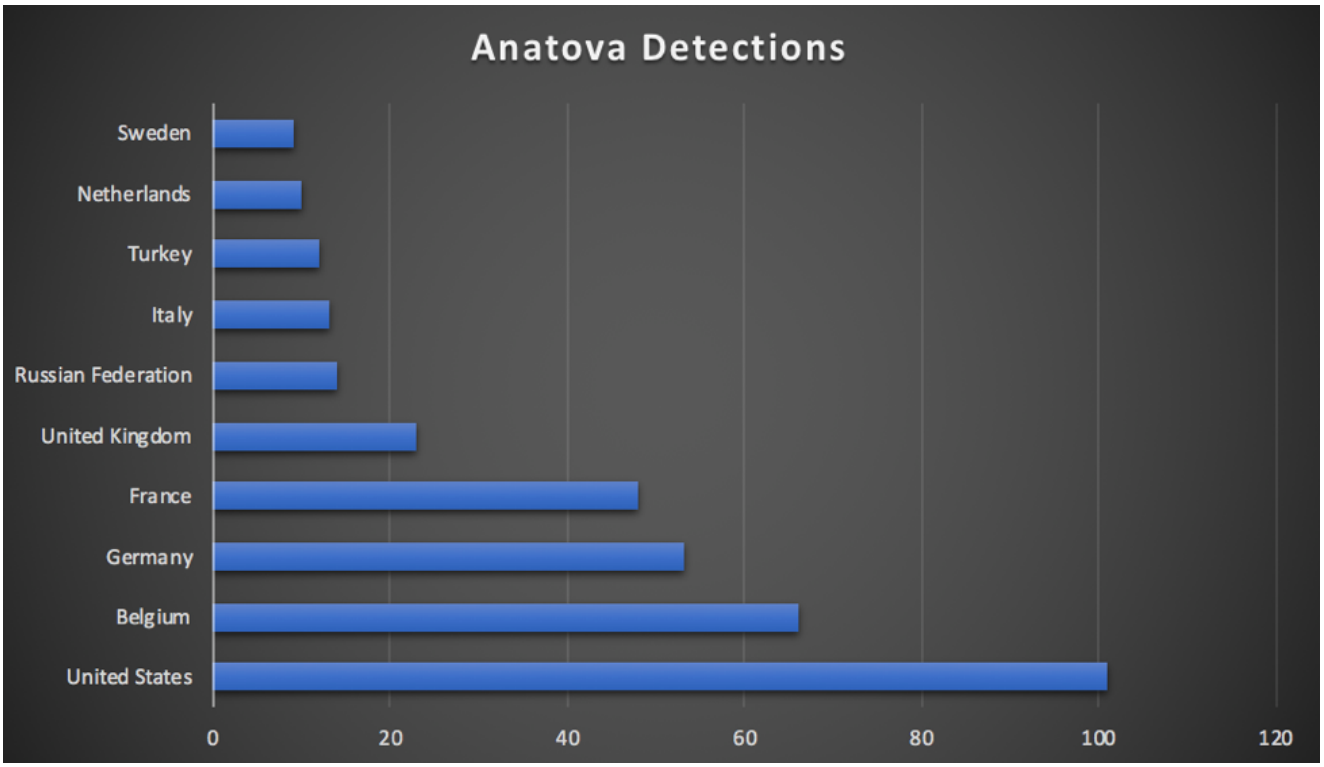
Happy New Year 2019! Anatova is here!

securingtomorrow.mcafee.com/other-blogs/mcafee-labs/happy-new-year-2019-anatova-is-here/

January 22, 2019

During our continuous hunt for new threats, we discovered a new ransomware family we call Anatova (based on the name of the ransom note). Anatova was discovered in a private peer-to-peer (p2p) network. After initial analysis, and making sure that our customers are protected, we decided to make this discovery public.



Our telemetry showed that although Anatova is relatively new, we already discovered a widespread detection of the thread around the globe

We believe that Anatova can become a serious threat since the code is prepared for modular extension.

Additionally, it will also check if network-shares are connected and will encrypt the files on these shares too. The developers/actors behind Anatova are, according our assessment, skilled malware authors. We draw this conclusion as each sample has its own unique key, as well as other functions we will describe, which we do not often see in ransomware families.

