

DISSECTING AND DEFEATING RANSOMWARE'S EVASION TACTICS

DefCon 32 Workshop

- “The 3 Malware Amigos”:
- Ryan Chapman
- Josh Stroschein
- Aaron Rosenmund

[\[@rj_chap\]](#)

[\[@jstrosch \]](#)

[\[@arosenmund\]](#)

ABOUT RYAN

Principal Threat Hunt Researcher

- SANS Author
 - FOR528: Ransomware for Incident Responders
- SANS Instructor
 - FOR610: Reverse Engineering Malware
- SANS Ransomware Summit Chair
 - for528.com/summit24
- CactusCon CFP Board
 - Former conference lead
- Pluralsight Author
- DFIR | Malware Analysis | Mouth Runner



@rj_chap

ABOUT JOSH

Reverse Engineer @ FLARE/Google

- The Cyber Yeti
 - Go check it out www.thecyberyeti.com
 - Discord, Github projects, etc
- Pluralsight Author
 - Malware/RE/Network stuff
- Occasional YouTuber
 - <https://www.youtube.com/@jstrosch>
- Regular trainer/speaker
 - Join me in Madrid for Suricon 2024!
- Oh, and I brought swag!



@jstrosch

ABOUT AARON

Sr. Dir. Content @ Pluralsight

- **Pluralsight**

- Create and manage the content and curriculum for IT/DEV & Cyber security training.
- Authored 120+ courses & labs. Including one about crowdstrike that just published.
- I have a website I never update:
Www.AaronRosenmund.com
- Twitter/X @arosenmnd
- Cyber Shield Red Team Lead ANG
 - Join our discord discord.gg/darkkittens
 - I'm @ironcat
- Threat Emulation various consulting groups:
 - Trulight
 - Ondefend



@arosenmund

What Are We Going to Cover?

- The lab environment
 - Ransomware Builder Analysis
 - Ryan
 - Reverse Engineering Ransomware to Find Evasion Techniques
 - Josh
 - Writing Custom Evasion Procedures
 - Aaron
 - Recap & Questions

The 3 Malware Amigos



Getting Started

All Instructions and Links are on the github associated with this lab:

github.com/arosenmund/defcon32_dissecting_defeating_ransomwares_evasion

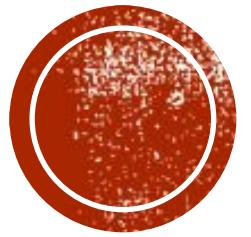
To launch the lab environment:

[Www.Pluralsight.com/free](https://www.pluralsight.com/free) (or use an existing account)

Once logged in search:

Defcon 32 Workshop: Dissecting and Defeating Ransomware's Evasion Tactics
and start the lab.





RANSOMWARE BUILDER ANALYSIS

Ryan Chapman

Ransomware Builders

- Facilitate the turnkey nature of RaaS operations
 - Developers create “builders” to generate samples for each affiliate/partner (i.e., they create customized payloads)
 - The builder generates customized core components:
 - Payload/cryptor
 - Encryption keys (often public/private key pair)
 - Decryptor
 - Dashboard access



LockBit 3.0's Web Builder Interface

Stealer LockBit RED **LockBit GREEN** LockBit BLACK Linux/ESXI Chat generation

LockBit GREEN

BUILD DATE ?
 15.08.23 08:26

COMMENT ?

COMPANY WEBSITE ?

REVENUE ?

Select the number of builds, the default is one. (Optional)
 ▼

Select encryption size
 ▼

RUNNING ONE ?

QUIET MODE ?

LOCAL DISKS ?

NETWORK SHARES ?

SAME ENCRYPTION KEY ?

MAXIMUM DECRYPTOR PROTECTION ?

○ **GET LOCKBIT GREEN**

DEFAULT SETTINGS



Options A-Plenty: Chaos Ransomware Builder

Chaos Ransomware Builder v5.2

—> Chaos is multi language ransomware. Translate your note to any language <—

All of your files have been encrypted

Your computer was infected with a ransomware virus. Your files have been encrypted and you won't be able to decrypt them without our help. What can I do to get my files back? You can buy our special decryption software, this software will allow you to recover all of your data and remove the ransomware from your computer. The price for the software is \$1,500. Payment can be made in Bitcoin only.

