

# RansomEXX Trojan attacks Linux systems

By Fedor Sinitsyn

Published: 2020-11-06 · Archived: 2026-04-05 21:37:11 UTC

We recently discovered a new file-encrypting Trojan built as an ELF executable and intended to encrypt data on machines controlled by Linux-based operating systems.

After the initial analysis we noticed similarities in the code of the Trojan, the text of the ransom notes and the general approach to extortion, which suggested that we had in fact encountered a Linux build of the previously known ransomware family RansomEXX. This malware is notorious for attacking large organizations and was most active earlier this year.

RansomEXX is a highly targeted Trojan. Each sample of the malware contains a hardcoded name of the victim organization. Moreover, both the encrypted file extension and the email address for contacting the extortionists make use of the victim's name.

Several companies have fallen victim to this malware in recent months, including the [Texas Department of Transportation](#) (TxDOT) and [Konica Minolta](#).

## Technical description

The sample we came across – [aa1ddf0c8312349be614ff43e80a262f](#) – is a 64-bit ELF executable. The Trojan implements its cryptographic scheme using functions from the open-source library mbedtls.

When launched, the Trojan generates a 256-bit key and uses it to encrypt all the files belonging to the victim that it can reach using the AES block cipher in ECB mode. The AES key is encrypted by a public RSA-4096 key embedded in the Trojan's body and appended to each encrypted file.

Additionally, the malware launches a thread that regenerates and re-encrypts the AES key every 0.18 seconds. However, based on an analysis of the implementation, the keys actually only differ every second.

Apart from encrypting the files and leaving ransom notes, the sample has none of the additional functionality that other threat actors tend to use in their Trojans: no C&C communication, no termination of running processes, no anti-analysis tricks, etc.

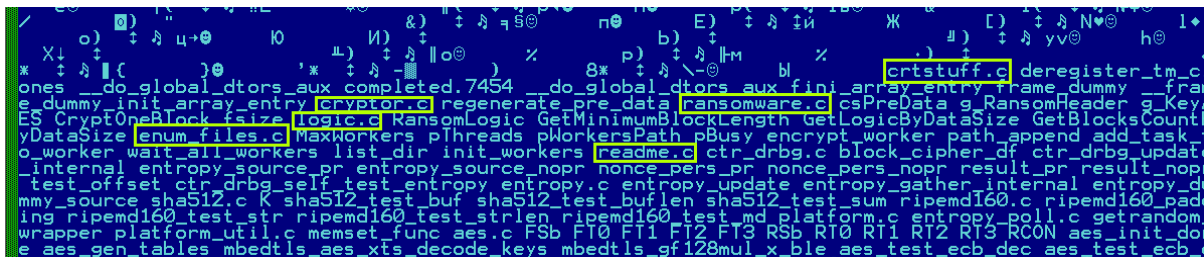
```

if ( a1 )
{
    v10 = strlen(s) + 277;
    v2 = alloca(16 * ((v10 + 23LL) / 0x10uLL));
    dest = (16 * ((&s + 7) >> 4));
    if ( dest )
    {
        strcpy(dest, s);
        strcat(dest, ".██████");
        v3 = rand();
        sprintf(src, "-%08x", v3);
        strcat(dest, src);
        if ( fsize(dest) == -1 )
        {
            stream = fopen64(s, "r+");
            if ( stream )
            {
                v9 = fsize(s);
                if ( v9 )
                {
                    if ( v9 > 15 )
                    {
                        mbedtls_aes_init(aes_ctx);
                        pthread_mutex_lock(&csPreData);
                        memcpy(ptr, &g_RansomHeader, 0x200uLL);
                        mbedtls_aes_setkey_enc(aes_ctx, &g_KeyAES, 256LL);
                        pthread_mutex_unlock(&csPreData);
                        if ( !fseek(stream, 0LL, SEEK_END)
                            && fwrite(ptr, 1uLL, 0x200uLL, stream)
                            && !fseek(stream, -512 - v9, 1)
                            && ProcessFileHandleWithLogic(stream, aes_ctx, a2, v9, CryptOneBlock) )
                        {
                            v13 = 1;
                        }
                    }
                }
            }
        }
    }
}

```

***Fragment of the file encryption procedure pseudocode; variable and function names are saved in the debug information and must match the original source code***

Curiously, the ELF binary contains some debug information, including names of functions, global variables and source code files used by the malware developers.



