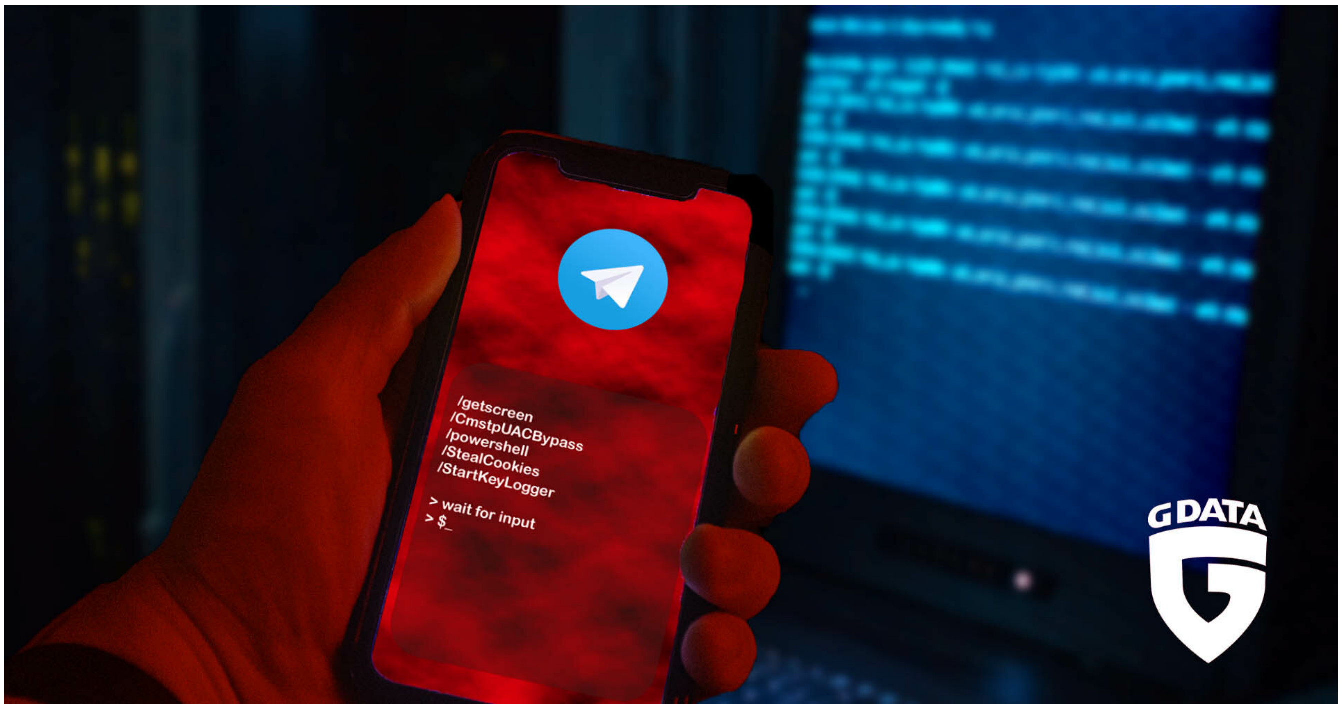
T-RAT 2.0: Malware control via smartphone

gdatasoftware.com/blog/trat-control-via-smartphone

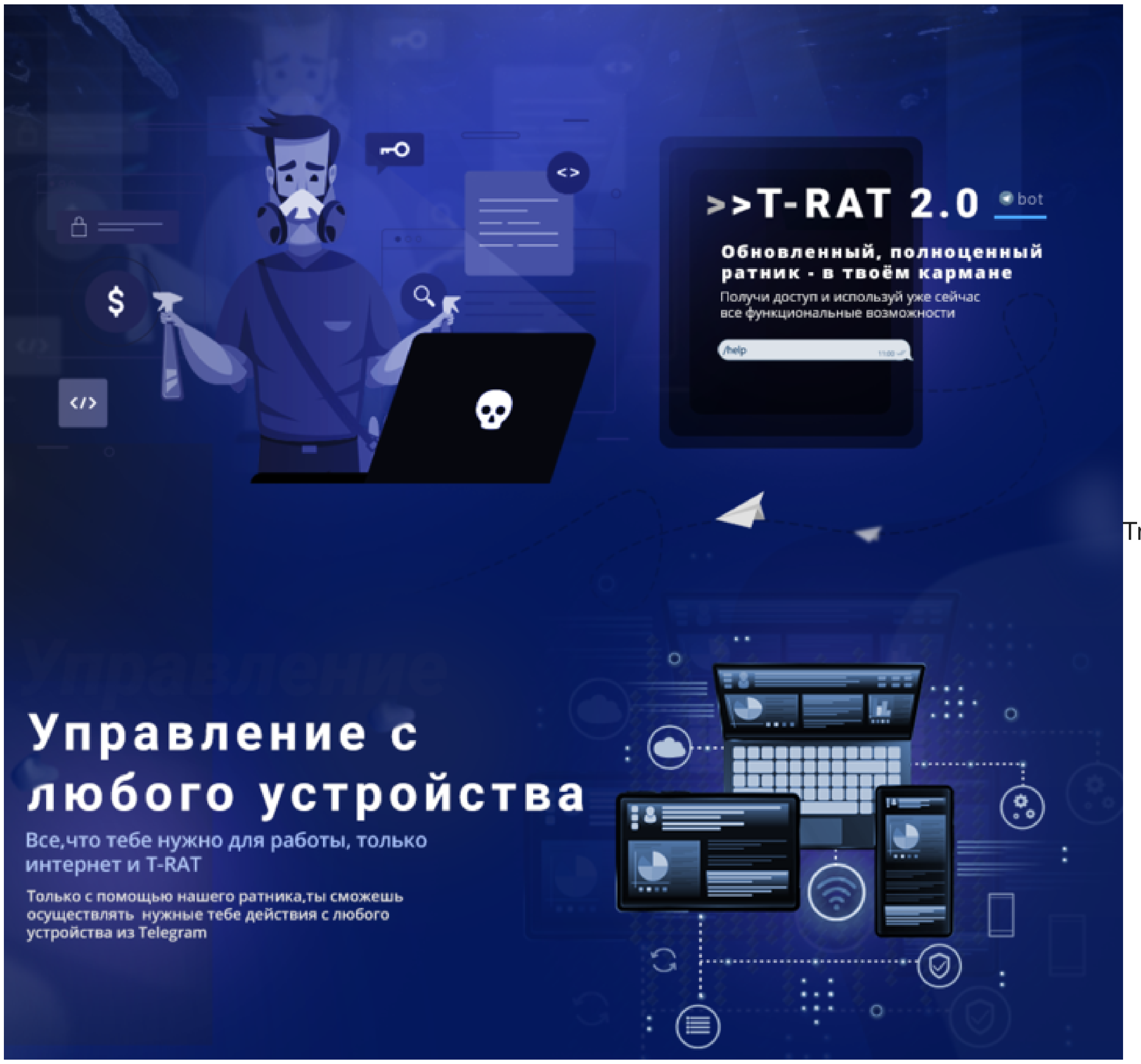


Malware sellers want to attract customers with convenience features. Now criminals can remote control malware during their bathroom routine by just using a smartphone and Telegram app.