How do I pay, where do I get Bitcoin?

Purchasing Bitcoin varies from country to country, you are best advised to do a quick google search yourself to find out how to buy Bitcoin.

Many of our customers have reported these sites to be fast and reliable:

Coinmama - <http://www.coinmama.com> Bitpanda - <http://www.bitpanda.com>

Payment information
Amount: 0.528 BTC
Bitcoin Address: for528r0ckzgiveuzyermoniezkhnxbitcoimplz0

Randomize file extension: Usb and network spread: Process Name: Dropped File Name

encrypted surprise svhost.exe read_it.txt

Delay second Add to startup 10 Select Icon

About Build

FileExtensions Advanced Options



The 3 Malware Amigos

Advanced Options

Resist for admin privileges

Delete all Volumes Shadow Copies

Delete the backup catalog

Disable windows recovery mode

Disable task manager

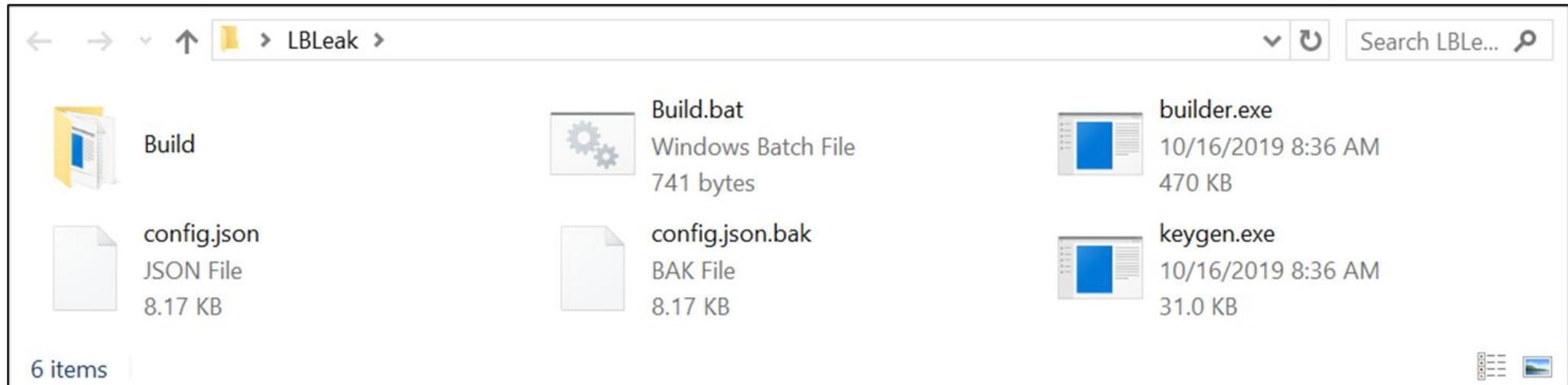
Change desktop wallpaper

Encrypt AES / RSA

Files will be encrypted with AES/RSA method

Decrypter Name

Our Focus: LockBit 3.0's Builder



LB3 Builder Components

- **config.json**
 - Configuration parameters in JavaScript Objection Notation (JSON) format
- **Build.bat**
 - Clears the Build directory
 - Runs keygen.exe to generate keys
 - Runs builder.exe to generate the ransomware payload(s)
- **keygen.exe**
 - Creates public and private key pairs
- **builder.exe**
 - Builds the actual ransomware – This is our focus!
 - Uses config.json for configuration



config.json

DEMO

```
{  
  "bot": {  
    "uid": "00000000000000000000000000000000",  
    "key": "00000000000000000000000000000000"  
  },  
  "config": {  
    "settings": {  
      "encrypt_mode": "auto",  
      "encrypt_filename": false,  
      "impersonation": true,  
      "skip_hidden_folders": false,  
      "language_check": false,  
      "local_disks": true,  
      "network_shares": true,  
      "kill_processes": true,  
      "kill_services": true,  
      "running_one": true,  
      "print_note": true,  
      "set_wallpaper": true,  
      "set_icons": true,  
      "send_report": false,  
      "self_destruct": true,  
      "script": ""  
    }  
  }  
}
```



