DISSECTING AND DEFEATING RANSOMWARE'S EVASION TACTICS

DefCon 32 Workshop

- **"The 3 Malware Amigos":**
- **Ryan Chapman [@rj_chap]**
- **Josh Stroschein [@jstrosch]**
- **Aaron Rosenmund [@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

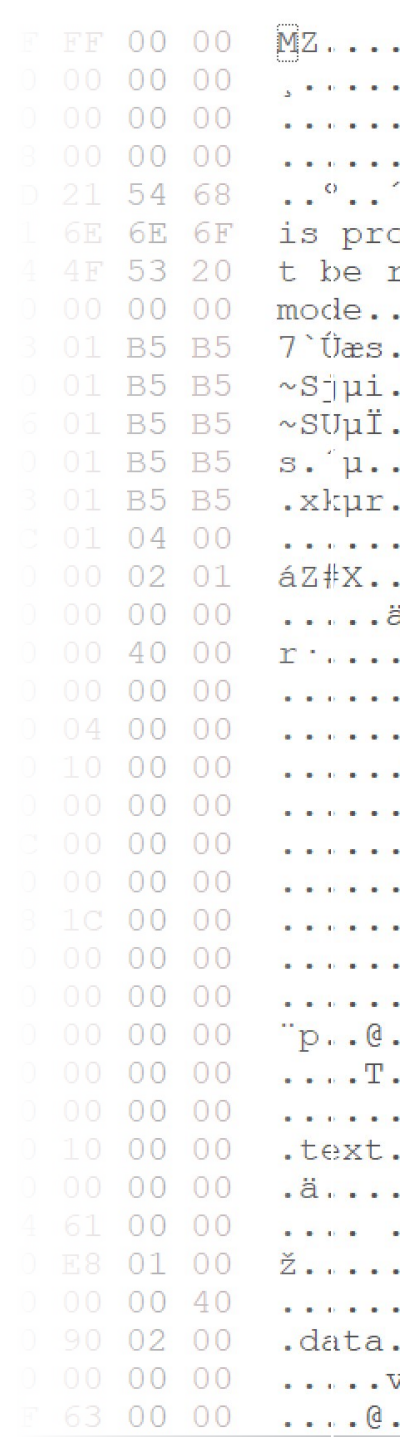
Sr. Dir. Content @ Pluralsight

- 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



Getting Started

