Dissecting REMCOS RAT: An in- depth analysis of a widespread 2024 malware, Part Four

elastic.co/security-labs/dissecting-remcos-rat-part-four

Subscribe

Detections, hunts using ES|QL, and conclusion

In previous articles in this multipart series [1] [2] [3], malware researchers on the Elastic Security Labs team decomposed the REMCOS configuration structure and gave details about its C2 commands. In this final part, you'll learn more about detecting and hunting REMCOS using Elastic technologies.

Detection and Hunt

The following Elastic Defend detections trigger on those techniques:

Persistence (Run key)

> Startup Persistence by a Low Reputation Process

Process Injection

Privilege Escalation (UAC Bypass)

> UAC Bypass via ICMLuaUtil Elevated COM Interface

Evasion (Disable UAC)

> Disabling User Account Control via Registry Modification (REMCOS spawns cmd.exe that uses reg.exe to disable UAC via registry modification)

Command and Control

Connection to Dynamic DNS Provider by an Unsigned Binary (although it's not a requirement but most of the observed samples use dynamic DNS)

File Deletion

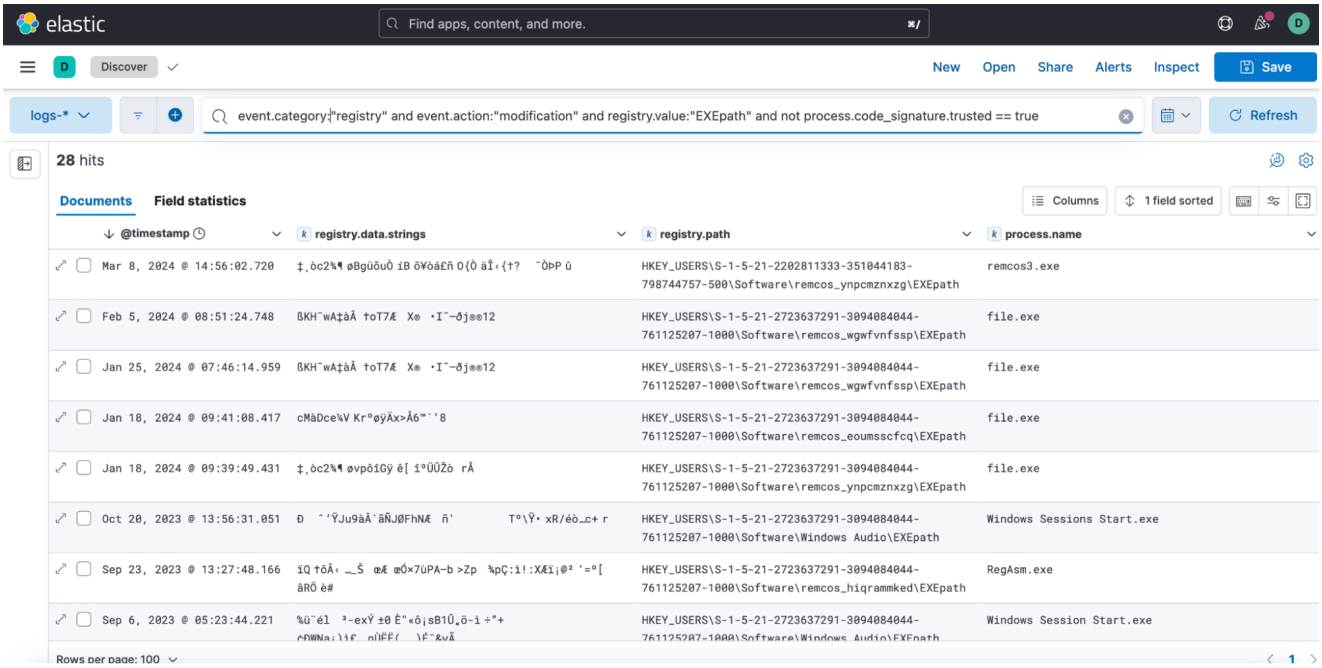
Remcos RAT INETCookies File Deletion

Modify Registry

Remcos RAT ExePath Registry Modification

The ExePath registry value used by the REMCOS watchdog process can be used as an indicator of compromise. Below is a KQL query example :

```
event.category:"registry" and event.action:"modification" and
registry.value:"EXEpath" and not process.code_signature.trusted:true
```