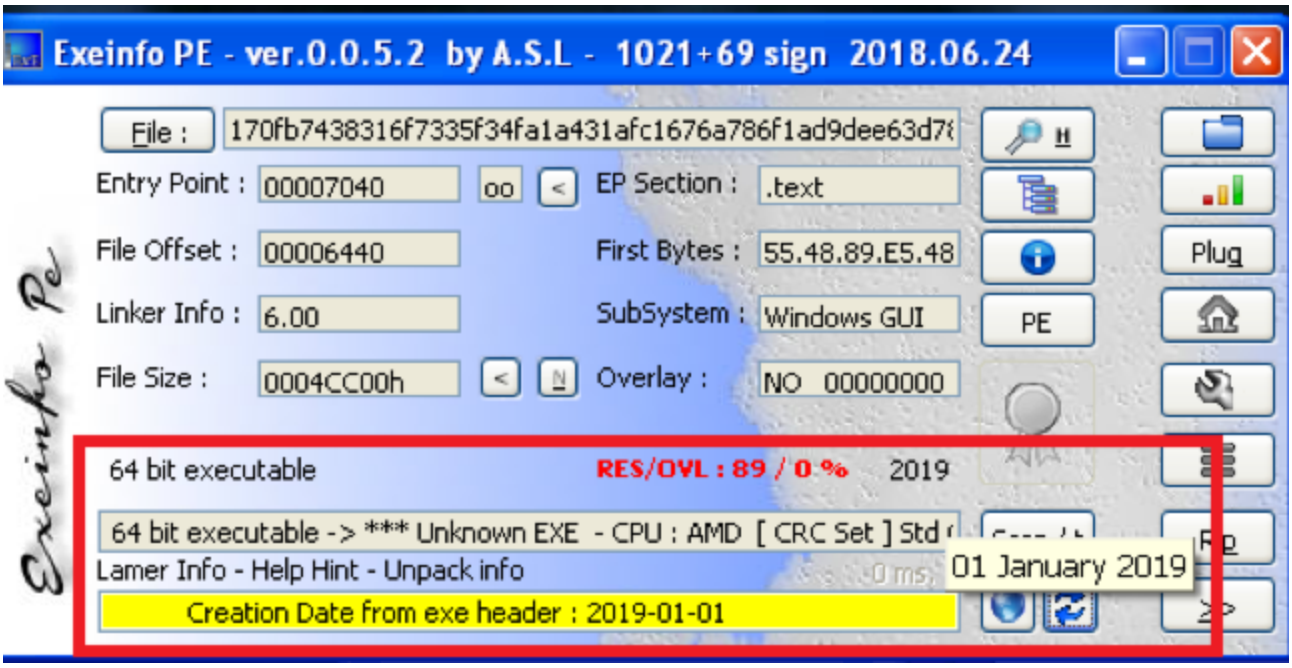
This post will explain the technical details of Anatova, as well as some interesting facts about this new ransomware family.

For the analysis we used this particular hash:
170fb7438316f7335f34fa1a431afc1676a786f1ad9dee63d78c3f5efd3a0ac0

The main goal of Anatova is to cipher all the files that it can before requesting payment from the victim.

Anatova Overview

Anatova usually uses the icon of a game or application to try and fool the user into downloading it. It has a manifest to request admin rights.



Information about the binary

The Anatova ransomware is a 64bits application with the compile date of January 1st, 2019. The file size of this particular hash is 307kb, but it can change due to the amount of resources used in the sample. If we remove all these resources, the size is 32kb; a very small program with a powerful mechanism inside.

Anatova has some strong protection techniques against static analysis which makes things slightly tricky:

- Most of the strings are encrypted (Unicode and Ascii), using different keys to decrypt them, embedded in the executable.
- 90% of the calls are dynamic;, they only use the following non-suspicious Windows API's and standard library of C- programming language: GetModuleHandleW, LoadLibraryW, GetProcAddress, ExitProcess and MessageBoxA.
- When we open the binary in IDA Pro (included the latest version of IDA) the functions are bad detected, and they finish being processed after 3 opcodes. We are not sure if this is a bug in IDA Pro or perhaps the malware authors created something to cause this on purpose (which we doubt).

```
AnatovaPrepareCheckSystemInstallLanguageIfIsBlackListed proc near
                                        ; CODE XREF: AnatovaMainFunction:_check_language↓p
                                        ; DATA XREF: .pdata:000000000040A198↓o
                push    rbp
                mov     rbp, rsp
                sub     rsp, 90h
AnatovaPrepareCheckSystemInstallLanguageIfIsBlackListed endp ; sp-analysis failed
```

Problem in IDA Pro 7.2 last version

Entry Vector

At the moment we don´t know all entry vectors that Anatova is using, or will be using, in the near future. Our initial finding location was in private p2p.