Build.bat

DEMO

```
ERASE /F /Q %cd%\Build\*.*  
keygen -path %cd%\Build -pubkey pub.key -privkey priv.key  
builder -type dec -privkey %cd%\Build\priv.key -config  
config.json -ofile %cd%\Build\LB3Decryptor.exe  
builder -type enc -exe -pubkey %cd%\Build\pub.key -config  
config.json -ofile %cd%\Build\LB3.exe  
builder -type enc -exe -pass -pubkey %cd%\Build\pub.key  
-config config.json -ofile %cd%\Build\LB3_pass.exe  
builder -type enc -dll -pubkey %cd%\Build\pub.key -config  
config.json -ofile %cd%\Build\LB3_Rundll32.dll  
builder -type enc -dll -pass -pubkey %cd%\Build\pub.key  
-config config.json -ofile %cd%  
\Build\LB3_Rundll32_pass.dll  
builder -type enc -ref -pubkey %cd%\Build\pub.key -config  
config.json -ofile %cd%  
\Build\LB3_ReflectiveDll_DllMain.dll  
exit
```



keygen.exe

- Generates private & public keys (priv.key & pub.key)
- Generates a decryption ID

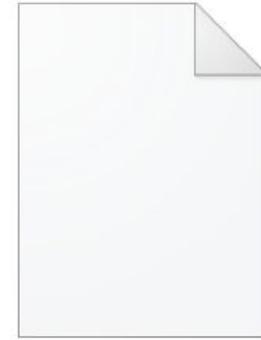
DEMO



DECRYPTION_ID.txt



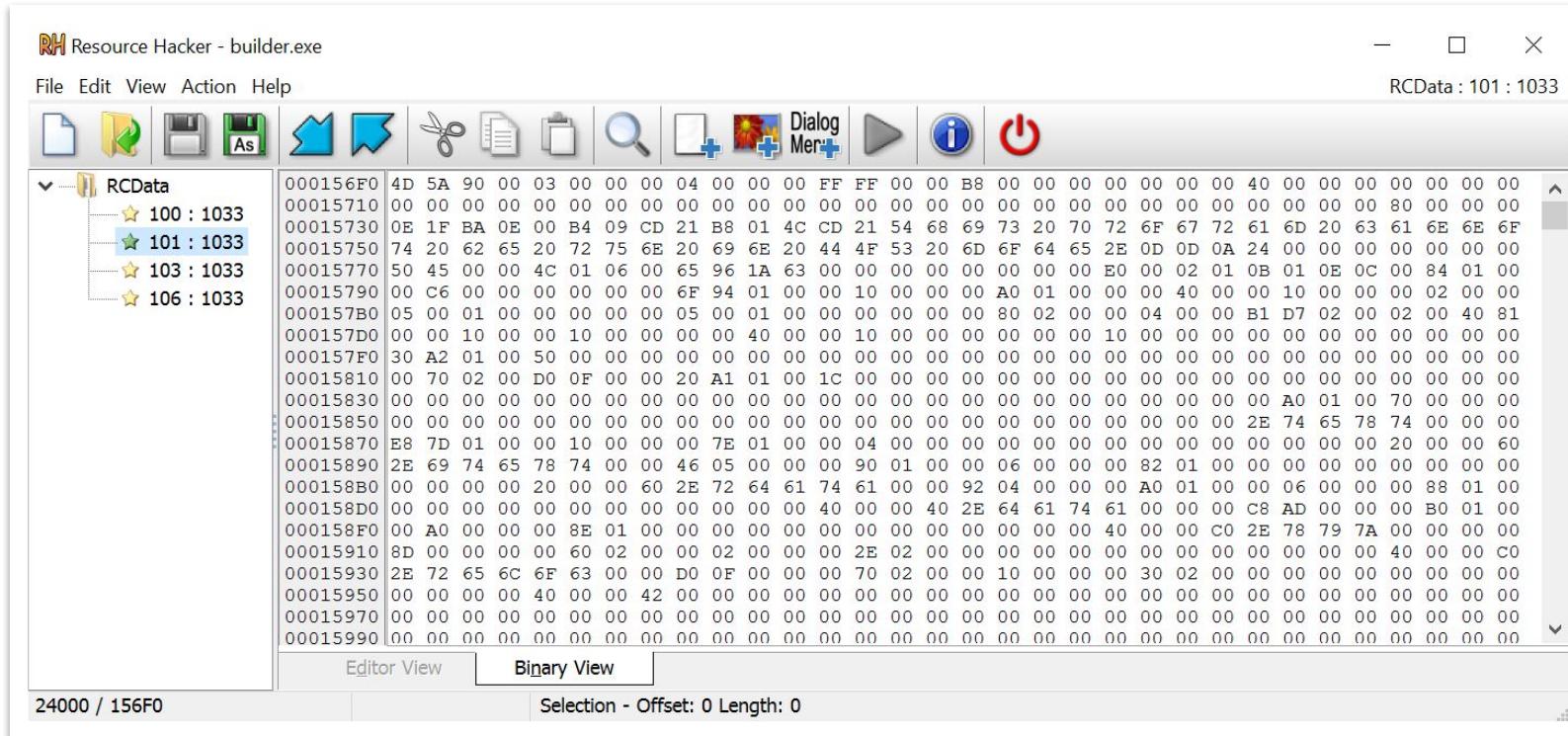
priv.key



pub.key

builder.exe Resources

- 100: Decryptor
- 101: EXE
- 103: DLL
- 106: Reflective DLL



builder.exe - Password

- pass parameter enables CLI password
 - If password is not on CLI, the payload won't run!

	FF	00	00	MZ
	00	00	00	.
	00	00	00	.
	00	00	00	.
	21	54	68	..°..
	6E	6E	6F	is pro
	4F	53	20	t be r
	00	00	00	mode..
	01	B5	B5	7`Ùæs..