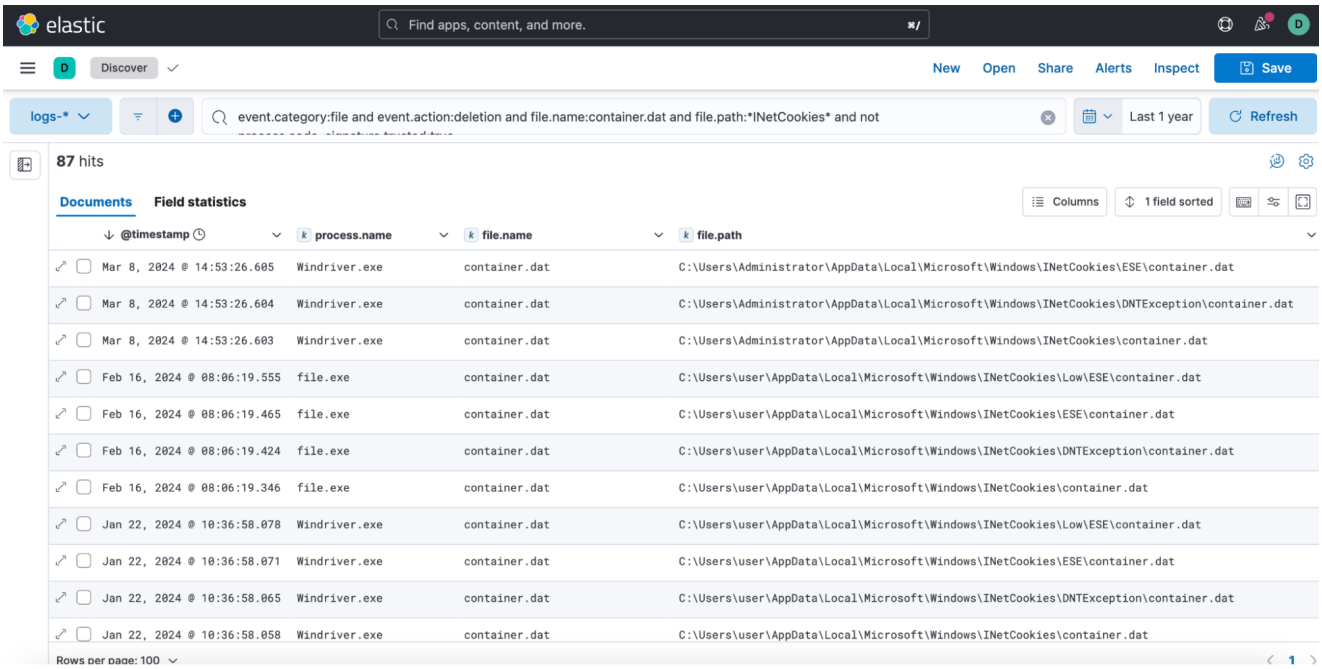


REMCOS includes three options for clearing browser data, possibly in an attempt to force victim users to re-enter their web credentials for keylogging:

- `enable_browser_cleaning_on_startup_flag`
- `enable_browser_cleaning_only_for_the_first_run_flag`
- `browser_cleaning_sleep_time_in_minutes`

This results in the deletion of browser cookies and history-related files. The following KQL query can be used to hunt for such behavior by an unsigned process:

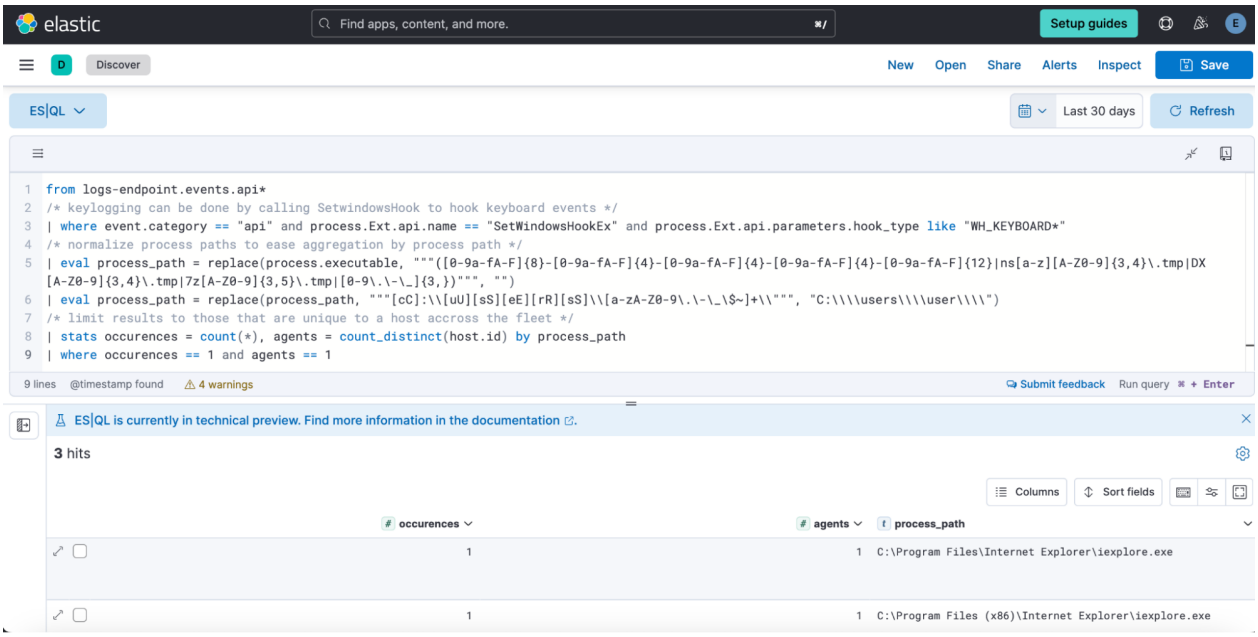
```
event.category:file and event.action:deletion and file.name:container.dat and
file.path:*INetCookies* and not process.code_signature.trusted:true
```



REMCOS also employs three main information collection methods. The first one is keylogging via SetWindowsHookEx API. The following ES|QL can be used to hunt for rare or unusual processes performing this behavior:

```
from logs-endpoint.events.api*

/* keylogging can be done by calling SetwindowsHook to hook keyboard events */

| where event.category == "api" and process.Ext.api.name == "SetWindowsHookEx" and
process.Ext.api.parameters.hook_type like "WH_KEYBOARD*"

/* normalize process paths to ease aggregation by process path */

| eval process_path = replace(process.executable, """([0-9a-fA-F]{8}-[0-9a-fA-F]{4}-
[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}|ns[a-z][A-Z0-9]{3,4}\.tmp|DX[A-Z0-9]
{3,4}\.tmp|7z[A-Z0-9]{3,5}\.tmp|[0-9\.\-\_]{3,})""", "")
| eval process_path = replace(process_path, """[cC]:\\[uU][sS][eE][rR][sS]\\[a-zA-Z0-
9\.\-\_\$~]+\\""", "C:\\\\users\\\\user\\\\")

/* limit results to those that are unique to a host across the agents fleet */

| stats occurrences = count(*), agents = count_distinct(host.id) by process_path
| where occurrences == 1 and agents == 1
```

Below is an example of matches on `iexplore.exe` (injected by REMCOS):