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

github.com/arosenmund/defcon32_dissecting_defeating_ransomwares_evasion

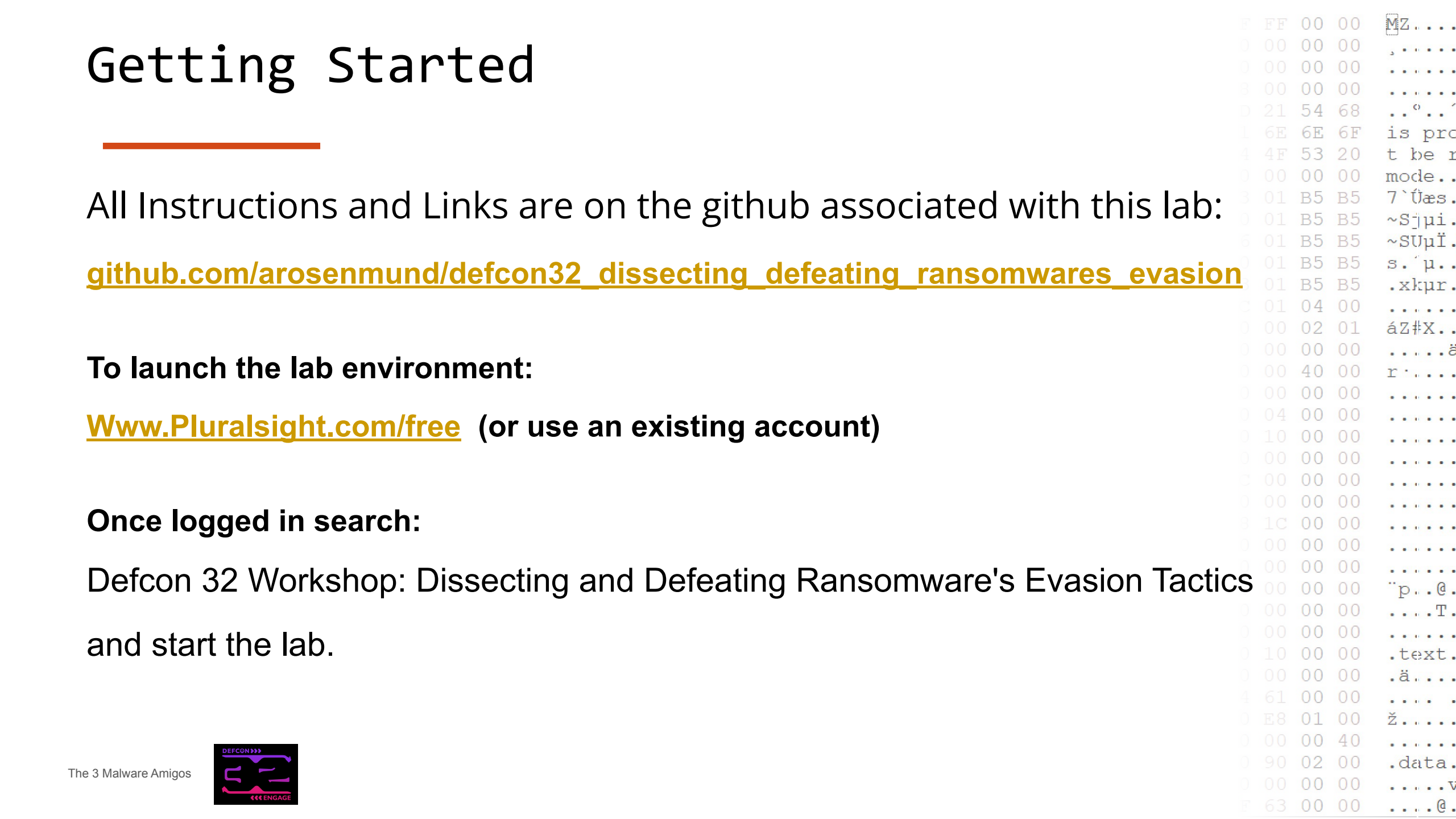
To launch the lab environment:

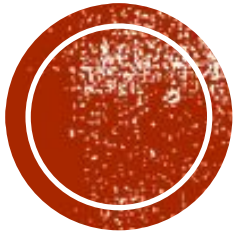
Www.Pluralsight.com/free **(or use an existing account)**

Once logged in search:

Defcon 32 Workshop: Dissecting and Defeating Ransomware's Evasion Tactics

and start the lab.