	01	B5	B5	~Sjµi..
	01	B5	B5	~SUµi..
	01	B5	B5	s. µ..
	01	B5	B5	.xkpr..
	01	04	00	.
	00	02	01	áz#X..
	00	00	00ä..
	00	40	00	r ..
	00	00	00	.
	04	00	00	.
	10	00	00	.
	00	00	00	.
	00	00	00	.
	00	00	00	.
	1C	00	00	.
	00	00	00	.
	00	00	00	.
	00	00	00	"p..@..
	00	00	00T..
	00	00	00	.
	10	00	00	.text..
	00	00	00	.ä..
	61	00	00	.
	E8	01	00	ž..
	00	00	40	.
	90	02	00	.data..
	00	00	00	.
	63	00	00	..@..



What We're About to Do

- Review the builder
 - Identify where the resources are pulled
 - Identify the password generator function
 - Review the function
 - Walk through password generation
 - Identify the function that pulls the template
 - Walk through ransomware binary writing



builder.exe Payload Encryption (-pass)

- "C:\Users\LegitUser\Desktop\LBLeak\builder.exe" -type enc -exe -pass -pubkey pub.key -config config.json -ofile LB3-dc32.exe

			MZ
F	FF	00	00
0	00	00	00
0	00	00	00
3	00	00	00
D	21	54	68
1	6E	6E	6F
4	4F	53	20
0	00	00	00
3	01	B5	B5
0	01	B5	B5
6	01	B5	B5
0	01	B5	B5
3	01	B5	B5
C	01	04	00
0	00	02	01
0	00	00	00
0	00	40	00
0	00	00	00
0	04	00	00
0	10	00	00
0	00	00	00
C	00	00	00
0	00	00	00
3	1C	00	00
0	00	00	00
0	00	00	00
0	00	00	00
0	00	00	00
0	00	00	00
0	00	00	00
0	00	00	00
0	00	00	00
0	00	00	00
0	10	00	00
0	00	00	00
0	00	00	00
4	61	00	00
0	E8	01	00
0	00	00	40
0	90	02	00
0	00	00	00
F	63	00	00

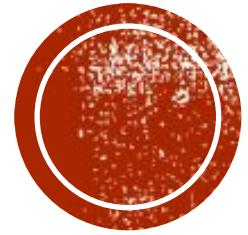


LAB TIME!



