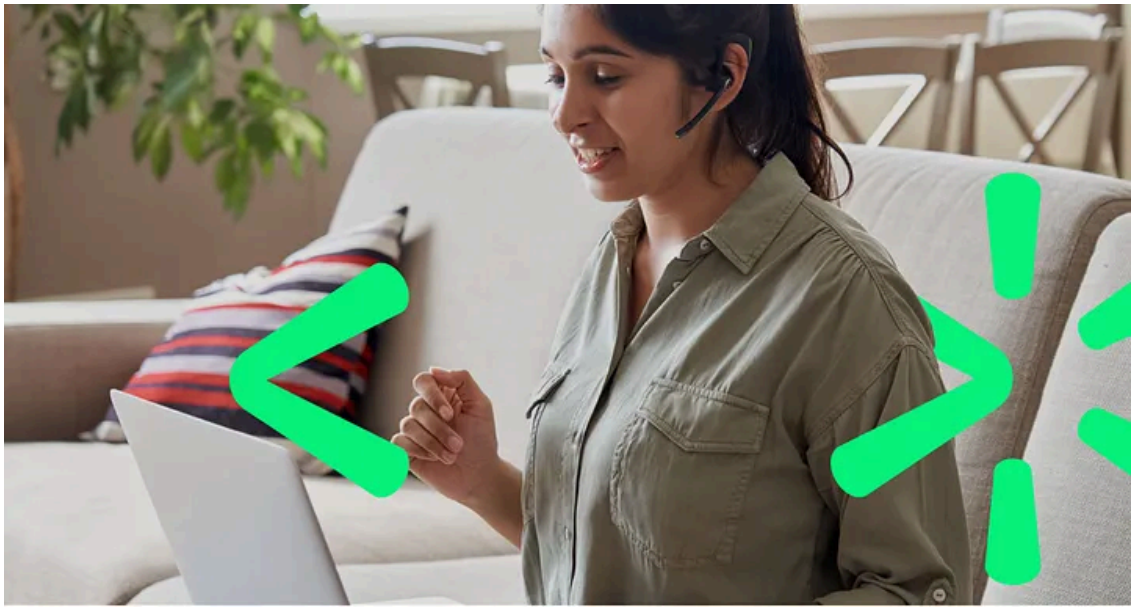
# CobaltStrike Stager Utilizing Floating Point Math

By Jason Reaves

Published: 2021-04-20 · Archived: 2026-04-05 18:32:24 UTC

Press enter or click to view image in full size



By: Jason Reaves and Joshua Platt

## Executive summary

1. New CobaltStrike stagers utilizing floating point mnemonics[1] to decode out stager shellcode.
2. Using raw sockets and date value from Google headers to check overwritten sleep values such as in some sandbox detonations.

## Date checking

The stager employs an interesting technique to check for being detonated in controlled environments such as sandboxes that might overwrite sleep values, at the same time it also checks for network connectivity.

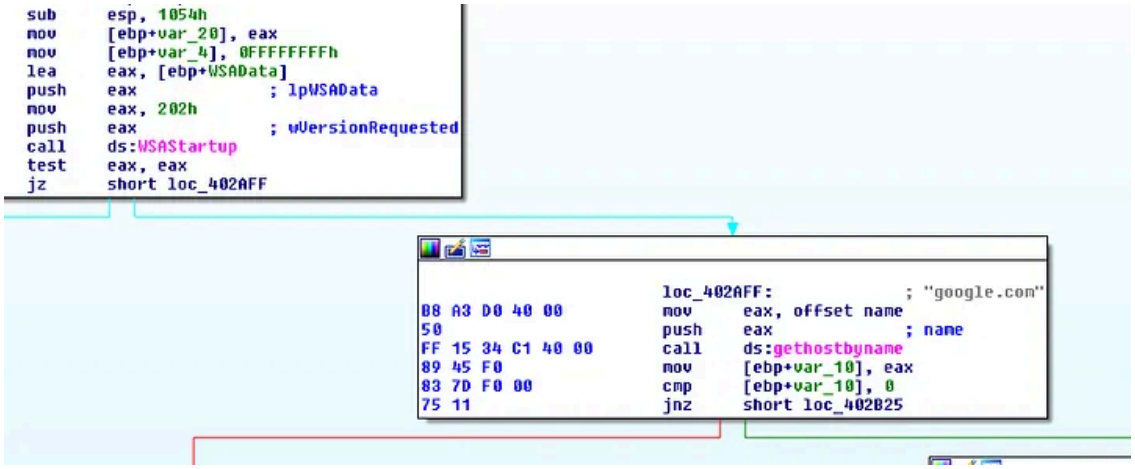
## Get Jason Reaves’s stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

The stager utilizes raw sockets to connect to ‘google.com’ over port 80 and send a GET request.