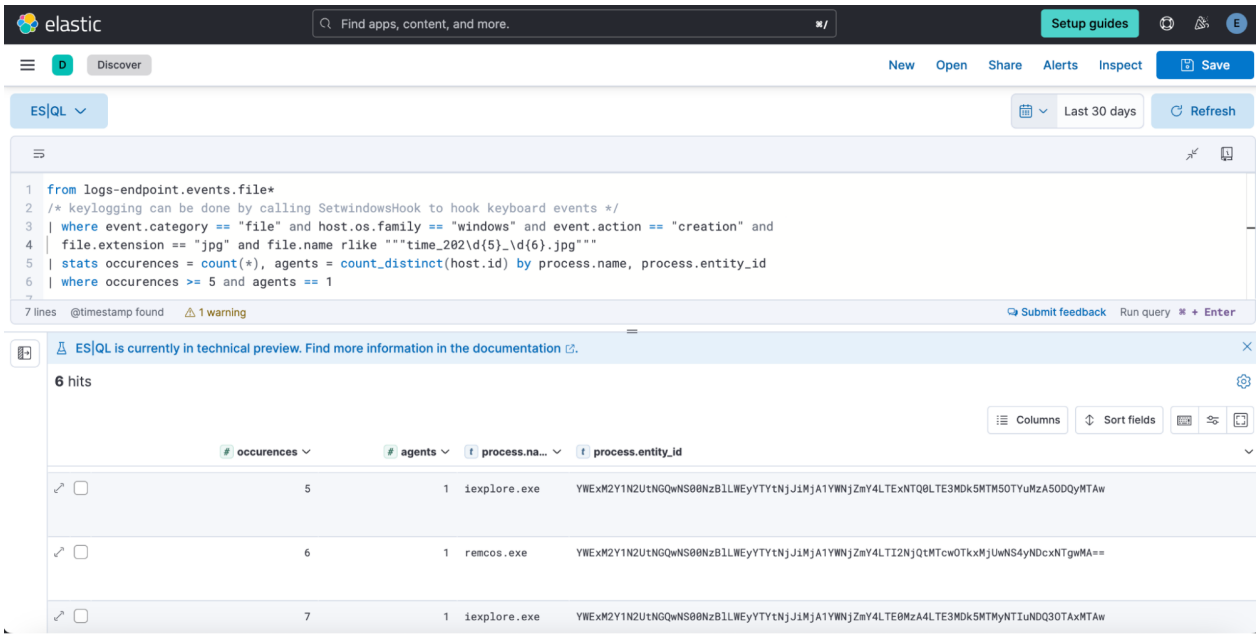


ES|QL hunt for rare processes calling SetWindowsHoook to hook keyboard events

The second method takes multiple screenshots and saves them as jpg files with a specific naming pattern starting with `time_year-month-day_hour-min-sec.jpb` (e.g. `time_20240308_171037.jpg`). The following ES|QL hunt can be used to identify suspicious processes with similar behavior :

```
from logs-endpoint.events.file*

/* remcos screenshots naming pattern */

| where event.category == "file" and host.os.family == "windows" and event.action == "creation" and file.extension == "jpg" and file.name rlike """time_202\d{5}_\d{6}.jpg"""
| stats occurrences = count(*), agents = count_distinct(host.id) by process.name, process.entity_id

 /* number of screenshots i more than 5 by same process.pid and this behavior is limited to a unique host/process */

| where occurrences >= 5 and agents == 1
```

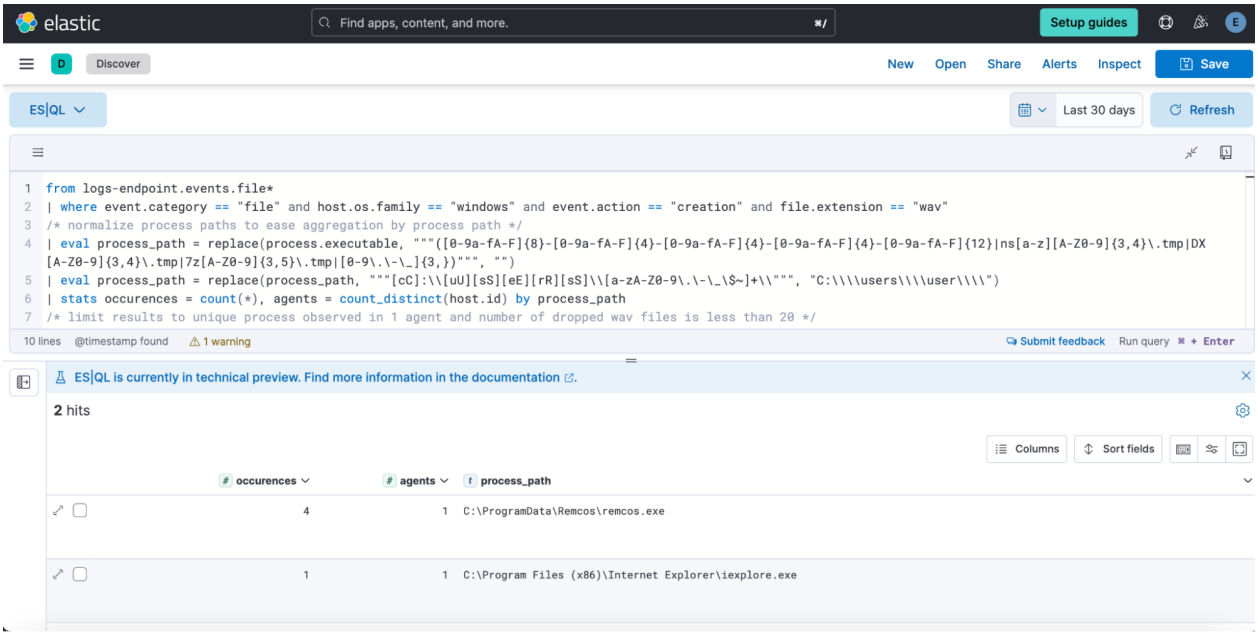
The following image shows both REMCOS and the injected iexplore.exe instance (further investigation can be done by pivoting by the process.entity_id):



