DRAKVUF Sandbox - Open-source, self-hosted malware sandbox in hypervisor

Adam Kliś, Michał Leszczyński

Confidence

2022 Cracow

$ whoami

Adam Kliś

- Former Specialist @ CERT.PL
  → Malware analysis automation / DRAKVUF research

- Researcher @ STM Cyber
  → Research: finding 0-days in IoT and enterprise software
  → Development: new red teaming tools and techniques

contact@bonusplay.pl
Twitter: @BonusPlay3

$ whoami

Michał Leszczyński

- Former Expert @ CERT.PL
  → Malware analysis automation / DRAKVUF research

- Founder @ ITSEC R&D Company
  → Engineering: NFC Solutions, Blockchain
  → Advisory: IT Security

ml@icedev.pl
Twitter: @icedevml

End goal demo

Assumptions:

- VM: Original Windows 7
  + Original game off the shelf

- Host: Xen/Ubuntu


[MEMOST] Current money 20000
[MEMOST] Current money 15000
[MEMOST] Current money 17000
[MEMOST] Current money 19000

Introduction

Malware 101

malware

malware

malware

Malware 101


10 tabletek
Paracetamol Polfa - Łódź
(Paracetamolum) 500 mg, tabletki
Paracetamol
Polfa - Łódź
Lek przeciwbólowy i przeciwgorączkowy, do stosowania doustnego w następujących wskazaniach: bóle różnego pochodzenia, między innymi:
bóle głowy (także migrenowe),
stawowe, mięśniowe,
miesiączkowe oraz występujące
po urazach i zabiegach chirurgicznych oraz stomatologicznych;
gorączka np. w stanach grypopodobnych i przeziębieniach.
50 tabletek
POLFA ŁÓDŹ

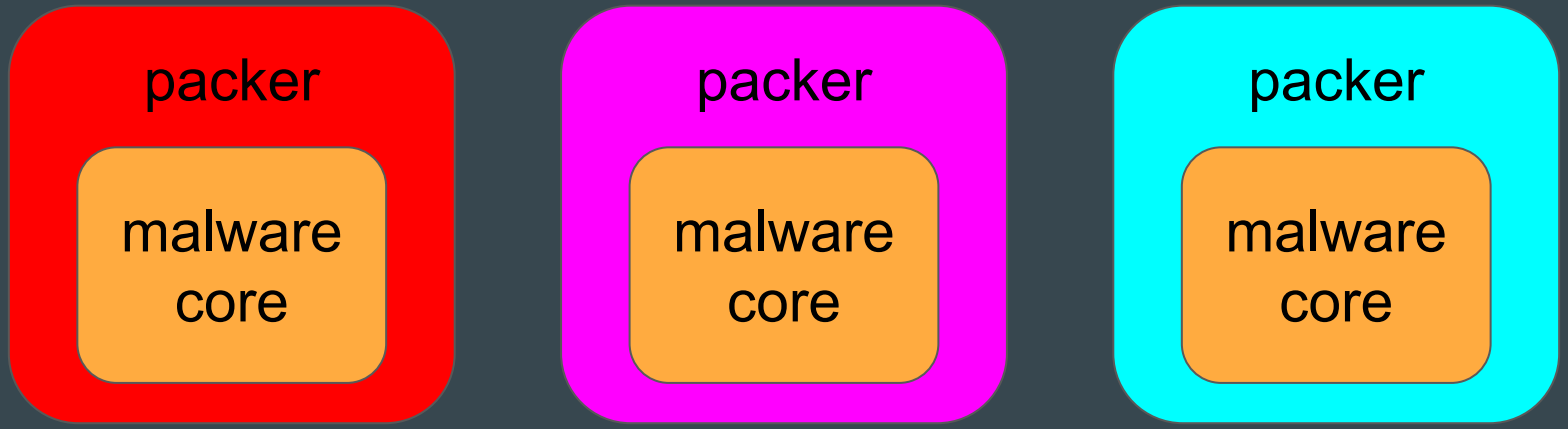

PARACETAMOL
BIOFARM®
Paracetamolum
500 mg
tabletki
20 tabletek
Lek przeciwbólowy i przeciwgorączkowy


APAP®
Paracetamolum 500 mg
LEK PRZECIWBÓLOWY, PRZECIWGORĄCZKOWY
bóle głowy
nerwobóle
bóle zębów
bóle mięśni
bóle menstruacyjne
gorączka w przebiegu
przeziębienia
grypy
Nie podrażnia błony śluzowej przewodu pokarmowego
12 tabletek powlekanych

Malware 101

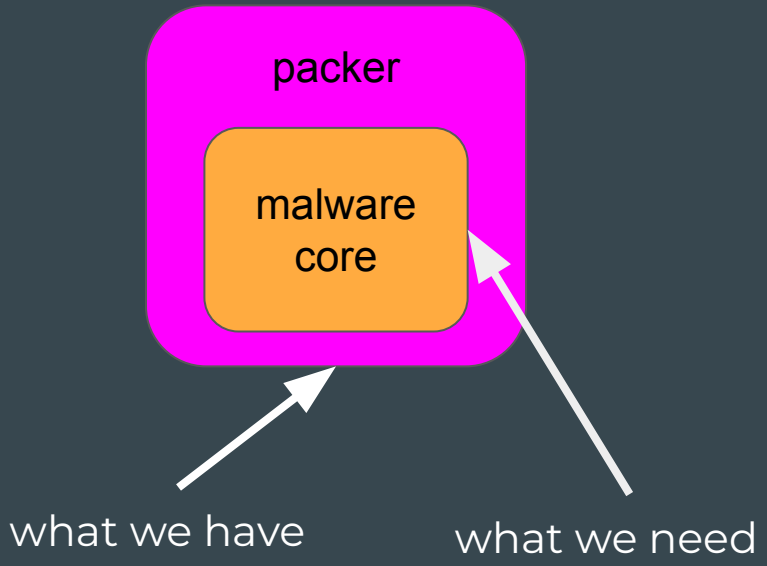
paracetamol
500mg

Malware 101


packer
malware core
packer
malware core
packer
malware core

Malware 101

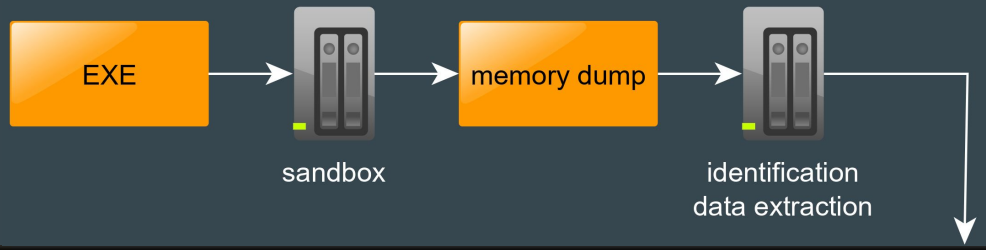
- different packers
- the same/similiar malware core
- malware core => easy identification
- ... also easy data extraction



Malware processing at CERT.PL

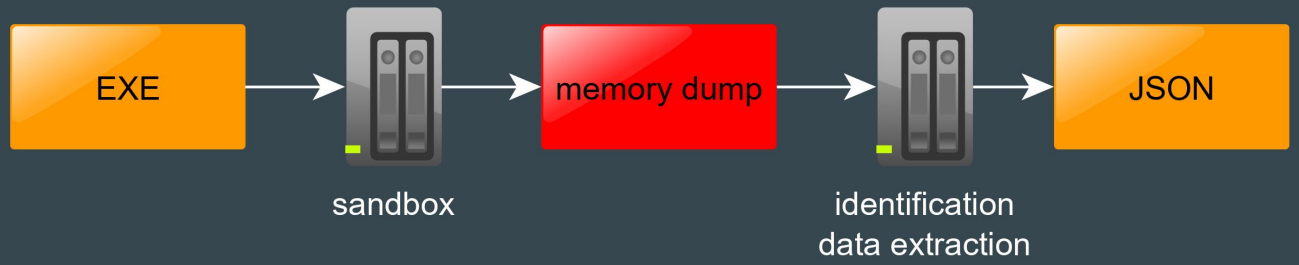
- malware unpacking
- extraction of some interesting stuff

Example:



Family	emotet
Config type	static
+ exe_words	["engine", "finish", "magnify", "resapi", "query", "skip", "wubi", "svcs", "router", "crypto", "backup", "ha…
+ public_key	-----BEGIN PUBLIC KEY----- MHwwDQYJKoZIhvcNAQEBBQADawAwaAJhALk+KlHgOKXm9eDkWu2yN9lanjwOm6W2 PV0tgr4msNVby2pOJ…
+ type	emotet
+ url_words	["teapot", "pnp", "tpt", "splash", "site", "codec", "health", "balloon", "cab", "odbc", "badge", "dma", "pse…
+ urls	[{ "cnc": "186.176.138.171", "port": 7080 }, { "cnc": "200.51.94.251", "port": 80 }, { "cnc": "46.105.131.87…
Upload time	Wed, 16 Oct 2019 11:40:07 GMT

Malware processing at CERT.PL



What is a memory dump?

- logical dump of the memory at given point of time

- metadata:
  - base address at which dump was made,
  - reason of the dump (e.g. malware made some interesting API call)

- profit? unpacked malware (at least sometimes)

Dynamic unpacking - theory

- when to make a memory dump?

Dynamic unpacking - theory

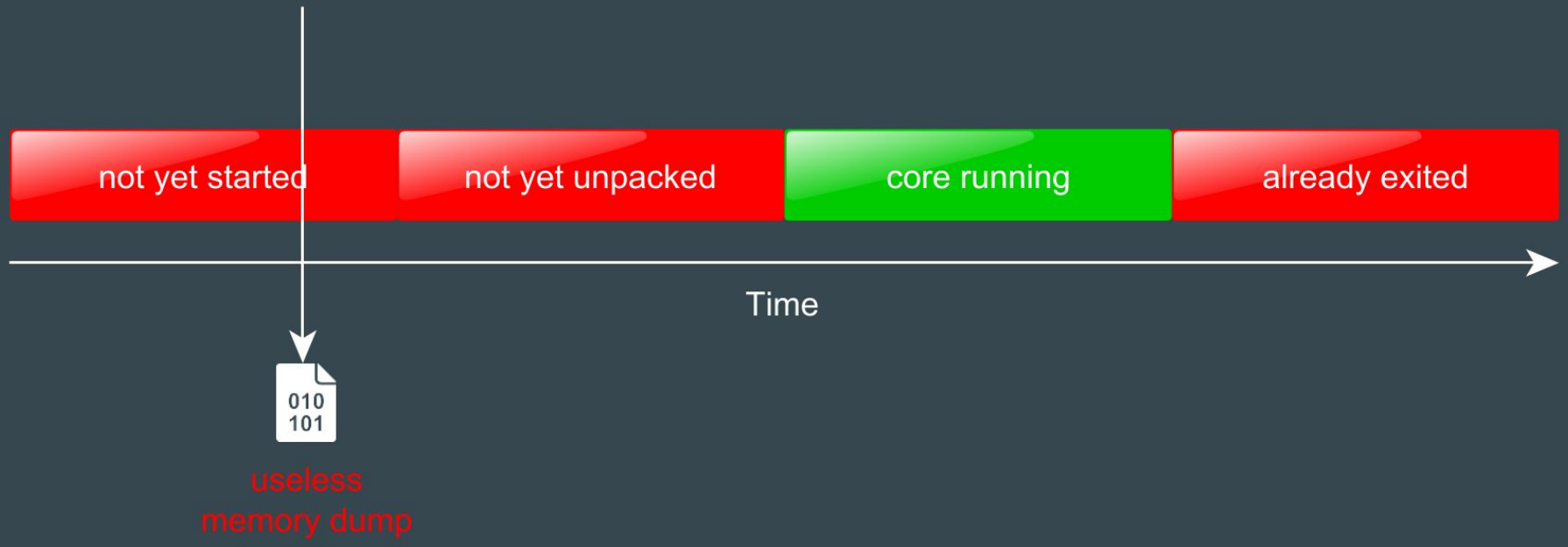
- when to make a memory dump?

randomly?