The 3 Malware Amigos





REVERSE ENGINEERING RANSOMWARE TO FIND EVASION TECHNIQUES

Josh Stroschein

First Up, Packing

- Builder creates two versions of the DLL/EXE
 - *_pass* added to file name

```
### Global Mode:  
LBB_pass.exe -pass 5dfd6c522cacd8de02d682e67f503d39
```

```
### Safe Mode:  
LBB_pass.exe -safe -pass 5dfd6c522cacd8de02d682e67f503d39
```

```
### Target Mode:  
LBB_pass.exe -path C:\file -pass 5dfd6c522cacd8de02d682e67f503d39  
LBB_pass.exe -path C:\folder -pass 5dfd6c522cacd8de02d682e67f503d39  
LBB_pass.exe -path C:\ -pass 5dfd6c522cacd8de02d682e67f503d39  
LBB_pass.exe -path \\?\Volume{11111111-2222-3333-4444-555555555555}\ -pass 5dfd6c522cacd8de02d682e67f503d39
```

```
public start

nop
nop    word ptr [eax+eax+00000000h]
call   nullsub_1
nop    dword ptr [eax+00h]
call   sub_40639C
xchg   ax, ax
call   sub_409990
nop    dword ptr [eax+00h]
call   sub_417458
```

```
op     aword ptr [eax+00001000]  
all    sub 40639C
```

b	66h, 90h)0	00	...
b	0E8h, 2, 5)1	00	ž...
d	1F0FFFFFh, 0C1E80040h, 66FFFFFFh, 441F0Fh,)0	40
d	4255C815h, 66666600h, 841F0Fh, 0)2	00	.data



Mash That Easy Button!

- Just analyze the non-packed version :)
- Or, you can break after unpacking function to start analyzing remaining functionality
- Uses XOR encryption with variable key
 - Decrypts *.text*, *.data*, and *.pdata* sections
- Could even analyze the builder logic...

The screenshot shows four windows of the IDA Pro debugger, each displaying assembly code:

- Top Window:** Shows code starting at .itext:00419063. It pushes a value of 0 onto the stack, then uses LEA to load the address of IMAGE_SECTION_HEADER.Name into EAX. A call to section_name_hash is made, followed by a CMP instruction comparing EAX with the value 76918075h (the address of the .text section). If they are equal, it jumps to short decrypt_text_section.
- Second Window:** Shows code starting at .itext:00419074. It compares EAX with the value 4A41Bh (the address of the .data section) and then jumps to short decrypt_text_section if they are equal.
- Third Window:** Shows code starting at .itext:0041907B. It compares EAX with the value 0B84B49Bh (the address of the .pdata section) and then jumps to short loc_41909A if they are equal.
- Bottom Window:** Shows the implementation of the decrypt_section function. It takes the virtual address of the section as input (ecx), adds some_int to it (eax), and then calls decrypt_text_section. It also pushes the section size and virtual address of the section onto the stack before making the call.

Determining Sections

Section	Virtual Address	Size	Type
.text	419063	00000000	Code
.data	419074	00000000	Data
.pdata	41907B	00000000	Data



Mash That Easy Button!

- Just analyze the non-packed version :)
 - Or, you can break after unpacking function to start analyzing remaining functionality
 - Uses XOR encryption with variable key
 - Decrypts *.text*, *.data*, and *.pdata* sections
 - Could even analyze the builder logic...

Determining Sections



API Hashing

Similar concept... but different

TLDR process

- Uses PEB to resolve *LdrLoadDLL* and *LdrGetProcAddress*
 - Uses precomputed hashes that are XORed before they are used and the module name is used as a seed
 - Function addresses are also XORed, then *trampolines* used to execute the function