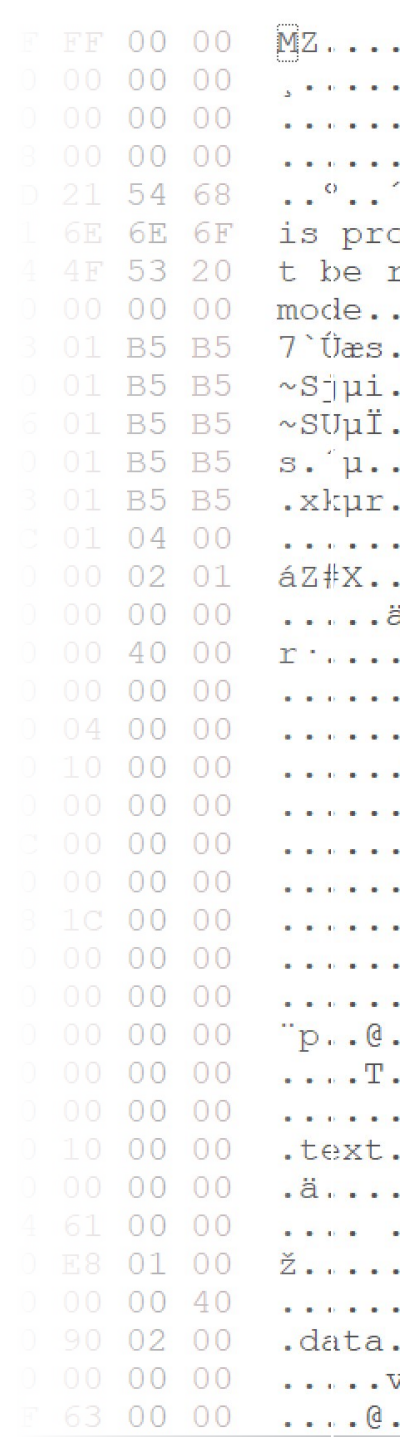


RANSOMWARE BUILDER ANALYSIS

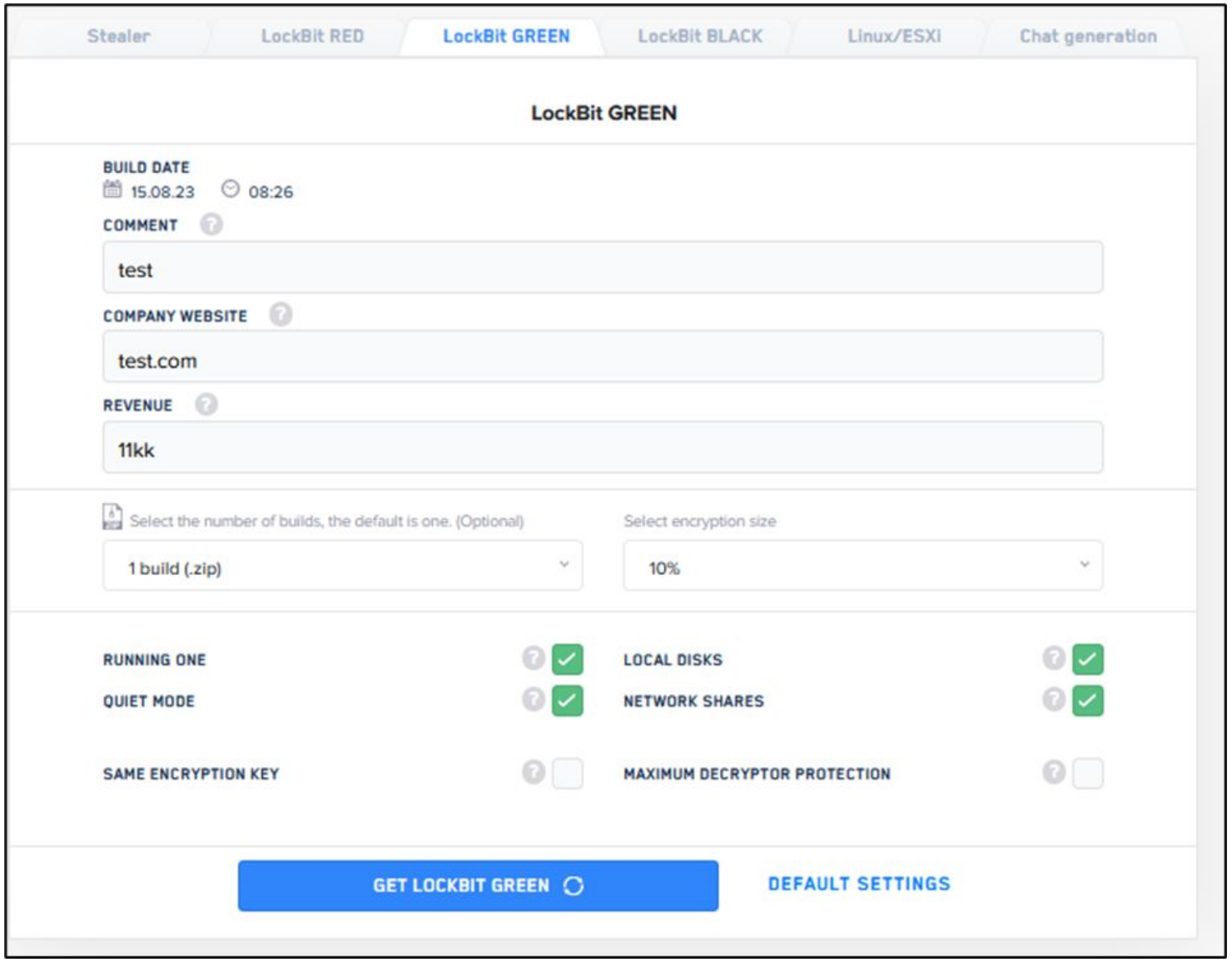
Ryan Chapman

Ransomware Builders

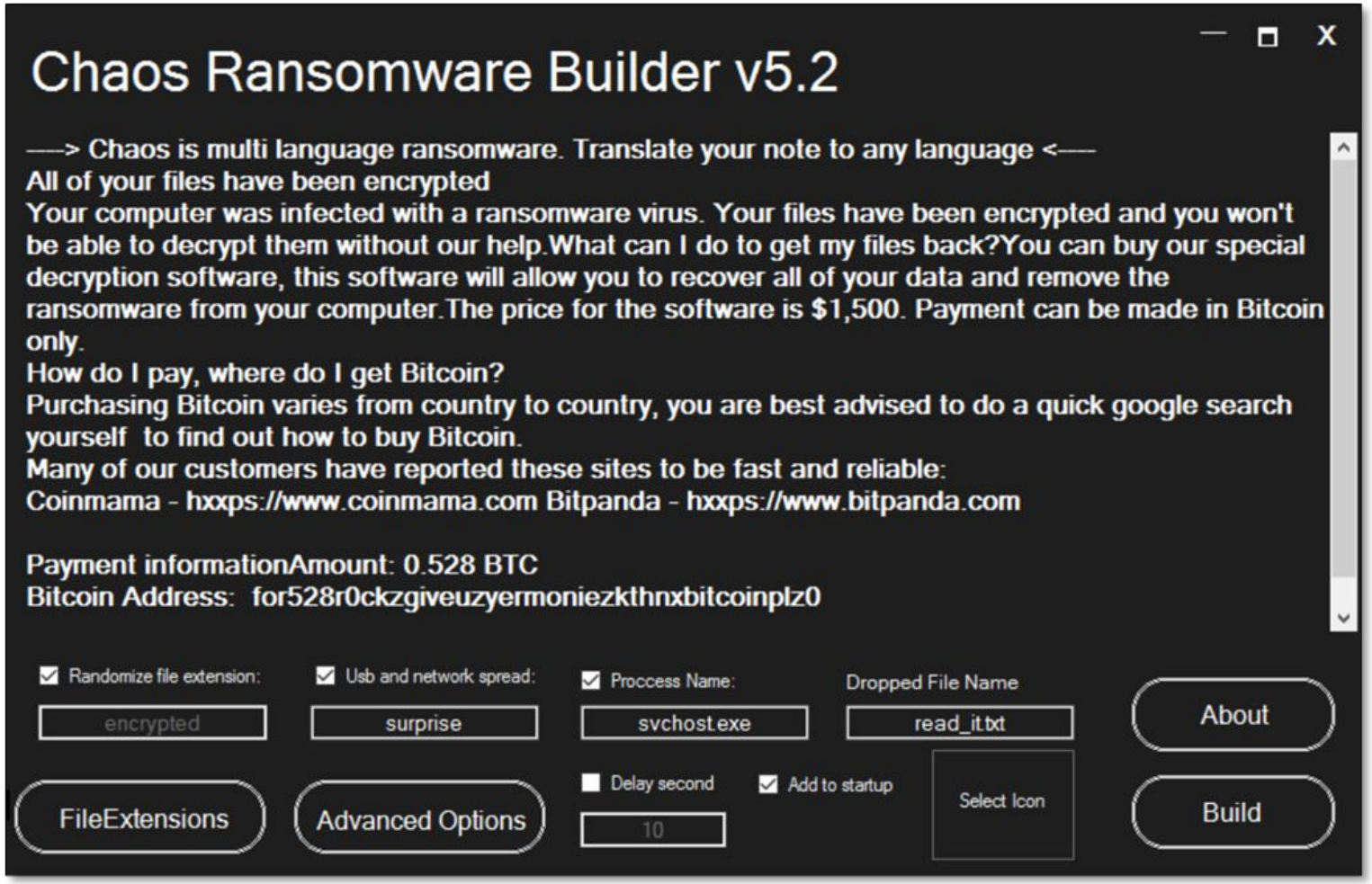
- Facilitate the turnkey nature of RaaS operations
- Developers create “builders” to generate samples for each affiliate/partner (i.e., they create customized payloads)
- The builder generates customized core components:
  - Payload/cryptor
  - Encryption keys (often public/private key pair)
  - Decryptor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

