Sneaky DoubleFinger loads GreetingGhoul targeting your cryptocurrency

securelist.com/doublefinger-loader-delivering-greetingghoul-cryptocurrency-stealer/109982/

Authors

- Expert GReAT
- Expert Sergey Lozhkin

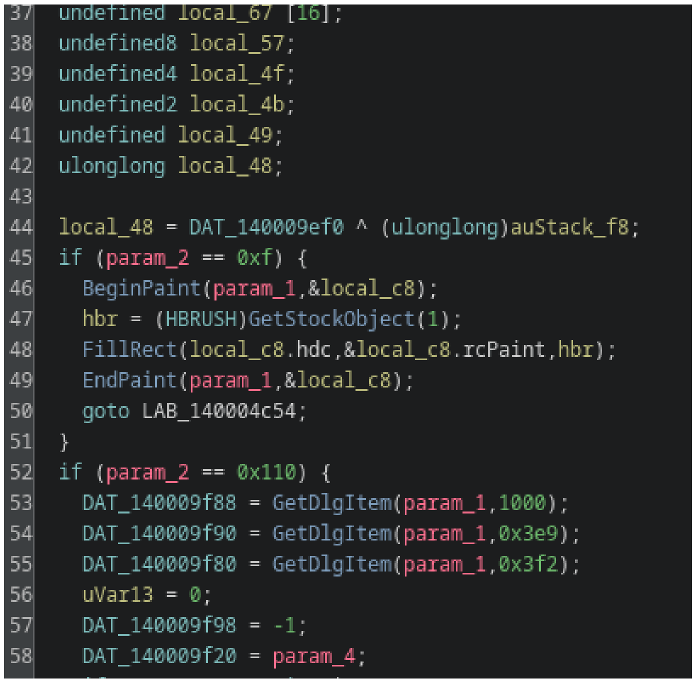
Introduction

Stealing cryptocurrencies is nothing new. For example, the Mt. Gox exchange was robbed of many bitcoins back in the beginning of 2010s. Attackers such as those behind the Coinvault ransomware were after your Bitcoin wallets, too. Since then, stealing cryptocurrencies has continued to occupy cybercriminals.

One of the latest additions to this phenomenon is the multi-stage DoubleFinger loader delivering a cryptocurrency stealer. DoubleFinger is deployed on the target machine, when the victim opens a malicious PIF attachment in an email message, ultimately executing the first of DoubleFinger's loader stages.

