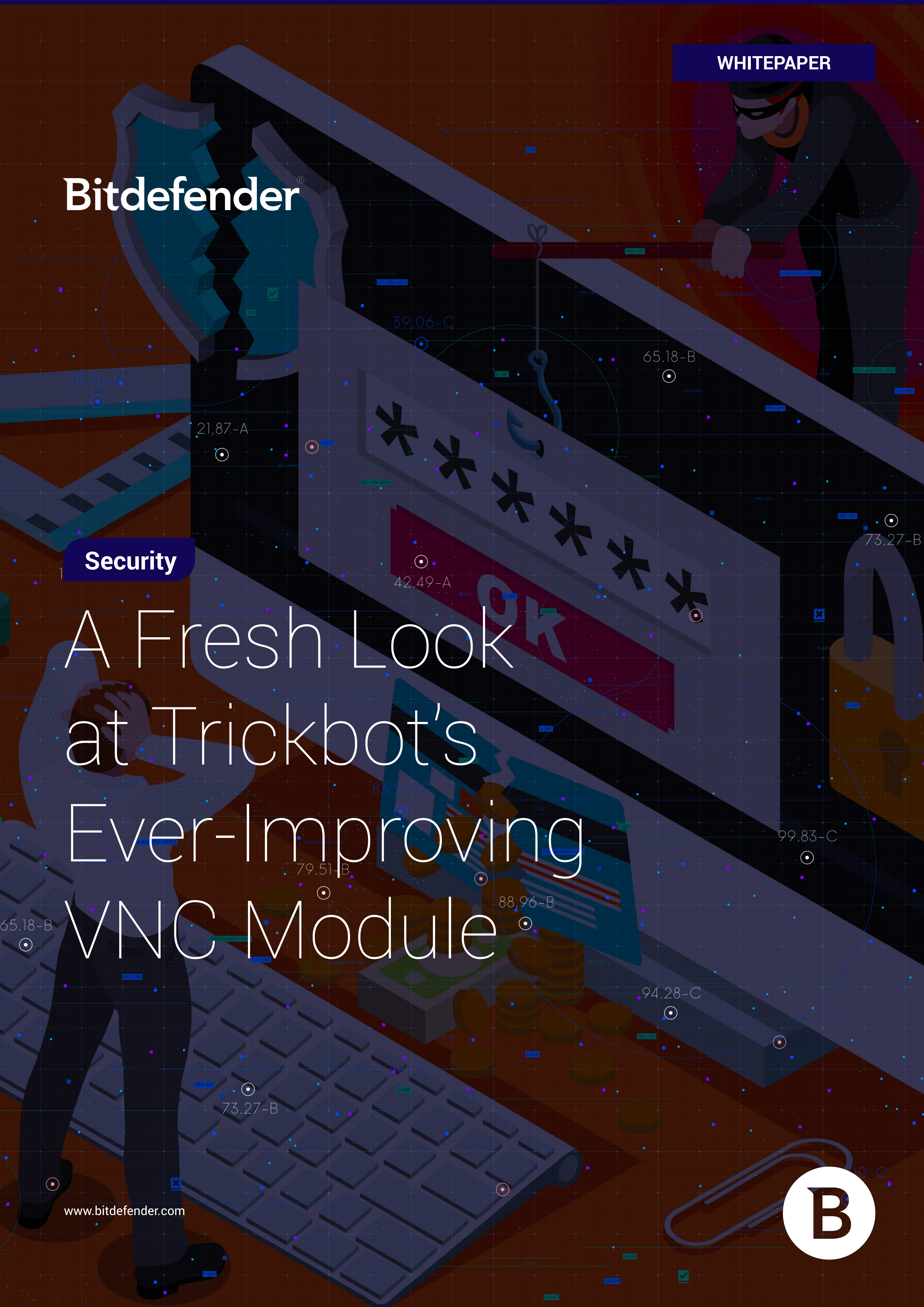
WHITEPAPER

Bitdefender®

Security

A Fresh Look at Trickbot's Ever-Improving VNC Module



www.bitdefender.com

Contents

Foreword ... 3

Key findings ... 3

A new update on the horizon ... 3

- Setup part ... 4
- Normal operation mode ... 5

Command and Control Infrastructure ... 11

Indicators of compromise ... 12

Authors:
Radu TUDORICA - Security Researcher

Foreword

The journey of Trickbot starts almost half a decade ago, when it appeared in the form of a banker and credential-stealing application. Drawing inspiration from Dyre (or Dyreza), Trickbot consists of an ecosystem of plugin modulesand helper components. As of late, the Trickbot group, which has managed to infect millions of computers worldwide, has played an active role in disseminating ransomware.

While most of the Dyre creators have gone dark, one person was recently charged for her role in this transnational cybercrime ring operating out of Russia, Belarus, Ukraine and Suriname. However, despite the indictment and the law enforcement takedown attempts, Trickbot shows no sign of slowing down.

Bitdefender researchers have been reporting on notable developments in Trickbot's lifecycle, with highlights including the analysis of one of its modules in late 2020 used to bruteforce RDP connections and the analysis of its new C2 infrastructure in the wake of the crackdown of its infrastructure.

This new research focuses on an updated VNC module, which includes new functionalities for monitoring and intelligence gathering.

Key findings

- Bitdefender researchers have discovered an updated VNC module that seems to be in active development, as its maintainers are updating it at a very fast pace;
- This module is now delivered under a new name; our observations also helped us map the attackers' network architecture
- Bitdefender researchers have identified he software application that the attackers use to connect to victims' computers. This tool, called VNCView, is described in a dedicated chapter.

A new update on the horizon

As of May 12, 2021, our monitoring systems started to pick up an updated version of the **vncDll** module used by Trickbot against select high-profile targets. This module is known as **tvncDll** and is used for monitoring and intelligence gathering**.** It seems to be still under development, since the group has a frequent update schedule, regularly adding new functionalities and bug fixes.

Our analysis focuses on identifying the communication protocol and the infrastructure behind it in correlation with the module's new functionalities.

During our investigation we also stumbled on an additional tool used by the Trickbot group to facilitate the access of other threat actors to the victims' computers.

Communication Protocol

This module, **vncDll / tvncDll,** uses a custom communication protocol, which only makes it harder to understand what data is being transmitted without prior knowledge. The module communicates with the C2 servers defined in its configuration file called *vncconf*and which includes a list of up to nine IP addresses. These servers act as mediators between the victims and attackers, one of their roles being to enable access to victims behind firewalls.