The goal of Anatova, as with other ransomware families, is to encrypt all or many files on an infected system and insist on payment to unlock them. The actor(s) demand a ransom payment in cryptocurrency of 10 DASH – currently valued at around $700 USD, a quite high amount compared to other ransomware families.

In-depth highlights of version 1.0

Since this is a novel family, we didn't find any version number inside the code, but let's call this version 1.0

The first action that the malware executes is to get the module handle of the library "kernel32.dll" and get 29 functions from it using the function "GetProcAddress".

```
mov     [rbp-10h], rax
mov     eax, 43h ; 'C'
mov     r11, rax
lea     rax, AnatovaGlobalVarStringKernel32UnicodeCrypted ; "("
mov     r10, rax
mov     rcx, r10
mov     rdx, r11
call    AnatovaPrepareDecryptUnicodeStringAndReturnPointerToTheString
mov     [rbp-10h], rax
mov     rax, [rbp-10h]
mov     r10, rax
mov     rcx, r10        ; lpModuleName
call    GetModuleHandleW
mov     [rbp-18h], rax
mov     rax, [rbp-10h]
mov     r10, rax
mov     rcx, r10        ; Memory
call    free
mov     rax, [rbp-18h]
cmp     rax, 0          ; check if have handle to kernel32.dll
jz      _exit
mov     rax, 0
mov     [rbp-20h], rax
lea     rax, aZeXymMCo  ; "Ze~xym`M``co"
mov     [rbp-108h], rax
lea     rax, aGxcedpWctt ; "Gxcedp}Wctt"
mov     [rbp-100h], rax
lea     rax, aWuhdbtt45anuts ; "Wuhdbtt45AnutsP"
```

Get kernel32 functions after decrypt strings

If the malware can´t get the module handle of kernel32, or some of the functions can´t be found, it will quit without executing any encryption.

Later, the malware will try to create a mutex with a hardcoded name (in this case: 6a8c9937zFIwHPZ309UZMZYVnwScPB2pR2MEx5SY7B1xgbruoO) but the mutex name changes in each sample. If the mutex is created, and gets the handle, it will call the "GetLastError" function and look if the last error is ERROR_ALREADY_EXISTS or ERROR_ACCESS_DENIED. Both errors mean that a previous instance of this mutex object exists. If that is the case, the malware will enter in a flow of cleaning memory, that we will explain later in this post, and finish.

```
mov     rax, cs:AnatovaGlobalVarPointerToVarWithMutexName
mov     r8, rax
mov     eax, 1
mov     r11, rax
mov     rax, 0
mov     r10, rax
mov     rcx, r10
mov     rdx, r11
mov     r11, cs:AnatovaGlobalVarCreateMutexAFunction
call    r11 ; AnatovaGlobalVarCreateMutexAFunction
mov     [rbp-18h], rax
mov     rax, [rbp-18h]
cmp     rax, 0
jz      _after_check_get_last_error
mov     r11, cs:AnatovaGlobalVarGetLastErrorFunction
call    r11 ; AnatovaGlobalVarGetLastErrorFunction
mov     [rbp-1Ch], eax
mov     eax, [rbp-1Ch]
cmp     eax, 0B7h ; '·' ; ERROR_ALREADY_EXISTS
jz      _mutex_exists
mov     eax, [rbp-1Ch]
cmp     eax, 5          ; ERROR_ACCESS_DENIED
jz      _mutex_exists
jmp     _after_check_get_last_error
```

Check mutex

After this check, Anatova will get some functions from the library "advapi32.dll", "Crypt32.dll" and "Shell32.dll" using the same procedure as in the kernel case. All text is encrypted and decrypted one per one, get the function, free the memory, and continue with the next one.

If it fails in getting some of these modules or some of the functions it needs, it will go to the flow of cleaning tool and exit.