Let's See This In Action!

```
.iext:0041946F
.iext:0041946F    public start
.iext:0041946F start:
.iext:0041946F    nop
.iext:00419470    nop    word ptr [eax+eax+00000000h]
.iext:00419479    call   unpack
.iext:0041947E    nop    dword ptr [eax+00h]
.iext:00419482    call   resolve_imports
...
```



```
new_heap_new_mem = sub_405AEC(0xF80F18E8);
if ( new_heap_new_mem )
{
    new_heap_new_mem = ((int (__stdcall *)(int, _DWORD, _DWORD, _DWORD, _DWORD,
266242,
0,
0,
0,
0,
0));
memory = new_heap_new_mem;
if ( new_heap_new_mem )
{
    if ( (((_DWORD *)new_heap_new_mem + 64) >> 28) & 4 ) != 0 )
        memory = __ROL4__(new_heap_new_mem, 1);
new_heap_new_mem = sub_405AEC(0x6E6047DB);
new_heap = new_heap_new_mem;
if ( new_heap_new_mem )
{
    resolve/apis/by/hash(&unk_42540C, dword_405EE8, memory, new_heap_new_mem);
    resolve/apis/by/hash(&unk_4254FC, dword_405FDC, memory, new_heap);
    resolve/apis/by/hash(&unk_4255EC, dword_4060D0, memory, new_heap);
    resolve/apis/by/hash(&unk_42568C, dword_406174, memory, new_heap);
    resolve/apis/by/hash(&unk_42569C, dword_406188, memory, new_heap);
    resolve/apis/by/hash(&unk_4256D4, dword_4061C4, memory, new_heap);
    resolve/apis/by/hash(&unk_425728, dword_40621C, memory, new_heap);
    resolve/apis/by/hash(&unk_42573C, dword_406234, memory, new_heap);
    resolve/apis/by/hash(&unk_425764, &byte_406260, memory, new_heap);
    resolve/apis/by/hash(&unk_42579C, dword_40629C, memory, new_heap);
    resolve/apis/by/hash(&unk_4257B0, dword_4062B4, memory, new_heap);
    resolve/apis/by/hash(&unk_4257B8, dword_4062C0, memory, new_heap);
    resolve/apis/by/hash(&unk_4257CC, sub_4062D8, memory, new_heap);
    resolve/apis/by/hash(&unk_4257F8, dword_406308, memory, new_heap);
    resolve/apis/by/hash(&unk_425810, dword_406324, memory, new_heap);
    resolve/apis/by/hash(&unk_42583C, dword_406354, memory, new_heap);
    resolve/apis/by/hash(&unk_42584C, dword_406368, memory, new_heap);
    resolve/apis/by/hash(&unk_425858, dword_406378, memory, new_heap);
    resolve/apis/by/hash(&unk_42586C, dword_406390, memory, new_heap);
    setinformationthread_hidedebugger(0);
    sub_417738(memory, (int (__stdcall *)(int, int, int))new_heap);
    return sub_40B470();
}
}
return new_heap_new_mem;
```



Example API Call

Hiding from a Debugger

The screenshot shows a debugger interface with assembly code. The code is as follows:

```
.text:0040B45F
.text:0040B45F loc_40B45F:
.text:0040B45F push    0
.text:0040B461 push    0
.text:0040B463 push    ecx
.text:0040B464 push    eax
.text:0040B465 call    dword_425498
.text:0040B46B pop     ebp
.text:0040B46C retn    4
.text:0040B46C setinformationthread_hidedebugger endp
.text:0040B46C
```

Address	Hex
00BD5498	98 37 6A 02
00BD54A8	D0 3C 6A 02
00BD54B8	A0 3F 6A 02
00BD54C8	E0 3E 6A 02
00BD54D8	28 3F 6A 02
00BD54E8	88 3F 6A 02
00BD54F8	F0 3D 6A 02
00BD5508	00 00 00 00
00BD5508	20 3E 6A 02
00BD5518	30 3D 6A 02
00BD5528	80 3E 6A 02
00BD5538	30 40 6A 02
00BD5548	B8 3F 6A 02
00BD5558	48 40 6A 02
00BD5568	A0 42 6A 02
00BD5578	A8 43 6A 02
00BD5588	B0 41 6A 02
00BD5598	00 43 6A 02
00BD55A8	E8 42 6A 02

026A3798	B8 189E3B58	mov eax, 583B9E18
026A379D	C1C0 09	rol eax, 9
026A37A0	- FFE0	jmp eax

EAX 773C30B0 <ntdll.ZwSetInformationThread>
EBX 00DF2000
ECX 00000011
EDX 02F85308
EBP 00F3FC48
ESP 00F3FC34
ESI 02F80000

Default (stdcall)

- 1: [esp+4] FFFFFFFE FFFFFFFE
- 2: [esp+8] 00000011 00000011
- 3: [esp+C] 00000000 00000000
- 4: [esp+10] 00000000 00000000
- 5: [esp+14] 00F3FC6C 00F3FC6C

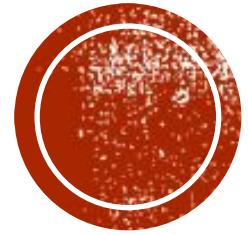


What Else
Can We
Find? !