ES|QL hunt for rare processes creating JPG files similar to REMCOS behavior

The third collection method is an audio recording saved as WAV files. The following ES|QL hunt can be used to find rare processes dropping WAV files:

```
from logs-endpoint.events.file*
| where event.category == "file" and host.os.family == "windows" and event.action ==
"creation" and file.extension == "wav"

/* normalize process paths to ease aggregation by process path */

| eval process_path = replace(process.executable, """([0-9a-fA-F]{8}-[0-9a-fA-F]{4}-
[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}|ns[a-z][A-Z0-9]{3,4}\.tmp|DX[A-Z0-9]
{3,4}\.tmp|7z[A-Z0-9]{3,5}\.tmp|[0-9\.\-\_]{3,})""", "")
| eval process_path = replace(process_path, """[cC]:\\[uU][sS][eE][rR][sS]\\[a-zA-Z0-
9\.\-\_\$~]+\\""", "C:\\\\users\\\\user\\\\")
| stats wav_files_count = count(*), agents = count_distinct(host.id) by process_path

/* limit results to unique process observed in 1 agent and number of dropped wav
files is less than 20 */

| where agents == 1 and wav_files_count <= 10
```

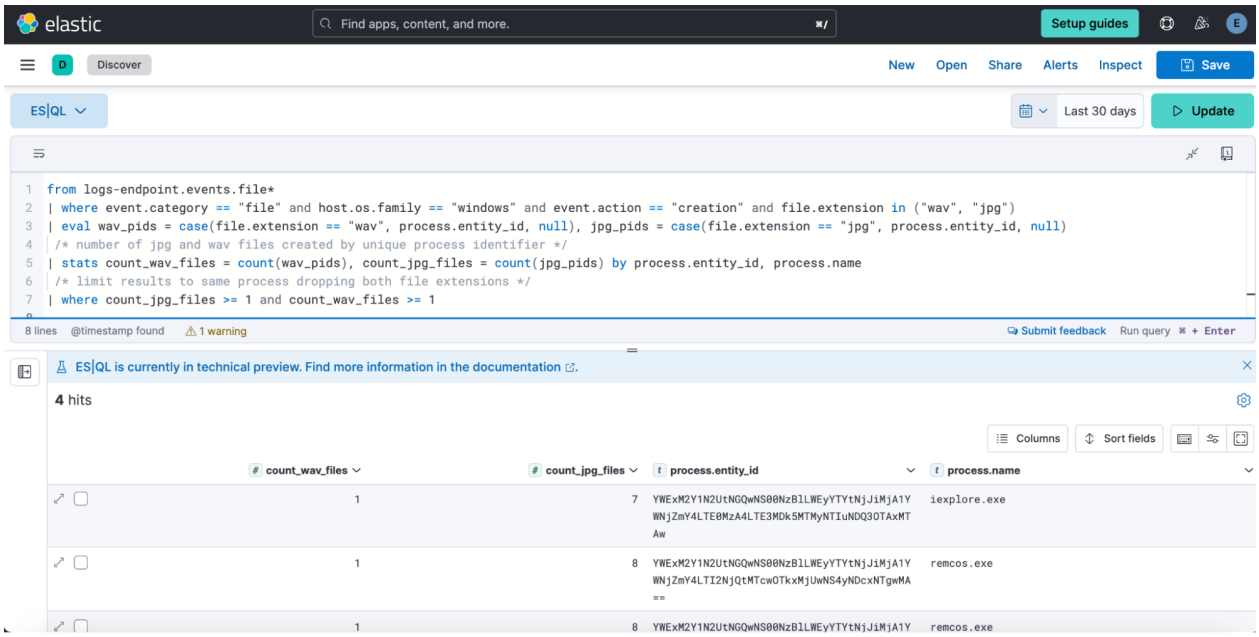


ES|QL hunt for rare processes creating WAV files

The following ES|QL hunt can also look for processes that drop both JPG and WAV files using the same `process.pid` :

```
from logs-endpoint.events.file*
| where event.category == "file" and host.os.family == "windows" and event.action ==
"creation" and file.extension in ("wav", "jpg") and

/* excluding privileged processes and limiting the hunt to unsigned
process or signed by untrusted certificate or signed by Microsoft */

not user.id in ("S-1-5-18", "S-1-5-19", "S-1-5-20") and
(process.code_signature.trusted == false or process.code_signature.exists == false or
starts_with(process.code_signature.subject_name, "Microsoft"))
| eval wav_pids = case(file.extension == "wav", process.entity_id, null), jpg_pids =
case(file.extension == "jpg", process.entity_id, null), others = case(file.extension
!= "wav" and file.extension != "jpg", process.entity_id, null)

/* number of jpg and wav files created by unique process identifier */

| stats count_wav_files = count(wav_pids), count_jpg_files = count(jpg_pids),
other_files = count(others) by process.entity_id, process.name

/* limit results to same process dropping both file extensions */

| where count_jpg_files >= 1 and count_wav_files >= 1
```

Examples of matches on both REMCOS and the injected `iexplore.exe` process:



ES|QL hunts for unique processes dropping image and audio files

Pivoting by process.entity_id to further investigate suspicious processes, installers, browsers, and decompression utilities are often the most observed false positives.

YARA rule

The REMCOS version 4.9.3 is detected statically using the following YARA rule produced by Elastic Security Labs