Advertisments on Russian forums

The researcher @3xp0rtblog discovered T-RAT 2.0 and posted about it on Twitter, including a sample hash and selling threads on Russian forums. One extravagant advertisment is shown below.

The images below show a section each of a 1000x5429 advertisment banner posted on lolz.guru (found and reported by 3xp0rtblog). The Russian text praises comfort and convenience while using T-RAT because it can be controlled via smartphone with Telegram app.


>>T-RAT 2.0
bot
Обновленный, полноценный ратник - в твоём кармане
Получи доступ и используй уже сейчас все функциональные возможности
/help
Управление с любого устройства
Все,что тебе нужно для работы, только интернет и T-RAT
Только с помощью нашего ратника,ты сможешь осуществлять нужные тебе действия с любого устройства из Telegram

Translation: "An updated, full-fledged RAT - in your pocket. Get access and use now all functionality. Control from any device, everything you need for it to work are the Internet and T-RAT"

Преимущества

Почему стоит задуматься о покупке?

Комфорт и удобство

Наш продукт не требует от тебя никаких действий по настройке. Тебе нужно всего лишь создать бота в Telegram

Простое управление

При покупке мы обязаны выдать тебе мануал по командам и взаимодействию с ратником.
Хотя мы считаем, что он тебе не понадобится, т ак как с управлением разберется даже ребенок

Огромный функционал по приятной стоимости

В нашем продукте собрано большое количество функционала, начиная от файлового менеджера, заканчивая функциями, которые есть в каждом ратнике

Анонимность и надежность

Наш продукт полностью анонимен ,и данные передаются только от твоего бота к жертве.
Он не использует никакие наши сервисы и хостинги, так что полностью независим от нас

Обновления и улучшения

Мы занимаемся улучшением нашего продукта и постоянно добавляем новые функции и исправляем ошибки

Очистка от детектов

Своевременная, бесплатная очистка от детектов АВ и отправка новых билдов клиентам - бесплатно

Translation:

"Advantages - why you should consider buying: comfort and convenience, simple control, huge functionality at a nice cost, anonymity and reliability, updates and improvements, cleaning from detectors"

Infection chain and persistence

The first known stage of infection is the downloader[4]. It obtains an encrypted file[6] from **hxxps://hgfhhdsf.000webhostapp.com/1DJjnw(dot)jpg** and saves it to **%TEMP%/gfdggfd.jpg**.

For decrypting the payload, the downloader applies XOR with the key 0x01. The resulting file is a ZIP archive which it saves to **%TEMP%/hrtghgesd.zip**. The downloader proceeds to delete **%TEMP%/gfdggfd.jpg** and extracts the ZIP archive. Sidenote: Both hardcoded names consist of characters whose keys are right besides each other on a QWERTY keyboard, so the threat actor likely just rolled a body part on the keyboard to create them.

The location of the extracted malware is determined as follows:

1) The downloader checks if the current user has administrator rights. If they have, the first part of the path is one of the following (chosen randomly)

- %APPDATA%\Microsoft\Windows\
- %USERPROFILE%\Windows\System32\
- %LOCALAPPDATA%\Microsoft\Windows\

If they don't have administrator rights, the first part of the path is one of the following

- %SYSTEM%\Microsoft\Protect\
- %COMMONAPPDATA%\Microsoft\Windows\

- %USERPROFILE%\AppData\LocalLow\Microsoft\Windows\
- C:\Windows\assembly\GAC\

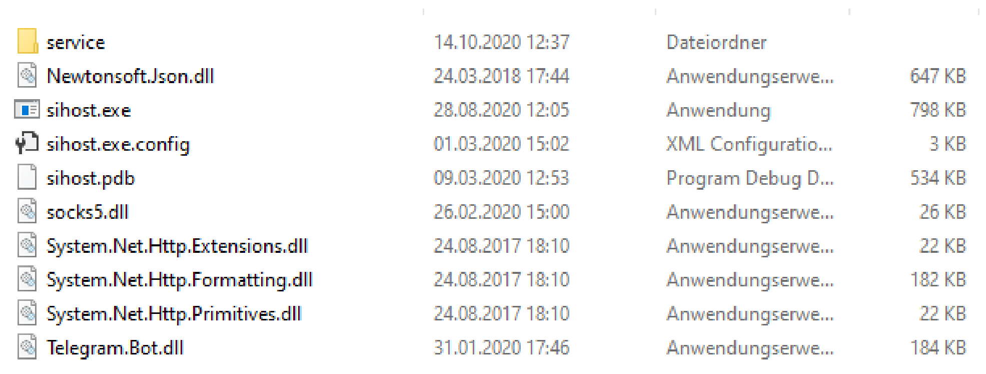
2) For the second part of the malware path the downloader generates a random number between 347 and 568203, converts that to a string, then generates the hash either using MD5, SHA1 or SHA256. It uses the hash's hexadecimal representation as second part of the malware path.

The archive contains the actual **T-RAT executable**, named **sihost.exe**, as well as several DLLs that the RAT needs. Some notable libraries are the **Telegram.Bot.dll** and **socks5.dll**.

A subfolder named **service** contains six more files (hashes are in the IoC listing):

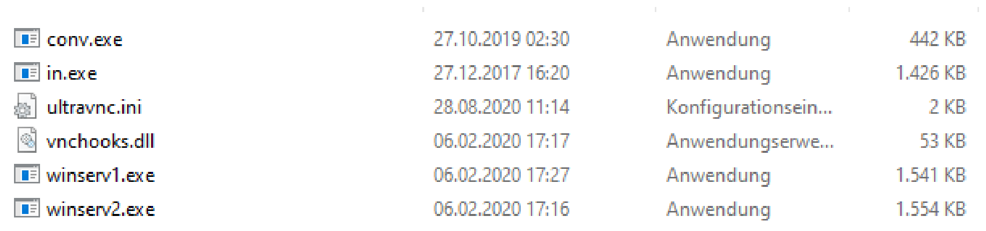
Filename	Description
conv.exe	High Performance MPEG 1.0/2.0/2.5 Audio Player
in.exe	RDP Wrapper
ultravnc.ini	UltraVNC configuration file
vnchooks.dll	UltraVNC - VNCHooks DLL
winserv1.exe	VNC Server 32 bit
winserv2.exe	VNC Server 64 bit

The downloader persists **sihost.exe** by scheduling a daily task. The name for the task is the processor ID of the system. If the current user has admin rights, it will set the run level to **HIGHEST**. Afterwards the downloader deletes itself with the help of a Batch file.



