Lorenz ransomware: analysis and a free decryptor

tesorion.nl/en/posts/lorenz-ransomware-analysis-and-a-free-decryptor

By Gijs Rijnders June 25, 2021

BleepingComputer recently wrote about a new ransomware family called Lorenz, that targets organizations worldwide. Like other well-known ransomware families, Lorenz also breaches their networks and performs double-extortion by stealing data before encrypting it. The victim is then threatened with their data being published if they decide not to pay the ransom. Ransom demands have been quite high, between $500.000 and $700.000.

Based on our analysis of the ransomware, we have been able to develop a process that can in some cases decrypt files affected by Lorenz without paying the ransom. In this blog post, we will discuss some details about the ransomware. Furthermore, we will announce a free decryptor for Lorenz. This decryptor will be available through the NoMoreRansom initiative soon to help victims free of charge.

Overview

A first glance at the ransomware reveals that it is likely written in C++ using Microsoft Visual Studio 2015. An interesting observation in the two samples we analyzed is that they are both compiled with debug information, even though they seem to be compiled in Release mode. Debug information can be useful as it tends to make the analysis slightly easier.

Like ransomware commonly does, Lorenz also creates a mutex at startup. The mutex is called ‘wolf’ and is used to prevent the ransomware from starting twice in parallel. Another interesting observation is that Lorenz sends the computer name of the infected system to a command & control server before starting encryption. In the samples we analyzed, the connections targeted the command & control server by IP-address on TCP port 55.

File Encryption

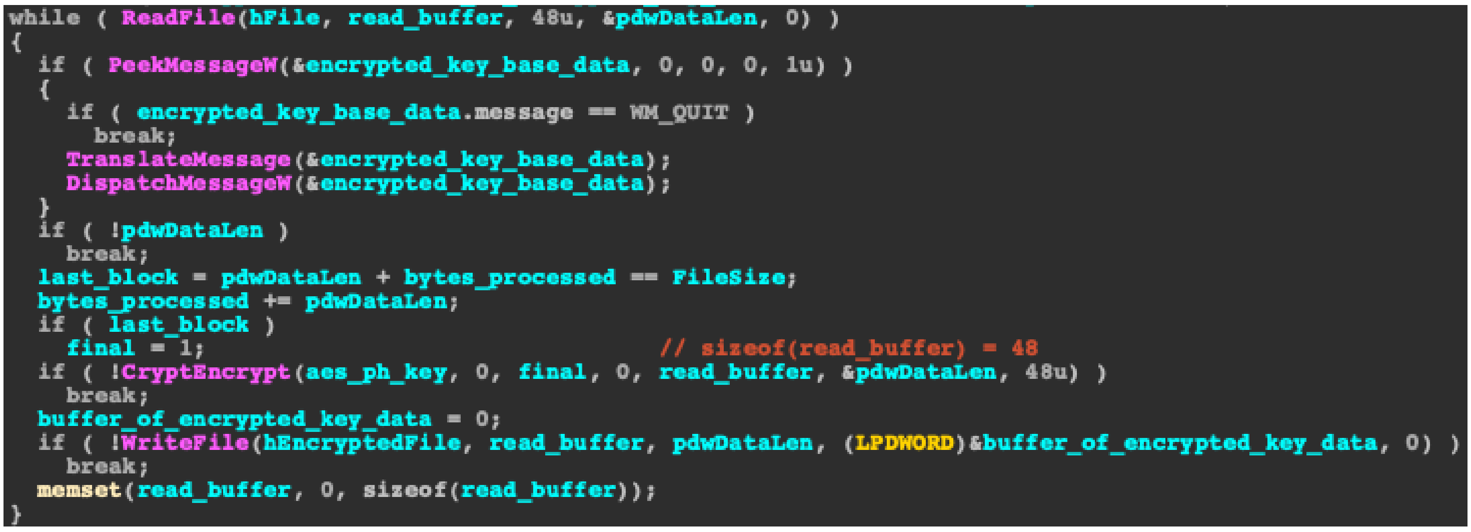
The Lorenz ransomware uses a combination of RSA and AES-128 in CBC mode to encrypt files on an infected system. A password is generated at random for each file, and an encryption key is then derived using the CryptDeriveKey function.

Files encrypted by ransomware commonly contain footers, as footers can be easily appended to a file. Lorenz places a header before the encrypted file instead. This makes the ransomware less efficient as it must copy the contents of every file. The header contains the magic value: ‘.sz40’, followed by the RSA-encrypted file encryption key. After writing the encrypted file header, every file is encrypted whole in rather small blocks of 48 bytes. Encrypted files get the file extension: ‘.Lorenz.sz40’.