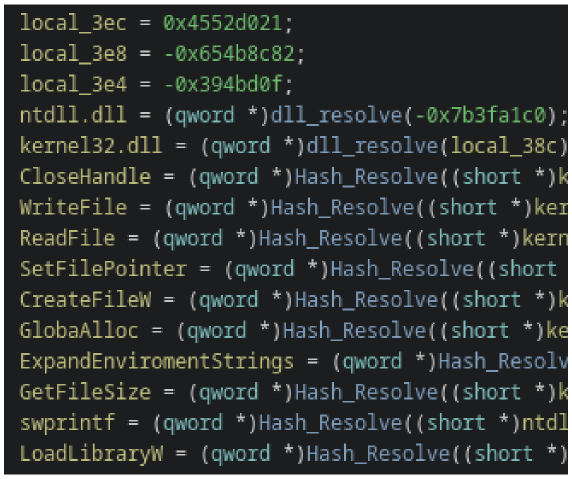
DoubleFinger stage 1

The first stage is a modified "espexe.exe" (MS Windows Economical Service Provider Application) binary, where the DialogFunc is patched so that a malicious shellcode is executed. After resolving API functions by hash, which were added to DialogFunc, the shellcode downloads a PNG image from Imgur.com. Next, the shellcode searches for the magic bytes (0xea79a5c6) in the downloaded image, locating the encrypted payload within the image.



```
undefined local_67 [16];
undefined8 local_57;
undefined4 local_4f;
undefined2 local_4b;
undefined local_49;
ulonglong local_48;

local_48 = DAT_140009ef0 ^ (ulonglong)auStack_f8;
if (param_2 == 0xf) {
  BeginPaint(param_1,&local_c8);
  hbr = (HBRUSH)GetStockObject(1);
  FillRect(local_c8.hdc,&local_c8.rcPaint,hbr);
  EndPaint(param_1,&local_c8);
  goto LAB_140004c54;
}
if (param_2 == 0x110) {
  DAT_140009f88 = GetDlgItem(param_1,1000);
  DAT_140009f90 = GetDlgItem(param_1,0x3e9);
  DAT_140009f80 = GetDlgItem(param_1,0x3f2);
  uVar13 = 0;
  DAT_140009f98 = -1;
  DAT_140009f20 = param_4;
```



```
local_3ec = 0x4552d021;
local_3e8 = -0x654b8c82;
local_3e4 = -0x394bd0f;
ntdll.dll = (qword *)dll_resolve(-0x7b3fa1c0);
kernel32.dll = (qword *)dll_resolve(local_38c)
CloseHandle = (qword *)Hash_Resolve((short *)k
WriteFile = (qword *)Hash_Resolve((short *)ker
ReadFile = (qword *)Hash_Resolve((short *)kern
SetFilePointer = (qword *)Hash_Resolve((short
CreateFileW = (qword *)Hash_Resolve((short *)k
GlobaAlloc = (qword *)Hash_Resolve((short *)ke
ExpandEnviromentStrings = (qword *)Hash_Resolv
GetFileSize = (qword *)Hash_Resolve((short *)k
swprintf = (qword *)Hash_Resolve((short *)ntdl
LoadLibraryW = (qword *)Hash_Resolve((short *)
```

Real DialogFunc function (left) and patched function with shellcode (right)

The encrypted payload consists of:

1. A PNG with the fourth-stage payload;
2. An encrypted data blob;
3. A legitimate java.exe binary, used for DLL sideloading;
4. The DoubleFinger stage 2 loader.

DoubleFinger stage 2

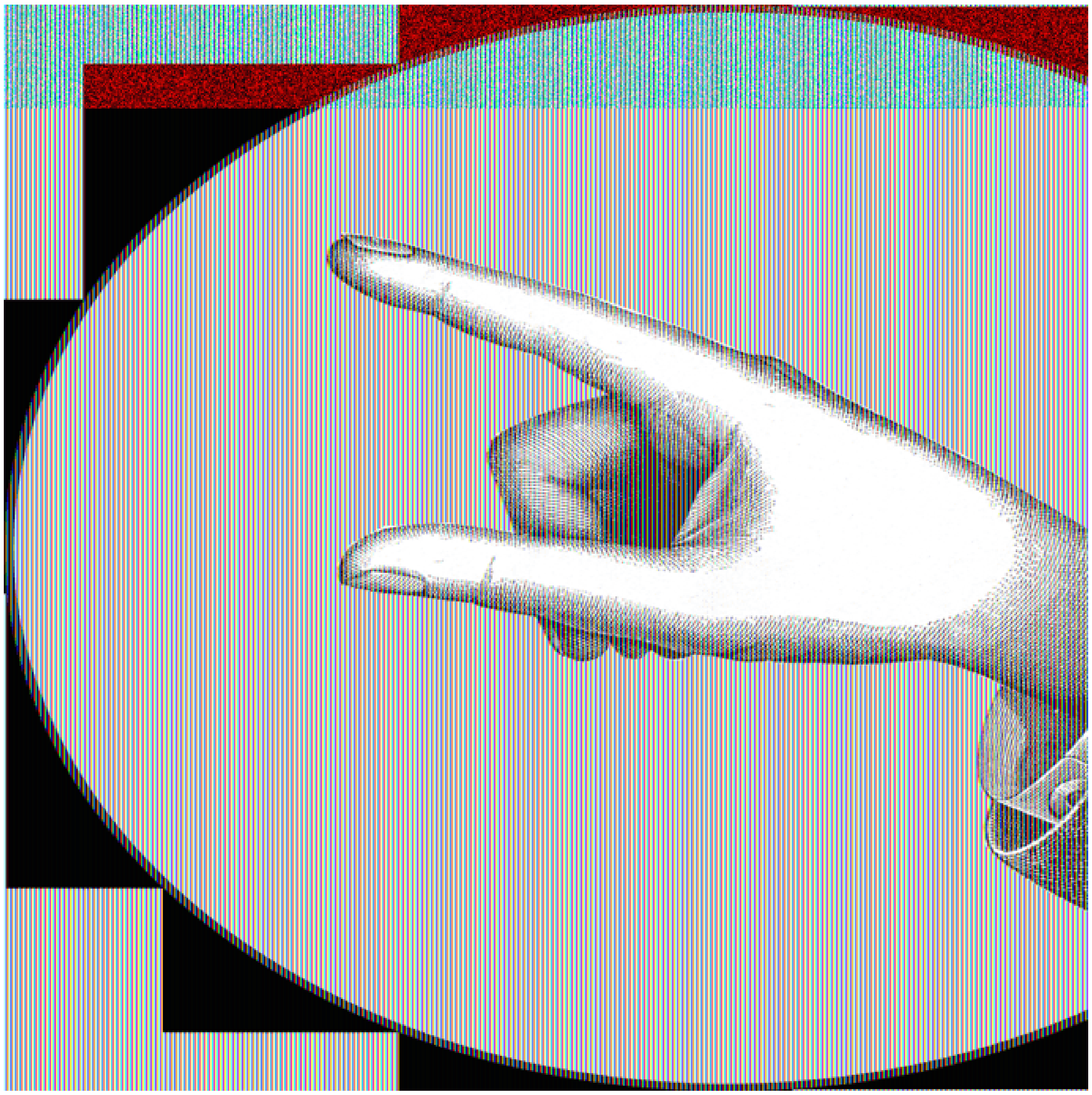
The second-stage shellcode is loaded by executing the legitimate Java binary located in the same directory as the stage 2 loader shellcode (the file is named msvcr100.dll). Just as the first stage, this file is a legitimate patched binary, having similar structure and functionality as the first stage.