service	14.10.2020 12:37	Dateiordner	
Newtonsoft.Json.dll	24.03.2018 17:44	Anwendungserwe...	647 KB
sihost.exe	28.08.2020 12:05	Anwendung	798 KB
sihost.exe.config	01.03.2020 15:02	XML Configuratio...	3 KB
sihost.pdb	09.03.2020 12:53	Program Debug D...	534 KB
socks5.dll	26.02.2020 15:00	Anwendungserwe...	26 KB
System.Net.Http.Extensions.dll	24.08.2017 18:10	Anwendungserwe...	22 KB
System.Net.Http.Formatting.dll	24.08.2017 18:10	Anwendungserwe...	182 KB
System.Net.Http.Primitives.dll	24.08.2017 18:10	Anwendungserwe...	22 KB
Telegram.Bot.dll	31.01.2020 17:46	Anwendungserwe...	184 KB

Content of ZIP archive [3]

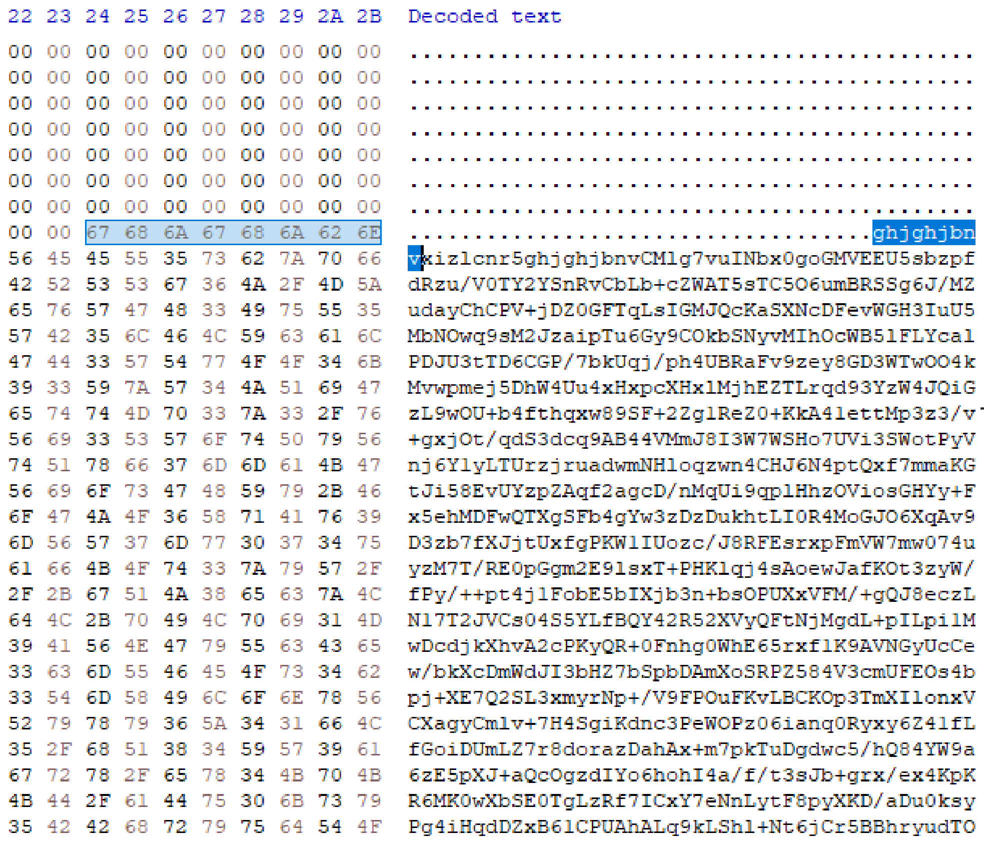


conv.exe	27.10.2019 02:30	Anwendung	442 KB
in.exe	27.12.2017 16:20	Anwendung	1.426 KB
ultravnc.ini	28.08.2020 11:14	Konfigurationsein...	2 KB
vnchooks.dll	06.02.2020 17:17	Anwendungserwe...	53 KB
winserv1.exe	06.02.2020 17:27	Anwendung	1.541 KB
winserv2.exe	06.02.2020 17:16	Anwendung	1.554 KB

Content of service folder in ZIP archive [3]

