DarkGate Keylogger Analysis: masterofnone

As cybercriminal threat actors evolve their tools to circumvent detection and to advance their attacks, it's critical to have experienced and well-equipped incident response firm at the ready to identify, contain and remove them from your environment.

In a recent investigation, Aon's Stroz Friedberg Incident Response Services ("Stroz Friedberg") encountered a group utilizing techniques similar to ScatteredSpider (a.k.a UNC3944, Roasted 0ktapus) and a malware called DarkGate. This blog post studies the new DarkGate string encryption functionality we identified in a more recent sample of the malware, and further describes the malware author's methodology for encrypting the keylog files created by DarkGate. Stroz Friedberg also has released a tool to decrypt these keylog files.

Background

In 2018, Fortinet observed DarkGate used in several crypto mining and ransomware campaigns. In 2020, Avast published a blog post about DarkGate in which they dub the malware "Meh". Aon's Stroz Friedberg Threat Intelligence identified an actor who started advertising DarkGate malware for sale on the dark web in May of 2023. The actor stated they had been working on this malware since 2017 and recently started renting it out. Another actor on the dark web confirmed that this malware was an earlier version of the malware analyzed by Fortinet in 2018.

The malware itself has a host of capabilities, including:

- hVNC (Hidden Virtual Network Computer) Access
- Cryptomining
- Information Stealing
- Reverse Shell Functionality
- Keylogging

Stroz Friedberg analyzed a sample of this DarkGate malware used in 2023 that was packed in a Microsoft Software Installer (MSI) file. The MSI file installed a compiled AutoIT script which had the DarkGate payload embedded inside it. This blog post focuses on the process to decrypt strings contained within the malware as well as analysis of the keylogger functionality of the DarkGate payload. To aid with investigations, Aon's Stroz Friedberg Incident Response team released two scripts:

- An IDA Python script to assist with string decryption of the DarkGate malware
- A Delphi program to decrypt corresponding keylog files

The Delphi program utilizes an AES-128 bit key, generated from the malware's key string "`masteroflog`", to decrypt the corresponding keylogger files. The string "`masteroflog`" is the same key string in this sample of DarkGate and observed by Avast in their analysis of Meh.

Technical Analysis – String Encryption Algorithm

The unpacked DarkGate malware sample that Stroz Friedberg analyzed is a 32-bit program written in Delphi. All strings related to the capabilities (I.e., hVNC, cookie theft, keylogging, etc) in the malware are encrypted with a single-byte XOR key, followed by a Base64 encoding using a custom character set. Decrypting the strings eases analysis of the DarkGate malware and allows a malware analyst to focus on important functionality of the malware.

The typical Base64 alphabet utilizes the standard character set of:

`ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=`

In this version of DarkGate, the malware utilizes the following custom character set to encode their encrypted strings:

`BHUPY4TaCANpsdt2zx9yXEnVkmcgl58wGbIRWSreJuKQO7fqLF3i0voZD+j1M6h=`

When decrypting the encrypted strings, the malware moves the address of the encrypted and encoded string into the `EAX` register before calling a function that uses the custom Base64 alphabet. This `MOV` and `CALL` can be seen at `0x469B1C` and `0x469B21` below.

```
CODE:00469AE8 loc_469AE8:                             ; CODE XREF: start+D↓j
CODE:00469AE8                 push    0
CODE:00469AEA                 push    0
CODE:00469AEC                 dec     ecx
CODE:00469AED                 jnz     short loc_469AE8
CODE:00469AEF                 push    ecx
CODE:00469AF0                 mov     eax, offset dword_469790
CODE:00469AF5                 call    sub_406968
CODE:00469AFA                 xor     eax, eax
CODE:00469AFC                 push    ebp
CODE:00469AFD                 push    offset loc_469D94
CODE:00469B02                 push    dword ptr fs:[eax]
CODE:00469B05                 mov     fs:[eax], esp
CODE:00469B08                 call    sub_402E84
CODE:00469B0D                 lea     eax, [ebp+var_14]
CODE:00469B10                 call    sub_45210C
CODE:00469B15                 mov     eax, [ebp+var_14]
CODE:00469B18                 push    eax
CODE:00469B19                 lea     edx, [ebp+var_1C]
CODE:00469B1C                 mov     eax, offset aJkiiug9m ; "JKIIug9M"
CODE:00469B21                 call    sub_4576E4
CODE:00469B26                 mov     eax, [ebp+var_1C]
CODE:00469B29                 lea     ecx, [ebp+var_18]
CODE:00469B2C                 mov     edx, offset aBbys ; "BbYs"
CODE:00469B31                 call    sub_45CE64
```