One interesting function we discovered was that Anatova will retrieve the username of the logged in and/or active user and compare with a list of names encrypted. If one of the names is detected, it will go to the cleaning flow procedure and exit.

The list of users searched are:

- LaVirulera
- tester
- Tester
- analyst
- Analyst
- lab
- Lab
- Malware
- malware

Some analysts or virtual machines/sandboxes are using these default usernames in their setup, meaning that the ransomware will not work on these machines/sandboxes.

After this user-check, Anatova will check the language of the system. When we say language, we mean the system language. When a user installs the Windows OS, they choose a language to install it with (though later the user could install a different language). Anatova checks for the first installed language on the system to ensure that a user cannot install one of these blacklisted languages to avoid encryption of the files.

The list of the countries that Anatova doesn't affect are:

- All CIS countries
- Syria
- Egypt
- Morocco
- Iraq
- India

It's quite normal to see the CIS countries being excluded from execution and often an indicator that the authors might be originating from one of these countries. In this case it was surprising to see the other countries being mentioned. We do not have a clear hypothesis on why these countries in particular are excluded.

```
mov     eax, 442h       ; Tatar
mov     [rbp-30h], eax
mov     eax, 816h       ; Moldova
mov     [rbp-2Ch], eax
mov     eax, 817h       ; Moldova (Russian)
mov     [rbp-28h], eax
mov     eax, 82Ah       ; Azerbaijan
mov     [rbp-24h], eax
mov     eax, 841h       ; Uzbekistan
mov     [rbp-20h], eax
mov     eax, 458h       ; Syria
mov     [rbp-1Ch], eax
mov     eax, 27FFh      ; Syria (Arabic)
mov     [rbp-18h], eax
mov     eax, 437h       ; India (Hindi)
mov     [rbp-14h], eax
mov     eax, 0BFFh      ; Egypt (Arabic)
mov     [rbp-10h], eax
mov     eax, 17FFh      ; Morocco (Arabic)
mov     [rbp-0Ch], eax
mov     eax, 7FFh       ; Iraq (Arabic)
mov     [rbp-8], eax
mov     r11, cs:AnatovaGlobalVarGetSystemDefaultUILanguageFunction
call    r11 ; AnatovaGlobalVarGetSystemDefaultUILanguageFunction
mov     [rbp-60h], eax
mov     eax, 0
mov     [rbp-64h], eax
```

Check system language

After the language check, Anatova looks for a flag that, in all samples we looked at, has the value of 0, but if this flag would change to the value of 1 (the current malware samples never change that value), it will load two DLLs with the names (after decryption) of “extra1.dll” and “extra2.dll”. This might indicate that Anatova is prepared to be modular or to be extended with more functions in the near future.

```
AnatovaLoadExtraModulesFunction proc near
                                        ; DATA XREF: .pdata:000000000040A1B0↓o
                mov     rax, 0
                mov     [rbp-8], rax
                mov     eax, 0Ch
                mov     r11, rax
                lea     rax, AnatovaGlobalVarExtra1UnicodeStringCrypted ; "i"
                mov     r10, rax
                mov     rcx, r10
                mov     rdx, r11
                call    AnatovaPrepareDecryptUnicodeStringAndReturnPointerToTheString
                mov     [rbp-8], rax
                mov     rax, [rbp-8]
                mov     r10, rax
                mov     rcx, r10        ; lpLibFileName
                call    LoadLibraryW
                mov     [rbp-10h], rax
                mov     rax, [rbp-8]
                mov     r10, rax
                mov     rcx, r10
                call    AnatovaPrepareReleaseMemoryWithVirtualFree
                mov     eax, 0Ch
                mov     r11, rax
                lea     rax, AnatovaGlobalVarExtra2UnicodeStringCrypted ; "itx~m>\"h``"
                mov     r10, rax
                mov     rcx, r10
                mov     rdx, r11
                call    AnatovaPrepareDecryptUnicodeStringAndReturnPointerToTheString
```

Load extra modules

After this, the malware enumerates all processes in the system and compares them with a large list including, for example “steam.exe”, “sqlserver.exe”, etc. If some of these processes are discovered, the malware will open them and terminate them. This action is typical of ransomware that attempts to unlock files that later will be encrypted, such as database files, game files, Office related files, etc.