Malware and MITRE ATT&CK

Elastic uses the MITRE ATT&CK framework to document common tactics, techniques, and procedures that advanced persistent threats use against enterprise networks.

Tactics

Tactics represent the *why* of a technique or sub-technique. It is the adversary's tactical goal: the reason for performing an action.

- Execution
- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Command and Control

Techniques

Techniques represent how an adversary achieves a tactical goal by performing an action.

- Windows Command Shell
- Visual Basic
- Registry Run Keys / Startup Folder
- Process Injection
- Credentials from Web Browsers
- Encrypted Channel
- System Binary Proxy Execution: CMSTP
- Bypass User Account Control

Conclusion

As the REMCOS continues to rapidly evolve, our in-depth analysis of version 4.9.3 offers critical insights that can significantly aid the malware research community in comprehending and combatting this pervasive threat.

By uncovering its features and capabilities in this series, we provide essential information that enhances understanding and strengthens defenses against this malicious software.

We've also shown that our Elastic Defend product can detect and stop the REMCOS threat. As this article demonstrates, our new query language, ES|QL, makes hunting for threats simple and effective.

Elastic Security Labs remains committed to this endeavor as part of our open-source philosophy, which is dedicated to sharing knowledge and collaborating with the broader cybersecurity community. Moving forward, we will persist in analyzing similar malware families, contributing valuable insights to bolster collective defense against emerging cyber threats.

Sample hashes and C2s

(Analysis reference)
0af76f2897158bf752b5ee258053215a6de198e8910458c02282c2d4d284add5

remchukwugixiemu4.duckdns[.]org:57844

remchukwugixiemu4.duckdns[.]org:57846

remchukwugix231fgh.duckdns[.]org:57844

remchukwugix231fgh.duckdns[.]org:57846

3e32447ea3b5f07c7f6a180269f5443378acb32c5d0e0bf01a5e39264f691587

122.176.133[.]66:2404

122.176.133[.]66:2667

8c9202885700b55d73f2a76fbf96c1b8590d28b061efbadf9826cdd0e51b9f26

43.230.202[.]33:7056

95dfdb588c7018babd55642c48f6bed1c281cecccbd522dd40b8bea663686f30

107.175.229[.]139:8087

517f65402d3cf185037b858a5cfe274ca30090550caa39e7a3b75be24e18e179

money001.duckdns[.]org:9596

b1a149e11e9c85dd70056d62b98b369f0776e11b1983aed28c78c7d5189cfdbf

104.250.180[.]178:7902

ba6ee802d60277f655b3c8d0215a2abd73d901a34e3c97741bc377199e3a8670

185.70.104[.]90:2404

185.70.104[.]90:8080

185.70.104[.]90:465

185.70.104[.]90:80

77.105.132[.]70:80

77.105.132[.]70:8080

77.105.132[.]70:2404

77.105.132[.]70:465

Research references