To no one's surprise, the shellcode loads, decrypts and executes the third stage shellcode.

DoubleFinger stage 3

The third-stage shellcode differs greatly from the first and second stages. For example, it uses low-level Windows API calls, and ntdll.dll is loaded and mapped in the process memory to bypass hooks set by security solutions.

Next step is to decrypt and execute the fourth-stage payload, located in the aforementioned PNG file. Unlike the downloaded PNG file, which does not display a valid image, this PNG file does. The steganography method used is, however, rather simple, as the data is retrieved from specific offsets.



The aa.png file with embedded Stage 4

DoubleFinger stage 4

The stage 4 shellcode is rather simple. It locates the fifth stage within itself and then uses the Process Doppelgänging technique to execute it.

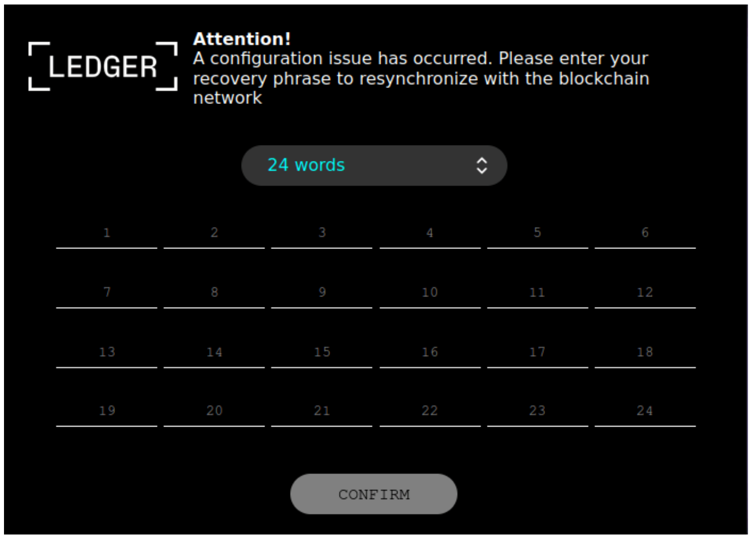
DoubleFinger stage 5

The fifth stage creates a scheduled task that executes the GreetingGhoul stealer every day at a specific time. It then downloads another PNG file (which is actually the encrypted GreetingGhoul binary prepended with a valid PNG header), decrypts it and then executes it.