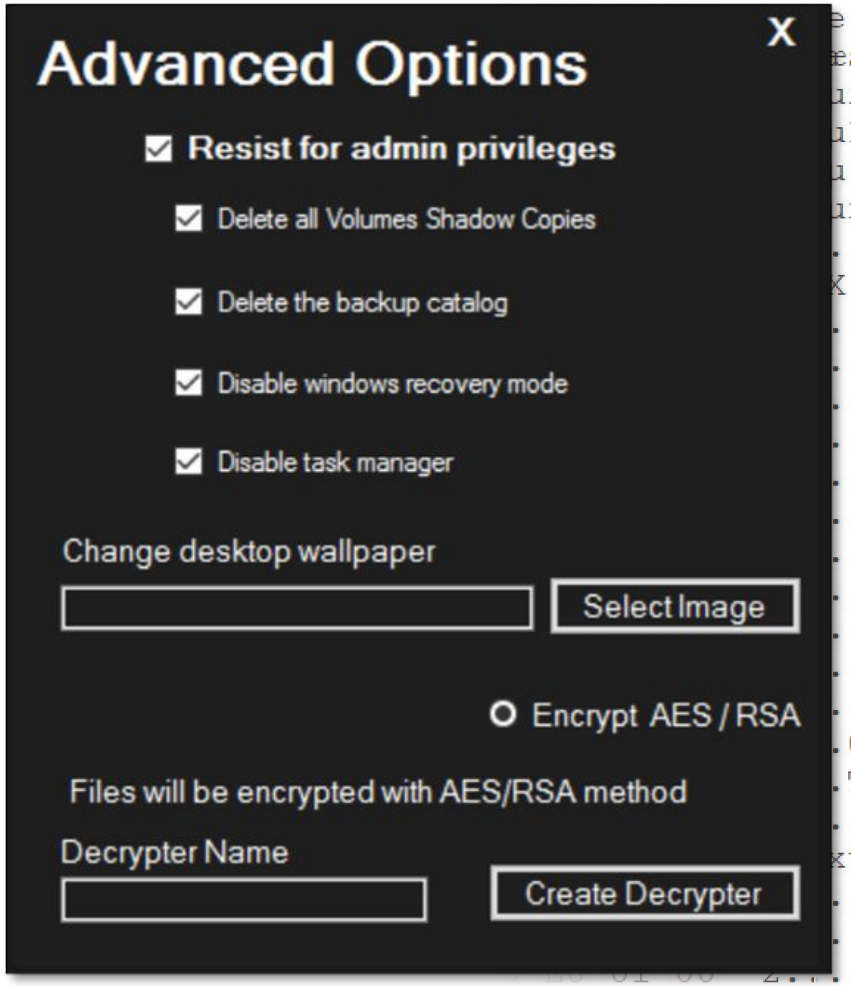


LockBit 3.0's Web Builder Interface

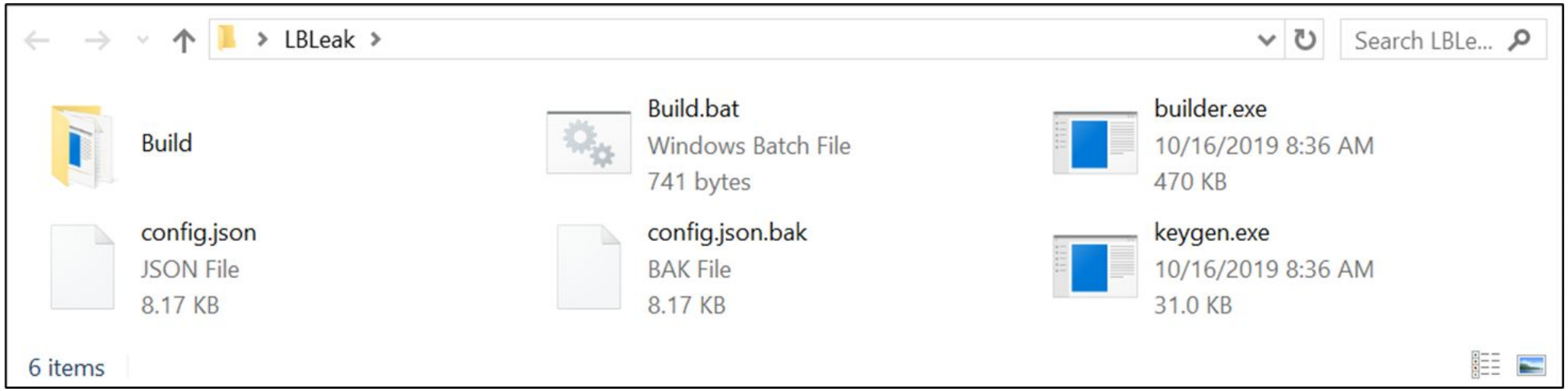


Options A-Plenty: Chaos Ransomware Builder



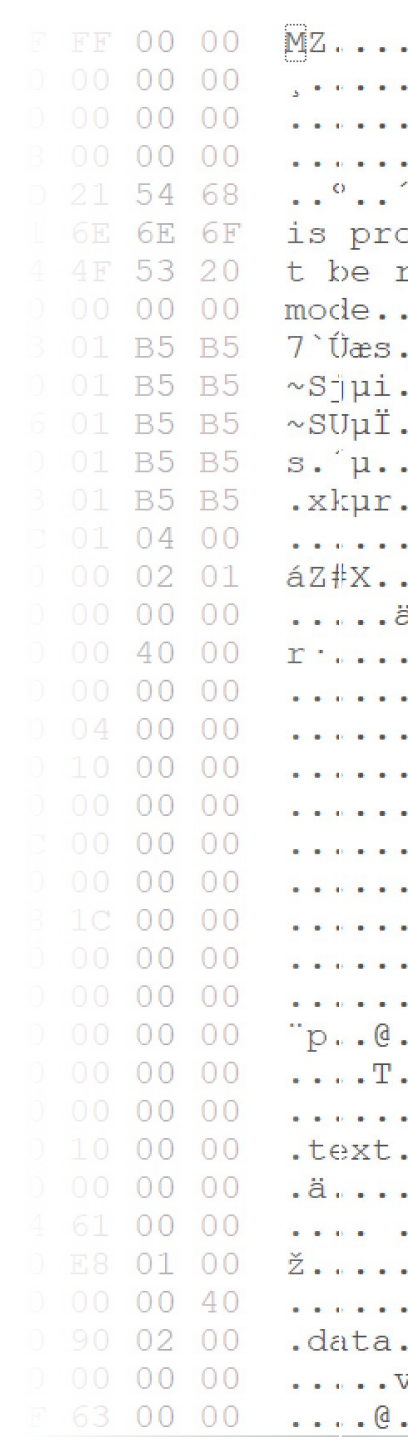


Our Focus: LockBit 3.0's Builder



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": "0000000000000000000000000000000000000000"
  },
"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,
```

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
```