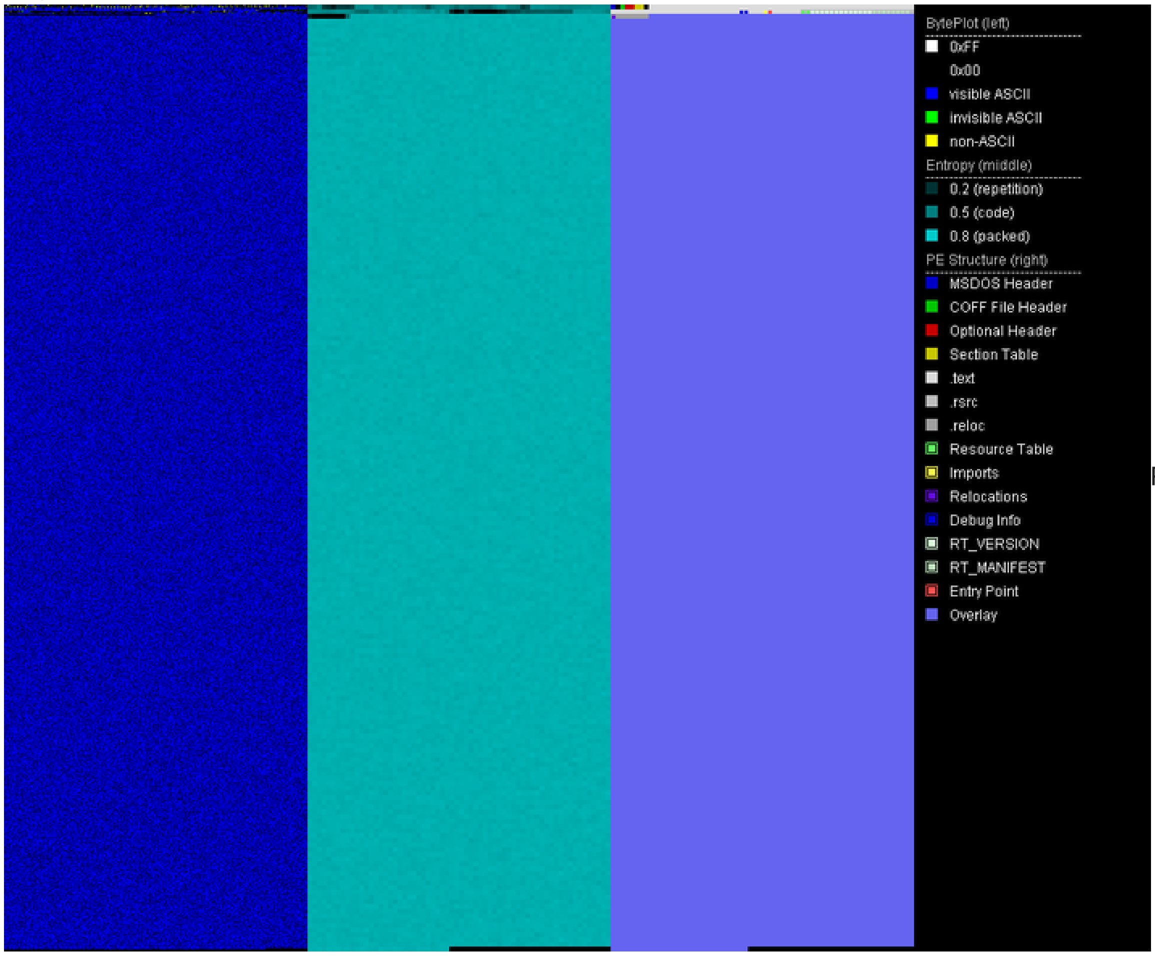
Packer and obfuscator

The original T-RAT sample[1] as well as the downloader[4] are .NET assemblies and packed the same way. The packed part is embedded as base64 string in the overlay of the file. Beginning and end of the strings are marked by the sequence "ghjghjbnv". The packer stub searches the sequence to find the packed image, decodes the base64 string and dynamically loads the resulting .NET assembly.



```
22 23 24 25 26 27 28 29 2A 2B  Decoded text
00 00 00 00 00 00 00 00 00 00  ........................................
00 00 00 00 00 00 00 00 00 00  ........................................
00 00 00 00 00 00 00 00 00 00  ........................................
00 00 00 00 00 00 00 00 00 00  ........................................
00 00 00 00 00 00 00 00 00 00  ........................................
00 00 00 00 00 00 00 00 00 00  ........................................
00 00 00 00 00 00 00 00 00 00  ........................................
00 00 67 68 6A 67 68 6A 62 6E  ................................ghjghjbn
56 45 45 55 35 73 62 7A 70 66  vxizlcnr5ghjghjbnvCMlg7vuINbx0goGMVEEU5sbzpf
42 52 53 53 67 36 4A 2F 4D 5A  dRzu/V0TY2YSnRvCbLb+cZWAT5sTC5O6umBRSSg6J/MZ
65 76 57 47 48 33 49 75 55 35  udayChCPV+jDZ0GFTqLsIGMJQcKaSXNcDFevWGH3IuU5
57 42 35 6C 46 4C 59 63 61 6C  MbNOwq9sM2JzaipTu6Gy9COkbSNyvMIhOcWB5lFLYcal
47 44 33 57 54 77 4F 4F 34 6B  PDJU3tTD6CGP/7bkUqj/ph4UBRaFv9zey8GD3WTwOO4k
39 33 59 7A 57 34 4A 51 69 47  Mvwpmej5DhW4Uu4xHxpcXHxlMjhEZTLrqd93YzW4JQiG
65 74 74 4D 70 33 7A 33 2F 76  zL9wOU+b4fthqxw89SF+2ZglReZ0+KkA4lettMp3z3/v
56 69 33 53 57 6F 74 50 79 56  +gxjOt/qdS3dcq9AB44VMmJ8I3W7WSHo7UVi3SWotPyV
74 51 78 66 37 6D 6D 61 4B 47  nj6YlyLTUrzjruadwmNHloqzwn4CHJ6N4ptQxf7mmaKG
56 69 6F 73 47 48 59 79 2B 46  tJi58EvUYzpZAqf2agcD/nMqUi9qplHhzOViosGHYy+F
6F 47 4A 4F 36 58 71 41 76 39  x5ehMDFwQTXgSFb4gYw3zDzDukhtLI0R4MoGJO6XqAv9
6D 56 57 37 6D 77 30 37 34 75  D3zb7fXJjtUxfgPKW1IUozc/J8RFEsrxpFmVW7mw074u
61 66 4B 4F 74 33 7A 79 57 2F  yzM7T/RE0pGgm2E9lsxT+PHKlqj4sAoewJafKOt3zyW/
2F 2B 67 51 4A 38 65 63 7A 4C  fPy/++pt4jlFobE5bIXjb3n+bsOPUXxVFM/+gQJ8eczL
64 4C 2B 70 49 4C 70 69 31 4D  N17T2JVCs04S5YLfBQY42R52XVyQFtNjMgdL+pILpilM
39 41 56 4E 47 79 55 63 43 65  wDcdjkXhvA2cPKyQR+0Fnhg0WhE65rxflK9AVNGyUcCe
33 63 6D 55 46 45 4F 73 34 62  w/bkXcDmWdJI3bHZ7bSpbDAmXoSRPZ584V3cmUFEOs4b
33 54 6D 58 49 6C 6F 6E 78 56  pj+XE7Q2SL3xmyrNp+/V9FPOuFKvLBCKOp3TmXIlonxV
52 79 78 79 36 5A 34 31 66 4C  CXagyCmlv+7H4SgiKdnc3PeWOPz06ianq0Ryxy6Z41fL
35 2F 68 51 38 34 59 57 39 61  fGoiDUmLZ7r8dorazDahAx+m7pkTuDgdwc5/hQ84YW9a
67 72 78 2F 65 78 34 4B 70 4B  6zE5pXJ+aQcOgzdIYo6hohI4a/f/t3sJb+grx/ex4KpK
4B 44 2F 61 44 75 30 6B 73 79  R6MK0wXbSE0TgLzRf7ICxY7eNnLytF8pyXKD/aDu0ksy
35 42 42 68 72 79 75 64 54 4F  Pg4iHqdDZxB61CPUAhALq9kLShl+Nt6jCr5BBhryudTO
```

The base64 string with marker sequence in a hex editor

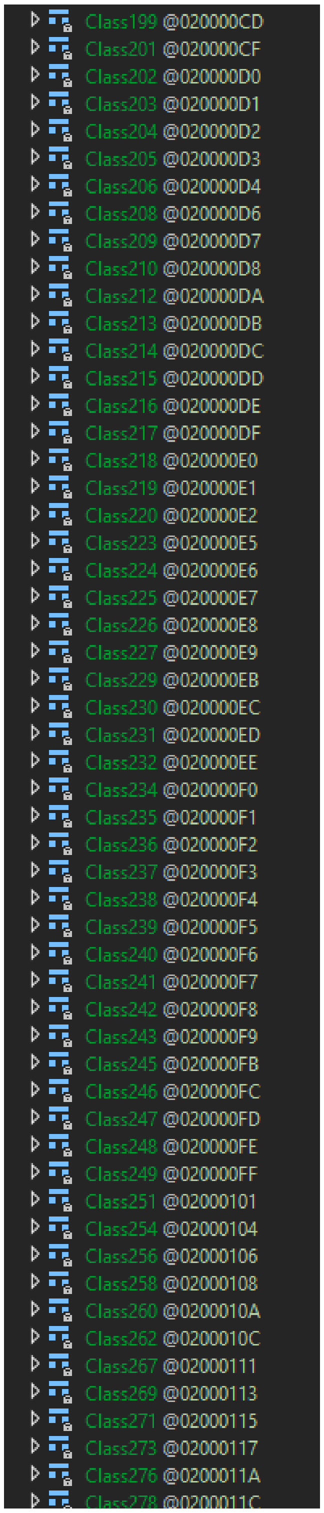


PortexAnalyzer visualization makes the base64 string visible as blue area in the byteplot.