***Original names of source files embedded in the trojan's body***

```

[06ac][06ac] << clone (,,, [0000000000006b1] {0000}
[06ac][06b1] << clone (,,, [000000000000000] {0000}
[06ac][06b1] >> set_robust_list (00007fa5e1ef49e0,0000000000000018) => 00007fa5f42cee44 {0000}
[06ac][06b1] << set_robust_list (,) [00000000] {0000}
[06ac][06b1] >> sched_setaffinity (00000000,00000000,00007fa5e1ef3e70 -> 0000000000000008) => 00007fa5f3f82c5f {0000}
[06ac][06ac] >> openat (ffffffffffff9c -> "",000055a3553bc0a0 -> "/root",00000000,0000) => 00007fa5f3fe8820 {0000}
[06ac][06ac] >> openat (,,, [0000000000000003 -> {Directory: "/root"}] {0000}
[06ac][06ac] >> stat (000055a3553bc0c0 -> "/root/INews_FOR_...!.txt",00007ffc35408c60) => 00007fa5f3fe82c5 {0000}
[06ac][06ac] >> stat (,00007ffc35408c60 -> (struct stat*){st_dev=0000000000000000,st_ino=0000000000000000,st_nlink=0000000000000000,st_mode=00000000,st_uid=
[06ac][06ac] >> open (000055a3553bc0c0 -> "/root/INews_FOR_...!.txt",00000241_01b6) => 00007fa5f3fe877d {0000}
[06ac][06ac] >> open (,, [0000000000000004 -> {File: "/root/INews_FOR_...!.txt"}] {0000}
[06ac][06ac] >> fstat (0000000000000004 -> {File: "/root/INews_FOR_...!.txt"},00007ffc35408ad0) => 00007fa5f3fe8314 {0000}
[06ac][06ac] >> fstat (,00007ffc35408ad0 -> (struct stat*){st_dev=0000000000000000,st_ino=0000000000000000,st_nlink=0000000000000001,st_mode=000081a4,st_uid=
[06ac][06ac] >> mmap (0000000000000000,000000000001000,00000003,00000022,ffffffffffffffff -> "",0000000000000000) => 00007fa5f3ff1d9a {0000}
[06ac][06ac] >> mmap (,,,, [00007fa5f4701000] {0000}
[06ac][06ac] >> write (0000000000000004 -> {File: "/root/INews_FOR_...!.txt"},00007fa5f4701000 -> "Greetings . . .!!!\r\n\r\nStudy this message REGARDFULLY
[06ac][06ac] >> write (,, [00000000000001ec] {0000}
[06ac][06ac] >> close (0000000000000004 -> {File: "/root/INews_FOR_...!.txt"}) => 00007fa5f3fe9050 {0000}
[06ac][06ac] << close () [0000000000000000] {0000}
    
```

### Execution log of the trojan in Kaspersky Linux Sandbox

## Similarities with Windows builds of RansomEXX

Despite the fact that previously discovered PE builds of RansomEXX use WinAPI (functions specific to Windows OS), the organization of the Trojan’s code and the method of using specific functions from the mbedtls library hint that both ELF and PE may be derived from the same source code.

In the screenshot below, we see a comparison of the procedures that encrypt the AES key. On the left is the ELF sample aa1ddf0c8312349be614ff43e80a262f; on the right is the PE sample fcd21c6fca3b9378961aa1865bee7ecb used in the TxDOT attack.