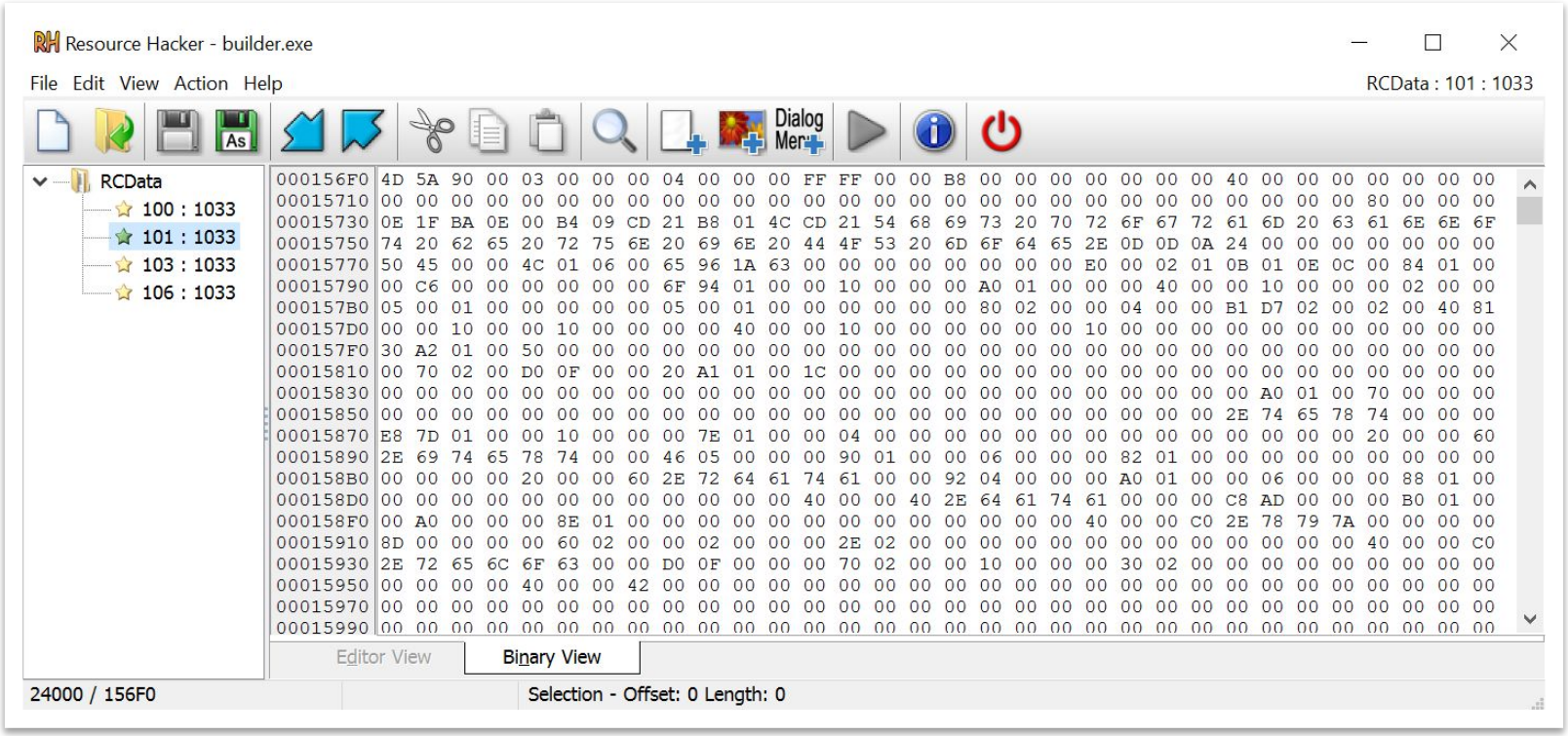
keygen.exe

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

DEMO

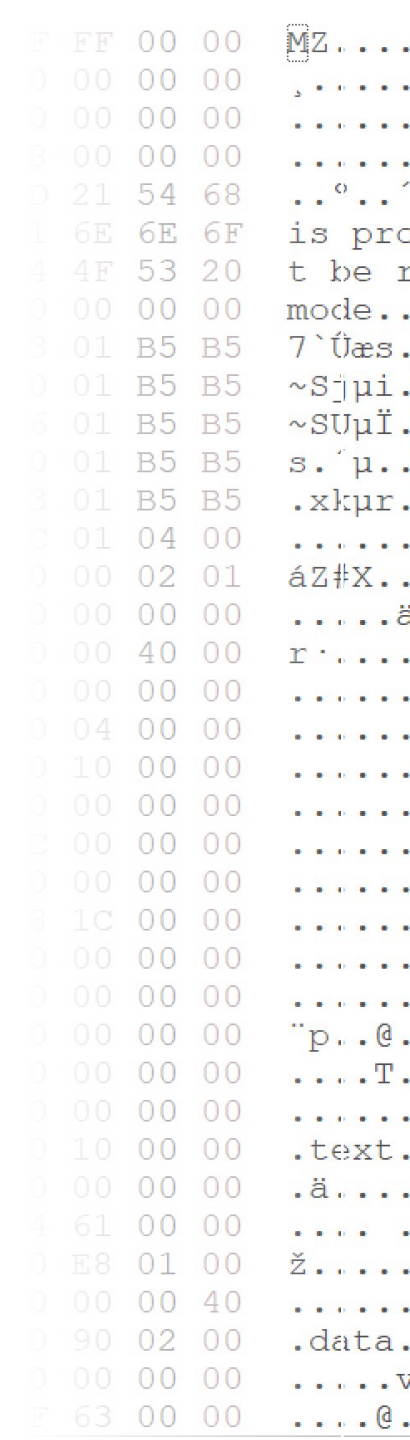
builder.exe Resources