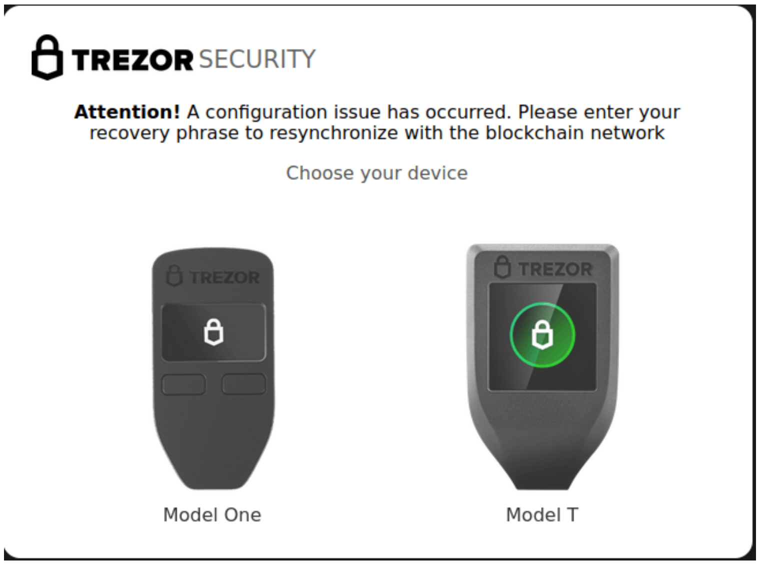
GreetingGhoul & Remcos

GreetingGhoul is a stealer designed to steal cryptocurrency-related credentials. It essentially consists of two major components that work together:

1. A component that uses MS WebView2 to create overlays on cryptocurrency wallet interfaces;
2. A component that detects cryptocurrency wallet apps and steals sensitive information (e.g. recovery phrases).



Attention!
A configuration issue has occurred. Please enter your recovery phrase to resynchronize with the blockchain network



Attention! A configuration issue has occurred. Please enter your recovery phrase to resynchronize with the blockchain network

Examples of fake windows

With hardware wallets, a user should never fill their recovery seed on the computer. A hardware wallets vendor will never ask for that.

Next to GreetingGhoul we also found several DoubleFinger samples that downloaded the Remcos RAT. Remcos is a well-known commercial RAT often used by cybercriminals. We've seen it being utilized in targeted attacks against businesses and organizations.

Victims & Attribution

We found several pieces of Russian text in the malware. The first part of the C2 URL is "Privetsvoyu" which is a misspelled transliteration of the Russian word for "Greetings." Secondly, we found the string "salamvsembratyamyazadehayustutlokeretodlyagadovveubilinashusferu." Despite the weird transliteration, it roughly translates to: "Greetings to all brothers, I'm suffocating here, locker is for bastards, you've messed up our area of interest."

Looking at the victims, we see them in Europe, the USA and Latin America. This is in accordance with the old adage that cybercriminals from CIS countries don't attack Russian citizens. Although the pieces of Russian text and the victimology are not enough to conclude that the ones behind this campaign are indeed from the post-Soviet space.

Conclusion

Our analysis of the DoubleFinger loader and GreetingGhoul malware reveals a high level of sophistication and skill in crimeware development, akin to advanced persistent threats (APTs). The multi-staged, shellcode-style loader with steganographic capabilities, the use of Windows COM interfaces for stealthy execution, and the implementation of Process Doppelgänging for injection into remote processes all point to well-crafted and complex crimeware. The use of Microsoft WebView2 runtime to create counterfeit interfaces of cryptocurrency wallets further underscores the advanced techniques employed by the malware.

To protect yourself against these threats, intelligence reports can help. If you want to stay up to date on the latest TTPs used by criminals, or have questions about our private reports, please contact crimewareintel@kaspersky.com.

Indicators of compromise

DoubleFinger
a500d9518bfe0b0d1c7f77343cac68d8
dbd0cf87c085150eb0e4a40539390a9a
56acd988653c0e7c4a5f1302e6c3b1c0
16203abd150a709c0629a366393994ea
d9130cb36f23edf90848ffd73bd4e0e0

GreetingGhoul
642f192372a4bd4fb3bfa5bae4f8644c
a9a5f529bf530d0425e6f04cbe508f1e

C2
cryptohedgefund[.]us

- crimeware
- Cryptocurrencies
- Cybercrime
- Financial malware
- Malware
- Malware Descriptions
- Malware Technologies
- RAT Trojan
- Russian-speaking cybercrime
- Trojan
- Trojan-stealer

Authors

- GReAT
- Sergey Lozhkin

Sneaky DoubleFinger loads GreetingGhoul targeting your cryptocurrency