The configuration file is sent to the module using the *Control* function, along with a command. In this case, there are two accepted commands, but only one is fully implemented. The commands are *SetConf* and *Listen.* While the latter is unimplemented, its existence suggests that the developers of the module were planning to implement a server component for victims who aren't behind firewalls. As its name implies, *SetConf* sets the configuration for the module, adding the IP addresses to the list of C2 servers. The module will use the first one it can connect to.

The port used to communicate with the servers is 443 to avoid arousing the suspicion of anyone observing the traffic. Although traffic on this port normally uses SSL or TLS, the data is sent unencrypted.

The communication protocol has two parts: the setup part and the normal operation mode.

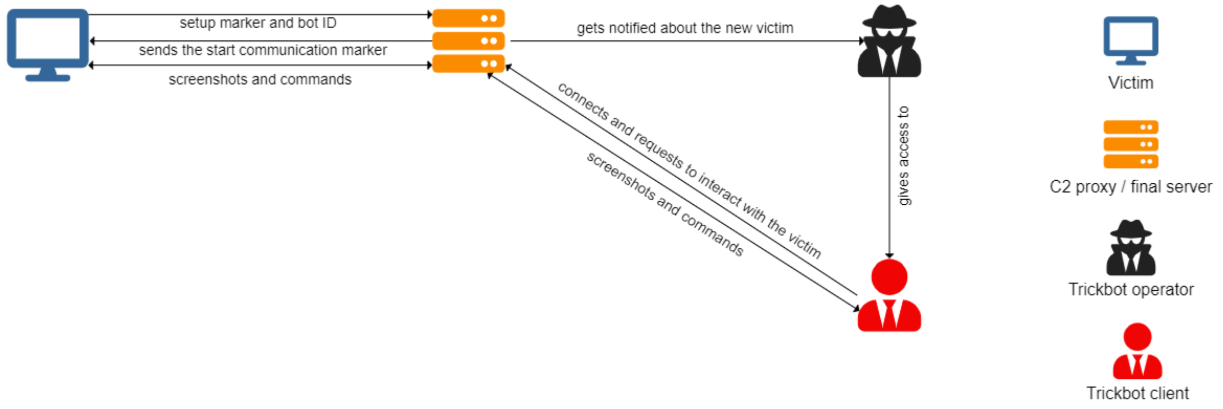


Figure 1: Attack overview

Setup part

The role of the setup part is to announce the C2 server of its existence and receive a set of commands while waiting for an attacker to connect to one of the servers and ask to be put in contact with one of the victims.

The setup part works as follows:

The client sends the marker "\x03\x02\x01" followed by a structure containing the *bot id* that uniquely identifies the victim and that is generated by Trickbot malware. It then waits for the server to reply with one of at least three possible commands, described as follows:

1. If the module receives **TS5T,** it will echo it back to the C2 server and wait for another command. This is used as a keep alive message while no attacker is requesting access to the victim.
2. If the module receives **LliK,** it will stop itself and ask Trickbot to unload it from memory.
3. If any other message is received, it will enter the normal operation mode, meaning that an attacker requested to communicate with the module and the C2 server is now just relaying messages between the two. It is also the moment when the module will create a new desktop that is fully controlled by the module and contains a custom interface.

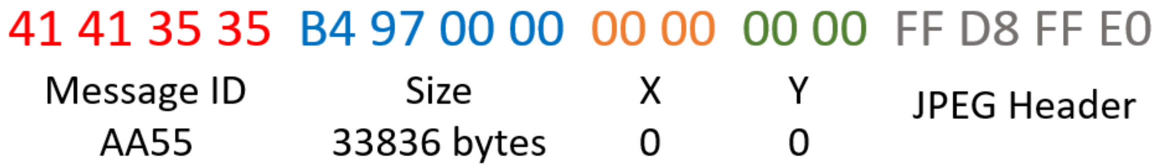
Normal operation mode

In the normal operation mode, the messages are exchanged as follows:

The communication is full duplex, the victim can send screenshots of the alternative desktop and the clipboard data, the server sends window messages that get decoded and processed.

The screenshot messages have the following format:

Message id (0x35354141 or "AA55"), followed by the size of the screenshot (4 bytes) and the X and Y coordinates of the top left corner (2 bytes for each) and followed by the screenshot. The screenshot is taken in the bitmap (bmp) format by the malware, but for network transportation it is converted to either jpeg or png, depending on its size.
If the image is under 40000 pixels it will be sent as png. Notably, the malware isn't sending full screenshots at a regular interval, but is instead sending a full screenshot at first, then it's sending only screenshots of the parts of the alternative desktop that changed, hence the use of the X and Y coordinates.

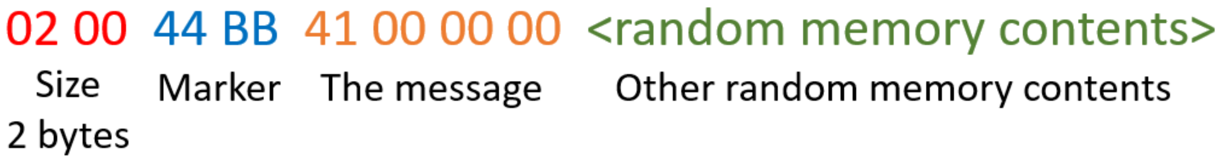


41 41 35 35	B4 97 00 00	00 00	00 00	FF D8 FF E0
Message ID AA55	Size 33836 bytes	X 0	Y 0	JPEG Header

Figure 2: First 16 bytes of a screenshot message

The clipboard messages have the following format:

> The size of the clipboard plus one encoded as a WORD, followed by the marker "\x44\xBB" and the clipboard encoded as wide characters truncated to a maximum of 512 (so, a max of 1024 1 byte characters).