Call to the custom Base64 wrapper function at 0x4576E4. Tool used in screenshot is IDA Pro.

Reviewing the call to `0x4576E4` shows that it is a wrapper of the function `0x433260`, which implements the Base64 algorithm with the custom alphabet. A subset of the code from `sub_433260`, in which the malware uses the custom Base64 alphabet, can be found below.

```
CODE:004332BE loc_4332BE:                          ; CODE XREF: sub_433260+57↑j
CODE:004332BE                 lea     eax, [ebp+var_18]
CODE:004332C1                 mov     edx, [ebp+var_4]
CODE:004332C4                 mov     dl, [edx+esi-1]
CODE:004332C8                 call    sub_4046E8
CODE:004332CD                 mov     eax, [ebp+var_18]
CODE:004332D0                 mov     edx, ds:off_47013C ; "BHUPY4TaCANpsdt2zx9yXEnVkmcgl58wGbIRWSr"...
CODE:004332D6                 call    sub_404AC4
CODE:004332DB                 mov     [ebp+var_C], eax
CODE:004332DE                 cmp     [ebp+var_C], 1
CODE:004332E2                 jge     short loc_4332EB
CODE:004332E4                 mov     [ebp+var_C], 1
```

The setup of the custom Base64 alphabet.

After the custom Base64 decoding is complete, the string is sent to a decryption function. This function accepts a buffer to the decoded string along with a key. The call to this function can be seen at `0x469B31` in the figure below. The key in this example is "`BbYs`", although each string in the sample has its own decryption key.

```
CODE:00469B26                 mov     eax, [ebp+var_1C]
CODE:00469B29                 lea     ecx, [ebp+var_18]
CODE:00469B2C                 mov     edx, offset aBbys ; "BbYs"
CODE:00469B31                 call    sub_45CE64
```

The call to the decryption function with the key "BbYs"

The malware decrypts the decoded string using a single-byte XOR key. The XOR key itself is generated in code, using a single byte rolling XOR algorithm, using the following process:

1. It performs an XOR operation of the length of the key and the first byte of the input buffer.
2. It then takes the resulting byte and performs an XOR with the second byte of the decoded string.
3. This continues until the end of the string is reached.
4. The final resulting byte is used as the XOR key to decrypt the fully decoded string.

The above methodology is used for each string in the binary. Stroz Friedberg has released an IDA Python script to extract the arguments to both the custom Base64 wrapper function and the decryption function to recover the plaintext strings from the binary.

Technical Analysis – Keylogger Output Encryption

Upon successful execution, the malware begins to record various details from the system, including keystrokes, clipboard data, and process information. After it encrypts those details, the malware writes them to a file located at:

```
C:\ProgramData\<random 7-character string>\<random 7 character string>\DD-MM-YYYY.log
```

The malware stores collected information in memory and writes the data out to the above file every 30 seconds. To encrypt the key log output, the malware relies on the Delphi library DCPCrypt, a library which implements various cryptographic function for Delphi programs. The library is used to create a block cipher by passing it a key string and a hashing algorithm for generation of the cipher. The Initialization Vector (IV), another component of AES Encryption, is created by the DCPCrypt library automatically based off the key string and hashing algorithm.