Despite being built by different compilers with different optimization options and for different platforms, the similarity is quite obvious.

<pre> v19 = mbedtls_mpi_read_string(     v16,     16LL,     "8FC02A208837E9B96A0ADFCCED1086888658672540E5408E0D9811F78C4EE1A0999EAD98B9D9006F9886     *FBFA2AF03748BF68C38C5962AD48F3787C3A5138C8409FACD015F38A1668001DA0922444F7A7487F     *6F56325A5E2032C6D178C671E45F4408C5FC034AF1F8F985E473FB20E09E73C2888562459FC8A050346     *1F9FE3153E1E024E4113CE7F940E764980505E3E0F0F72982D0061D028F0943665E293823AF248276     *818988F5CAA331F4039F47070501F10518E49684018E6AC79C28888040528358E6C1A1400003F161     *D0AACD4DC7CF089F68921ABE6E4CF89E529A5A706738497C715031AE3C2BCD7085C21F205484386905     *6D0AC69C4E221E076187DE4E65E609E8862285DE804F63258A098928024D478769CF7E383A44CFD32     *B0872728EE4D406309F9419F84C1E50C73F8728EC828890A9878C49395EE4D18EC0D419CE9F3D445898     *1857A4AF8F3E331042616045AD71D8F866260087861C9617D73C979884B7244F759E9F8FF3AE0DF8EE39     *0390844DF8F35D8507CC6E9544A1AA237"); if ( v19 )     return v21; v19 = mbedtls_mpi_read_string(&amp;v17, 16LL, "010001"); if ( v19 )     return v21; v15 = (mbedtls_mpi_bitlen(v16) + 7) &gt;&gt; 3; v19 = mbedtls_rsa_pkcs1_encrypt(&amp;v14, mbedtls_ctr_dbg_random, v12, 0, 32LL, v8, v18); if ( v19 )     return v21; pthread_mutex_lock(&amp;csPreData); *rg_KeyAES = *v8; *rg_KeyAES[8] = v9; *rg_KeyAES[16] = v10; *rg_KeyAES[24] = v11; _memp_memcpy(rg_RansomHeader, v18, v15); pthread_mutex_unlock(&amp;csPreData); v21 = 1; mbedtls_rsa_free(&amp;v14); mbedtls_ctr_dbg_free(v13); mbedtls_entropy_free(v13); return v21;     </pre>	<pre> 93  &amp;&amp; mbedtls_mpi_read_string( 94  &amp;v20[2], 95  "8B45A5C97A75102E2C0030C89A4851D89026721CA327A27A7E0645FD427586ACF468430C87719F3C060 96  *30D600870899C01C3F9C0E8303F2A780029F071E2DC792D6A791503A95450719F408896E9FB08E4 97  *1AD44F8E8996EA7F09501A5E5F7F803F5742395AC3C776808957C5D20695868F68883F1F5519C0A0C 98  *97A626F2FD3F792088D4C34722033895ED0487E85ACDC8714C357F0ACAD8BE48C4682AA2FE85D0B 99  *961E2D20983501C7A316710129A84567581931C43C7016F7F9944557D10005E7676D008A6608800B 100  *8F04C2A0D0802C0650E2D487380548F8E486564E53ECC2E447D03166090831854F274C935A 101  *968E4DF581D88E644E2AF607F03115A1A4ZD89A4487CC161A65268C28FDEAF8979772D0D0858D 102  *CASACAB8288F4374F5DA9398130080428F901A1080B51E535164A601164D440A9F8E24F4929691CC9 103  *C6E180988FAC205C7A5DC4A4CD186980C8A2F58AA4FD79C014669A232FC3C8A0694E8B0C4C1E28E7 104  *86AC9628DF08F09321C0C808996A41CAFE60E1D849A7E808F"); 105  &amp;&amp; mbedtls_mpi_read_string(&amp;v20[5], "010001"); 106  { 107  v20[1] = (mbedtls_mpi_bitlen(&amp;v20[2]) + 7) &gt;&gt; 3; 108  if ( !v20[4] ) 109  { 110  v8 = mbedtls_rsa_pkcs1_encrypt(v17, v20, v16, v20); 111  LABEL_27: 112  if ( !v8 ) 113  { 114  EnterCriticalSection(&amp;crit_sect); 115  memcpy(&amp;aes_key, v16, 0x20u); 116  memcpy(&amp;aes_key_encrypted, v20, v20[1]); 117  LeaveCriticalSection(&amp;crit_sect); 118  v12 = 1; 119  mbedtls_rsa_free(v20); 120  memset(v17, 0, 0x140u); 121  memset(v23, 0, sizeof(v23)); 122  v24 = 0; 123  memset(v25, 0, 0x190u); 124  v22 = 0; 125  } 126  goto LABEL_29; 127  }     </pre>
--	--