The next action is to create an RSA Pair of Keys using the crypto API that will cipher all strings. This function is the same as in other ransomware families, such as GandCrab or Crysis, for example. It makes sure that the keys that will be used, are per user and per execution.

If the malware can´t create the keys, it will go to the clean flow and exit.

After this, Anatova will make a random key of 32 bits and another value of 8 bytes using the function of the crypto API “CryptGenRandom” to encrypt using the Salsa20 algorithm and the private previous blob key in runtime.

During the encryption process of the files, it will decrypt the master RSA public key of the sample of 2 layers of crypto, the first one is a XOR with the value 0x55 and the second one is to decrypt it using a hardcoded key and IV in the sample using the Salsa20 algorithm.

```
_decrypt:                               ; CODE XREF: AnatovaGenerateKeyForSalsa20AndIVAndCr
                mov     eax, [rbp-28h]
                movsxd  rax, eax
                lea     rcx, AnatovaGlobalBufferWithRSAPublicMasterSampleKey
                add     rcx, rax
                movzx   eax, byte ptr [rcx]
                xor     eax, 55h        ; key for the first layer
                mov     [rcx], al
                jmp     short _decrypt_master_key_first_layer_loop
; ---------------------------------------------------------------------------
```

Decrypt from first layer the master RSA public key of sample

After this, it will import the public key and with it, will encrypt the Salsa20 key and IV used to encrypt the private RSA key in runtime.

The next step is to prepare a buffer of memory and with all of the info encrypted (Salsa20 key, Salsa20 IV, and private RSA key). It makes a big string in BASE64 using the function “CryptBinaryToStringA”. The ransomware will later clean the computer’s memory of the key, IV, and private RSA key values, to prevent anyone dumping this information from memory and creating a decrypter.

This BASE64 string will be written later in the ransom note. Only the malware authors can decrypt the Salsa20 key and IV and the private RSA key that the user would need to decrypt the files.

If this does not work, Anatova will delete itself, enter in the clean flow and exit.

When the keys are encrypted in the memory buffer, Anatova will enumerate all logic units and will search for all existing instances of the type DRIVE_FIXED (a normal hard disk for example) or DRIVE_REMOTE (for remote network shares that are mounted). Anatova will try to encrypt the files on each of those locations. This means that one corporate victim can cause a major incident when files on network-shares are being encrypted.

```
                jz      _release_memory_and_exit

_check_if_units_exists:                 ; CODE XREF: AnatovaGetAllLogicalUnitsAnd
                mov     eax, [rbp-14h]
                cmp     eax, 0
                jz      _release_memory_and_exit
                mov     eax, [rbp-14h]
                and     eax, 1
                cmp     eax, 0
                jz      _check_if_need_check_more_units
                mov     rax, [rbp-10h]
                mov     r10, rax
                mov     rcx, r10
                mov     r11, cs:AnatovaGlobalVarGetDriveTypeWFunction
                call    r11 ; AnatovaGlobalVarGetDriveTypeWFunction
                mov     [rbp-18h], eax
                mov     eax, [rbp-18h]
                cmp     eax, 3          ; DRIVE_FIXED
                jz      _prepare_to_start_enumerate_files
                mov     eax, [rbp-18h]
                cmp     eax, 4          ; DRIVE_REMOTE
                jz      _prepare_to_start_enumerate_files
                jmp     _check_if_need_check_more_units
; ---------------------------------------------------------------------------
```

Check all logic units

For each mounted drive – hard disk or remote share, Anatova will get all files and folders. It will later check if it is a folder and, if it is, will check that the folder name doesn't have the name of "." and "..", to avoid the same directory and the previous directory.