Press enter or click to view image in full size




Raw socket to google.com

The request is not parsed as an HTTP request in most utilities including Wireshark[2] and Suricata[3] because it is incomplete with just a newline and no carriage return.

```
mov     eax, offset aGetDrv ; "GET drv\n"
push    eax
lea     eax, [ebp+WSAData]
push    eax
call    sub_4030C0
add     esp, 8
push    0               ; flags
lea     eax, [ebp+WSAData]
call    sub_4030F0
push    eax             ; len
lea     eax, [ebp+WSAData]
push    eax             ; buf
mov     eax, [ebp+s]
push    eax             ; s
call    ds:send
```

Incomplete request

The request is enough to retrieve the 404 response from the webserver and then the malware begins parsing the values out of the date, specifically it parses out the day, year and time values.

```
add     esp, 8
mov     edx, offset aDate ; "Date: "
lea     eax, [ebp+WSAData]
call    loc_403110
mov     [ebp+var_C], eax
mov     edx, ','
mov     eax, [ebp+var_C]
call    FindChar_4031C0
mov     [ebp+var_C], eax
mov     eax, [ebp+var_C]
inc     [ebp+var_C]
mov     eax, [ebp+var_C]
inc     [ebp+var_C]
mov     edx, offset aGmt ; "GMT"
mov     eax, [ebp+var_C]
call    loc_403110
```

Parse values from response

After parsing out the values it converts it to seconds but without accounting for the month.

```
83 C4 1C                add     esp, 1Ch
8D 45 AC                lea     eax, [ebp+var_54]
E8 AB 04 00 00          call    ConvertDate_403230
```

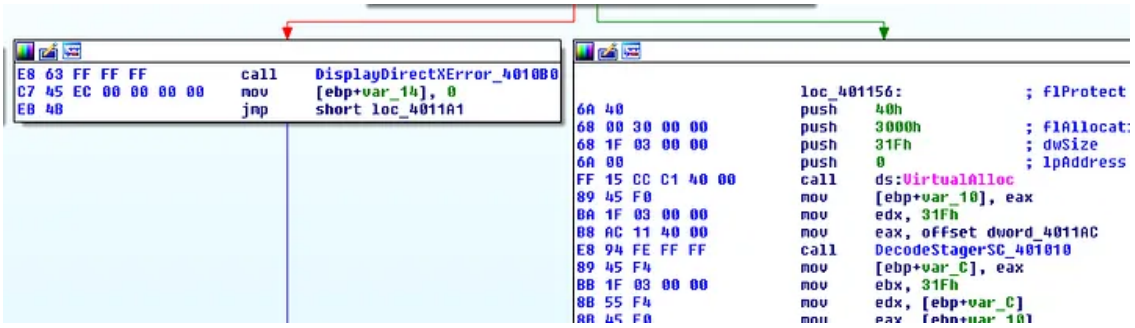
Convert values to seconds

Press enter or click to view image in full size


Time Check

Above you can see a sleep call is sandwiched by two of these calls to the function responsible for retrieving the converted value from a google request, the sleep is 30 seconds and then it checks if the values differ less than 28. It is checking if the process took less than 28 seconds or not.

Press enter or click to view image in full size

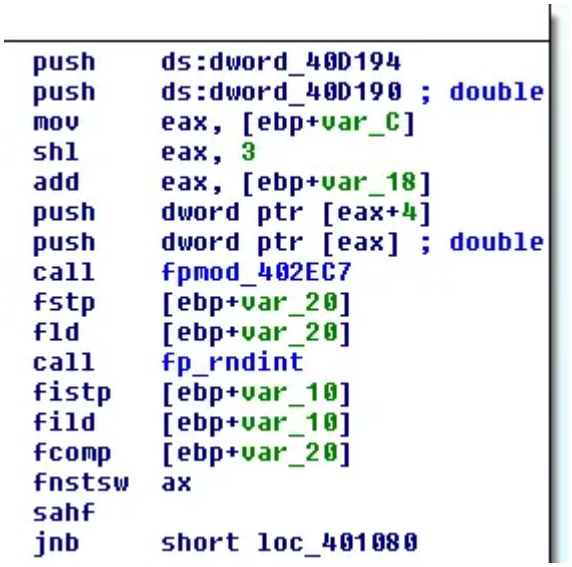



Error or decode logic

If the check fails then a fake DirectX error message is displayed, otherwise the process for decoding the stager shellcode begins.

