture, leading us to suspect that there might be a larger scheme at play.

The Abrupt Halt of the Campaign

We have a relatively comprehensive view of Wpeeper's recent activities.

- April 17: Wpeeper was first uploaded to VirusTotal.
- April 18: The system alerted us, and we began analysis.
- April 19: We started tracking commands and received 36 new C2 servers.
- April 22 at 8:31 AM: We received the last command.

The final command, with function number 12, was to delete itself. Initially, we thought our command tracking had been exposed, but changing tracking IPs proved ineffective; subsequently, the downloader also ceased providing sample downloads. The entire campaign seemed to have been abruptly halted.

Why would the creators of Wpeeper do this? Perhaps they decided to give up, but we consider another possibility: the repackaged APKs served as downloaders for

the Wpeeper backdoor, successfully evading antivirus detection. However, as long as there is network activity, there's a chance of detection. It might be strategically better to voluntarily stop network services, allowing the APKs to maintain their "innocent" status in the eyes of antivirus software, increase their installation numbers, and only then reveal Wpeeper's true capabilities.

Wild-Scale Distribution

We don't have direct data on the scale of Wpeeper's distribution, but based on Google and Passive DNS (PDNS) results, the infection seems to be at the thousand-level, without widespread propagation.

- APK Installation Numbers:
  A search on Google for the MD5 hash of the repackaged UPtodown APK yielded some results: a website named aapks.com offered it for download; another site named Android-apk.org appeared to have statistics, with the count on April 20 being 1,743, and the number of downloads continues to increase.
- Downloader's PDNS:
  The APK includes two downloader domain names, whose resolution counts in the PDNS system can indirectly reflect its wild-scale distribution.

APK Analysis:

Currently, we have captured one repackaged Uptodown application, and its basic information is as follows:

```
Family: Wpeeper Downloader
MD5: 3dab5a687ab46dbbd80189d727637542
Type: Android
```

```
Package Name: com.uptodown
Version: 5.92
```

The attackers added the following code in the APK's MainActivity to create a new thread (MainActivity.F3) for downloading the malicious ELF file.

The MainActivity.F3 function eventually calls the MainActivity.h5 function. The code is not obfuscated, making its functionality clear: it downloads an ELF file named "android" from two domains, renames it to "com.uptodownload.libs," and finally initiates its execution.

ELF Analysis

The malicious Wpeeper samples downloaded from the two domains are identical, and their basic information is as follows:

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

Packer: None
```

Wpeeper does not employ anti-analysis techniques, making its reverse engineering relatively straightforward. Its functionality is quite simple. In summary, when it runs on a victim's device, it checks for the presence of a file named `store.lock` in the same directory, which contains configuration information including C2 servers, Cookie, Interval, etc. If the `store.lock` file exists, it uses AES CBC mode to decrypt the file to obtain the C2s. It then constructs a POST request using libcurl to establish communication with C2 server and awaits commands. If the `store.lock` file does not exist, it decodes the C2 servers embedded in the sample using Base64, encrypts them, and saves them to the `store.lock` file before proceeding to establish communication and execute

commands in the same manner. The following discussion will delve into these aspects to dissect the functionality of Wpeeper.

Obtaining C2

Wpeeper uses an array known as `c2_list` to store the current C2 servers, supporting up to 30 entries, initially empty. Its primary task upon execution is to populate this `c2_list`, which can be filled from two sources:

Source 1: Hardcoding

The bot decodes the embedded C2 within the sample to populate the `c2_list`. The embedded C2 is base64 encoded and decodes to a total of nine entries (see IOC C2 Hardcoded for details).

Source 2: store.lock

The bot reads and decrypts the `store.lock` file to fill the `c2_list`.

The first 32 bytes of `store.lock` are the AES key, with the remainder being ciphertext. For example, considering a `store.lock` generated after Wpeeper has run on a test device for some time, the decryption is as follows:

In addition to C2, the PlainText contains other information such as id, cookie, and interval. The flag is a fixed value of 0x000004, interval controls the sleep duration, and cookie and id are used to construct the HTTP header Cookie field.

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

Besides these two filling methods, `c2_list` also supports runtime updates. When Wpeeper receives a command type 3, it updates the `c2_list` using the payload delivered by the C2.

Network Communication

Wpeeper employs the libcurl library to construct POST requests for interacting with the C2 server. Within the header, the Cookie and Session fields are particularly critical as they are used to identify different types of requests.

Consider the example of Bot communication with the server via `http[s]://snipsnack.com/T8Q2BN/`:

BOT ---> C2

The value in the Cookie field, after base64 decoding, is "nrjzekrxohxw0", where "nrjzekrxohxw" is a randomly generated lowercase string of length 12, and "0" is the initial id. The Session, decoded, is `{"auth": 1, "type": "search"}`, a combination of id and session, indicating that this is the Bot's first network join, and the request is a beacon.

Wpeeper's server differentiates between types of requests precisely through the Session field.

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

The C2 responds with a JSON object containing two fields: id and data. The id is used to generate a Cookie in conjunction with a random string, while the data contains commands issued by the C2.

When the Bot receives the data, it first decodes it using base64 to obtain the raw_data. The first 64 bytes of the raw_data are the digital signature, followed by 32 bytes which are the AES key. The remaining part is the ciphertext.

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

The Bot proceeds with AES decryption and executes the corresponding commands only after the digital signature verification is successful.

Wpeeper uses compromised WordPress sites as relay C2s, effectively concealing the true C2 server but also introducing reliability issues, as the relay C2s are not under direct control of the attackers.
Suppose an administrator of a compromised site (relay C2) discovers traces of Wpeeper, they could potentially issue commands. If the Bot does not validate the "legitimacy" of commands, the entire network could be easily compromised. Wpeeper's **signature verification mechanism** eliminates this possibility, ensuring the network remains secure from **poisoning, takeover, or destruction**.

Upon successful decryption of the ciphertext using AES CBC mode, `cmd_context` is obtained, structured as follows:

```
struct cmd_context
{
uint32 id;
```

```
uint32 cmd;
uint32 payload_len;
uint8[cmd_len] payload;
uint32 randstr_len;
uint8[randstr_len] cookie;
}
```

Finally, the Bot executes various commands based on the `cmd` value in `cmd_context` and returns the results to the C2.

Currently, Wpeeper supports a total of 13 commands, listed below with their numbers and corresponding functionalities.

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	self-destruction
13	download from arbitrary URL, then exec

We have implemented Python and CyberChef scripts in the Appendix for decrypting raw data (readers can directly copy the CyberChef script from the Appendix into their browser, which intuitively displays the decryption process). Taking the above message from C2 as an example, the decrypted plaintext reveals a `cmd` value of 3. When the Bot receives this message, it executes command 3 to update the `c2_list` with the domain names found in the payload.

Command Tracking

Previous reverse engineering of Wpeeper answered the question of **what it can do**, but we were still in the dark about **what it was actually doing**. To address this, we implemented the Wpeeper network protocol within the `XLab Command Tracking System`, which soon provided some insights.

Although the volume of commands for Wpeeper is not extensive, deeper analysis revealed some interesting findings. Statistical data shows that Wpeeper's commands are heavily concentrated in commands 1, 2, 3, and 4, indicating that Wpeeper is actively collecting sensitive information from victim devices, updating C2 servers, among others. Additionally, command 13 allowed us to capture a new ELF file, **AppInstallerEx**, which also functions to gather device information.

Insight 1: New C2

By sending beacon requests to the nine hardcoded C2 servers within the Wpeeper sample, we received 3500 instances of command 3, capturing 3500 sets of new C2 servers, resulting in 36 unique entries after deduplication (details seen in IOC C2 NEW).

Insight 2: AppInstallerEx

On April 21, we received command 13, which instructed the download and execution of a program named AppInstallerEx.

AppInstallerEx functions to collect device information and report it to `eamdomai.com`.

C2 Role Analysis

In this operation, Wpeeper utilized 45 C2 servers. The design behind these servers suggests that the attackers are highly experienced and clearly follow the "best practices" of cyber attacks.

Most of the 45 C2s are various compromised WordPress sites covering themes like cuisine, medicine, sports, and adult content. It's highly unlikely that the C2 servers are directly deployed on these compromised sites due to the high risk of exposure. More accurately, these are not C2s but C2 Redirectors— their role is to forward the Bot's requests to the real C2, aimed at shielding the actual C2 from detection (for ease of discussion, we will still refer to these 45 domains as C2s).

Initially, we categorized both the hardcoded and new C2s as the same tier (Tier3). Over time, we observed that after executing command 3 to update C2s, the new C2s would respond to the Bot's "request cmd" with commands 1, 2, and 4, while the hardcoded C2s only responded to initial check-in messages with command 3. Their patterns are distinctly different; the hardcoded C2s merely serve to lead to the new C2s, which truly fit the Tier3 definition: Used for general commands, enumeration, scanning, data exfiltration, etc.

Since the hardcoded C2s are fixed in the malicious samples and cannot be changed, there's a risk of losing control if these C2s are solely compromised websites. Thus, among these nine hardcoded C2s, some are likely servers owned by the attackers themselves. By analyzing the registration dates and whether they host legitimate pages, we believe that tartarcusp.com is likely an asset of the attackers.

Conclusion

The encryption, signature verification, C2 Redirectors, and other mechanisms employed by Wpeeper all reflect the creators' professional proficiency. Even its current mysterious "silence" could likely be part of their attack strategy, aiming to enter the AI learning sample set of antivirus software as a trusted entity. Once Wpeeper's characteristics are learned by AI as normal behavior, such threats could remain hidden for longer.

This article represents the basic intelligence we currently hold on Wpeeper. We invite peers with unique perspectives and administrators of websites affected by this backdoor Trojan to provide further clues. Interested readers can also contact us through Twitter for more detailed information.

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
```

What do you think?

5 Responses

 Upvote

 Funny

 Love

 Surprised

 Angry

 Sad

0 Comments Login ▾

G

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS

Name

♡ Share

Best Newest Oldest