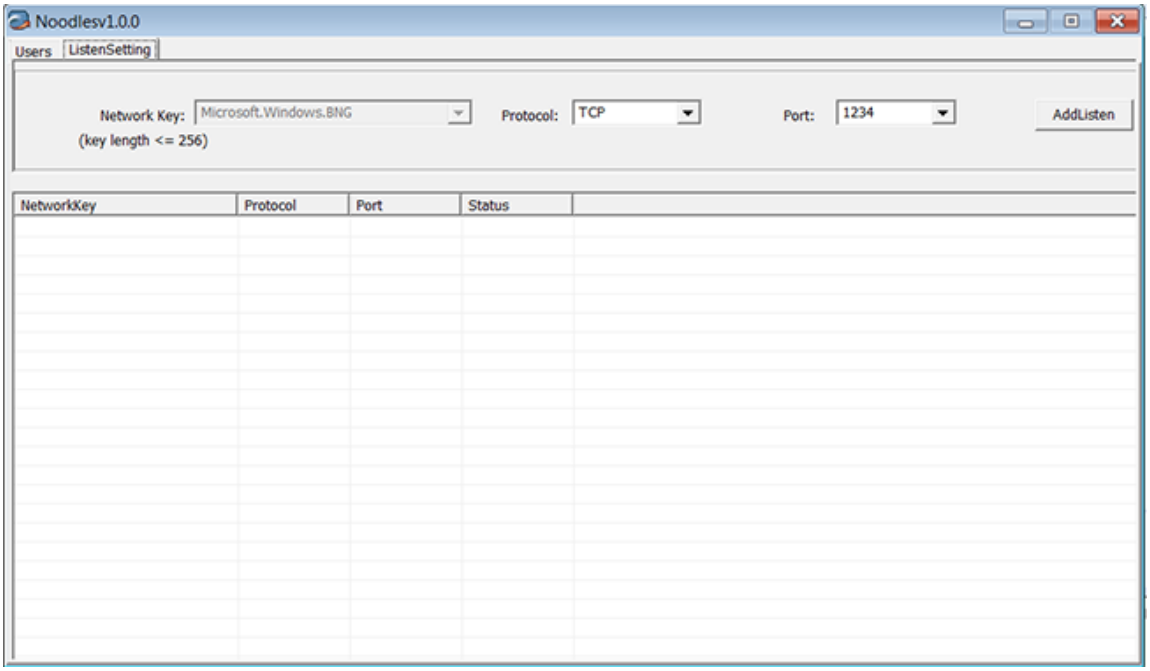
Decoding network data from a Gh0st RAT variant

research.nccgroup.com/2018/04/17/decoding-network-data-from-a-gh0st-rat-variant/

April 17, 2018



During a forensic investigation in March 2018 we were able to retrieve some files which appeared to be linked with a well-known group named Iron Tiger.

From our research, we believe that the perpetrator hasn't shown any advanced technical capabilities in this attack. In fact, the main goal was to mine cryptocurrency. During the investigation we found several tools such as password dumpers, Monero cryptocurrency miners, portable executable (PE) injectors and a modified version of Gh0st RAT. Even though Bitdefender and TrendMicro have published reports [1] [2] describing some of the tools used by the group, we were not able to find any references to this specific modified version of Gh0st RAT. Therefore, the purpose of this blog is to briefly describe the modified Gh0st RAT version that is used by the group.

The malicious payload

Firstly, a malicious executable file is executed which will drop a batch file (install.bat) and a cabinet file (data.cab) under a new folder in C:\ProgramData with a random name. The cabinet file includes two files: the malicious shellcode which is partially encrypted and a Dynamic-link library (DLL) which will execute the malicious shellcode. The malicious executable file will then execute the batch file, which will decompress and execute the DLL file. Persistence is achieved by creating a new service or a new registry key, depending on the privileges that the malware has.

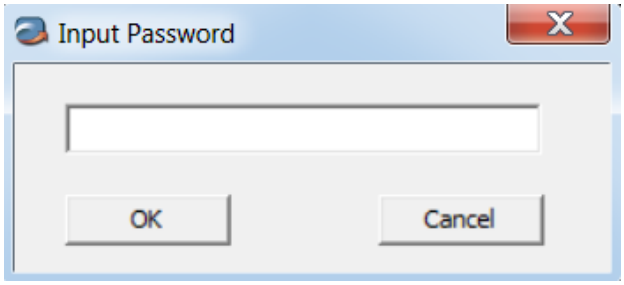
Once the execution is passed to the shellcode, it will decrypt the rest of the encrypted data using a single byte as the key in an eXclusive OR (XOR) loop. After decryption, the following interesting string is observed:

```
Microsoft.Windows.BNG|[C&C IP address]:443;|1;1;1;1;1;1;1;|00-24;|1
```

The main goal of the shellcode is to load and execute the attacker's plugins in memory.