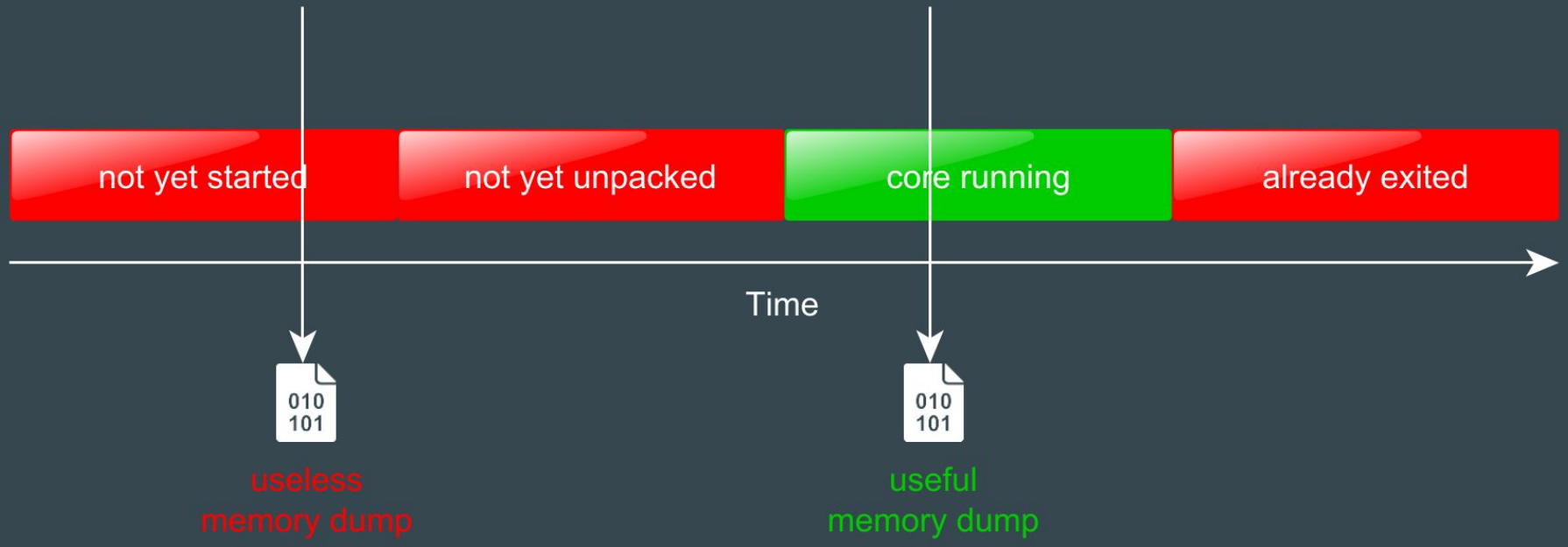
Dynamic unpacking - theory

- when to make a memory dump?

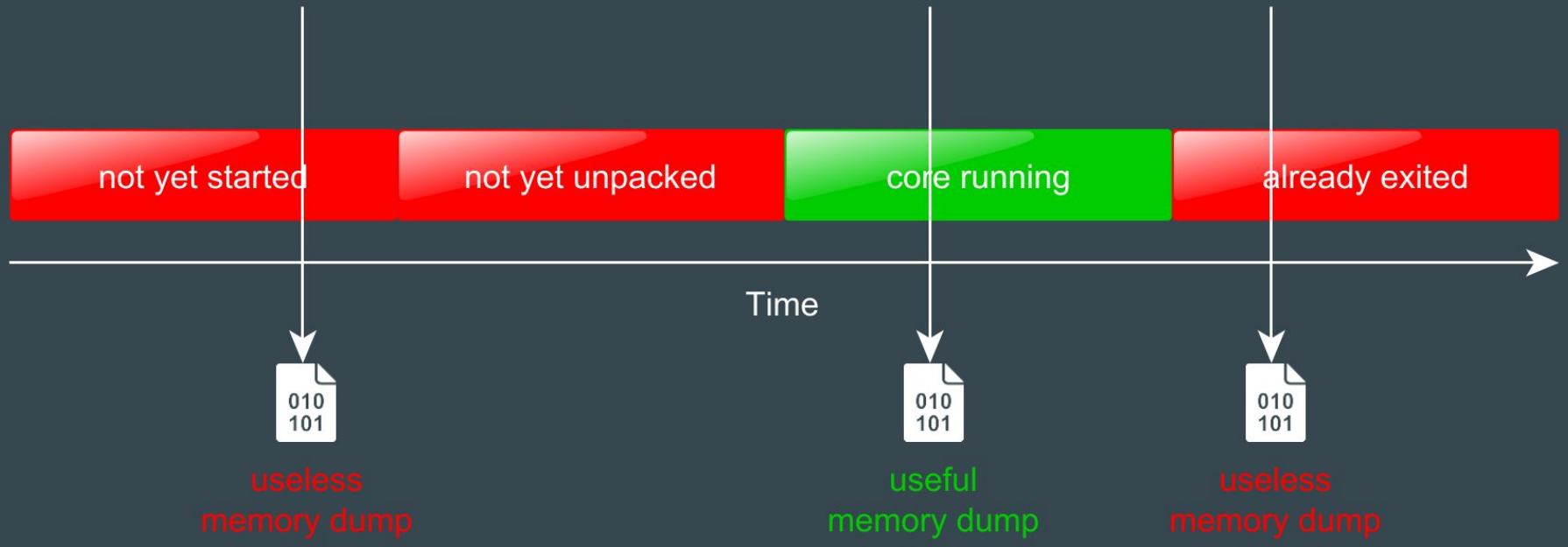
Dynamic unpacking - theory



Dynamic unpacking - theory



Dynamic unpacking - theory



Dynamic unpacking - theory

- in order to have good memory dumps, you need good heuristics
- good heuristics need good behavioral monitoring
- why can’t you just use an ordinary sandbox?
  - we do, but...

Malware monitoring problems

Example #1 - trickbot (1c81272ffc)

Example #1 - trickbot (1c81272ffc)

- Well known trojan / stealer
- Packed x86/x64 binaries
- Process hollowing using direct system calls

Sample:
https://mwdb.cert.pl/sample/1c81272ffc28b29a82d8313bd74d1c6030c2af1ba4b165c44dc8ea6376679d9f

References:
https://www.cyberbit.com/blog/endpoint-security/latest-trickbot-variant-has-new-tricks-up-its-sleeve/
https://www.cert.pl/en/news/single/detricking-trickbot-loader/

Example #1 - trickbot (1c81272ffc)

Directly making syscalls - not visible on conventional sandboxes

```
10002600    8B D4        mov edx, esp
10002602    0F 34        sysenter
10002604    C3           ret
```

References:
https://www.cyberbit.com/blog/endpoint-security/latest-trickbot-variant-has-new-tricks-up-its-sleeve/

Example #2 - remcos (60c07bac07)

Example #2 - remcos (60c07bac07)

- Remote Access Trojan
- Packed x86/x64 binaries
- Hollowing **svchost.exe** using WriteProcessMemory()

Sample:
https://mwdb.cert.pl/sample/60c07bac07c7e2f2f3e03817addb88b38b8fbcd893d4b41b5007d984e8ba1fc5

Example #2 - remcos (60c07bac07)

This is how **Cuckoo** hooks **ntdll.dll** (for Windows 7 x86):

```
static int hook_api_jmp_direct(hook_t *h, unsigned char *from,
    unsigned char *to)
{
    // unconditional jump opcode
    *from = 0xe9;

    // store the relative address from this opcode to our hook function
    *(unsigned long *)(from + 1) = (unsigned char *) to - from - 5;
    return 0;
}
```

TLDR: replace first 5 bytes of the hooked function with a 0xE9 jump

Example #2 - remcos (60c07bac07)

This is how **Cuckoo** hooks **ntdll.dll** (for Windows 7 x86):

```
static int hook_api_jmp_direct(hook_t *h, unsigned char *from,
    unsigned char *to)
{
    // unconditional jump opcode
    *from = 0xe9;

    // store the relative address from this opcode to our hook function
    *(unsigned long *)(from + 1) = (unsigned char *) to - from - 5;
    return 0;
}
```

TLDR: replace first 5 bytes of the hooked function with a 0xE9 jump

Example #2 - remcos (60c07bac07)

This is how **Cuckoo** hooks **ntdll.dll** (for Windows 7 x86):

```
static int hook_api_jmp_direct(hook_t *h, unsigned char *from,
    unsigned char *to)
{
    // unconditional jump opcode
    *from = 0xe9;

    // store the relative address from this opcode to our hook function
    *(unsigned long *)(from + 1) = (unsigned char *) to - from - 5;
    return 0;
}
```

TLDR: replace first 5 bytes of the hooked function with a 0xE9 jump

Example #2 - remcos (60c07bac07)

… and this is how remcos unhooks:

```
94    v16 = v26;
95    while ( ++v16 != v27 + v26 )
96    {
97      if ( *(_BYTE *)v16 == 0xB8 && !*(_DWORD *)(v16 + 1) && *(_BYTE *)(v16 + 5) == 0xB9 )
98      {
99        v17 = v16 + 0xA;
100       v18 = 0;
101       v19 = 1;
102       do
103       {
104         ++v18;
105         if ( *(_DWORD *)++v17 == 0x424548D )
106         {
107           if ( *(_WORD *)(v17 - 2) != 0x33C9 && *(_BYTE *)(v17 - 5) == 0xB9 )
108           {
109             *(_BYTE *)(v17 - 10) = 0xB8;
110             *(_DWORD *)(v17 - 9) = v19++;
111           }
112           else
113           {
114             *(_BYTE *)(v17 - 7) = 0xB8;
115             *(_DWORD *)(v17 - 6) = v19++;
116           }
117         }
118       }
119       while ( v18 != 0x3000 );
120       goto LABEL_47;
121     }
122   }
```

```
Disassembly - C:\Users\janusz\Desktop\mlwr.exe - WinDbg:10.0.19041.1 AMD64
Offset: 77aef890
77aef875 6f                outs    dx,dword ptr [esi]
77aef876 6c                ins     byte ptr es:[edi],dx
77aef877 007763            add     byte ptr [edi+63h],dh
77aef87a 7374              jae     ntdll32!NtReadFile+0x10 (77aef8f0)
77aef87c 6f                outs    dx,dword ptr [esi]
77aef87d 6d                ins     dword ptr es:[edi],dx
77aef87e 627300            bound   esi,qword ptr [ebx]
77aef881 7763              ja      ntdll32!NtReadFile+0x6 (77aef8e6)
77aef883 7374              jae     ntdll32!NtReadFile+0x19 (77aef8f9)
77aef885 6f                outs    dx,dword ptr [esi]
77aef886 756c              jne     ntdll32!NtReadFile+0x14 (77aef8f4)
77aef888 009090909090      add     byte ptr [eax-6F6F6F70h],dl
77aef88e 90                nop
77aef88f 90                nop
ntdll32!ZwMapUserPhysicalPagesScatter:
77aef890 b800000000        mov     eax,0
77aef895 b90a000000        mov     ecx,0Ah
77aef89a 8d542404          lea     edx,[esp+4]
77aef89e 64ff15c0000000    call    dword ptr fs:[0C0h]
77aef8a5 83c404            add     esp,4
77aef8a8 c20c00            ret     0Ch
77aef8ab 90                nop
ntdll32!NtWaitForSingleObject:
77aef8ac b801000000        mov     eax,1
77aef8b1 b90d000000        mov     ecx,0Dh
77aef8b6 8d542404          lea     edx,[esp+4]
77aef8ba 64ff15c0000000    call    dword ptr fs:[0C0h]
```

manually unpacked by @nazywam - thx

Example #2 - remcos (60c07bac07)