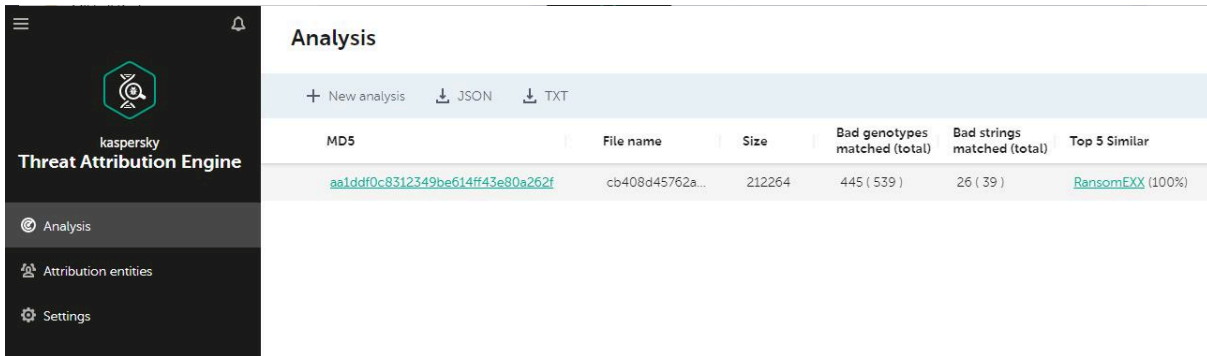
We also observe resemblances in the procedure that encrypts the file content, and in the overall layout of the code.

What’s more, the text of the ransom note is also practically the same, with the name of the victim in the title and equivalent phrasing.

## Parallels with a recent attack in Brazil

As [reported](#) by the media, one of the country’s government institutions has just been attacked by a targeted ransomware Trojan.





The screenshot shows the Kaspersky Threat Attribution Engine interface. On the left is a dark sidebar with the Kaspersky logo and menu items: Analysis, Attribution entities, and Settings. The main area is titled 'Analysis' and contains a table of results. The table has columns for MD5, File name, Size, Bad genotypes matched (total), Bad strings matched (total), and Top 5 Similar. One entry is visible, showing a match to RansomEXX with 100% similarity.

MD5	File name	Size	Bad genotypes matched (total)	Bad strings matched (total)	Top 5 Similar
<a href="#">aa1ddf0c8312349be614ff43e80a262f</a>	cb408d45762a...	212264	445 ( 539 )	26 ( 39 )	<a href="#">RansomEXX</a> (100%)

*Kaspersky Threat Attribution Engine identifies Ransomexx malware family*

## Indicators of compromise

Recent Linux version: [aa1ddf0c8312349be614ff43e80a262f](#)

Earlier Windows version: [fcd21c6fca3b9378961aa1865bee7ecb](#)

---

Source: <https://securelist.com/ransomexx-trojan-attacks-linux-systems/99279/>