02 00	44 BB	41 00 00 00	<random memory contents>
Size 2 bytes	Marker	The message	Other random memory contents

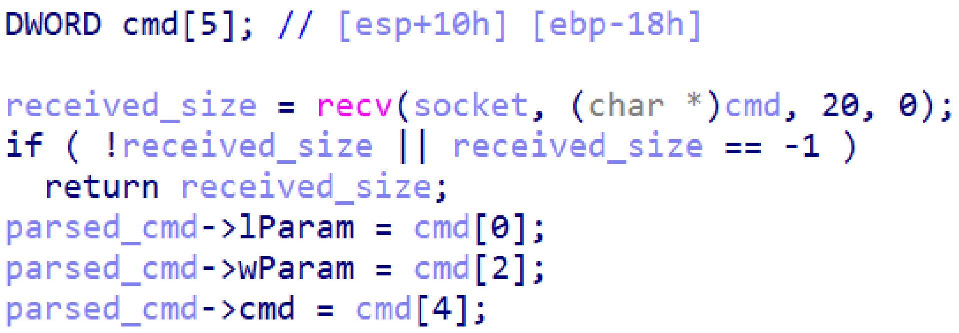
Figure 3: Example clipboard message where the clipboard would contain only "A"

One more thing to mention is that the module always sends 1024 bytes after the header, leading to out of bounds memory reads and the leaking of memory contents from the victim's computer.

There is one other message that can be sent (0xBB440001), that shows the module wasn't able to create the alternative desktop, after which it closes the connection.

The server sends its commands as follows:

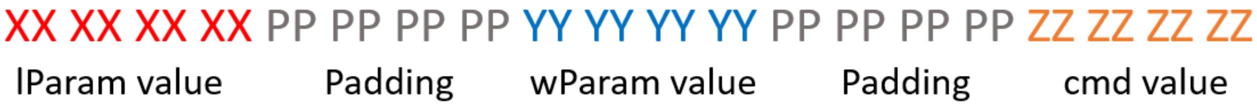
lParam (4 bytes), 4 unused / padding bytes, wParam (4 bytes), another 4 unused / padding bytes and then the message (4 bytes) and is decoded in the following way.



```
DWORD cmd[5]; // [esp+10h] [ebp-18h]

received_size = recv(socket, (char *)cmd, 20, 0);
if ( !received_size || received_size == -1 )
  return received_size;
parsed_cmd->lParam = cmd[0];
parsed_cmd->wParam = cmd[2];
parsed_cmd->cmd = cmd[4];
```

Figure 4:The decompiled code that does the decoding of the message



XX XX XX XX	PP PP PP PP	YY YY YY YY	PP PP PP PP	ZZ ZZ ZZ ZZ
lParam value	Padding	wParam value	Padding	cmd value

Figure 5:Message format

The window messages are processed as expected, simulating mouse clicks or key presses on the virtual desktop that was created. Besides the usual messages, the module processes a few control messages, probably coming from the attacker's interface, such as forcing the module to resend a screenshot, or gracefully exiting the session.

Using this knowledge, a client can be implemented to test its functionality.

During normal operation, the alternate desktop is created and fully controlled by the module, copying the icons from the desktop, creating a custom taskbar for managing its processes and creating a custom right click menu, containing custom functionality.

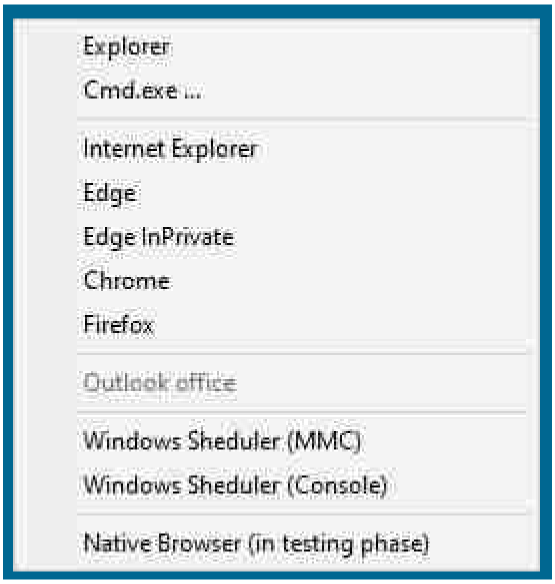
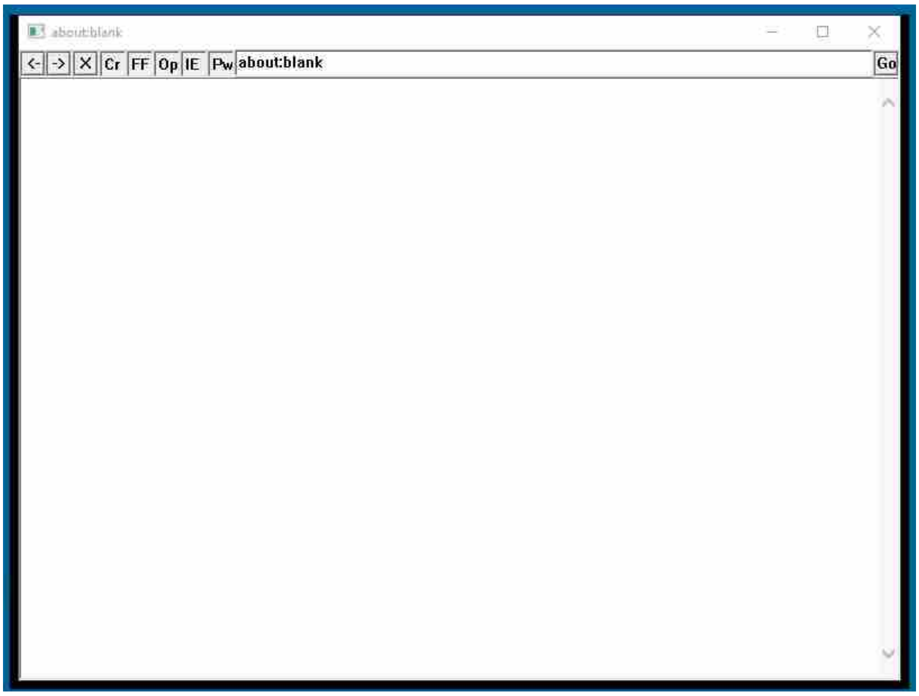


Figure 6: The menu that appears if a right click command is sent.

Most of the options just open programs from the machine, but through **Cmd.exe** the threat actors can perform several high-impact actions leveraging PowerShell, such as:

- download new payloads to further propagate the attack inside the network;
- open different documents or the email inbox;
- upload data from victims' computers to the command-and-control servers.