… and this is how remcos unhooks:

binary-match first export

```
94    v16 = v26;
95    while ( ++v16 != v27 + v26 )
96    {
97      if ( *(_BYTE *)v16 == 0xB8 && !*(_DWORD *)(v16 + 1) && *(_BYTE *)(v16 + 5) == 0xB9 )
98      {
99        v17 = v16 + 0xA;
100       v18 = 0;
101       v19 = 1;
102       do
103       {
104         ++v18;
105         if ( *(_DWORD *)++v17 == 0x424548D )
106         {
107           if ( *(_WORD *)(v17 - 2) != 0x33C9 && *(_BYTE *)(v17 - 5) == 0xB9 )
108           {
109             *(_BYTE *)(v17 - 10) = 0xB8;
110             *(_DWORD *)(v17 - 9) = v19++;
111           }
112           else
113           {
114             *(_BYTE *)(v17 - 7) = 0xB8;
115             *(_DWORD *)(v17 - 6) = v19++;
116           }
117         }
118       }
119       while ( v18 != 0x3000 );
120       goto LABEL_47;
121     }
122   }
```

```
Disassembly - C:\Users\janusz\Desktop\mlwr.exe - WinDbg:10.0.19041.1 AMD64
Offset: 77aef890
77aef875 6f                outs    dx,dword ptr [esi]
77aef876 6c                ins     byte ptr es:[edi],dx
77aef877 007763            add     byte ptr [edi+63h],dh
77aef87a 7374              jae     ntdll32!NtReadFile+0x10 (77aef8f0)
77aef87c 6f                outs    dx,dword ptr [esi]
77aef87d 6d                ins     dword ptr es:[edi],dx
77aef87e 627300            bound   esi,qword ptr [ebx]
77aef881 7763              ja      ntdll32!NtReadFile+0x6 (77aef8e6)
77aef883 7374              jae     ntdll32!NtReadFile+0x19 (77aef8f9)
77aef885 6f                outs    dx,dword ptr [esi]
77aef886 756c              jne     ntdll32!NtReadFile+0x14 (77aef8f4)
77aef888 009090909090      add     byte ptr [eax-6F6F6F70h],dl
77aef88e 90                nop
77aef88f 90                nop
ntdll32!ZwMapUserPhysicalPagesScatter:
77aef890 b800000000        mov     eax,0
77aef895 b90a000000        mov     ecx,0Ah
77aef89a 8d542404          lea     edx,[esp+4]
77aef89e 64ff15c0000000    call    dword ptr fs:[0C0h]
77aef8a5 83c404            add     esp,4
77aef8a8 c20c00            ret     0Ch
77aef8ab 90                nop
ntdll32!NtWaitForSingleObject:
77aef8ac b801000000        mov     eax,1
77aef8b1 b90d000000        mov     ecx,0Dh
77aef8b6 8d542404          lea     edx,[esp+4]
77aef8ba 64ff15c0000000    call    dword ptr fs:[0C0h]
```

manually unpacked by @nazywam - thx

Example #2 - remcos (60c07bac07)

… and this is how remcos unhooks:

```
94    v16 = v26;
95    while ( ++v16 != v27 + v26 )
96    {
97      if ( *(_BYTE *)v16 == 0xB8 && !*(_DWORD *)(v16 + 1) && *(_BYTE *)(v16 + 5) == 0xB9 )
98      {
99        v17 = v16 + 0xA;
100       v18 = 0;
101       v19 = 1;
102       do
103       {
104         ++v18;
105         if ( *(_DWORD *)++v17 == 0x424548D )
106         {
107           if ( *(_WORD *)(v17 - 2) != 0x33C9 && *(_BYTE *)(v17 - 5) == 0xB9 )
108           {
109             *(_BYTE *)(v17 - 10) = 0xB8;
110             *(_DWORD *)(v17 - 9) = v19++;
111           }
112           else
113           {
114             *(_BYTE *)(v17 - 7) = 0xB8;
115             *(_DWORD *)(v17 - 6) = v19++;
116           }
117         }
118       }
119       while ( v18 != 0x3000 );
120       goto LABEL_47;
121     }
122   }
```

"for each export"

```
ntdll32!ZwMapUserPhysicalPagesScatter:
77aef890 b800000000      mov     eax,0
77aef895 b90a000000      mov     ecx,0Ah
77aef89a 8d542404        lea     edx,[esp+4]
77aef89e 64ff15c0000000  call    dword ptr fs:[0C0h]
77aef8a5 83c404          add     esp,4
77aef8a8 c20c00          ret     0Ch
77aef8ab 90              nop
ntdll32!NtWaitForSingleObject:
77aef8ac b801000000      mov     eax,1
77aef8b1 b90d000000      mov     ecx,0Dh
77aef8b6 8d542404        lea     edx,[esp+4]
77aef8ba 64ff15c0000000  call    dword ptr fs:[0C0h]
```

manually unpacked by @nazywam - thx

Example #2 - remcos (60c07bac07)

... and this is how remcos unhooks:

```
94    v16 = v26;
95    while ( ++v16 != v27 + v26 )
96    {
97      if ( *(_BYTE *)v16 == 0xB8 && !*(_DWORD *)(v16 + 1) && *(_BYTE *)(v16 + 5) == 0xB9 )
98      {
99        v17 = v16 + 0xA;
100       v18 = 0;
101       v19 = 1;
102       do
103       {
104         ++v18;
105         if ( *(_DWORD *)++v17 == 0x424548D )
106         {
107           if ( *(_WORD *)(v17 - 2) != 0x33C9 && *(_BYTE *)(v17 - 5) == 0xB9 )
108           {
109             *(_BYTE *)(v17 - 10) = 0xB8;
110             *(_DWORD *)(v17 - 9) = v19++;
111           }
112           else
113           {
114             *(_BYTE *)(v17 - 7) = 0xB8;
115             *(_DWORD *)(v17 - 6) = v19++;
116           }
117         }
118       }
119       while ( v18 != 0x3000 );
120       goto LABEL_47;
121     }
122   }
```

override first 5 bytes to ensure we're unhooked

```
ntdll32!ZwMapUserPhysicalPagesScatter:
77aef890 b800000000      mov     eax,0
77aef895 b90a000000      mov     ecx,0Ah
77aef89a 8d542404        lea     edx,[esp+4]
77aef89e 64ff15c0000000  call    dword ptr fs:[0C0h]
77aef8a5 83c404          add     esp,4
77aef8a8 c20c00          ret     0Ch
77aef8ab 90              nop
ntdll32!NtWaitForSingleObject:
77aef8ac b801000000      mov     eax,1
77aef8b1 b90d000000      mov     ecx,0Dh
77aef8b6 8d542404        lea     edx,[esp+4]
77aef8ba 64ff15c0000000  call    dword ptr fs:[0C0h]
```

manually unpacked by @nazywam - thx

Example #2 - remcos (60c07bac07)

... and this is how remcos unhooks:

```
94    v16 = v26;
95    while ( ++v16 != v27 + v26 )
96    {
97      if ( *(_BYTE *)v16 == 0xB8 && !*(_DWORD *)(v16 + 1) && *(_BYTE *)(v16 + 5) == 0xB9 )
98      {
99        v17 = v16 + 0xA;
100       v18 = 0;
101       v19 = 1;
102       do
103       {
104         ++v18;
105         if ( *(_DWORD *)++v17 == 0x424548D )
106         {
107           if ( *(_WORD *)(v17 - 2) != 0x33C9 && *(_BYTE *)(v17 - 5) == 0xB9 )
108           {
109             *(_BYTE *)(v17 - 10) = 0xB8;
110             *(_DWORD *)(v17 - 9) = v19++;
111           }
112           else
113           {
114             *(_BYTE *)(v17 - 7) = 0xB8;
115             *(_DWORD *)(v17 - 6) = v19++;
116           }
117         }
118       }
119       while ( v18 != 0x3000 );
120       goto LABEL_47;
121     }
122   }
```

override first 5 bytes to ensure we're unhooked

```
ntdll32!ZwMapUserPhysicalPagesScatter:
77aef890 b800000000      mov     eax,0
77aef895 b90a000000      mov     ecx,0Ah
77aef89a 8d542404        lea     edx,[esp+4]
77aef89e 64ff15c0000000  call    dword ptr fs:[0C0h]
77aef8a5 83c404          add     esp,4
77aef8a8 c20c00          ret     0Ch
77aef8ab 90              nop
ntdll32!NtWaitForSingleObject:
77aef8ac b801000000      mov     eax,1
77aef8b1 b90d000000      mov     ecx,0Dh
77aef8b6 8d542404        lea     edx,[esp+4]
77aef8ba 64ff15c0000000  call    dword ptr fs:[0C0h]
```

manually unpacked by @nazywam - thx

Unhooking

Of course you can implement anti$^{(2n - 1)}$-unhooking...

Unhooking

Of course you can implement anti$^{(2n - 1)}$-unhooking...

... and they would implement anti$^{(2n)}$-unhooking ...

Unhooking

Of course you can implement anti$^{(2n - 1)}$-unhooking...

... and they would implement anti$^{(2n)}$-unhooking ...

(Valid for $n \in Z^{+}$)

Example #3 - kronos (6a8419d81f)

Example #3 - kronos (6a8419d81f)

- Banking malware
- Packed x86/x64 binaries
- API hammering

Sample:
https://mwdb.cert.pl/sample/6a8419d81fb645c073439e284a988ab540cd514a933ce2b6ee4b776aa50b50ac

Example #3 - kronos (6a8419d81f)

API hammering, pretty long sequence of operations:

- manipulating registry keys
  `\\REGISTRY\\MACHINE\\Software\\Wow6432Node\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\occidentalconvertors`
- creating directories
- etc.