**HANDS-ON TIME!
LET'S HAVE SOME FUN!**





WRITING CUSTOM EVASION PROCEDURES

Aaron Rosenmund

DumpLsass.exe Signatures

ee0696b8768775e6278261b5a56571c6b87f541d1b3d792169e27e8cc461a510

2 / 75 Community Score

2 / 75 security vendors flagged this file as malicious

ee0696b8768775e6278261b5a56571c6b87f541d1b3d792169e27e8cc461a510
dumpLsass.exe
peexe overlay 64bits

Size 145.66 KB Last Analysis Date a moment ago EXE

REANALYZE SIMILAR MORE

DETECTION DETAILS BEHAVIOR COMMUNITY

Crowdsourced YARA rules *i*

- ⚠ Matches rule CreateMiniDump from ruleset CreateMiniDump at <https://github.com/bartblaze/Yara-rules> by @bartblaze
 - ↳ Identifies CreateMiniDump, tool to dump LSASS. - a moment ago
- ⚠ Matches rule INDICATOR_TOOL_PWS_LSASS_CreateMiniDump from ruleset indicator_tools at <https://github.com/ditekshen/detection> by ditekSHen
 - ↳ Detects CreateMiniDump tool - a moment ago

Security vendors' analysis *i*

Do you want to automate checks?			
DeepInstinct	⚠ MALICIOUS	Microsoft	⚠ HackTool:Win32/DumpLsass.T
Acronis (Static ML)	✓ Undetected	AhnLab-V3	✓ Undetected
Alibaba	✓ Undetected	AliCloud	✓ Undetected

Dump1sass Yara Ruleset 1

These rulesets help define what you need to do for evasion.

Notice the condition.

Ruleset: CreateMiniDump

[ps://github.com/bartblaze/Yara-rules](https://github.com/bartblaze/Yara-rules)

[Go to matched rule](#) [Copy ruleset](#) [Download](#)

```
1 rule CreateMiniDump
2 {
3     meta:
4         id = "kMNDXhwJQURe8ehD0ueqk"
5         fingerprint = "b391a564b4730559271e11de0b80dce1562a9038c230a2be729a896913c7"
6         version = "1.0"
7         creation_date = "2021-03-01"
8         first_imported = "2021-12-30"
9         last_modified = "2021-12-30"
0         status = "RELEASED"
1         sharing = "TLP:WHITE"
2         source = "BARTBLAZE"
3         author = "@bartblaze"
4         description = "Identifies CreateMiniDump, tool to dump LSASS."
5         category = "HACKTOOL"
6         tool = "CREATEMINIDUMP"
7         reference = "https://www.ired.team/offensive-security/credential-access-and-"
8
9
0     strings:
1         $ = "[+] Got lsass.exe PID:" ascii wide
2         $ = "[+] lsass dumped successfully!" ascii wide
3         $ = { 40 55 57 4? 81 ec e8 04 00 00 4? 8d ?? ?4 40 4? 8b fc b9 3a 01 00 00
4             8b 05 ?? ?? ?? ?? 4? 33 c5 4? 89 8? ?? ?? ?? ?? c7 4? ?? 00 00 00 00 4? c7 4? ???
5             c7 44 ?? ?? 00 00 00 00 c7 44 ?? ?? 80 00 00 00 c7 44 ?? ?? 02 00 00 00 45 33 c9
6             00 00 10 4? 8d 0d ?? ?? ?? ?? ff 15 ?? ?? ?? ?? 4? 89 4? ?? 33 d2 b9 02 00 00 00
7             4? 89 4? ?? 4? 8d ?? 90 00 00 00 4? 8b f8 33 c0 b9 38 02 00 00 f3 aa c7 8? ?? ???
8             00 4? 8d 05 ?? ?? ?? ?? 4? 89 ?? ?? ?? ?? ?? 4? 8d ?? 90 00 00 00 4? 8b 4? ?? e8
9             c0 74 ?? 4? 8d 15 ?? ?? ?? ?? 4? 8b ?? ?? ?? ?? ?? ?? ?? ff 15 ?? ?? ?? ?? ?? 85 c0 74 ???
0             00 00 4? 8b 4? ?? e8 ?? ?? ?? ?? 4? 8d ?? bc 00 00 00 4? 89 8? ?? ?? ?? ?? ?? 8b 8?
1
2     condition:
3         any of them
4 }
```