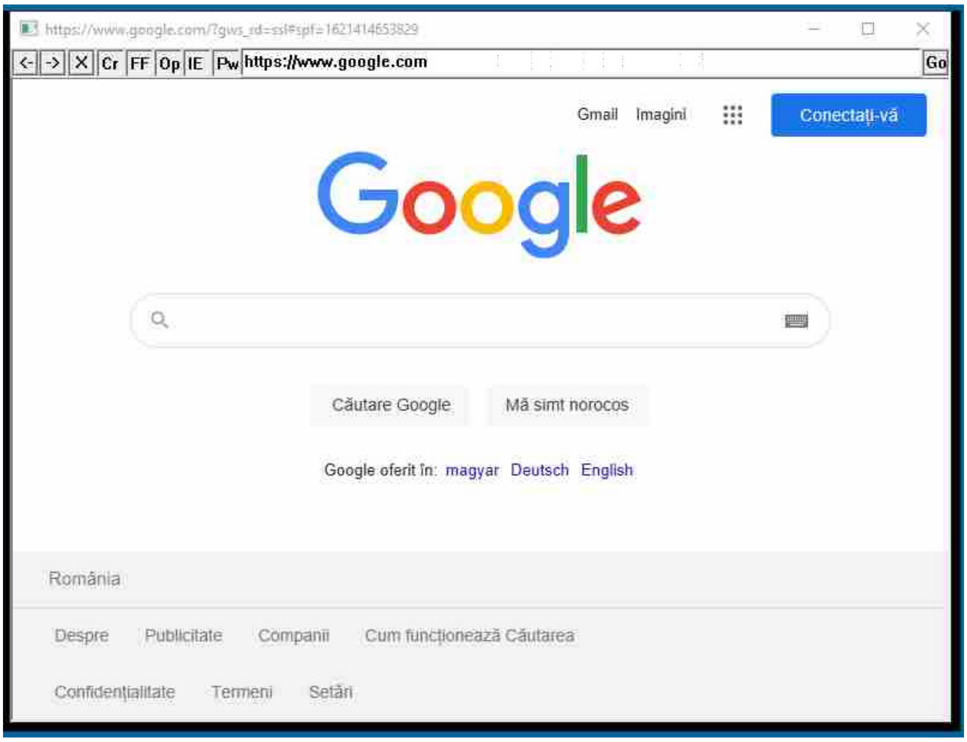
The last option (Native Browser) adds a password dumping functionality and is in active development, with multiple weekly updates.



about:blank

Figure 7: The window that appears if a left click on the "Native Browser" button is sent

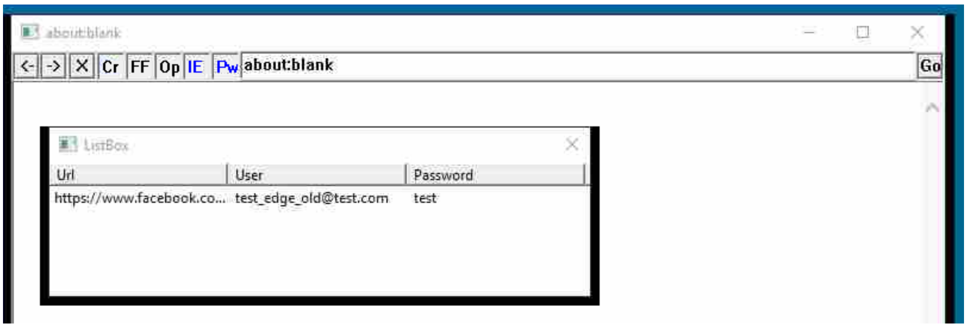
By default, it creates its own browser using the OLE automation feature for Internet Explorer.



https://www.google.com

Figure 8: Working browser window created using OLE automation

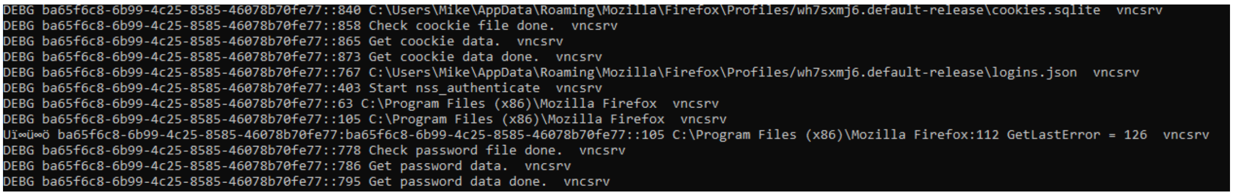
The buttons on the left of the navigation bar are supposed to be used for password dumping, and should work for Chrome, Firefox, Opera and Internet Explorer, but this functionality properly works for Internet Explorer only.



Url	User	Password
https://www.facebook.co...	test_edge_old@test.com	test

Figure 9: The password dumping panel triggered by selecting IE and Pw in the native browser panel

In the last update that we know for this module, the password dumping for Firefox was in the works, but for now, it doesn't seem to function as expected.



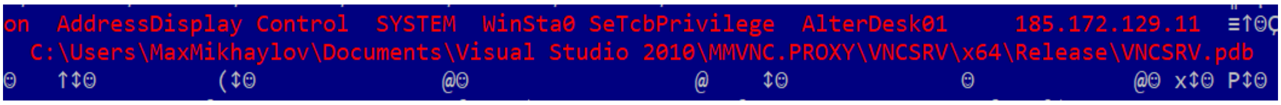
```
DEBG ba65f6c8-6b99-4c25-8585-46078b70fe77::840 C:\Users\Mike\AppData\Roaming\Mozilla\Firefox\Profiles/wh7sxmj6.default-release\cookies.sqlite  vncsrv
DEBG ba65f6c8-6b99-4c25-8585-46078b70fe77::858 Check coockie file done.  vncsrv
DEBG ba65f6c8-6b99-4c25-8585-46078b70fe77::865 Get coockie data.  vncsrv
DEBG ba65f6c8-6b99-4c25-8585-46078b70fe77::873 Get coockie data done.  vncsrv
DEBG ba65f6c8-6b99-4c25-8585-46078b70fe77::767 C:\Users\Mike\AppData\Roaming\Mozilla\Firefox\Profiles/wh7sxmj6.default-release\logins.json  vncsrv
DEBG ba65f6c8-6b99-4c25-8585-46078b70fe77::403 Start nss_authenticate  vncsrv
DEBG ba65f6c8-6b99-4c25-8585-46078b70fe77::63 C:\Program Files (x86)\Mozilla Firefox  vncsrv
DEBG ba65f6c8-6b99-4c25-8585-46078b70fe77::105 C:\Program Files (x86)\Mozilla Firefox  vncsrv
Uï∞ü∞ö ba65f6c8-6b99-4c25-8585-46078b70fe77:ba65f6c8-6b99-4c25-8585-46078b70fe77::105 C:\Program Files (x86)\Mozilla Firefox:112 GetLastError = 126  vncsrv
DEBG ba65f6c8-6b99-4c25-8585-46078b70fe77::778 Check password file done.  vncsrv
DEBG ba65f6c8-6b99-4c25-8585-46078b70fe77::786 Get password data.  vncsrv
DEBG ba65f6c8-6b99-4c25-8585-46078b70fe77::795 Get password data done.  vncsrv
```

Figure 10: Output from the module

The vncDll module was used by the Trickbot threat actors since 2018 (based on the compiled time for the files found in our malware collection), but the new updated version was released in the wild on May 11th 2021 (based on the same background research).