Example #3 - kronos (6a8419d81f)

API hammering:

```
$ cat drakmon.log \
    | grep NtCreateKey \
    | grep occidentalconvertors \
    | wc -l
40484
```

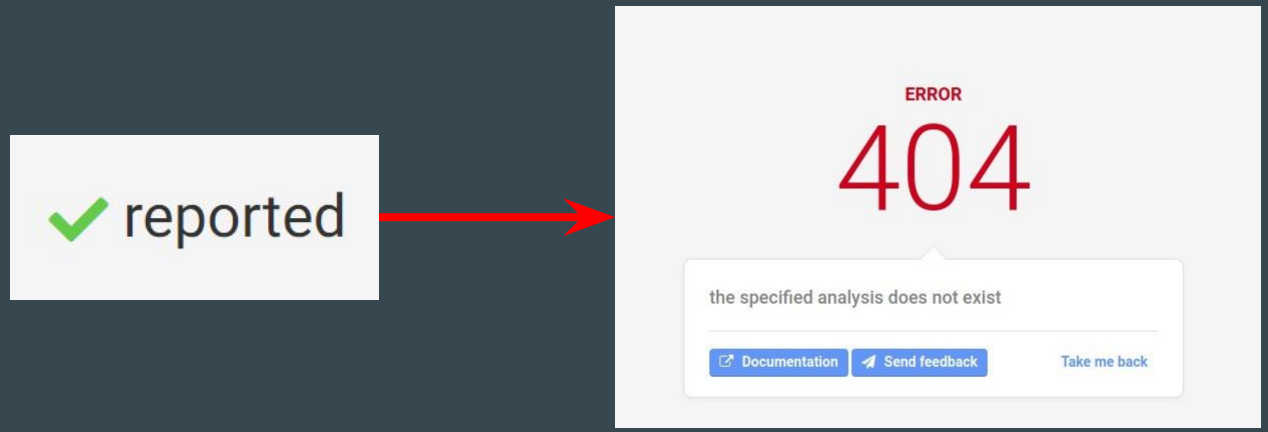
Example #3 - kronos (6a8419d81f)

API hammering:

```
$ cat drakmon.log | grep NtCreateKey | grep occidentalconvertors | head -n1
{
  "Plugin": "regmon",
  "TimeStamp": "1596380139.796501",
  "ProcessName": "\\Device\\HarddiskVolume2\\Users\\janusz\\Desktop\\MALWAR.EXE",
  "UserName": "SessionID",
  "UserId": 1,
  "PID": 1584,
  "PPID": 804,
  "Method": "NtCreateKey",
  "Key":
"\\REGISTRY\\MACHINE\\Software\\Wow6432Node\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\
occidentalconvertors"
}
```

Example #3 - kronos (6a8419d81f)

After uploading to **cuckoo.cert.ee**:



Example #3 - kronos (6a8419d81f)

Our old&rusty Cuckoo 1:

```
sie 02 17:08:08 rex python[9179]: 2020-08-02 17:08:08,536 [lib.cuckoo.core.guest]
INFO: Starting analysis on guest (id=m, ip=192.168.122.31)
sie 02 17:10:33 rex python[9179]: 2020-08-02 17:10:33,621 [lib.cuckoo.core.scheduler]
ERROR: Analysis failed: [Errno 10054] An existing connection was forcibly closed by
the remote host
sie 02 17:10:35 rex python[9179]: 2020-08-02 17:10:35,608 [lib.cuckoo.core.scheduler]
INFO: Task #132707: analysis procedure completed
```

(exact reason not known)

Registry

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88
89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132
133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220
221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264
265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308
309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352
353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396
397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440
441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484
485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528
529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572
573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616
617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660
661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704
705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748

Let’s do it on our own

HOW sandboxes PROLIFERATE:

SITUATION: THERE ARE 14 COMPETING sandboxes

14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL sandbox THAT COVERS EVERYONE'S USE CASES.

YEAH!

SOON:

SITUATION: THERE ARE 15 COMPETING sandboxes


CERT.PL
That sign can't stop me because I can't read!

Let’s do it on our own

- ~~user mode~~ (problems already mentioned)
- kernel mode
- hypervisor

Let’s do it on our own

- ~~user mode~~ (problems already mentioned)
- ~~kernel mode~~ (already done by others)
- hypervisor

Let’s do it on our own

- ~~user mode~~ (problems already mentioned)
- ~~kernel mode~~ (already done by others)
- **hypervisor**

Virtual Machine Introspection

What is VMI? (simplified)

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

What is VMI? (simplified)

```
$ vmi-process-list windows7-sp1
Process listing for VM windows7-sp1-x86 (id=7)
[    4] System (struct addr:84aba980)
[  220] smss.exe (struct addr:85a44020)
[  300] csrss.exe (struct addr:85f67a68)
[  336] wininit.exe (struct addr:8601e030)
```

DRAKVUF

What is DRAKVUF?

- blackbox binary analysis system
- **very clunky** “gdb/strace” for Virtual Machines

What is DRAKVUF?

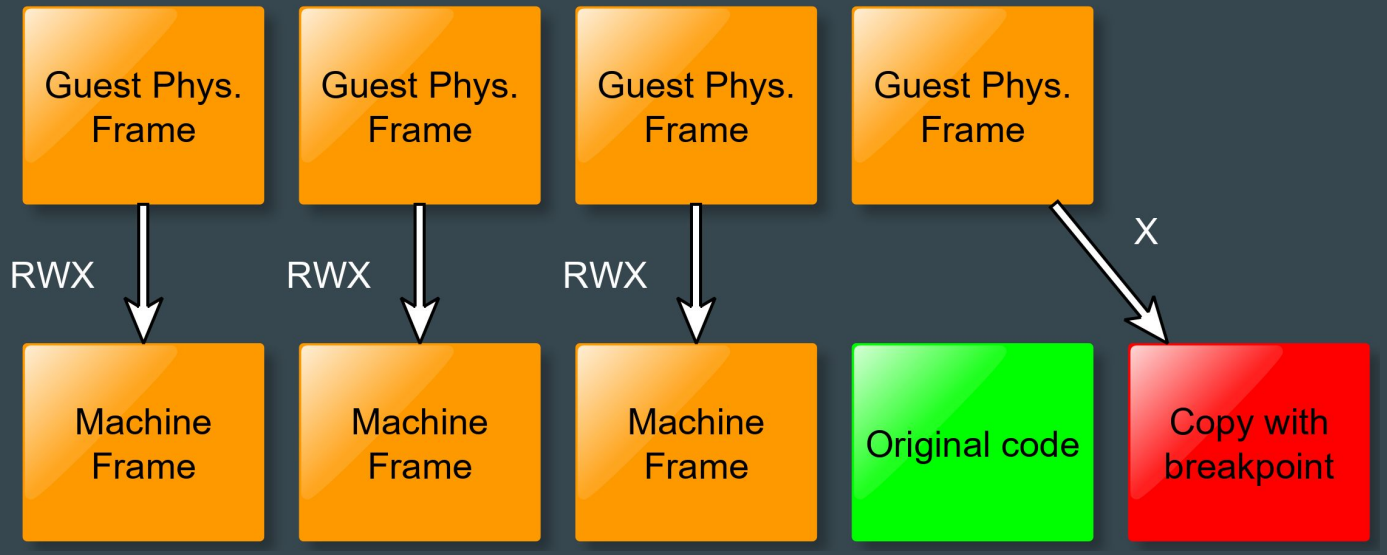
```
$ drakvuf -d windows7-sp1 …

[SYSCALL] TIME:1571248115.605033 VCPU:1
CR3:0x56ca5000,"\Device\HarddiskVolume2\Windows\System32\WindowsPowerShell\
v1.0\powershell.exe" SessionID:1 ntoskrnl.exe!NtProtectVirtualMemory
Arguments: 5
    IN HANDLE ProcessHandle: 0xffffffffffffffff
    INOUT PVOID *BaseAddress: 0x13cd08
    INOUT PSIZE_T RegionSize: 0x13cd10
    IN WIN32_PROTECTION_MASK NewProtectWin32: 0x4
    OUT PULONG OldProtect: 0x13cfb0

[SYSCALL] TIME:1571248171.517430 VCPU:0 …
```

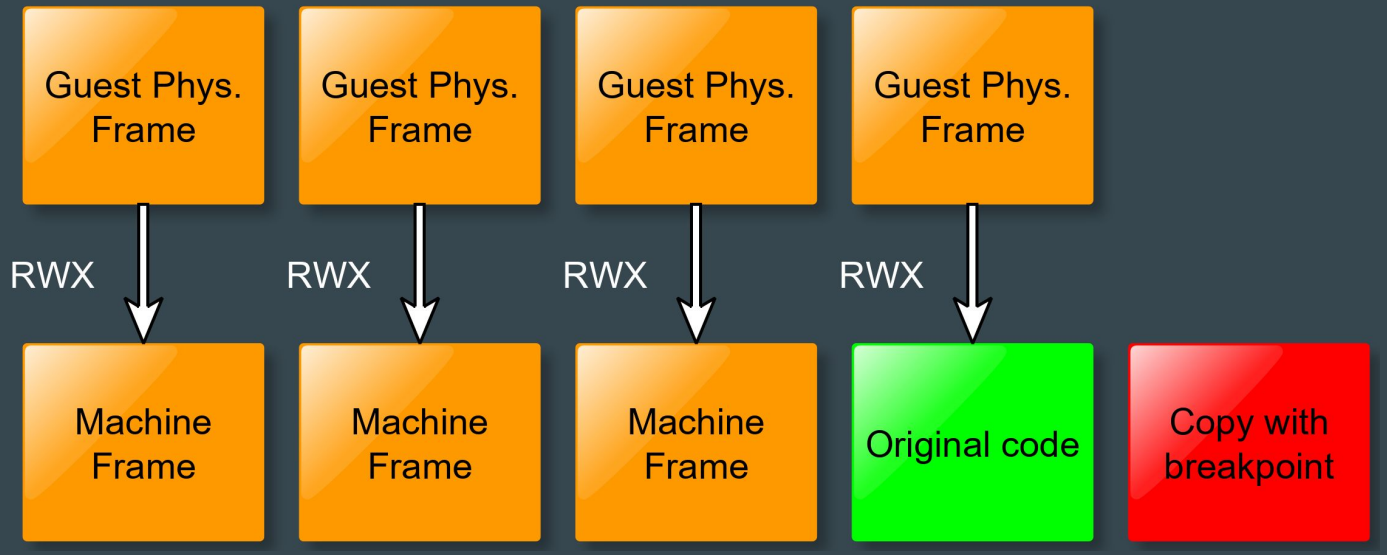
DRAKVUF's hooks (simplified)

Default altp2m view during execution



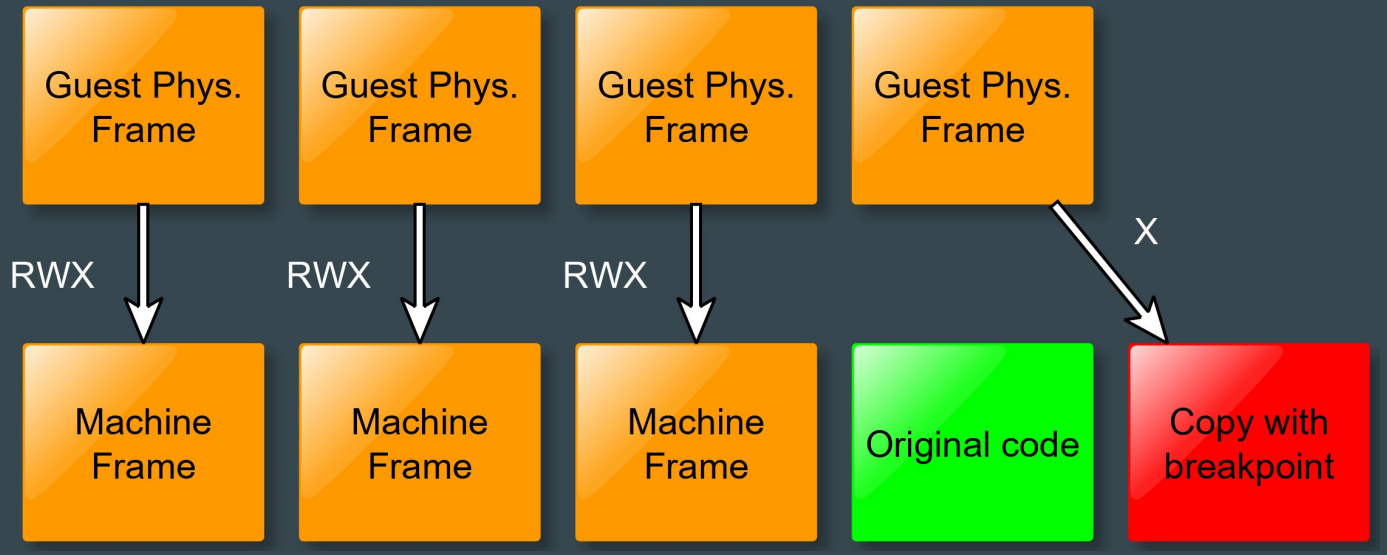
DRAKVUF's hooks (simplified)

"Normal view" - used only during single-step



DRAKVUF's hooks (simplified)

Back to default altp2m view after single-step



Memory dumps

the “technical” part

Heuristics

Hook `NtProtectVirtualMemory(process_handle, base_addr, ...)`:

```
if (process_handle == ~0ULL) {
    char buf[2];
    read_vm_memory(base_addr, buf, 2);

    if (buf[0] == 'M' && buf[1] == 'Z') {
        dump_memory(base_addr, "possible binary detected");
    }
}
```

Heuristics

Hook `NtFreeVirtualMemory(process_handle, base_addr, ...)`:

```
if (process_handle == ~0ULL) {
    if (page_table_check_rwx(base_addr)) {
        dump_memory(base_addr, "free called on RWX memory");
    }
}
```

Memory dumps

How to map a single pointer into a corresponding memory region?

```
dump_memory(mem_base_address, "possible binary detected");
```

→ Look inside Virtual Address Descriptors.