In the list of gathered folder names, Anatova checks against a list of blacklisted names such as "Windows", "Program Files", "Program Files(x86)", etc. This is usual in many ransomware families, because the authors want to avoid destroying the Operating System, instead targeting the high value files. Anatova does the same for file-extensions .exe, .dll and .sys that are critical for the Operating system as well.

```
arg_28          = qword ptr  30h

                mov     [rbp+10h], rcx
                mov     [rbp+18h], rdx
                mov     eax, 0
                mov     [rbp-1], al
                mov     rax, [rbp+18h]
                mov     r10, rax
                mov     rcx, r10
                call    AnatovaPrepareToCheckIfFileNameIsBlackListedFunction
                movzx   eax, al
                mov     [rbp-1], al
                movzx   eax, byte ptr [rbp-1]
                cmp     eax, 0
                jz      _prepare_to_check_extension
                jmp     _exit
; ---------------------------------------------------------------------------

_prepare_to_check_extension:            ; CODE XREF: AnatovaStartingManageFileToCryptItFunction+2C↑j
                mov     rax, [rbp+18h]
                mov     r10, rax
                mov     rcx, r10
                call    AnatovaPrepareToCheckIfTheFileExtensionIsBlackListedFunction
                movzx   eax, al
                mov     [rbp-1], al
                movzx   eax, byte ptr [rbp-1]
                cmp     eax, 0
                jz      _create_file
                jmp     _exit
; ---------------------------------------------------------------------------
```

Check file name and extension

If this check is passed, Anatova will open the file and get its size, comparing it to1 MB. Anatova will only encrypt files1 MB or smaller to avoid lost time with big files; it wants to encrypt fast. By setting pointers at the end of the encrypted files, Anatova makes sure that it does not encrypt files that are already encrypted.

Next, Anatova will create a random value of 32bits as a key for the Salsa20 algorithm and another value of 8 bytes that will be used as IV for Salsa20.

With these values, it will read all files in memory or files with a maximum size of 1 MB and encrypt this information with the key and IV using the Salsa20 algorithm (this is very popular lately because it is a very quick algorithm and has open source implementations).

```
_set_eax_to_1_mega:                     ; CODE XREF: AnatovaStartingManageFileTc
                mov     rax, 100000h
                jmp     _crypt_file
; ---------------------------------------------------------------------------

_set_eax_to_file_size:                  ; CODE XREF: AnatovaStartingManageFileTc
                mov     rax, [rbp-18h]

_crypt_file:                            ; CODE XREF: AnatovaStartingManageFileTc
                mov     [rsp+arg_20], rax
                mov     rax, [rbp-20h]
                mov     [rsp+arg_18], rax
                mov     eax, 0
                mov     r9, rax
                mov     rax, [rbp-58h]
                mov     r8, rax
                mov     eax, 0
                mov     r11, rax
                mov     rax, [rbp-50h]
                mov     r10, rax
                mov     rcx, r10
                mov     rdx, r11
                call    AnatovaPrepareToSalsa20MainCryptAndDecryptFunction
                mov     [rbp-5Ch], eax
                mov     eax, [rbp-5Ch]
                cmp     eax, 0
                jz      _import_key
                jmp     _release_context
```

Encryption of files function

It will import the RSA public key created in runtime and with it, encrypt the key and IV used to encrypt the file. Next, it will write the encrypted content in the same file from the beginning of the file and then it will set the pointer to the end of the file and write the next things:

- The block encrypted of the Salsa20 key is ciphered with the public RSA key.
- The block encrypted of the Salsa20 IV is ciphered with the public RSA key.
- The size of the file is smaller than 1 MB.
- A special hardcoded value for each sample that will appear in the ransom note.
- A special hardcoded value in the sample that is the mark of infection checked before to avoid encrypting the same file twice.

When this is completed, Anatova will write a ransom note in the same folder. So, if Anatova can´t encrypt at least something in a folder, it won't create a ransom note in this folder, only in the affected folders.

This behavior is different from other ransomware families that write a ransom note in all folders.

The ransom note text is fully encrypted in the binary, except for the mail addresses to contact the author(s) and the dash address to pay.

Anatova doesn't overwrite the ransom note if it already exists in a folder in order to save time.The ransom note contains the base64 block with all encrypted information that is needed to decrypt the files in a block that start with the string "—-KEY—-", as well asthe id number.