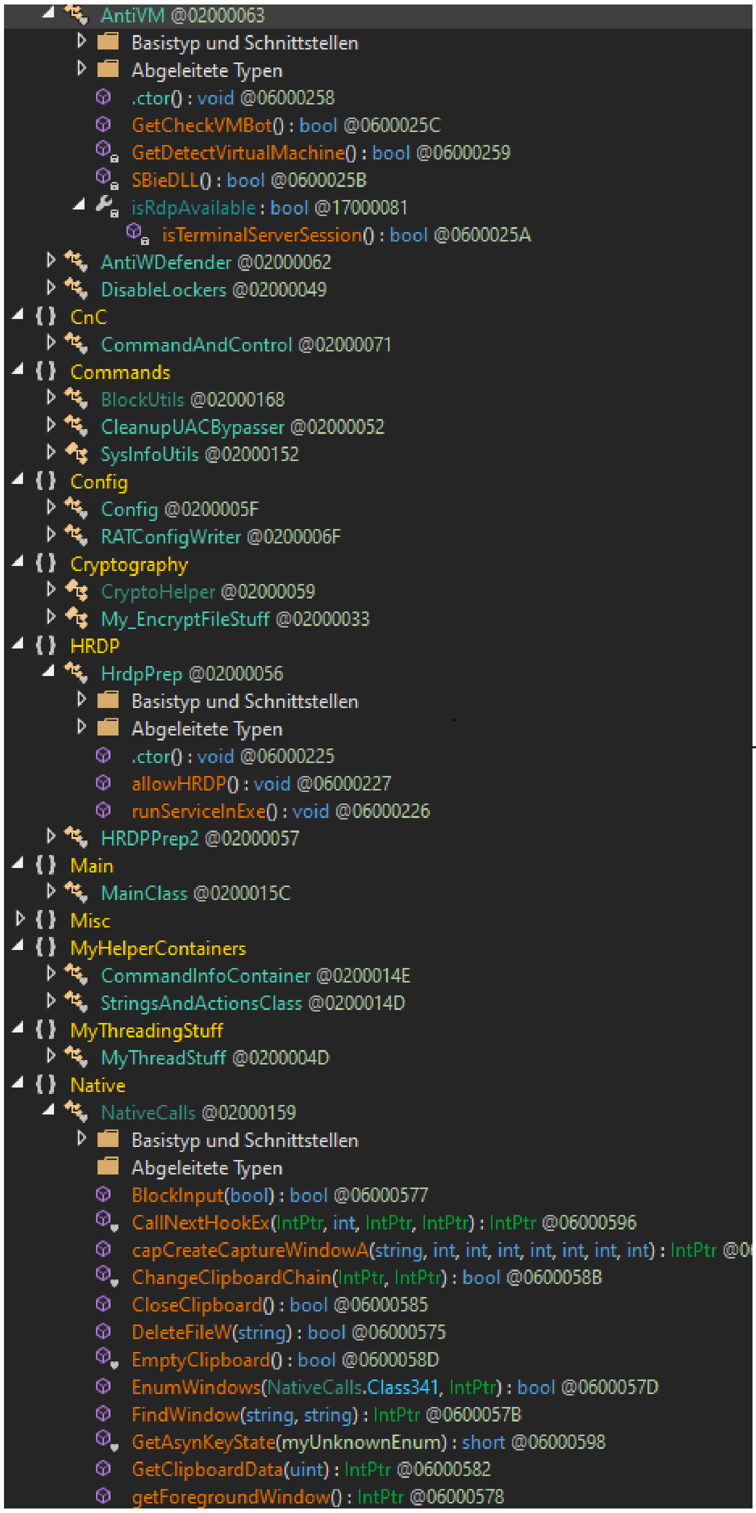
The unpacked .NET assemblies[2][4] are obfuscated with a variant of ConfuserEx. Some Russian strings are visible but most of the referenced strings are base64 encoded.

After deobfuscating the assembly with NoFuserEx, the base64 encoded strings remain. I wrote a small Python script to do the rest (see Appendix A). It replaces the IL code for calls to **FromBase64String** with NOPs and replaces the base64 strings with their decoded counterparts. Since the decoded strings are shorter, the remainder is filled with **U+200B** which is the **zero width space** Unicode character. (Sidenote: this is a rather lazy solution which does not create a perfectly working executable but is good enough for continuing static analysis.)

The most time-intensive part of deobfuscation for this assembly cannot be automated. The symbol names for methods and classes were eradicated by the obfuscator. So while analysing the code of T-RAT, I added my own names along the way. The code base is comparably on the large side with 98 different commands to control the T-RAT client.



```
Class199 @020000CD
Class201 @020000CF
Class202 @020000D0
Class203 @020000D1
Class204 @020000D2
Class205 @020000D3
Class206 @020000D4
Class208 @020000D6
Class209 @020000D7
Class210 @020000D8
Class212 @020000DA
Class213 @020000DB
Class214 @020000DC
Class215 @020000DD
Class216 @020000DE
Class217 @020000DF
Class218 @020000E0
Class219 @020000E1
Class220 @020000E2
Class223 @020000E5
Class224 @020000E6
Class225 @020000E7
Class226 @020000E8
Class227 @020000E9
Class229 @020000EB
Class230 @020000EC
Class231 @020000ED
Class232 @020000EE
Class234 @020000F0
Class235 @020000F1
Class236 @020000F2
Class237 @020000F3
Class238 @020000F4
Class239 @020000F5
Class240 @020000F6
Class241 @020000F7
Class242 @020000F8
Class243 @020000F9
Class245 @020000FB
Class246 @020000FC
Class247 @020000FD
Class248 @020000FE
Class249 @020000FF
Class251 @02000101
Class254 @02000104
Class256 @02000106
Class258 @02000108
Class260 @0200010A
Class262 @0200010C
Class267 @02000111
Class269 @02000113
Class271 @02000115
Class273 @02000117
Class276 @0200011A
Class278 @0200011C
```