Memory dumps

VAD - Virtual Address Descriptor

```
[1] dump.mem 18:15:32> vad(eprocess=0xfa8002992060)
  VAD            lev start            end          com   type     exe    protect
  -------------- --- --------------- ------------- ----- ------- ------ --------------------
  0xfa80020076d0   8  0x7fef4020000  0x7fef405ffff     3 Mapped  Exe    EXECUTE_WRITECOPY
C:\Windows\System32\tapi32.dll
  0xfa80016d6d80   6  0x7fef4060000  0x7fef4097fff     2 Mapped  Exe    EXECUTE_WRITECOPY
C:\Windows\System32\WinSCard.dll
  ...
  0xfa80016cbc40   6  0x7fefd020000  0x7fefd036fff     2 Mapped  Exe    EXECUTE_WRITECOPY
C:\Windows\System32\cryptsp.dll
  0xfa8003022a00   7  0x7fefd680000  0x7fefd68efff     2 Mapped  Exe    EXECUTE_WRITECOPY
C:\Windows\System32\cryptbase.dll
```

Memory dumps

What if we don’t have any pointer provided as an argument?

E.g. `NtTerminateProcess` is not memory-related but it’s still interesting to know the caller.

Memory dumps

What if we don’t have any pointer provided as an argument?

E.g. `NtTerminateProcess` is not memory-related but it’s still interesting to know the caller.

→ Perform stack walk.

Memory dumps

Known: current CPU context inside syscall

Unknown: 64 bit stack, 32 bit stack (SYSWOW64)

64 bit: `_KTHREAD->TrapFrame->Rsp`

Memory dumps

Known: current CPU context inside syscall

Unknown: 64 bit stack, 32 bit stack (SYSWOW64)

64 bit: `_KTHREAD->TrapFrame->Rsp`

32 bit: `(WOW_CONTEXT*)(_KTHREAD->Teb->TlsSlots[1] + 4)->Esp/Ebp`

```
                                        ^^^^^^^^^^^^^^^^
```

??? but why ???

Memory dumps

Stack unwinding?

```
for (int i = 0; i < 500; i++) {
    addr_t ptr = *(rsp+i);

    if (looks_legit(ptr))
        add_stack_entry(ptr);
}
```

Memory dumps

Stack unwinding?

```
for (int i = 0; i < 500; i++) {
    addr_t ptr = *(rsp+i);

    if (looks_legit(ptr))
        add_stack_entry(ptr);
}
```

```
// TODO: fix
```

Usermode hooking

Usermode hooking

But why?

- hooks on syscalls are too low-level for us
- there are WinAPI functions that are not doing any syscalls at all
- usermode calls needed for behavioral analysis

DRAKVUF Demo #2: Crypto API

Assumptions:

- VM: Original Windows 7
  + Brand-new CLion

- Magic program on the host

```
1573737543.910192 Activating remapped gfns in the altp2m views!
1573737543.910275                Trap added @ PA 0x29445e4 RPA 0xff0075e4 Page
10564 for NtTerminateProcess.
1573737543.910294        ntoskrnl.exe @ 0xfffff800[illegible]
1573737543.910307 Physmap populated? 0
1573737543.910353 Copied trapped page to new location
1573737543.910362 Activating remapped gfns in the altp2m views!
1573737543.910445                Trap added @ PA 0x2932850 RPA 0xff018950 Page
10546 for NtWriteVirtualMemory.
1573737543.910509        ntoskrnl.exe @ 0xfffff800[illegible]
1573737543.910604        ntoskrnl.exe @ 0xfffff800[illegible]
1573737543.910620 Physmap populated? 0
1573737543.910674 Copied trapped page to new location
1573737543.910683 Activating remapped gfns in the altp2m views!
1573737543.910783                Trap added @ PA 0x29a89e0 RPA 0xff019e60 Page
10660 for NtMapViewOfSection.
1573737543.910802        ntoskrnl.exe @ 0xfffff800[illegible]
1573737543.910815 Physmap populated? 0
1573737543.910867 Copied trapped page to new location
1573737543.910874 Activating remapped gfns in the altp2m views!
1573737543.910957                Trap added @ PA 0x2[illegible]4c0 RPA 0xff[illegible]4c0 Page
9966 for KiSystemServiceHandler.
1573737543.910973        ntoskrnl.exe @ 0xfffff800[illegible]
1573737543.910988        ntoskrnl.exe @ 0xfffff800[illegible]
1573737543.911008 Physmap populated? 0
1573737543.911049 Copied trapped page to new location
1573737543.911050 Activating remapped gfns in the altp2m views!
1573737543.911141                Trap added @ PA 0x[illegible] RPA 0xff01b8e0 Page
9959 for MiCopyOnWrite.
1573737543.911154 Starting plugin memdump finished
1573737543.911159 Beginning DRAKVUF loop
1573737543.911163 Started DRAKVUF loop
```

Usermode hooking

Which syscalls are issued when a new DLL is loaded?

Usermode hooking

Which syscalls are issued when a new DLL is loaded?

Closest call: `NtMapViewOfSection` / `NtProtectVirtualMemory`

Usermode hooking

Which syscalls are issued when a new DLL is loaded?

Closest call: `NtMapViewOfSection` / `NtProtectVirtualMemory`

DLLs are loaded...
But they don’t exist in the physical memory (yet).

Usermode hooking

DRAKVUF can’t add breakpoint on a memory which is not yet mapped :(

So...

Usermode hooking

Inject a page fault through VMX from the Xen hypervisor level:

Add vmi_request_page_fault to libvmi

Browse files

Signed-off-by: Alexandru Isaila <aisaila@bitdefender.com>

master (#762)

aisaila committed on 2 May

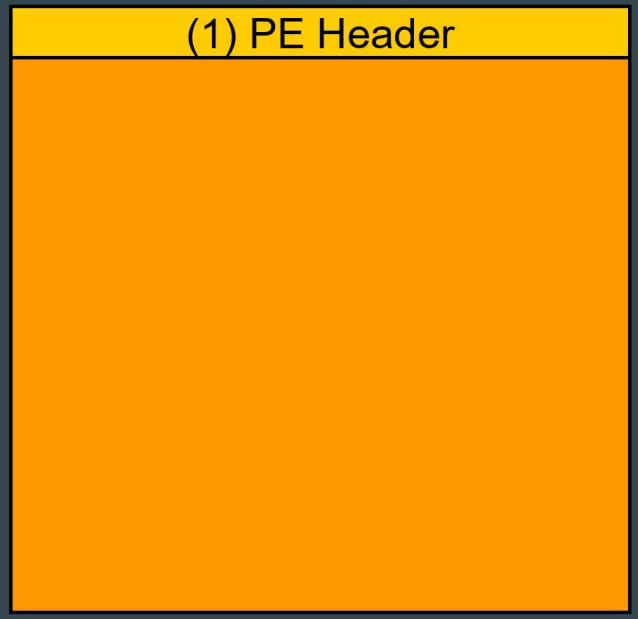
1 parent 23b05b0 commit 34ec2e5df0c0d0eba4d835dae8fa49f38215c440

Thanks BitDefender! :)

Usermode hooking

How to reach the interesting DLL export?

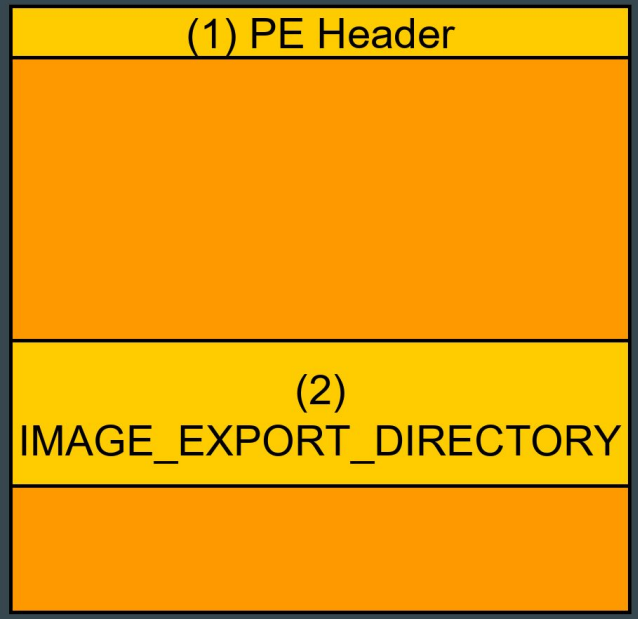
1. Parse the PE header



Usermode hooking

How to reach the interesting DLL export?

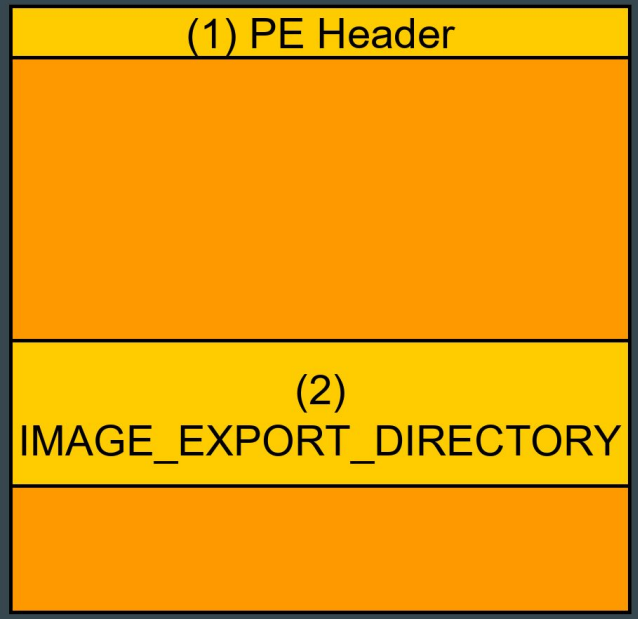
1. Parse the PE header
2. Find image export directory



Usermode hooking

How to reach the interesting DLL export?