The viewer tool

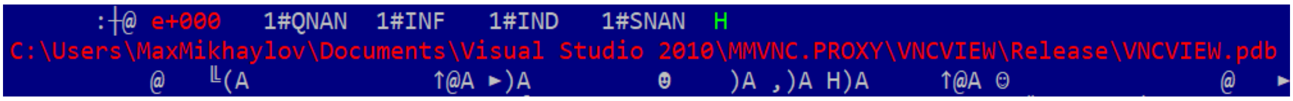
While searching for older versions of the module, we stumbled upon a very old one (e2916d8061258fc1c52739209b192406), with the compile time of 2019-04-29 08:19:47, referencing a PDB with a very unusual path:



```
on  AddressDisplay Control  SYSTEM  WinSta0 SeTcbPrivilege  AlterDesk01     185.172.129.11  ≡↑ΘÇ
  C:\Users\MaxMikhaylov\Documents\Visual Studio 2010\MMVNC.PROXY\VNCSRV\x64\Release\VNCSRV.pdb
Θ   ↑↕Θ         (↕Θ             @Θ               @    ↕Θ             Θ              @Θ x↕Θ P↕Θ
```

Figure 11: PDB path of the module

This led us to find another file (3bd25b1989db5802d0cf68ae3d532e87) with a similar path.



```
      :┼@ e+000    1#QNAN   1#INF    1#IND    1#SNAN   H
C:\Users\MaxMikhaylov\Documents\Visual Studio 2010\MMVNC.PROXY\VNCVIEW\Release\VNCVIEW.pdb
           @    ╙(A              ↑@A ►)A          Θ     )A ,)A H)A      ↑@A Θ              @    ►
```

Figure 12: PDB path of the viewer tool

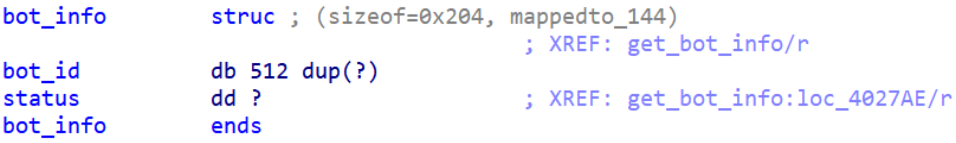
This second file (that we will further call "*the viewer*") is the client the attackers use to interact with the victims through the C2 servers. This gave us a "behind the scenes" look at how Trickbot operators are facilitating access on the victims' computers to their clients and on how their tools for interactions with the victims are working.

The viewer seems to be using a hardcoded IP address to communicate with the victims (even if the interface indicates otherwise, the connection function rewrites the user input). This server is behaving similarly to the other C2 servers, the setup part of the protocol being used to either retrieve the list of victims and their state, or to initiate the forwarding of messages between the clients and the victims.

The viewer can obtain the bot id of a victim either by asking the hardcoded server for a list of victims, or by user interaction, through the interface or by accessing a webserver handled by the viewer at http://127.0.0.1:80/vncstart.

When the viewer needs to obtain the list of victims, it sends the marker "\x01\x02\x03" to the baked in server, and the server responds with an array of structures detailing the bot id and the state of the bot. This array is parsed, displayed in a list of bots and stored locally in a file called *vnclist* located in the same directory as the viewer itself.

The structure for one entry in the array looks as follows:



```
bot_info        struc ; (sizeof=0x204, mappedto_144)
                                        ; XREF: get_bot_info/r
bot_id          db 512 dup(?)
status          dd ?                    ; XREF: get_bot_info:loc_4027AE/r
bot_info        ends
```

Figure 13: Bot information structure

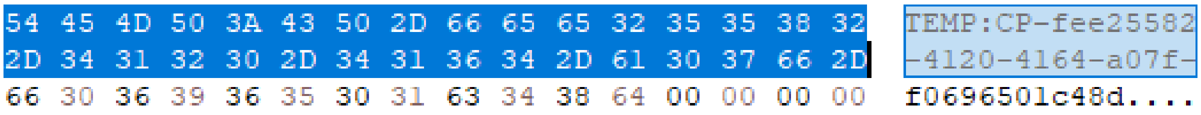
The status field is one of the following possible values:

Value of the status field	Status description (taken literal from the code itself)	Description
0	??? - unknown	No information is currently available
1	offline	The victim is offline
2	busy	Someone is already connected to the victim
3	blocked	Victim is somehow banned
10	online	The victim is online and ready to be interacted with

Table 1: Status fields

The most common values for the status field are 1, 2 and 10 (offline, busy and online), the value of 0 is set when the user enters a *bot id*.

If we connect to the server and ask for the list of victims, we can observe something odd: some values for the bot id field don't have the format of a bot id.



```
54 45 4D 50 3A 43 50 2D 66 65 65 32 35 35 38 32  TEMP:CP-fee25582
2D 34 31 32 30 2D 34 31 36 34 2D 61 30 37 66 2D  -4120-4164-a07f-
66 30 36 39 36 35 30 31 63 34 38 64 00 00 00 00  f0696501c48d....
```

Figure 14: Hex view of a response from then server when asking for a list of victims

A bunch of entries in the list start with "TEMP.CP-" followed by an UUID. This could be a developer of the module testing it using a custom module loader tool.

The *bot id* is then used to communicate with that victim. This is done by sending the marker "\x05\x01\x00" (technically it would be "\x05\x01\x00\x01", but the send function is specified to only send 3 bytes) and the bot id.

At this point, if the victim is online and no one else is connected to the server, the attacker will send a marker to the victim to start the screen sharing process and it will now only be used to forward packets between the client and the victim.