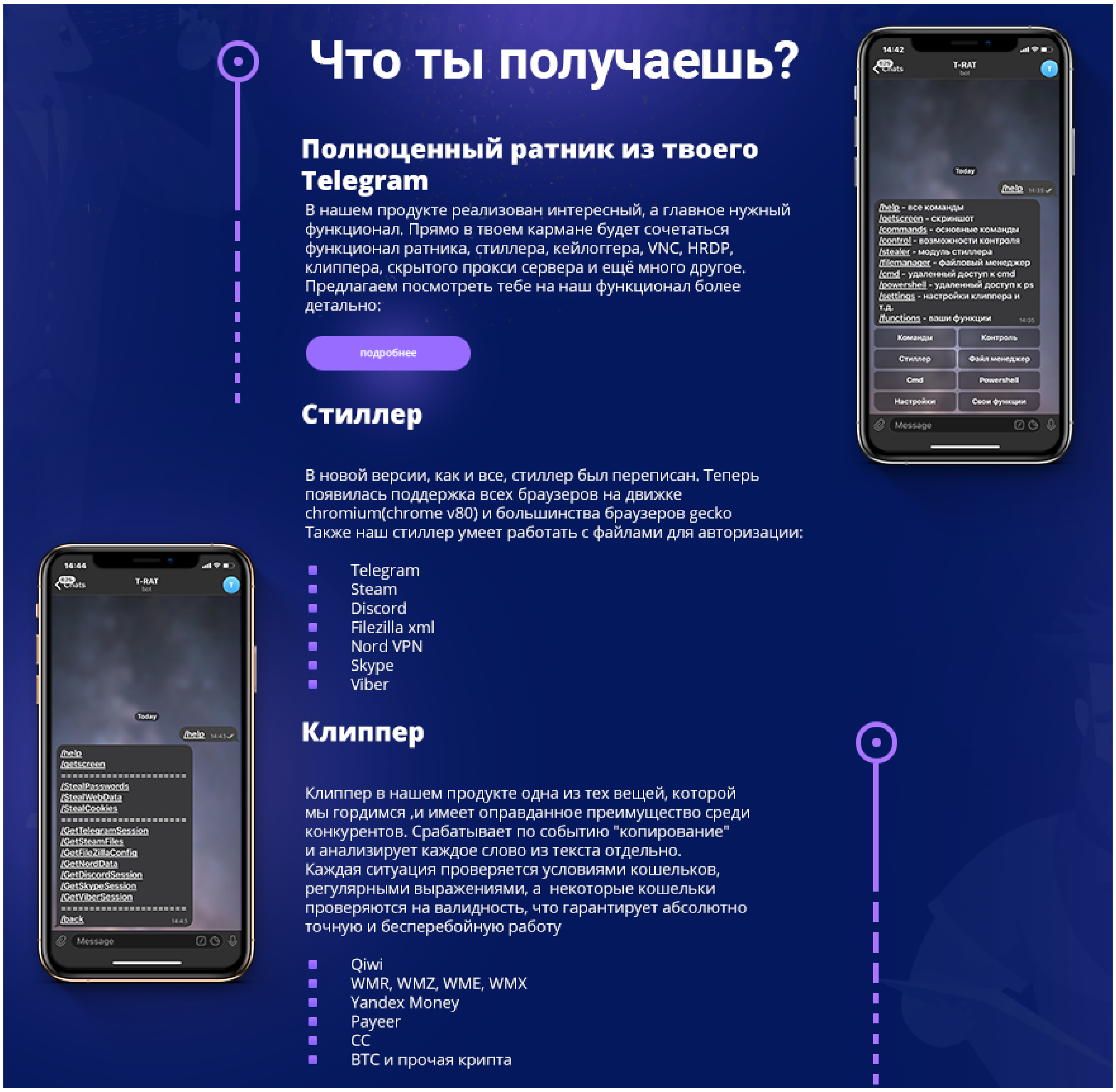


```
    AntiVM @02000063
        Basistyp und Schnittstellen
        Abgeleitete Typen
        .ctor() : void @06000258
        GetCheckVMBot() : bool @0600025C
        GetDetectVirtualMachine() : bool @06000259
        SBieDLL() : bool @0600025B
        isRdpAvailable : bool @17000081
            isTerminalServerSession() : bool @0600025A
    AntiWDefender @02000062
    DisableLockers @02000049
{ } CnC
    CommandAndControl @02000071
{ } Commands
    BlockUtils @02000168
    CleanupUACBypasser @02000052
    SysInfoUtils @02000152
{ } Config
    Config @0200005F
    RATConfigWriter @0200006F
{ } Cryptography
    CryptoHelper @02000059
    My_EncryptFileStuff @02000033
{ } HRDP
    HrdpPrep @02000056
        Basistyp und Schnittstellen
        Abgeleitete Typen
        .ctor() : void @06000225
        allowHRDP() : void @06000227
        runServiceInExe() : void @06000226
    HRDPPrep2 @02000057
{ } Main
    MainClass @0200015C
{ } Misc
{ } MyHelperContainers
    CommandInfoContainer @0200014E
    StringsAndActionsClass @0200014D
{ } MyThreadingStuff
    MyThreadStuff @0200004D
{ } Native
    NativeCalls @02000159
        Basistyp und Schnittstellen
        Abgeleitete Typen
        BlockInput(bool) : bool @06000577
        CallNextHookEx(IntPtr, int, IntPtr, IntPtr) : IntPtr @06000596
        capCreateCaptureWindowA(string, int, int, int, int, int, int, int) : IntPtr @0
        ChangeClipboardChain(IntPtr, IntPtr) : bool @0600058B
        CloseClipboard() : bool @06000585
        DeleteFileW(string) : bool @06000575
        EmptyClipboard() : bool @0600058D
        EnumWindows(NativeCalls.Class341, IntPtr) : bool @0600057D
        FindWindow(string, string) : IntPtr @0600057B
        GetAsynKeyState(myUnknownEnum) : short @06000598
        GetClipboardData(uint) : IntPtr @06000582
        getForegroundWindow() : IntPtr @06000578
```

T-RAT sample before and after manual deobfuscation

Functionality overview

The attacker controls T-RAT via Telegram using text based commands and command buttons provided by the RAT. The commands are in English, the help messages mostly Russian. One section of the advertisment banner demonstrates the controls and how they look like on the phone (see picture below).



Что ты получаешь?

Полноценный ратник из твоего Telegram

В нашем продукте реализован интересный, а главное нужный функционал. Прямо в твоем кармане будет сочетаться функционал ратника, стиллера, кейлоггера, VNC, HRDP, клиппера, скрытого прокси сервера и ещё много другое. Предлагаем посмотреть тебе на наш функционал более детально:

Стиллер

В новой версии, как и все, стиллер был переписан. Теперь появилась поддержка всех браузеров на движке chromium(chrome v80) и большинства браузеров gecko
Также наш стиллер умеет работать с файлами для авторизации:

- Telegram
- Steam
- Discord
- Filezilla xml
- Nord VPN
- Skype
- Viber

Клиппер

Клиппер в нашем продукте одна из тех вещей, которой мы гордимся ,и имеет оправданное преимущество среди конкурентов. Срабатывает по событию "копирование" и анализирует каждое слово из текста отдельно.
Каждая ситуация проверяется условиями кошельков, регулярными выражениями, а некоторые кошельки проверяются на валидность, что гарантирует абсолютно точную и бесперебойную работу

- Qiwi
- WMR, WMZ, WME, WMX
- Yandex Money
- Payeer
- CC
- BTC и прочая крипта

```
/help 14:35
/help - все команды
/getscreen - скриншот
/commands - основные команды
/control - возможности контроля
/stealer - модуль стиллера
/filemanager - файловый менеджер
/cmd - удаленный доступ к cmd
/powershell - удаленный доступ к ps
/settings - настройки клиппера и т.д.
/functions - ваши функции 14:35
```

```
/help 14:43
/help
/getscreen
=====================
/StealPasswords
/StealWebData
/StealCookies
=====================
/GetTelegramSession
/GetSteamFiles
/GetFileZillaConfig
/GetNordData
/GetDiscordSession
/GetSkypeSession
/GetViberSession
=====================
/back 14:43
```

Translation for first passage: "What do you get? A full-fledged RAT for Telegram. Our product implements interesting and most importantly necessary functionality. The functionality of a RAT, stealer, keylogger VNC, HRDP, clipper, hidden proxy server will be combined right in your pocket and so much more. We invite you to look at our functionality in more detail." The next passages explain stealer and clipper.
T-RAT has 98 commands. Instead of describing every single command within the main article, I categorized them into groups which are explained below. The full command listing is in Appendix B.