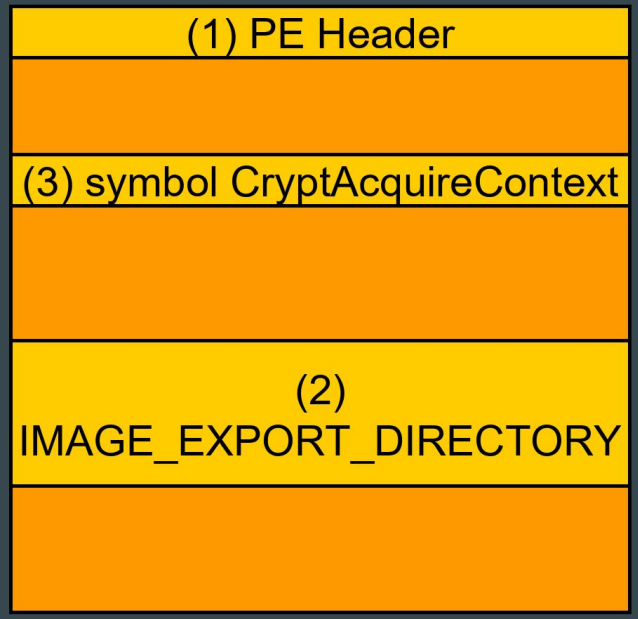
1. Parse the PE header
2. Find image export directory
3. Not readable? Page fault the export directory



Usermode hooking

How to reach the interesting DLL export?

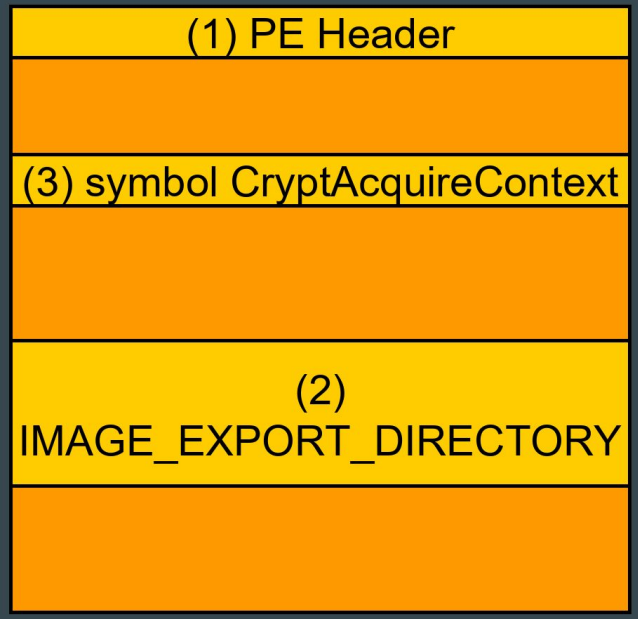
1. Parse the PE header
2. Find image export directory
3. Not readable? Page fault the export directory
4. Find out the RVA of export



Usermode hooking

How to reach the interesting DLL export?

1. Parse the PE header
2. Find image export directory
3. Not readable? Page fault the export directory
4. Find out the RVA of export
5. The first instruction of the exported function is not accessible? Page fault



Usermode hooking

What if the DLL would be (purposely?) corrupted and the pointer to `IMAGE_EXPORT_DIRECTORY` would be invalid?

Usermode hooking

What if the DLL would be (purposely?) corrupted and the pointer to `IMAGE_EXPORT_DIRECTORY` would be invalid?

Our injected page fault would crash the whole Windows system.

Usermode hooking

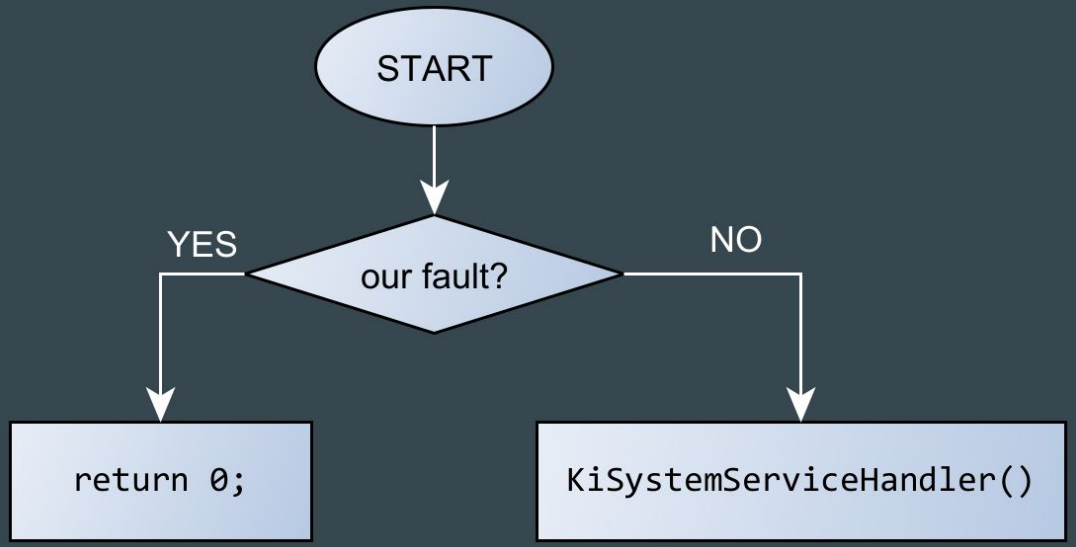
What if the DLL would be (purposely?) corrupted and the pointer to `IMAGE_EXPORT_DIRECTORY` would be invalid?

Our injected page fault would crash the whole Windows system.

Let's hook `KiSystemServiceHandler` ("BSOD handler") and pretend that nothing has happened.

Usermode hooking

Hook the kernel's exception handler:



Usermode hooking

Hook the kernel's exception handler:

Usermode hooking

Malware could attempt to override it’s own WinAPI function

- DLLs are shared between processes, Copy On Write occurs when they are overridden
- solution: hook `MiCopyOnWrite`
- rewrite hooks to the new physical page

iexplore.exe - the best test program

iexplore.exe - overriding it’s own DLLs

```
.text:7666FBD1                 ; int __stdcall MessageBoxIndirectA(const MSGBOXPARAMSA
.text:7666FBD1                                 public _MessageBoxIndirectA@4
.text:7666FBD1                 _MessageBoxIndirectA@4 proc near        ; DATA XREF: .te
.text:7666FBD1
.text:7666FBD1                 var_68          = byte ptr -68h
.text:7666FBD1                 MultiByteString = dword ptr -5Ch
.text:7666FBD1                 var_58          = dword ptr -58h
.text:7666FBD1                 var_8           = dword ptr -8
.text:7666FBD1                 P               = dword ptr -4
.text:7666FBD1                 lpmbp           = dword ptr  8
.text:7666FBD1
.text:7666FBD1 8B FF                           mov     edi, edi
.text:7666FBD3 55                              push    ebp
.text:7666FBD4 8B EC                           mov     ebp, esp
.text:7666FBD6 83 EC 68                        sub     esp, 68h
.text:7666FBD9 53                              push    ebx
.text:7666FBDA 56                              push    esi
.text:7666FBDB 57                              push    edi
.text:7666FBDC 33 DB                           xor     ebx, ebx
.text:7666FBDE 6A 60                           push    60h ; '`'       ; Size
.text:7666FBE0 8D 45 98                        lea     eax, [ebp+var_68]
.text:7666FBE3 53                              push    ebx             ; Val
```

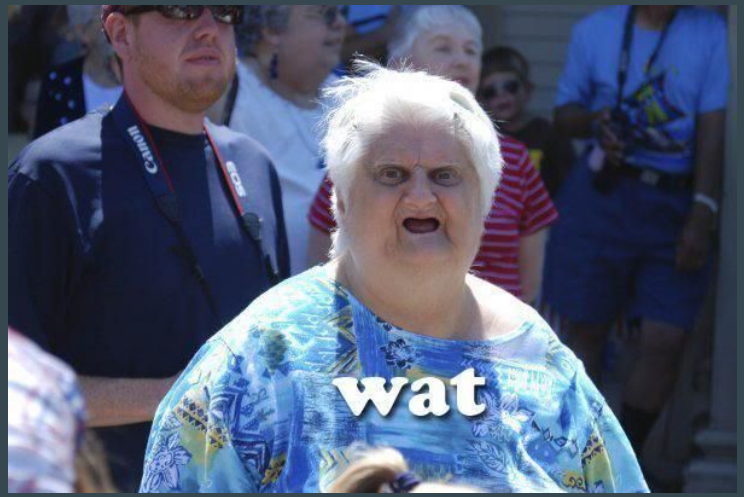
```
7666FBCC    90              nop
7666FBCD    90              nop
7666FBCE    90              nop
7666FBCF    90              nop
7666FBD0    90              nop
7666FBD1    E9 34E08AFA     jmp ieframe.70F1DC0A
7666FBD6    83EC 68         sub esp,68
7666FBD9    53              push ebx
7666FBDA    56              push esi
7666FBDB    57              push edi
7666FBDC    33DB            xor ebx,ebx
```

regular vs internet explorer

iexplore.exe - overriding it’s own DLLs

DLLs overridden by IE:

- `comdlg32.dll`
- `ole32.dll`
- `oleaut32.dll`
- `user32.dll`
- `comctl32.dll`



Traffic decryption PoC

Super secret data

Intel Processor Trace (work in progress)

#xen-devel

15:22 <andyhhp__> oh wow - we've got Cert.pl implementing a VM feature which we couldn't even perusade Intel to do