Dumplsass Yara Ruleset 2

```
711
712 rule INDICATOR_TOOL_PWS_LSASS_CreateMiniDump {
713     meta:
714         author = "ditekSHen"
715         description = "Detects CreateMiniDump tool"
716     strings:
717         $s1 = "lsass.dmp" fullword wide
718         $s2 = "lsass dumped successfully!" ascii
719         $s3 = "Got lsass.exe PID:" ascii
720         $s4 = "\\experiments\\CreateMiniDump\\CreateMiniDump\\" ascii
721         $s5 = "MiniDumpWriteDump" fullword ascii
722     condition:
723         uint16(0) == 0x5a4d and 2 of them
724 }
```

No Yara, No Microsoft

The screenshot shows a VirusTotal analysis page for the file `c19d901684871791873a5030cc42bc53686b34f9547c29443ac8348be234a945`. The file is identified as `dumpless-evade.exe`. The analysis summary indicates that 2 out of 75 security vendors flagged it as malicious. The file is a 64-bit executable (PEEXE) with an overlay. It was last analyzed a moment ago and has a size of 145.68 KB. The community score is 2/75. The interface includes tabs for DETECTION, DETAILS, BEHAVIOR, and COMMUNITY. Under the DETECTION tab, the 'Security vendors' analysis' section lists vendor results:

Vendor	Result	Action
DeepInstinct	Malicious	Automate
Acronis (Static ML)	Undetected	Automate
Alibaba	Undetected	Automate
SecureAge	Malicious	Automate
AhnLab-V3	Undetected	Automate
AliCloud	Undetected	Automate

At the bottom right, there is a link to 'Do you want to automate checks?'. The VirusTotal logo is visible in the top right corner.



“Malicious” API Calls

b696282412e510672d52ba7a4e9b92485c55e8ff6eae4f7d21506625891519f2 | Aaron

6 / 58
Community Score

6/58 security vendors flagged this file as malicious

b696282412e510672d52ba7a4e9b92485c55e8ff6eae4f7d21506625891519f2
api-calls.exe
peexe overlay 64bits

Size 261.54 KB | Last Analysis Date 2 minutes ago |

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY

Security vendors' analysis

			Do you want to automate checks?
Bkav Pro	W64.AIDetectMalware	CrowdStrike Falcon	Win/malicious_confidence_90% (D)
Elastic	Malicious (high Confidence)	McAfee Scanner	Ti!B696282412E5
SecureAge	Malicious	Trapmine	Suspicious.low.ml.score
Acronis (Static ML)	Undetected	AhnLab-V3	Undetected



Top WinAPIs Used in Malware

isDebuggerPresent

Others:

CreateThread

CreateRemoteThread

VirtualAlloc

ShellExecute

WinAPI Hashing

The screenshot shows a malware analysis interface with the following details:

- File Hash:** 9e4ce9fcdf4eb2a074b222da20f287caeb09f7ce4cc7946330c7996fd38c8446
- Community Score:** 2 / 75
- Vendor Analysis:** 2/75 security vendors flagged this file as malicious.
- File Details:** 9e4ce9fcdf4eb2a074b222da20f287caeb09f7ce4cc7946330c7996fd38c8446, all-api-hash-call.exe, peexe, overlay, 64bits.
- File Statistics:** Size 12.32 MB, Last Analysis Date a moment ago.
- File Type:** EXE
- Tab Navigation:** DETECTION (selected), DETAILS, BEHAVIOR, COMMUNITY.
- Security Vendors' Analysis:**

Vendor	Result	Notes
Bkav Pro	W64.AIDetectMalware	SecureAge Malicious
Acronis (Static ML)	Undetected	AhnLab-V3 Undetected
Alibaba	Undetected	AliCloud Undetected
- Automation:** Do you want to automate checks?



LAB TIME!



The 3 Malware Amigos





THANK YOU!