1. Menu navigation

These are commands to enter or exit certain modules like the file manager. They help to make controls via smartphone more convenient.

2. File manager

T-RAT can navigate on the file system, show information about the drives and available space, folder contents and modify files and folders. It can also send files to the attacker. Interestingly it mixes in Unix command names. E.g., the file listing is done with **ls**.

3. Stealer

This module allows to obtain passwords, cookies, autofill data from browsers, session or config data of Telegram, Discord, Steam, Nord, Viber, Skype and Filezilla. Most of the data files are either saved besides the T-RAT executable in text files or to a ZIP archive in **%TEMP%/winsys/** before being sent to Telegram.

4. Clipper

The clipper checks the clipboard for coin addresses and replaces them, thus, any digital currency is sent to the attacker's wallet. It supports Qiwi, WMR, WMZ, WME, WMX, Yandex money, Payeer, CC, BTC, BTCG, Ripple, Doge and Tron. The attackers uses the clipper commands to save their addresses for the specified crypto currency and to start or stop execution of the clipper.

5. Monitoring and spying

Enables the attacker to run a keylogger, create screenshots, record audio via the microphone, take pictures via webcam, send clipboard contents.

6. Evasion

T-RAT has various methods to bypass UAC, including Fodhelper, Cmstp, Cleanup, Computerdefaults. It can disable Windows Defender and Smart Screen notifications. It can disable various security settings, e.g., Association policies can be changed to set ".exe" as a low-risk file extension, and ZoneIdentifiers can be turned off. It has a check for sandboxes and virtual machines.

7. Disruption

These commands kill processes, block websites via the hosts file, block and redirect programs by setting a debugger via Image File Execution Options (for blocking the debugger is one that doesn't exist), disable the taskbar and the task manager.

8. Remote control

T-RAT provides a **Powershell** or **CMD** terminal via Telegram. Remote control can also be done via **HRDP** or **VNC**.

T-RAT runs the HRDP client named **service\in.exe** which resides in the executable's location. Then it will create a new user account with a randomized password and name and send the credentials to the attacker. It adds the newly created user to the **Remote Desktop Users** group and enables remote access by setting **fDenyTSConnections** to "0".

The VNC server is **service\winserv1.exe** on 32 bit systems and **service\winserv2.exe** on 64 bit systems.

Indicators of Compromise

Sample hashes

Sample	Filename	SHA256
[1] T-RAT, packed	Update Service.exe sihost.exe	dfa35a3bed8aa7e30e2f3ad0927fa69adecb5b6f4c8a8535b05c28eacbd0dad8
[2] T-RAT, unpacked from [1]	NA	0388c08ae8bf8204ed609a4730a93a70612d99e66f1d700c2edfb95197ab7cc9
[3] ZIP archive containing [1][7-11]	%TEMP%/hrtghgesd .zip	9fe677aa81790414db3187bba2e159c5aafda6dc0411fbd5d4786b7e596143f3

Sample	Filename	SHA256
[4] T-RAT downloader	Update Service.exe	b6093289ff0470053bd7dde771fa3a6cd21dae99fc444bfebcd33eb153813263
[5] T-RAT downloader, unpacked from [4]	NA	e7604cc2288b27e29f1c0b2aeade1af486daee7b5c17b0478ce336dcdbeee2f1
[6] Raw download	1DJjnw.jpg %TEMP%/gfdggfd.jpg	27dcb69c1d010da7d1f359523b398e14e0af0dd5bad1a240734a31ffce8b9262
[7] Audio Player	conv.exe	96ba1d40eb85f60a20224e199c18126b160fe165e727b7dee268890dc5148c68
[8] RDP Wrap	in.exe	ac92d4c6397eb4451095949ac485ef4ec38501d7bb6f475419529ae67e297753
[9] VNC Server	winserv1.exe	c1316ac68d5f3f5ec080d09ffc7c52670a7c42672f0233b9ef50e4b739bd0586
[10] VNC Server	winserv2.exe	912913d897dd2f969fbcbdb54dde82e54f287ade97725380232dce664417c46c
[11] Ultra VNC Hooks DLL	vnchooks.dll	c8164ccc0cf04df0f111d56d7fb717e6110f8dee77cfc3ef37507f18485af04d

IoCs for downloader[4]

Download URL	hxxps://hgfhhdsf.000webhostapp.com/1DJjnw.jpg
Download location	%TEMP%/gfdggfd.jpg
Decrypted download	%TEMP%/hrtghgesd.zip
Mutex	dwm
Scheduled task	for sihost.exe[1], task name is the processor ID of infected system

IoCs for T-RAT[1]

File name	sihost.exe
Mutex	srvhost
Creates processes	winserv1.exe, winserv2.exe, in.exe
IFEO Debugger	fghdshdzfhgsdfh.exe
User account on system	usr[1000-10000], e.g., usr3432
Data folder	%TEMP%/winsys/

Appendix A: Deobfuscation script

```
#!/usr/bin/env python2.7
import re
import base64
import sys
import os
import argparse
from shutil import copyfile
def isBase64(s): try: return base64.b64encode(base64.b64decode(s)) == s except Exception: return False
def searchAndReplace(search, replace, binfile): content = ""    with open(binfile,"rb") as bif: content =
bif.read()      new_content = content.replace(search, replace)  if new_content == content:      print
"Search string not found."      return  with open(binfile,"wb+") as wif:        wif.write(new_content)
if __name__ == "__main__":      parser = argparse.ArgumentParser(description='Decode and replace base64
strings in binary. Karsten Hahn @ G DATA CyberDefense') parser.add_argument('str_listing', help='Text file
with strings listing of sample. E.g. use Sysinternals strings.exe')     parser.add_argument('sample',
help='Sample file where base64 strings should be replaced')     args = parser.parse_args()      inputfile =
args.sample     outputfile = args.sample + ".decoded"   copyfile(inputfile, outputfile) base64Regex =
re.compile(r'^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?$') str_listing =
args.str_listing        with open(str_listing) as ref_file:     print 'Extracting base64 strings...'
base_strings = []       for line in ref_file:   base_strings += base64Regex.findall(line)       print
"Replacing base64 strings..."   for base_str in sorted(base_strings, key=len, reverse=True):    if
len(base_str) > 3 and isBase64(base_str):       decoded_string = base64.b64decode(base_str)
decoded_bytes = bytearray(str(decoded_string).decode('utf-8').encode("utf-16le"))       base_bytes =
bytearray(str(base_str).decode('utf-8').encode("utf-16le"))     while len(decoded_bytes) < len(base_bytes):
decoded_bytes.extend(b'\x0B\x20')       #print decoded_bytes    searchAndReplace(base_bytes, decoded_bytes,
outputfile)     print "Replacing calls to decode Base64..."     # Optional: remove calls to Base64
conversion, this is specific to the sample      # for T-RAT     # searchAndReplace(b'\x28\x27\x00\x00\x0A',
b'\x00\x00\x00\x00\x00', outputfile)    # for T-RAT downloader  # searchAndReplace(b'\x28\x17\x00\x00\x0A',
b'\x00\x00\x00\x00\x00', outputfile)    print 'All done'        print 'Deobfuscated file written to',
outputfile
```

Appendix B: T-RAT Commands

These are all T-RAT 2.0 commands and a description for some of them.