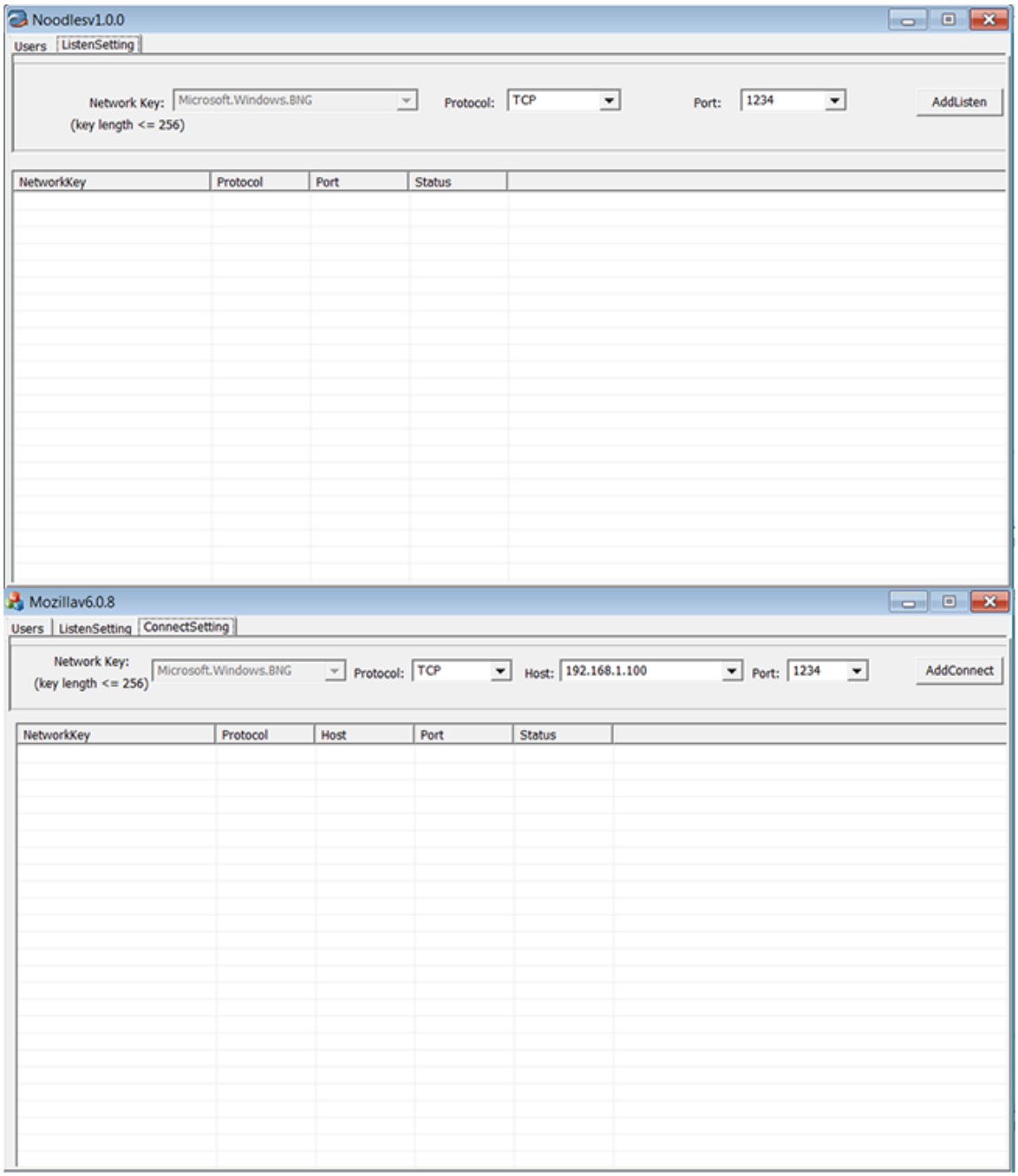
Modified Gh0st RAT

While analysing the previous files, we found a folder named 'Plugins' with some interesting DLLs inside and two files which required a password upon execution (example in Figure 1).