15:22 <andyhhp__> this is going to be interesting

Intel Processor Trace

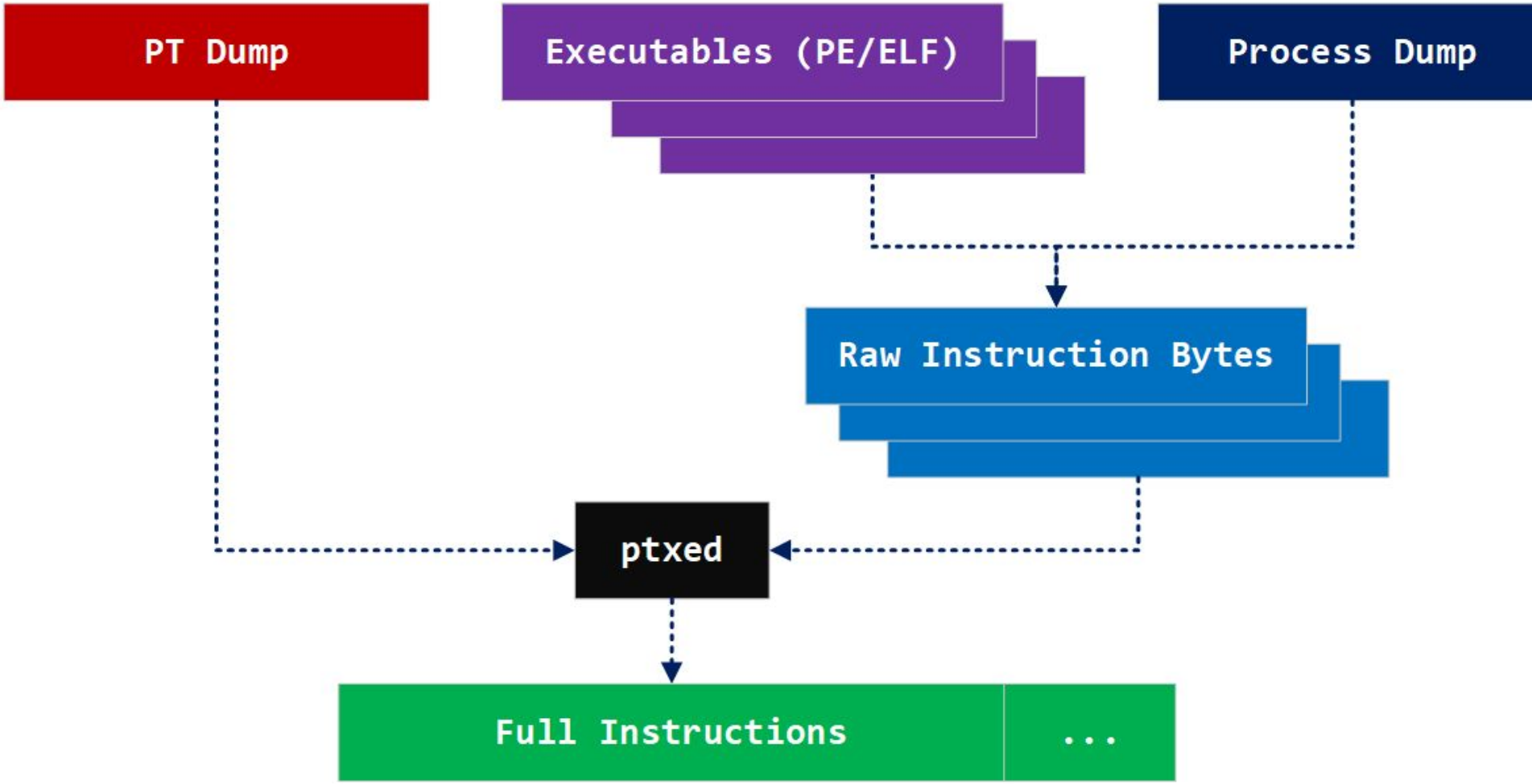
Decompile: FUN_00401250 - (netwalker....

```
21  if (DAT_00411264 == (int *)0x0) {
22    return DAT_00411260;
23  }
24  iVar4 = FUN_00401000(iVar1,-0x5e2ba68c);
25  *DAT_00411264 = iVar4;
26  iVar4 = FUN_00401000(iVar1,-0x50ee43dc);
27  DAT_00411264[1] = iVar4;
28  iVar4 = FUN_00401000(iVar1,-0x468c4724);
29  DAT_00411264[2] = iVar4;
30  iVar4 = FUN_00401000(iVar1,-0x7b9c69f6);
31  DAT_00411264[3] = iVar4;
32  iVar4 = FUN_00401000(iVar1,-0x2ebe502d);
33  DAT_00411264[4] = iVar4;
34  iVar4 = FUN_00401000(iVar1,0x57f17b6b);
35  DAT_00411264[5] = iVar4;
36  iVar4 = FUN_00401000(iVar1,0x23398d9a);
37  DAT_00411264[6] = iVar4;
38  iVar4 = FUN_00401000(iVar1,-0x4298ca3d);
39  DAT_00411264[9] = iVar4;
40  iVar4 = FUN_00401000(iVar1,-0x6ff09592);
41  DAT_00411264[10] = iVar4;
42  iVar4 = FUN_00401000(iVar1,-0x57518bee);
43  DAT_00411264[7] = iVar4;
44  iVar4 = FUN_00401000(iVar1,0x4896a43);
45  DAT_00411264[8] = iVar4;
46  iVar4 = FUN_00401000(iVar1,0x4c8a5b22);
47  DAT_00411264[0xb] = iVar4;
48  iVar4 = FUN_00401000(iVar1,0x61e2048f);
49  DAT_00411264[0xc] = iVar4;
```

Decompile: FUN_00401250 - (netwalker....

```
21  if (DAT_00411264 == (int *)0x0) {
22    return DAT_00411260;
23  }
24  iVar4 = FUN_00401000(iVar1,-0x5e2ba68c);
25  *DAT_00411264 = iVar4;
26  iVar4 = FUN_00401000(iVar1,-0x50ee43dc);
27  DAT_00411264[1] = iVar4;
28  iVar4 = FUN_00401000(iVar1,-0x468c4724);
29  DAT_00411264[2] = iVar4;
30  iVar4 = FUN_00401000(iVar1,-0x7b9c69f6);
31  DAT_00411264[3] = iVar4;
32  iVar4 = FUN_00401000(iVar1,-0x2ebe502d);
33  DAT_00411264[4] = iVar4;
34  iVar4 = FUN_00401000(iVar1,0x57f17b6b);
35  DAT_00411264[5] = iVar4;
36  iVar4 = FUN_00401000(iVar1,0x23398d9a);
37  DAT_00411264[6] = iVar4;
38  iVar4 = FUN_00401000(iVar1,-0x4298ca3d);
39  DAT_00411264[9] = iVar4;
40  iVar4 = FUN_00401000(iVar1,-0x6ff09592);
41  DAT_00411264[10] = iVar4;
42  iVar4 = FUN_00401000(iVar1,-0x57518bee);
43  DAT_00411264[7] = iVar4;
44  iVar4 = FUN_00401000(iVar1,0x4896a43);
45  DAT_00411264[8] = iVar4;
46  iVar4 = FUN_00401000(iVar1,0x4c8a5b22);
47  DAT_00411264[0xb] = iVar4;
48  iVar4 = FUN_00401000(iVar1,0x61e2048f);
49  DAT_00411264[0xc] = iVar4;
```

```
qmemcpy(gdiplus_dll_str, "GDIPLUS", 7);
gdiplus_dll_str[7] = 0xE;
strcpy(v159, "DLL");
do
  gdiplus_dll_str[v0++] ^= v157;
while ( v0 < 0xB );
v1 = KERNEL32_DLL;
v159[3] = 0;
if ( !KERNEL32_DLL )
{
  v1 = Resolve_Kernel32();
  KERNEL32_DLL = v1;
}
LoadLibraryA = LoadLibraryA_0;
if ( !LoadLibraryA_0 )
{
  LoadLibraryA = Resolve_LoadLibraryA(v1);
  LoadLibraryA_0 = LoadLibraryA;
}
(LoadLibraryA)(gdiplus_dll_str);
```


PT Dump
Executables (PE/ELF)
Process Dump
Raw Instruction Bytes
ptxed
Full Instructions
...

Instructions
push
mov
add
cmp
je .label
mov
.label:
call (edx) // virtual function

PT trace
Taken
0x407e1d8

Decoded
push
mov
add
cmp
je .label
mov
.label:
call (0x407e1d8)

Execution

```
0000000077b117cd  pop r14
0000000077b117cf  pop rsi
0000000077b117d0  ret
0000000077b13f11  test eax, eax
0000000077b13f13  jns 0x77b54241
0000000077b13f19  xor ebx, ebx
0000000077b13f1b  mov rcx, qword ptr [rsp+0x88]
0000000077b13f23  call 0x77b21400
0000000077b21400  mov r10, rcx
0000000077b21403  mov eax, 0xc
0000000077b21408  syscall
```

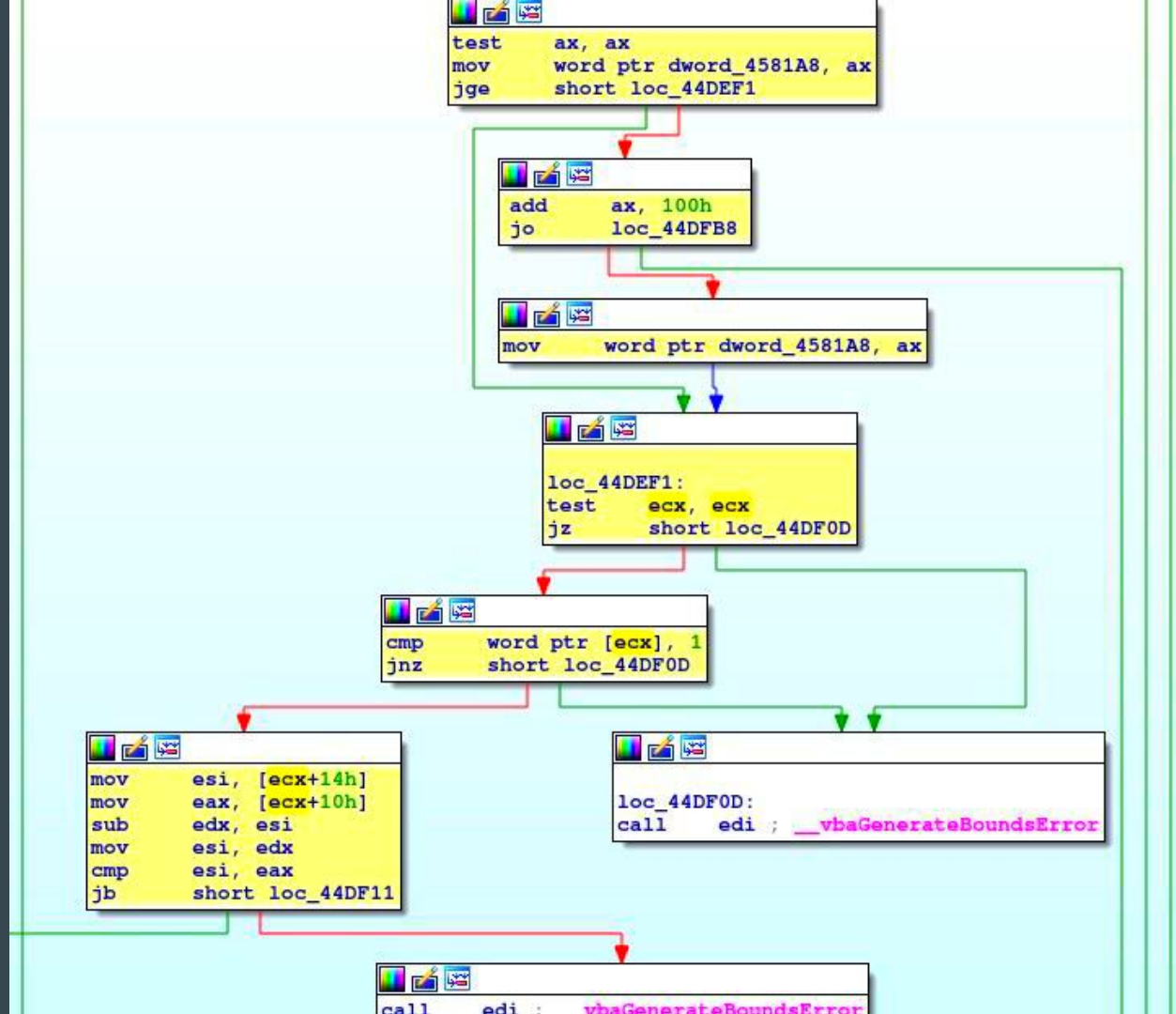
IPT disassembly

```
[disabled]
[drakvuf_event: {
  "Plugin": "syscall",
  "TimeStamp": "1602506121.043932",
  "PID": 2200,
  "PPID": 2080,
  "TID": 2204,
  "UserName": "SessionID",
  "UserId": 1,
  "ProcessName": "\\Device\\HarddiskVolume2\\Windows\\System32\\wermgr.exe",
  "Method": "NtClose",
  "EventUID": "0xe7218",
  "Module": "nt",
  "vCPU": 0,
  "CR3": "0xa8ff0000",
  "Syscall": 12,
  "NArgs": 1,
  "Handle": "0x8"
}]
```