In the case of the DarkGate malware, the code implements the DCPrijndael block cipher class initialized with the key string “`masteroflog`” and the hashing algorithm of SHA1. This can be seen in the figure below. The malware calls blockcipher_create with an argument of the DCPrijndael structure. Finally, it passes the DCPrijndael class, the DCP_sha1 constant, and the key string as arguments to the InitStr function of the DCPrijndael class. The InitStr function is where DCPCrypt generates the AES-128 bit key and the IV.

```
CODE:0042170C                 mov     eax, VMT_41F498_TDCP_rijndael
CODE:00421711                 call    TDCP_blockcipher_Create ; 'TDCP_rijndael.Create'
CODE:00421716                 mov     [esi], eax
CODE:00421718
CODE:00421718 loc_421718:                             ; CODE XREF: InitStrWrapper+26↑j
CODE:00421718                 mov     al, [ebp+DCPHash_const_Copy]
CODE:0042171B                 sub     al, 2
CODE:0042171D                 jnz     short loc_42172F
CODE:0042171F                 mov     eax, [esi]      ; TDCP_cipher_funcs_arg
CODE:00421721                 mov     ecx, VMT_420164_TDCP_sha1 ; TCDP_sha1_arg
CODE:00421727                 mov     edx, [ebp+string] ; key_arg
CODE:0042172A                 call    TDCP_Cipher_InitStr
```

Initialization of DCPrijndael class with a SHA1 algorithm.

Prior to the call to InitStr, the malware decrypts the key string, “masteroflog” using the procedure described in the first section of this blog post. The decryption of this key string can be seen in the figure below.

```
CODE:0045C05B                 mov     eax, offset aWHdiufswmi9wmw ; "W+hdIufsWmI9WmW"
CODE:0045C060                 call    decode_custom_base64_wrap ; bytearray(b'masteroflog')
CODE:0045C065                 mov     eax, dword ptr [ebp+buff] ; encbuffer
CODE:0045C068                 lea     ecx, [ebp+xor_key_ptr] ; xor_key
CODE:0045C06B                 mov     edx, offset aChflec ; "chfLEc"
CODE:0045C070                 call    decrypt_string
```

Decrypted version of the string “masteroflog”.

The malware then utilizes the EncryptString and DecryptString functions of the DCPrijndael class to encrypt and decrypt the contents of the key log file. The EncryptString and DecryptString functions from the DCPCrypt library return and take encrypted Base64 strings as seen in the figure below.

```
    function EncryptString(const Str: UnicodeString): UnicodeString; overload; virtual;
      { Encrypt a Unicode string and return Base64 encoded }
    function DecryptString(const Str: UnicodeString): UnicodeString; overload; virtual;
      { Decrypt a Base64 encoded Unicode string }
```

Comments in the DCPCrypt library explaining functionality of EncryptString and DecryptString calls.

On the backend of these functions, they implement the string encryption/decryption using the cipher in CFB8Bit mode. Some sample Delphi code has been provided below which can be leveraged to setup the cipher and decrypt a Base64 string.

```
KeyStr := 'masteroflog';
Base64String := '<encrypted Base64 string>'
Cipher :=  TDCP_rijndael.Create(nil);
Cipher.InitStr(KeyStr, TDCP_sha1);
Cipher.DecryptString(Base64String);
```

Tool Release

Aon's Stroz Friedberg Incident Response team released code that's designed to decrypt the strings in the binary, as well as a released a Delphi executable which implements the full decryption of key log files.

Sample Analyzed

File Name	SHA256 Hash
au3 Script	FADABBF0EF32B7295B5C0DC1830816C35BCE50EC9256D01A600D06F346A161D7
Unnamed payload	457767A1726BBC1AF05175B5A61612A4E1AD29D633E32A887E241ACCED72A006

Author: Zachary Reichert
August 2, 2023

Cyber security services offered by Stroz Friedberg Inc. and its affiliates. Insurance products and services offered by Aon Risk Insurance Services West, Inc., Aon Risk Services Central, Inc., Aon Risk Services Northeast, Inc., Aon Risk Services Southwest, Inc., and Aon Risk Services, Inc. of Florida and their licensed affiliates. Aon UK Limited is authorised and regulated by the Financial Conduct Authority in respect of insurance distribution services. FP.AGRC.238.JJ The following products or services are not regulated by the Financial Conduct Authority:

- Cyber risk services provided by Aon UK Limited and its affiliates
- Cyber security services provided by Stroz Friedberg Limited and its affiliates.