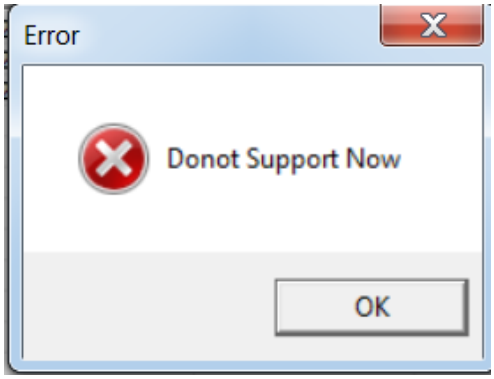


After reversing the binary, we found out that the password is not hardcoded. Instead, the password is based on the current year and month. For example, the password for the month of March in 2018 is '201803'.

The first file which is named 'Noodles' seems to be an old modified version of Gh0st RAT based on compilation date and features. The second file named 'Mozilla' is the primary tool which was used for this attack. Below you can see how both panels look.



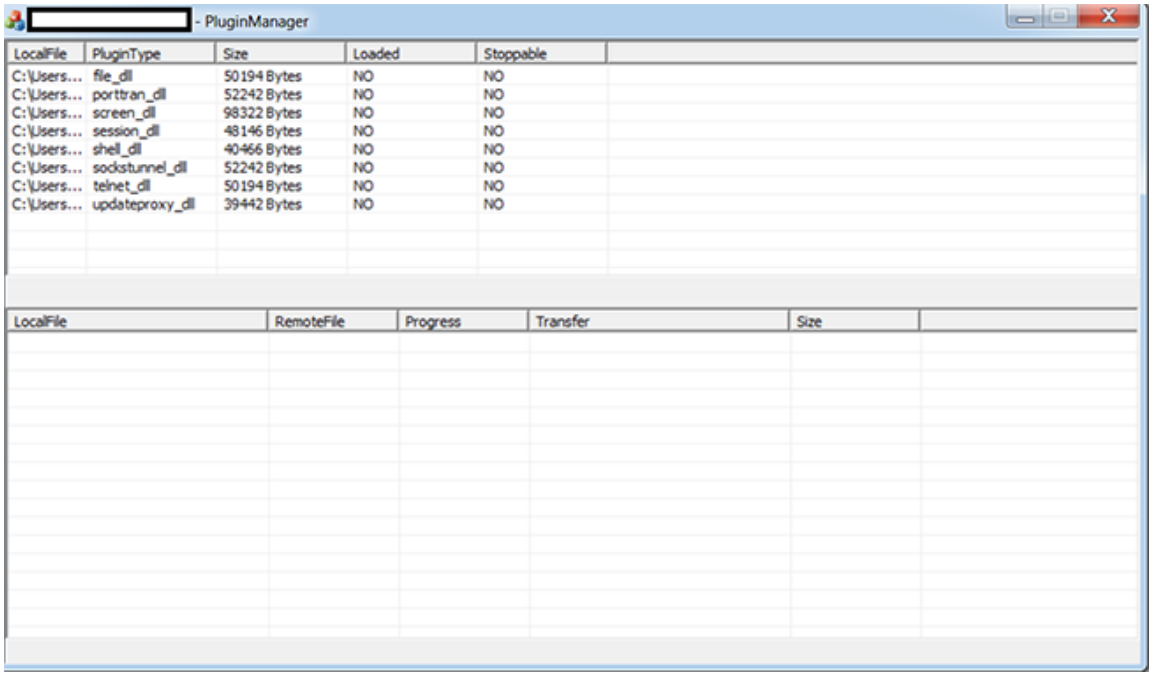
Currently, both tools can listen on any given ports but only 'Mozilla' can connect to a bind port. The supported protocols include Secure Sockets Layer (SSL) and Transmission Control Protocol (TCP). One of the protocols in the list is named, according to the malware authors, 'WINNET' but this is not supported yet and an error message is displayed. This might suggest that this tool is still in development and there are plans to add additional functionality.



Additionally, analysis of the Mozilla tool identified a Program database (PDB) path inside the binary:

K:\Mozilla\Mozillav6.0.x_dll\WorkActive\Release\Mozilla.pdb

The tool heavily relies on plugins. When there is a new victim connection, the attacker can use the PluginManager to load new plugins to the infected machine.