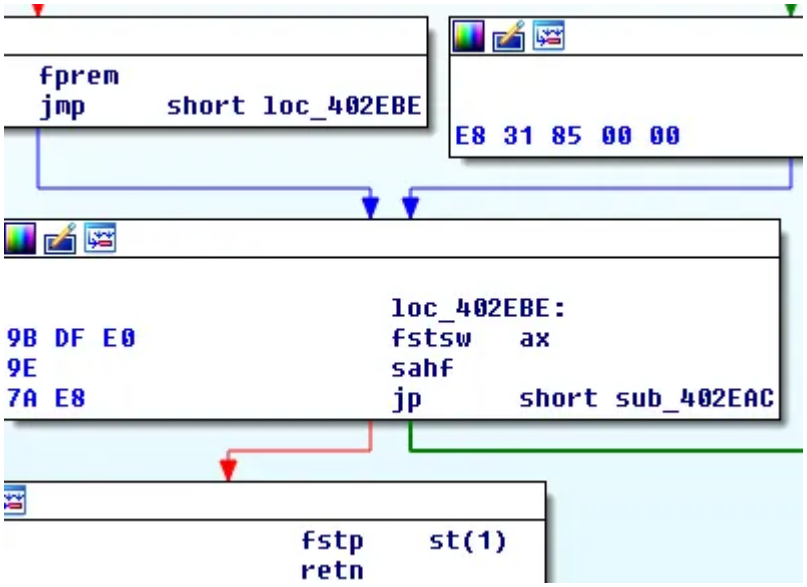
## Shellcode decode

The shellcode is decoded by utilizing floating point mnemonics, judging by some of the actors testing this appears to be pretty good at bypassing static detection engines.




Decode loop

The process involved begins with floating point modulus against a table of data using a key value that is hardcoded.




fpmod

After the modulus the value is rounded to an int value. Example python code for decoding the data can be seen below:

```
def fpmod_decode(key, data, l):
 out = ""
 for i in range(l):
  temp = struct.unpack_from('<d', data[i*8:])[0]
  if temp > int(temp%key):
   out += chr((ord(struct.pack('<Q', int(temp%key))[0])+1)&0xff)
  else:
   out += chr((ord(struct.pack('<Q', int(temp%key))[0]))&0xff)
 return out
```

Using our decode code we can quickly enumerate samples for decoding out the shellcode and harvesting IOCs.

## Indicators of compromise

```
cda7edc9414814ef57c31e473ce87e489bcd6f1ed8d81a504e960e184fce1609
abaf70728e6f940195e35e689cae40e0d598f2e85e2c881f8b558a45bb57cce5
7793c2fd34248236e83206fdd01b547436e966bcb6cae21adcbf61550b62daea
5d4fd3e3fef4e46cff33d0772f0c0c2c13ab7ba50cdd95f0761401652bb898de
9ee75f2d28d93c90e2cf0da6d6d0d39fe9c12ce65c7ba6b880cf1c2c94add657
7c047718e71e393bebd8147889087a4d207b125b98099b9effbf78f2291d2a68
603e112de99388f8aea461a539ae57e38ca83faf2bd43984036eb3b7080c24be
aca0a3e30d83e10197ebf1bf0fc2e7557e4e07f45066d6d1b3e997ca78d683f6
6c6e49e0e822618c21d04ffd02ab26a0cb20b296d9d5a4e0cb27a5809a416089
13177de544464a87be341fda62b7c62efd34adc858728893963d5169d2763b1f
b4dceaded7b0184ebefbdac8b6d6af543b19b248a64754ffe8cee02473cefa83adsec[.]pro
manageupdaternetwork[.]com
```

```
192.99.250[.]7
192.95.16[.]237
195.123.234[.]60
aloogi[.]com
45.141.86[.]9
185.4.65[.]139
107.181.187[.]96
5.34.179[.]35alert tcp $HOME_NET any -> $EXTERNAL_NET 80 (msg:"CS stager time check 1"; dsize:8; con
```

## References


1. https://www.felixcloutier.com/x86/index.html
2. https://www.wireshark.org/
3. https://suricata-ids.org/


---