DRAKVUF tracing

```
[resumed]
[exec mode: 64-bit]
0000000077b2140a  ret
0000000077b13f28  cmp ebx, 0xffffffff
0000000077b13f2b  jz 0x77b5427d
0000000077b13f31  cmp ebx, 0x1
0000000077b13f34  jz 0x77b5426e
0000000077b13f3a  cmp ebx, 0x2
0000000077b13f3d  jz 0x77b5425f
```

IPT disassembly


test ax, ax
mov word ptr dword_4581A8, ax
jge short loc_44DEF1
add ax, 100h
jo loc_44DFB8
mov word ptr dword_4581A8, ax
loc_44DEF1:
test ecx, ecx
jz short loc_44DF0D
cmp word ptr [ecx], 1
jnz short loc_44DF0D
mov esi, [ecx+14h]
mov eax, [ecx+10h]
sub edx, esi
mov esi, edx
cmp esi, eax
jb short loc_44DF11
loc_44DF0D:
call edi ; __vbaGenerateBoundsError
call edi ; __vbaGenerateBoundsError

Decompile: FUN_00401250 - (netwalker....

```
21  if (DAT_00411264 == (int *)0x0) {
22    return DAT_00411260;
23  }
24  iVar4 = FUN_00401000(iVar1,-0x5e2ba68c);
25  *DAT_00411264 = iVar4;
26  iVar4 = FUN_00401000(iVar1,-0x50ee43dc);
27  DAT_00411264[1] = iVar4;
28  iVar4 = FUN_00401000(iVar1,-0x468c4724);
29  DAT_00411264[2] = iVar4;
30  iVar4 = FUN_00401000(iVar1,-0x7b9c69f6);
31  DAT_00411264[3] = iVar4;
32  iVar4 = FUN_00401000(iVar1,-0x2ebe502d);
33  DAT_00411264[4] = iVar4;
34  iVar4 = FUN_00401000(iVar1,0x57f17b6b);
35  DAT_00411264[5] = iVar4;
36  iVar4 = FUN_00401000(iVar1,0x23398d9a);
37  DAT_00411264[6] = iVar4;
38  iVar4 = FUN_00401000(iVar1,-0x4298ca3d);
39  DAT_00411264[9] = iVar4;
40  iVar4 = FUN_00401000(iVar1,-0x6ff09592);
41  DAT_00411264[10] = iVar4;
42  iVar4 = FUN_00401000(iVar1,-0x57518bee);
43  DAT_00411264[7] = iVar4;
44  iVar4 = FUN_00401000(iVar1,0x4896a43);
45  DAT_00411264[8] = iVar4;
46  iVar4 = FUN_00401000(iVar1,0x4c8a5b22);
47  DAT_00411264[0xb] = iVar4;
48  iVar4 = FUN_00401000(iVar1,0x61e2048f);
49  DAT_00411264[0xc] = iVar4;
```

Decompile: FUN_00401250 - (netwalker....

```
21    if (DAT_00411264 == (int *)0x0) {
22      return DAT_00411260;
23    }
24    iVar4 = FUN_00401000(iVar1,-0x5e2ba68c);
25    *DAT_00411264 = iVar4;
26    iVar4 = FUN_00401000(iVar1,-0x50ee43dc);
27    DAT_00411264[1] = iVar4;
28    iVar4 = FUN_00401000(iVar1,-0x468c4724);
29    DAT_00411264[2] = iVar4;
30    iVar4 = FUN_00401000(iVar1,-0x7b9c69f6);
31    DAT_00411264[3] = iVar4;
32    iVar4 = FUN_00401000(iVar1,-0x2ebe502d);
33    DAT_00411264[4] = iVar4;
34    iVar4 = FUN_00401000(iVar1,0x57f17b6b);
35    DAT_00411264[5] = iVar4;
36    iVar4 = FUN_00401000(iVar1,0x23398d9a);
37    DAT_00411264[6] = iVar4;
38    iVar4 = FUN_00401000(iVar1,-0x4298ca3d);
39    DAT_00411264[9] = iVar4;
40    iVar4 = FUN_00401000(iVar1,-0x6ff09592);
41    DAT_00411264[10] = iVar4;
42    iVar4 = FUN_00401000(iVar1,-0x57518bee);
43    DAT_00411264[7] = iVar4;
44    iVar4 = FUN_00401000(iVar1,0x4896a43);
45    DAT_00411264[8] = iVar4;
46    iVar4 = FUN_00401000(iVar1,0x4c8a5b22);
47    DAT_00411264[0xb] = iVar4;
48    iVar4 = FUN_00401000(iVar1,0x61e2048f);
49    DAT_00411264[0xc] = iVar4;
```

```
25    *DAT_00411264 = iVar4;
26    iVar4 = FUN_00401000(iVar1,-0x50ee43dc);
27                        /* ntdll.RtlFreeHeap */
28    DAT_00411264[1] = iVar4;
29    iVar4 = FUN_00401000(iVar1,-0x468c4724);
30                        /* ntdll.RtlReAllocateHeap */
31    DAT_00411264[2] = iVar4;
32    iVar4 = FUN_00401000(iVar1,-0x7b9c69f6);
33                        /* ntdll.memset */
34    DAT_00411264[3] = iVar4;
35    iVar4 = FUN_00401000(iVar1,-0x2ebe502d);
36                        /* ntdll.memcpy */
37    DAT_00411264[4] = iVar4;
38    iVar4 = FUN_00401000(iVar1,0x57f17b6b);
39                        /* ntdll.memcmp */
40    DAT_00411264[5] = iVar4;
41    iVar4 = FUN_00401000(iVar1,0x23398d9a);
42                        /* ntdll.sprintf */
43    DAT_00411264[6] = iVar4;
44    iVar4 = FUN_00401000(iVar1,-0x4298ca3d);
45                        /* ntdll.strcpy */
46    DAT_00411264[9] = iVar4;
47    iVar4 = FUN_00401000(iVar1,-0x6ff09592);
48                        /* ntdll.strcat */
49    DAT_00411264[10] = iVar4;
50    iVar4 = FUN_00401000(iVar1,-0x57518bee);
51                        /* ntdll.strchr */
52    DAT_00411264[7] = iVar4;
53    iVar4 = FUN_00401000(iVar1,0x4896a43);
```

DRAKVUF Sandbox

DRAKVUF Sandbox

Wrapper for DRAKVUF Engine with:

- web interface
- easy installation
- sample queueing
- … much more coming soon!

DRAKVUF
Sandbox

SANDBOX

- Upload sample
- Analyses

ANALYSIS

- Report
- API calls
- Logs

Report

Process tree

- unnamed process (0)
- unnamed process (368)
  - csrss.exe (384)
    - conhost.exe (2716)
  - winlogon.exe (424)
- unnamed process (1456)
  - explorer.exe (1544)
    - cmd.exe (1384)
    - test.exe (1620)

Metadata

SHA256	ebd2f6fa793e97fd1f48b8e5f03fafb183bf606747bed0863e3f950411a3824d
Magic bytes	PE32+ executable (console) x86-64, for MS Windows
Start command	C:\Users\janusz\Desktop\test.exe
Started at	2020-11-23 10:41:18
Finished at	2020-11-23 10:42:33

DRAKVUF
Sandbox

SANDBOX

Upload sample

Analyses

ANALYSIS

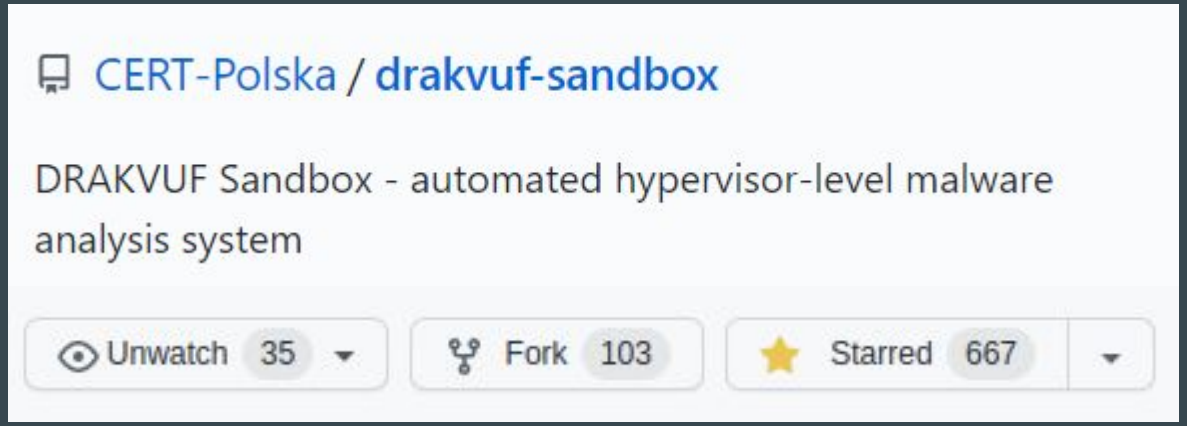
Report

API calls

1594640303.923439	LdrGetProcedureAddress	0x77670000 0x1af220 0x0 0x1af248
1594640303.923658	LdrGetProcedureAddress	0x77670000 0x1af220 0x0 0x1af248
1594640303.953502	WriteConsoleW	0x7 0x2823f0 0x25 0x1af650 0x0
1594640303.953871	LdrLoadDll	0x27f4e0 0x1af670 0x1af628:"dhcpcsvc.dll" 0x1af688
1594640303.954152	LdrGetProcedureAddress	0x7fefbc00000 0x1af650 0x0 0x1af678
1594640303.965090	LdrLoadDll	0x27f4e0 0x1af050 0x1af008:"dhcpcsvc6.DLL" 0x1af068
1594640303.965328	LdrGetProcedureAddress	0x7fefbc80000 0x1af080 0x0 0x1af0a8
1594640303.965545	LdrLoadDll	0x27f4e0 0x1aed70 0x1aed28:"IPHLPAPI.DLL" 0x1aed88
1594640303.965811	LdrGetProcedureAddress	0x7fefbe00000 0x1aeda0 0x0 0x1aedc8
1594640303.966066	LdrLoadDll	0x27f4e0 0x1ae6f0 0x1ae6a8:"rpcrt4.dll" 0x1ae708

GitHub project

Fully open-source and free ;)



DRAKVUF IS NOT

- magical box that tells you if your email attachment is a malware

DRAKVUF IS NOT

- magical box that tells you if your email attachment is a malware

DRAKVUF IS

- sandbox which provides crucial information for malware analyst

Summary

GitHub

- LibVMI
  https://github.com/libvmi/libvmi

- DRAKVUF
  https://github.com/tklengyel/drakvuf

- DRAKVUF Sandbox
  https://github.com/CERT-Polska/drakvuf-sandbox

Kudos

- **CERT.PL Reverse Engineers - nazywam, psrok1, msm**
  → for many important remarks and hints about malware monitoring

- **chivay, konstantyc**
  → co-development of DRAKVUF/DRAKVUF Sandbox

Kudos

- **Maciej “mak” Kotowicz**
  → for providing many good heurstics for memory dumping (and some hints about them)

- **Tamas K. Lengyel**
  → a lots of helpful remarks during our research
  → creator/maintainer of DRAKVUF project na GitHub

- **Mathieu (Wenzel) Tarral**
  → libvmi maintainer
  → gathering VMI community together

Thank you!

Slides:
https://icedev.pl/confidence22