Most of the available plugins are based on the Gh0st RAT source code and a summary of them can be found below:

Name	Description	Original Filename
File Manager	A file explorer which can download, rename and delete files	Plugin_File.dll
PortTran Manager	Port forward ports	Plugin_HPort.dll
ScreenSpy Manager	Capture victim's screen remotely	Plugin_ScreenSpy.dll
Session Manager	Offers the attacker the option to use a subset of plugins against other logged on users on the same machine	Plugin_Session.dll
Shell Manager	Provides a remote shell to the attacker	Plugin_Shell.dll
Telnet Manager	Opens a telnet session between the victim and the attacker	Plugin_Telnet.dll
Socks Manager	Runs a socks proxy on the victim's machine	Plugin_Socks.dll
ProxyInfo Update	Updates proxy	Plugin_UpdateProxy.dll

Network communication

The network traffic between the victim and the attacker is encrypted using Rivest Cipher 4 (RC4). The key is unique for each request and is encrypted using 'XOR' and 'AND' instructions. The key is stored in the first 28 bytes of the request. We wrote a Python script that takes as input a network capture (PCAP) and decrypts it, which can be found in our Github repository here: https://github.com/nccgroup/Cyber-Defence/tree/master/Scripts/gh0st_variant_c2

For example, below we can see the initial connection between a victim and the C2 server, where the machine name is sent:

Data to server..

```
[i] Found key: C8410061440A01c762FA9000
00000000: 0501570049004E00  2D00510047004F00  ..W.I.N.-.Q.G.O.
00000010: 3400430051004E00  49004F004E003500  4.C.Q.N.I.O.N.5.
```

And below the distinctive start of a PE file is seen, as a plugin is transferred to the client.

Data to client..

```
[i] Found key: C841006804EC089c84EA9020
00000000: 4D5A900003000000  04000000FFFF0000  MZ..............
00000010: B800000000000000  4000000000000000  ........@.......
00000020: 0000000000000000  0000000000000000  ................
00000030: 0000000000000000  00000000F0000000  ................
00000040: 0E1FBA0E00B409CD  21B8014CCD215468  ........!..L.!Th
00000050: 69732070726F6772  616D2063616E6E6F  is program canno
00000060: 742062652072756E  20696E20444F5320  t be run in DOS
00000070: 6D6F64652E0D0D0A  2400000000000000  mode....$.......
```

IOCs

Command and Control (C&C) IPs:

- 23.227.207.137
- 89.249.65.194

Malicious files directory:

- C:\ProgramData\HIDMgr
- C:\ProgramData\Rascon
- C:\ProgramData\TrkSvr

Malicious service name:

- HIDMgr
- RasconMan
- TrkSvr

Registry key for persistence:

> ‘rundll32.exe_malicious_DLL_path’ in ‘HKCU\Software\Microsoft\Windows\CurrentVersion\Run’

File names and hashes:

Filename	Hash(SHA-256)
Mozilla.exe	EE04B324F7E25B59D3412232A79D1878632D6817C3BB49500B214BF19AFA4E2C
Updateproxy.dll	0BA49FEB7784E6D33D821B36C5C669D09E58B6795ACA3EEBBF104B763B3B3C20
Telnet.dll	33B7407E534B46BF8EC06D9F45ECD2D3C7D954340669E94CD7CEDCBAE5BAD2DD

Filename	Hash(SHA-256)
Socks.dll	6160AF383794212B6AD8AB9D6D104BBE7AEFB22410F3AB8EA238F98DABFC48B7
Shell.dll	C63B01C40038CA076072A35913F56D82E32FCEE3567650F3392B5C5DA0004548
Session.dll	D51EC4ACEAFA971E7ABD0CF4D27539A4212A448268EF1DB285CD9CE9024D6EB3
Screen.dll	BD8086DE44E16EFDD380E23E49C4058D956538B01E1AE999B679B6B76B643C7D
Port.dll	B44A9545B697B4D46D5B96862A6F19EA72F89FED279F56309B2F245AC8380BE0
File.dll	F4DF97108F18654089CFB863F2A45AA41D17A3CE8A44CCCC474F281A20123436
ConEmu.exe	D31D38403E039F5938AE8A5297F35EB5343BB9362D08499B1E07FAD3936CE6F7
Noodles.exe	A591D4D5B8D23FF12E44A301CE5D4D9BF966EBA0FC0068085B4B4EC3CE352963
Coal.exe (Malicious executable)	EEBFF21DEF49AF4E85C26523AF2AD659125A07A09DB50AC06BD3746483C89F9D
Abg.exe (Malicious executable)	97B9D7E16CD6B78A090E9FA7863BD9A57EA5BBE6AE443FA788603EEE5DA0BFC3
23d.exe (Malicious executable)	B6C21C26AEF75AD709F6C9CFA84BFA15B7EE709588382CE4BC3544A04BCEB661
89d.exe (Malicious executable)	DB9B9FA9EFA53662EC27F4B74B79E745F54B6C30C547A4E5BD2754E9F635F6DB

References

Published date: 17 April 2018

Written by: Nikolaos Pantazopoulos

1. 
   Thank you for the research done on these security cameras and the app used. You saved me a lot of…