A statistic that we were able to extract during our investigation shows that the interval that a victim was in the online state (waiting for an attacker to connect) was between 14 and 46 minutes and the interval an attacker was connected to a victim to send commands was from 1 minute to 29 minutes. More details are illustrated in table below:

DAY	BOTS ONLINE	AVG NUMBER OF MINUTES SPENT ONLINE	AVG NUMBER OF MINUTES SPENT BUSY
5/21/2021	7A7BB7EE73B39FB5D0907FCF691552BB	14	
5/24/2021	8179540F734B51B0ABB3C6B9F7DBB7D6 B6582EEF39568179253764078FF7B37D BBAB9DBFD53B1DDFDFBC33B4D25F414B FFDF73B813BB48B95399B7F4C33750CB B17675C6F3FBF8595482BB6B71B7643F BB33B3BB33BB3B3B3BBCE7D734465B89	37.33333333	29.5
5/25/2021	B6582EEF39568179253764078FF7B37D 02099117A64BB6D09D5952EB37FF7AA5 BB3251DD6DBFF9397B94CBB9D1395ACB B17675C6F3FBF8595482BB6B71B7643F 5A8B3667BB3083FF87FBB395B37563B3 BB33B3BB33BB3B3B3BBCE7D734465B89 338F5B8B3B38BBDB25F773DDF7DCBBAF EF33B5DF15385B799C171D2BB9E50F3B	30	9.666666667
5/27/2021	BB7137A177FC5F951D3383B19CF7B721 F13F468DBDC8BBABFB711ABFBF91DF62 BE0A9BF57B3BA407170633BB99750CD3	32	4
5/28/2021	320B9DD0233B13F7F99F7FE0BF5B3FBB BBC773FB7437B96B359BB3B3067D933B BB342F9D6D2FFB4C991A44BBA56199E2	30	
6/5/2021	4BB94B32E237B383F5458B3E71F3D3D3	30	
6/7/2021	31EE4BB68D3591F73CD3BB9E15B9B2EB 338F5B8B3B38BBDB25F773DDF7DCBBAF EF33B5DF15385B799C171D2BB9E50F3B	46.33333333	9.5
6/9/2021	3FFB81F3AFFBBCF55F7683FB9D557DC2	40	1
6/15/2021	0CDBB4A9F9141337AA9FF4403B3A9511 7B17B3FF33D9F7B540BB093151A2E7B3 BB3251DD6DBFF9397B94CBB9D1395ACB 5BB3BC0BBF93F31F1B77EE9DF39334EA 9D7FBF76155241DB8915D97715531DBB 1FF2D737B737D16233BA7857250BB773 77B1BBAA31525073F1AFF9C3B3313FDF	45.42857143	4

Table 2: Online/Busy sessions

We also found more viewers, most of whom have little to no change to their functionality. The main differences between them are the hardcoded IP and, in some samples, the PDB location, suggesting the code is inherited and shared between members of the Trickbot Group.

Hash	Compile Time	PDB Path	IP
d383b6d26fe448d7e8a0266c7f90725a	2018-05-29 20:59:37	D:\Max\MMVNC.PROXY\VNCVIEW\x64\Release\VNCVIEW.pdb	198.12.95.248
3bd25b1989db5802d0cf68ae3d532e87	2018-09-25 11:12:08	C:\Users\MaxMikhaylov\Documents\Visual Studio 2010\MMVNC.PROXY\VNCVIEW\Release\VNCVIEW.pdb	217.172.179.14
39900f2e07d565a49c46e29f9ec51075	2018-10-26 12:34:59	C:\Users\MaxMikhaylov\Documents\Visual Studio 2010\MMVNC.PROXY\VNCVIEW\Release\VNCVIEW.pdb	198.27.87.34
647d941c35271fb685a6a4796d12efa2	2019-02-26 15:35:55	C:\Users\MaxMikhaylov\Documents\Visual Studio 2010\MMVNC.PROXY\VNCVIEW\Release\VNCVIEW.pdb	109.94.208.62
baff4d3263804341d80208d6fc1eb300	2019-11-11 10:41:21	D:\Projects\MMVNC.PROXY\VNCVIEW\Release\VNCVIEW.pdb	45.141.101.253

Table 3: Viewer Tools

Command and Control Infrastructure

In our research, we observed the three C2 servers found in the *vncconf* configuration file have the same *bot ids*. Correlating this with the fact that only one server is baked in the viewer and that it uses a different port than the normal servers (port 5900, while the other C2s use 443), we concluded that all public C2 servers are proxying data (directly or through a few layers) to the server in the viewer. Thus, the network has the following architecture:

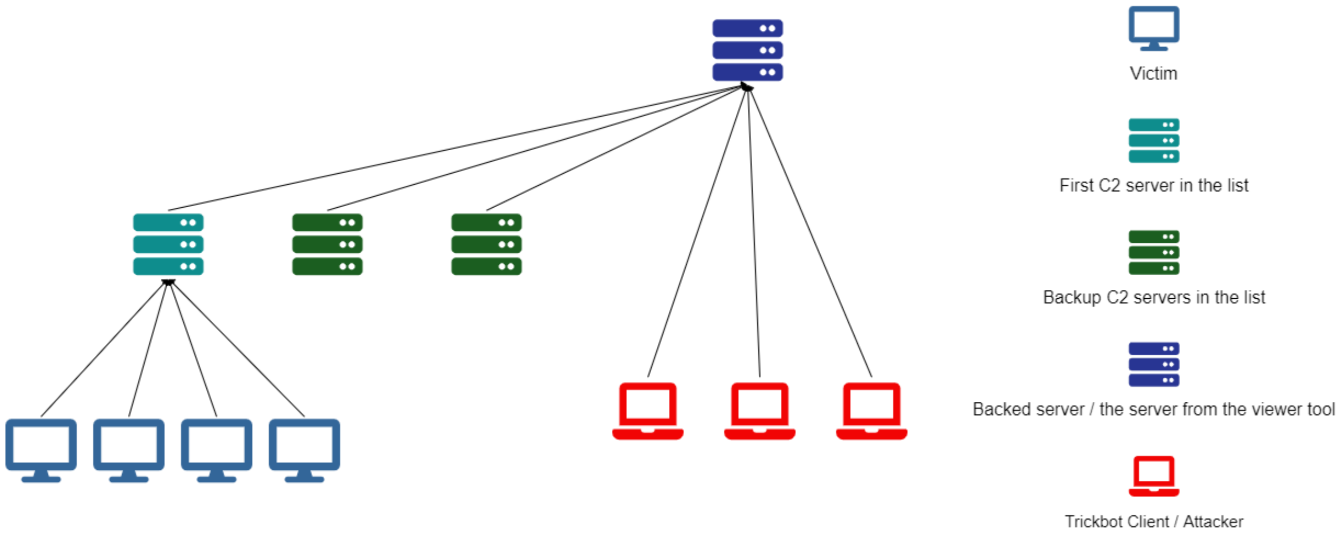
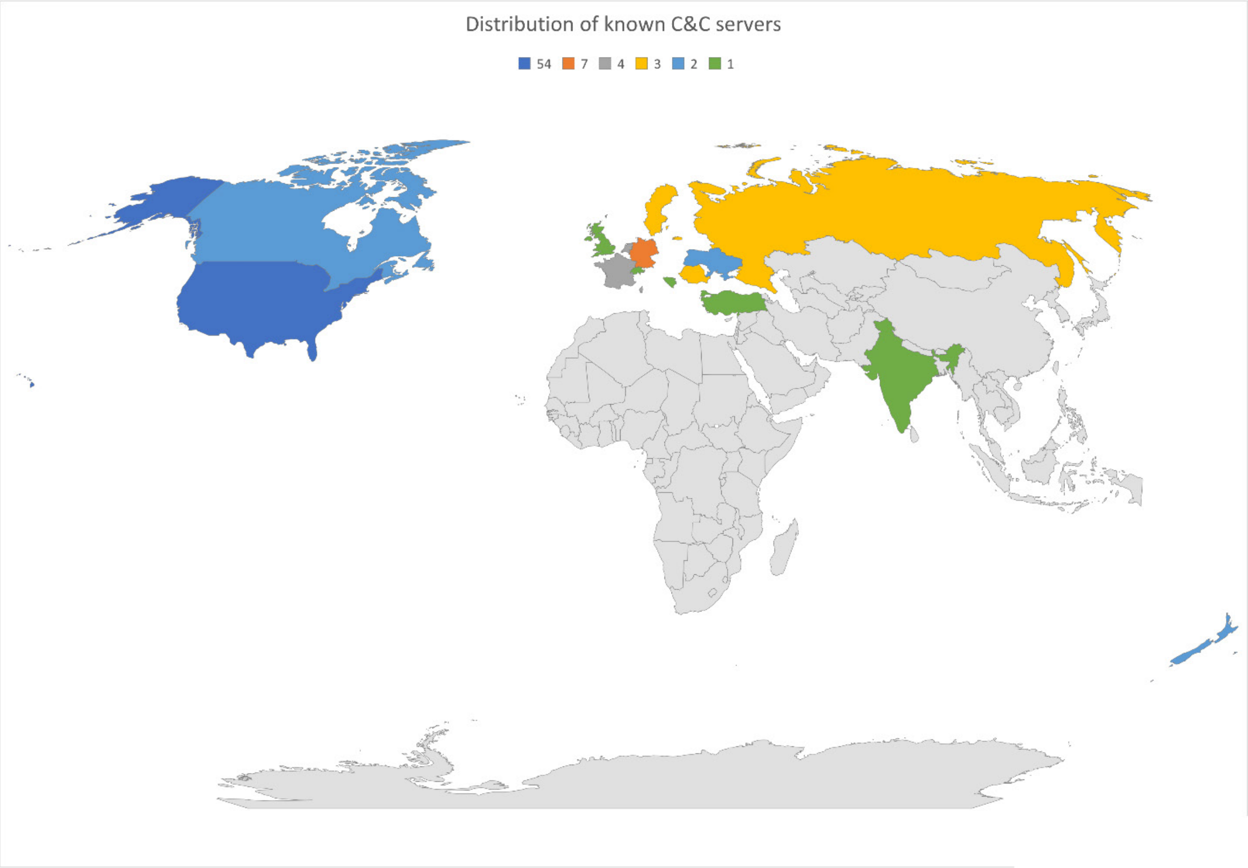

Victim
First C2 server in the list
Backup C2 servers in the list
Backed server / the server from the viewer tool
Trickbot Client / Attacker

Figure 15: Network structure of the module



Indicators of compromise

An up-to-date and complete list of indicators of compromise is available to Bitdefender Advanced Threat Intelligence users. More information about the program is available at https://www.bitdefender.com/oem/advanced-threat-intelligence.html

```
rule vncview
{
   meta:
      author = “Radu Tudorica”
      description = “Trickbot’s internal tool for interacting with vncDll/tvncDll victims”
   strings:
      $html = “<head><meta http-equiv=\”refresh\” content=\”0; URL=%s\”/></head>”
      $url = “http://127.0.0.1:80/vncstart” wide
      $status1 = “???”
      $status2 = “off-line”
      $status3 = “busy”
      $status4 = “blocked”
      $status5 = “on-line”
      $s = “VNC Server ID:” wide
   condition:
      ($html and $url) or ((all of ($status*)) and $s)
}
```

```
import “pe”
rule trickbot_vnc_module
{
    meta:
        author = “Radu Tudorica”
        description = “Trickbot’s vnc module”
    condition:
            (pe.exports(“Start”) and pe.exports(“Control”) and pe.exports(“FreeBuffer”) and
pe.exports(“Release”)) and
            (pe.exports(“NetServerStart”) and pe.exports(“NetServerStop”))
}
```