Command	Description
/help	Print available commands (shows different commands depending on the state of the menu)
/getscreen	Takes a screenshot and sends as photo to Telegram
/webcam	Takes a picture using the webcam and sends as photo to Telegram
/record	Records audio using the microphone. Saves it to record.wav in the executable's folder.
/sysinfo	Shows: username, IP, MAC, computername, processor model, number of cores, processor size, graphics card model, RAM, operating system, architecture, system directory, antivirus, firewall, drive info and available space
/isadmin	Checks if executable has admin rights
/activewindow	
/openwindows	
/programs	Shows list of installed programs by obtaining all DisplayName values for all subkeys of SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
/processlist	
/killprocess [process]	

Command	Description
/run [path]	Creates a hidden folder in %TEMP% named **winsys**. Puts a VBScript file named <random_digits>.vbs in this folder. The VBScript uses ShellExecute to run the file given in [path] parameter. It uses a template called "Run2" in the resources for the VBScript and replaces "lmao" with [path]
/clipboard	Posts clipboard content to Telegram
/location	
/path	
/blocksite [example google.com]	Blocks a site via hosts file redirection to localhost
/redirectprogram [first] [second]	Sets the second parameter as debugger for the first via Image File Execution Options (IFEO)
/blockprogram [name] [block|unblock]	**block:** Sets a non-existant debugger ("fghdshdzfhgsdfh.exe") for the program via Image File Execution Options (IFEO) **unblock:** Removes the IFEO debugger from registry
/CmstpUACBypass	UAC bypass via cmstp.exe
/CleanupUACBypass	UAC bypass via SilentCleanup
/FodHelperUACBypass	UAC bypass via fodhelper.exe
/ComputerDefUACBypass	UAC bypass via computerdefaults.exe
/OffCertChecking	In Attachment Policies sets: HideZoneInfoOnProperties to "1" and SaveZoneInformation to "2" (= Off) In Associations Policies sets: DefaultFileTypeRisk to "6152" (= Low) and LowRiskFileTypes to ".exe" (yes, only ".exe")
/DisableWindowsDefender	Disables TamperProtection; enables DisableAntiSpyware, DisableBehaviorMonitoring, DisableOnAccessProtection and DisableScanOnRealtimeEnable
/OffAvNotification	Disables SmartScreen and sets registry values to "0" for: EnableLUA, ConsentPromptBehaviorAdmin, PromptOnSecureDesktop
/cmd	Provides a remote cmd terminal
/powershell	Provides a remote powershell console
/settings	
/disconnect	
/opencd	Calls **mciSendStringA** with "set cdaudio door open"
/closecd	Calls **mciSendStringA** with "set cdaudio door closed"
/exploreroff	Sets DisableTaskMgr to "1"
/exploreron	Deletes subkey tree for Software\Microsoft\Windows\CurrentVersion\Policies\System
/hidetaskbar	Calls user32.dll **ShowWindow** with **SW_HIDE** parameter

Command	**Description**
/showtaskbar	Calls user32.dll **ShowWindow** for **Shell_TrayWnd**
/wallpaper	Asks the user to send a picture to set as wallpaper
/collapsewindows	
/reboot	
/kill	
/suicide	
cd [directory]	Sets working directory
back	Goes one step back in the command listing
ls	
drives	
action [name]	Provides file operations: info, run, delete, read, send, cd
mkdir [NameFileInFolder]	Creates a directory
remove [NameFileInFolder], [AnotherDirectory]	
rename [NameFileInFolder], [NewName]	
/hrdp	1) Runs service\in.exe from executable folder. 2) Sets fDenyTSConnections to "0" 3) Creates new user account named **usr<rand_nr1000-10000>** with password **<rand_nr10000-20000>** 4) Adds new user to Remote Desktop Users group 5) Prints credentials for new user to Telegram
/StartProxyServer	Starts a Socks5 proxy using port 5901
/StopProxyServer	Stops above proxy
/StartVNC	Runs service\winserv1.exe for 32 bit architecture, or service\winserv2.exe for 64 bit architecture. Both reside in the executable folder.
/StopVNC	Kills any process with a name containing the substring **winserv1** (32 bit)or **winserv2** (64 bit)
/CheckVNC	Returns if a process name containing **winserv1** or **winserv2** exists
/commands	Menu navigation
/control	Menu navigation
/stealer	Menu navigation
/filemanager	Menu navigation
/StealPasswords	
/StealWebData	Searches for **Web Data** folder in the %LOCALAPPDATA% directory and extracts autofill information. This folder is part of Chrome.

Command	Description
/StealCookies	Saves cookies to **Cookies.txt** in the executable folder and uploads it to Telegram
/GetTelegramSession	Steal Telegram data
/GetSteamFiles	Steal Steam data
/GetNordData	Steal Nord data
/GetFilezillaConfig	Steal Filezilla configuration
/GetSkypeSession	Saves skype appdata folder contents to %TEMP%/winsys/**Skype.zip** and uploads this file to Telegram
/GetDiscordSession	Saves Discord\Local Storage\leveldb folder contents to %TEMP%/winsys/**Discord.zip** and uploads this to Telegram
/GetViberSession	Steal Viber data
/SetQiwi [wallet]	Set Qiwi wallet for clipper
/SetWMR [wallet]	Set WMR wallet for clipper
/SetWMZ [wallet]	Set WMZ wallet for clipper
/SetWME [wallet]	Set WME wallet for clipper
/SetWMX [wallet]	Set WMX wallet for clipper
/SetYandexMoney [wallet]	Set Yandex Money wallet for clipper
/SetCC [wallet]	Set CC wallet for clipper
/SetPayeer [wallet]	Set Payeer wallet for clipper
/SetRipple [wallet]	Set Ripple wallet for clipper
/SetDogechain [wallet]	Set Doge wallet for clipper
/SetTron [wallet]	Set Tron wallet for clipper
/SetBTCG [wallet]	Set BTCG wallet for clipper
/SetBTC [wallet]	Set BTC waller for clipper
/wallets	
/SaveConfig	
/SendConfig	
/StartScreenLogger	
/StartKeyLogger	
/SendLog	
/StopKeyLogger	
/SendScreenshots	
/StopScreenLogger	
/ClipperStart	

Command	Description
/ClipperStop	
/ClipboardLoggerStart	
/ClipboardLoggerSend	
/ClipboardLoggerStop	
/clipboard	
/functions	
/exit	Menu navigation