According to BleepingComputer, the Lorenz ransomware appears to be a variant of the ThunderCrypt ransomware. We have not analyzed any ThunderCrypt samples and therefore, we do not know whether the file encryption is similar or not.

Encryption bug

Lorenz encrypts every file whole in blocks of 48 bytes. It first reads the next 48 bytes (or whatever is available) from the original file. The freshly obtained data block is then encrypted and written to the encrypted file. This encryption algorithm is displayed in the screenshot below.



```
while ( ReadFile(hFile, read_buffer, 48u, &pdwDataLen, 0) )
{
  if ( PeekMessageW(&encrypted_key_base_data, 0, 0, 0, 1u) )
  {
    if ( encrypted_key_base_data.message == WM_QUIT )
      break;
    TranslateMessage(&encrypted_key_base_data);
    DispatchMessageW(&encrypted_key_base_data);
  }
  if ( !pdwDataLen )
    break;
  last_block = pdwDataLen + bytes_processed == FileSize;
  bytes_processed += pdwDataLen;
  if ( last_block )
    final = 1;                          // sizeof(read_buffer) = 48
  if ( !CryptEncrypt(aes_ph_key, 0, final, 0, read_buffer, &pdwDataLen, 48u) )
    break;
  buffer_of_encrypted_key_data = 0;
  if ( !WriteFile(hEncryptedFile, read_buffer, pdwDataLen, (LPDWORD)&buffer_of_encrypted_key_data, 0) )
    break;
  memset(read_buffer, 0, sizeof(read_buffer));
}
```

If we look closer, the problem lies within the usage of the CryptEncrypt function. Relevant parameters are the 5th, 6th and 7th. They respectively represent the buffer containing the freshly read data, length of this available data and the total size of ‘read_buffer’.

As we can see, CryptEncrypt is told that ‘read_buffer’ is 48 bytes in size. Therefore, it is not allowed to write more than 48 bytes when encrypting this block. If we take a closer look at the documentation of the CryptEncrypt function, we read the following:

“If a large amount of data is to be encrypted, it can be done in sections by calling CryptEncrypt repeatedly. The Final parameter must be set to TRUE on the last call to CryptEncrypt, so that the encryption engine can properly finish the encryption process. The following extra actions are performed when Final is TRUE:

If the key is a block cipher key, the data is padded to a multiple of the block size of the cipher. If the data length equals the block size of the cipher, one additional block of padding is appended to the data.”

Recall that Lorenz uses the AES-128 algorithm, which is a block cipher having a block size of 16 bytes. If the size of ‘read_buffer’ is smaller than 48 bytes, the data inside it would be padded to at most 48 bytes. However, if the available data is exactly 48 bytes in size, CryptEncrypt would append an additional block of padding. The ‘read_buffer’ would then be required to be at least 64 bytes in size. As a result, CryptEncrypt fails, breaking out of the encryption loop. The last block of the file is hence never written, meaning it is lost. Of course, the last block of padding is only appended when ‘final’ (the 3rd parameter for CryptEncrypt) is set to TRUE. If this was not the case, all other blocks would fail as well.

The result of this bug is that for every file which’s size is a multiple of 48 bytes, the last 48 bytes are lost. Even if you managed to obtain a decryptor from the malware authors, these bytes cannot be recovered.

A free decryptor

Based on our analysis of the Lorenz ransomware we have come to the conclusion that we can decrypt (non-corrupted) affected files in some cases without paying the ransom. Supported file types include Microsoft Office documents, PDF files and some image and movie types. We built a decryptor that we are providing to victims free of charge. The decryptor is available for download via the NoMoreRansom initiative.

Indicators of compromise (IoCs)

Indicator	Description
71cdbbc62e10983db183ca60ab964c1a3dab0d279c5326b2e920522480780956	Lorenz ransomware
4b1170f7774acfdc5517fbe1c911f2bd9f1af498f3c3d25078f05c95701cc999	Lorenz ransomware
157[.]90[.]147[.]28	C2
172[.]86[.]75[.]63	C2

For more information on the Lorenz samples:

https://otx.alienvault.com/indicator/file/71cdbbc62e10983db183ca60ab964c1a3dab0d279c5326b2e920522480780956
https://otx.alienvault.com/indicator/file/4b1170f7774acfdc5517fbe1c911f2bd9f1af498f3c3d25078f05c95701cc999