Responding victims are then allowed to decrypt one .jpg file of maximum size 200kb free of charge, as proof that they the decrypted files can be retrieved.

```
All your files are crypted. Only us can decrypt your files, you need pay 10 DASH in the address:

XpRvUwSjSeHfJqLePsRfQtCKa1VMwaXh12

After the payment send us the address used to make the payment to one of these mail addresses:

anatova2@tutanota.com
anatoday@tutanota.com

Later wait for our reply with your decryptor. If you want can send us ONE JPG FILE ONLY max 200kb to decrypt per free before of payment.

Dont try fuck us, in this case you NEVER will recover your files. Nothing personal, only business.

Send this file untouched with your payment or/and free file!

481

---KEY---
+jX2pgsDcCgzZK+aPFw0n274D5rdIL85nSaVA7w3XvmBiUjykkfkRDB4iU4b9bKF
MeoR7mykyyGHRlWqgNErYE0nI4IUU6hvE6YU3FLdjSDW1jIAFoZB9/4GXEc2UP4l
zGfHe1STTHg2nGuxOPDpeeu9rwCHxqxdNssfVYgxOUk9IQJUn/CDBrLhS8/RMuW3
LLHApBdBs7ntOi2MmoT23JOAaAe8DGOlTuWShSDCWpLpmlKwuSZAInogfSqd1/f2
Py2E6GsGNai+acgkKtkh1aXD4tWvto0czgBXXL0r1q1k17Hp80SQPzg16PJjOhzu
O2Nx4fNE4HqD2XLDNp19YwZKwCVx0Xsuo0i6ijqZsGmi+2XDbvdbXsdBZSIxLdW/
v4VTVzbVDtsR5ZizcJt8yKPxzDYcEs3bXxSd1ezJCcENi16GelzVcPbnoWDv/fHC
ZEuFtyeQ0s4W85tEQN/ua2ni/RhPaTe0kAJYuoaHCY9RI9QD7Kk0k892JaJoPmuj
h6Zg46Q0enlxwS3ImwfbjQOePxx19poeWZMB/8SjsaujTdGqPWvoDeoV9K/p1E0G
5bDayZ484Zj8aisIB5+EFrQlyRal9Q2QwwoBLGRREBLzPJEH6jaBE3YMK4KQsGq6
ENjLS6V/dl6NPyGBk68kNU4I6phkk7RGuoTvIdFcXJoPtmEi3/WdJ67zEQctn+yF
```

Example of ransom note

When all this is done, Anatova will destroy the Volume Shadow copies 10 times in very quick succession. Like most ransomware families, it is using the vssadmin program, which required admin rights, to run and delete the volume shadow copies.

```
                mov     rax, [rbp-20h]
                cmp     rax, 0
                jz      _release_memory
                mov     rax, [rbp-18h]
                cmp     rax, 0
                jz      _release_memory
                mov     rax, [rbp-18h]
                mov     r11, rax
                mov     rax, [rbp-8]
                mov     r10, rax
                mov     rcx, r10
                mov     rdx, r11
                mov     r11, cs:AnatovaGlobalValstrcatWFunction
                call    r11 ; AnatovaGlobalValstrcatWFunction
                mov     eax, 0
                mov     [rbp-2Ch], eax

_check_if_reach_the_max_of_launchs:     ; CODE XREF: AnatovaDestroyShadowVolumesFunction+130↓j
                mov     eax, [rbp-2Ch]
                cmp     eax, 0Ah        ; will launch vssadmin to delete shadow volumes 10 times,
                jge     _release_memory_
                jmp     _delete_shadow_volumes
; ---------------------------------------------------------------------------
```

Delete of Shadow Volumes 10 times

Finally, when all steps are completed, the ransomware will follow the flow of cleaning code, as described earlier, mainly to prevent dumping memory code that could assist in creating a decryption tool.

COVERAGE

Customers of McAfee gateway and endpoint products are protected against this version. Detection names include Ransom-Anatova![partialhash].

INDICATORS OF COMPROMISE

The samples use the following MITRE ATT&CK™ techniques:

- Execution through API
- Application processes discovery
- File and directory discovery: to search files to encrypt
- Encrypt files
- Process discovery: enumerating all processes on the endpoint to kill some special ones
- Create files
- Elevation of privileges: request it to run.
- Create mutants

Hashes

2a0da563f5b88c4d630aefbcd212a35e

366770ebfd096b69e5017a3e33577a94

9d844d5480eec1715b18e3f6472618aa

61139db0bbe4937cd1afc0b818049891

596ebe227dcd03863e0a740b6c605924

Alexandre Mundo
Alexandre Mundo, Senior Malware Analyst is part of Mcafee's Advanced Threat Research team. He reverses the new threads in advanced attacks and make research of them in a daily basis....