vncDll/tvncDll modules

MD5 hash	Compile time
6aef3296553e0aaf0e4f0659947c8497	2018-12-20 11:40:06
e2916d8061258fc1c52739209b192406	2019-04-29 08:19:47
0ab29fb3e27c67f082328550e55d7b2b	2019-08-30 17:57:16
6042e039ef260a7385d72ee947846172	2019-08-30 17:57:29
f1102b24fee1a98df9e8b847532f22d5	2019-09-04 17:34:15
e946bad3a605b1df178fe70b5b001235	2019-09-11 18:20:38
987de9ce70e272a86c063fefe43e41bf	2019-09-16 12:08:35
4cb28de0bc97dc8ae67311516694d6c5	2019-12-19 18:37:50
05566a2f11aa1b3d8d2b322878b46ff7	2019-12-19 18:38:13
d9214badc75c4fd9d5302f65410e3edb	2020-01-10 10:47:02
30804bbfe985e5ecd685a6cfcd77a3f7	2020-01-10 10:47:21
203f0082dc1944bf0978603a608b2ae8	2020-01-10 15:57:51
018cf114c397b6e69a7ee1456a4dfc84	2020-01-10 15:58:11
4723d1ba7eaa9d27e47005ebbf4c40e0	2020-01-15 13:07:42
596c67af0e1eede4ad25b68244e8b564	2020-01-15 13:07:55
c0a56755d6d2e69faf34ef0f35accb42	2020-01-17 13:52:19
78c6ff8c5deb60411fcbecea6cec51ea	2020-01-17 13:57:21
7dd68ccb4ed06d3cc832158390b36d30	2020-02-05 19:50:20
41269382fb65617d9246b4761ed4f2cb	2020-02-05 19:50:36
42fd0ff40929dee63dea51352a111310	2020-02-11 10:34:31
b2f59339039fbd6eee71b6a07c2b6e2e	2020-02-11 10:34:31
1e48b373a9b55c4b76b82dda58b61f14	2020-02-11 10:34:46
e31f0609fa727592013052f1849795a8	2020-02-11 10:34:46
22d875ba9ca40e96058f934237fc03ec	2020-07-13 18:22:24
ff6f4557e8bf7663fb5802719af716be	2020-07-13 18:31:05
bc88b1f77969823a4f04aecb5c861b48	2020-07-17 12:05:47
3866fe4eae9b1bbc35e0b2fa67501eda	2020-07-17 12:09:19
86042a47266e0258f261ba93161c76c0	2020-07-27 16:52:13
0d881b0dc664c9ba389d12a416667b83	2020-07-27 16:56:33
2cd9734d02a58847a40098a52b3b6541	2020-10-01 12:01:45
ee2bbce67b77e10727e8064d56a1b185	2020-10-01 12:01:45
e6428f29d769601491bb32682871f3e5	2020-10-13 00:46:31
2b7bc36504fbbe134e8a2203ae9e7eb6	2020-12-24 22:18:17
7ec473f5b5841f649fa14431bc830915	2021-01-15 17:00:54

MD5 hash	Compile time
e325d8f9f815aa31d522f275e9e98636	2021-01-21 14:12:33
0fd0db8274d43b6fc77f1b9ebbdb039d	2021-03-22 14:56:05
8c8a59b04114189139db177c1831eeea	2021-03-22 14:56:05
43d8978d656f47c7a20a305245a5d61e	2021-03-22 15:22:42
7c1b41a592dc9656dcaab82998dcff01	2021-03-22 15:22:42
f14af5e2dd034bfae624b06f7445dc7c	2021-03-24 16:18:27
85f43963a5e4f3e808654d2f6ba8e7ba	2021-03-24 16:18:49
b099de385032c4186adf27467516ed0d	2021-04-19 14:19:00
1483dfa7c71da01458b691482724abf0	2021-04-19 14:20:39
08a8a5afc6af0608842c0b4162771084	2021-04-20 15:38:26
d70971d537a14857d72390a616e13303	2021-04-20 15:38:26
64a063f2a0218d1d70001f38503b0788	2021-04-20 15:39:54
a8fa658313f88db2b2476081794d8079	2021-04-20 15:39:54
c54f958376f35520e248c277488b9cb0	2021-05-11 11:30:49
856207959766b05d30f7f7d3125fe599	2021-05-11 12:24:53
95d30e6ccf6982146523276a097ac6d3	2021-05-12 22:20:07
9aa06c006431ff37e000675c1913a812	2021-05-12 22:20:07
0b5e4a0acb571a5c53db89d09f8d173b	2021-05-12 22:20:53
35fd5b78f8a2b60b060fa7a66d1e44e1	2021-05-12 22:20:53
7d55d26f840437e1c34c20da08822013	2021-05-17 17:49:57
3c94cc2f3c1629bad971b2f69c36c252	2021-05-18 18:05:55
857c71b682d1d60d455ae660c0bbcfdf	2021-05-18 18:05:55
289d14c89b3ec655b29a7e656261ad91	2021-05-18 18:09:17
d9e30fef6b6c1fac44ea1b124a057178	2021-05-18 18:09:17
53eea0b33eaabdf910f3410bb4aa7b47	2021-05-26 23:42:26
e2852eb113ec8c227ce4f3527a613648	2021-05-26 23:42:26
28ac5562ecf5d4aca3a96f47b4d0f1e6	2021-05-27 13:58:39
447ddd3673a01f4fffb955545a13f8f8	2021-05-27 13:58:39

Why Bitdefender

Proudly Serving Our Customers

Bitdefender provides solutions and services for small business and medium enterprises, service providers and technology integrators. We take pride in the trust that enterprises such as **Mentor, Honeywell, Yamaha, Speedway, Esurance or Safe Systems** place in us.

Leader in Forrester's inaugural Wave™ for Cloud Workload Security
NSS Labs "Recommended" Rating in the NSS Labs AEP Group Test
SC Media Industry Innovator Award for Hypervisor Introspection, 2nd Year in a Row
Gartner® Representative Vendor of Cloud-Workload Protection Platforms

Dedicated To Our +20.000 Worldwide Partners

A channel-exclusive vendor, Bitdefender is proud to share success with tens of thousands of resellers and distributors worldwide.

CRN 5-Star Partner, 4th Year in a Row. Recognized on CRN's Security 100 List. CRN Cloud Partner, 2nd year in a Row
More MSP-integrated solutions than any other security vendor
3 Bitdefender Partner Programs - to enable all our partners – resellers, service providers and hybrid partners – to focus on selling Bitdefender solutions that match their own specializations

Trusted Security Authority

Bitdefender is a proud technology alliance partner to major virtualization vendors, directly contributing to the development of secure ecosystems with **VMware, Nutanix, Citrix, Linux Foundation, Microsoft, AWS, and Pivotal.**

Through its leading forensics team, Bitdefender is also actively engaged in countering international cybercrime together with major law enforcement agencies such as FBI and Europol, in initiatives such as NoMoreRansom and TechAccord, as well as the takedown of black markets such as Hansa. Starting in 2019, Bitdefender is also a proudly appointed CVE Numbering Authority in MITRE Partnership.

RECOGNIZED BY LEADING ANALYSTS AND INDEPENDENT TESTING ORGANIZATIONS

CRN AV-TEST AV Gartner 451 Research FORRESTER IDC GLOBAL

TECHNOLOGY ALLIANCES

Microsoft NUTANIX aws Pivotal Cloud Foundry CITRIX

Bitdefender®

Founded 2001, Romania
Number of employees 1800+

Headquarters
Enterprise HQ – Santa Clara, CA, United States
Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES
USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA
Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS
Australia: Sydney, Melbourne

UNDER THE SIGN OF THE WOLF

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.