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



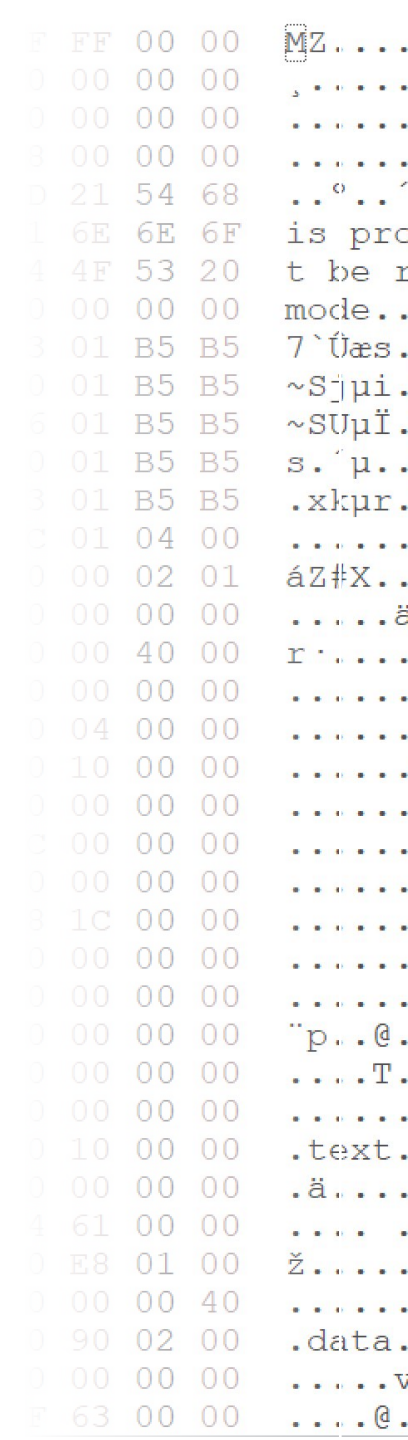
builder.exe - Password

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



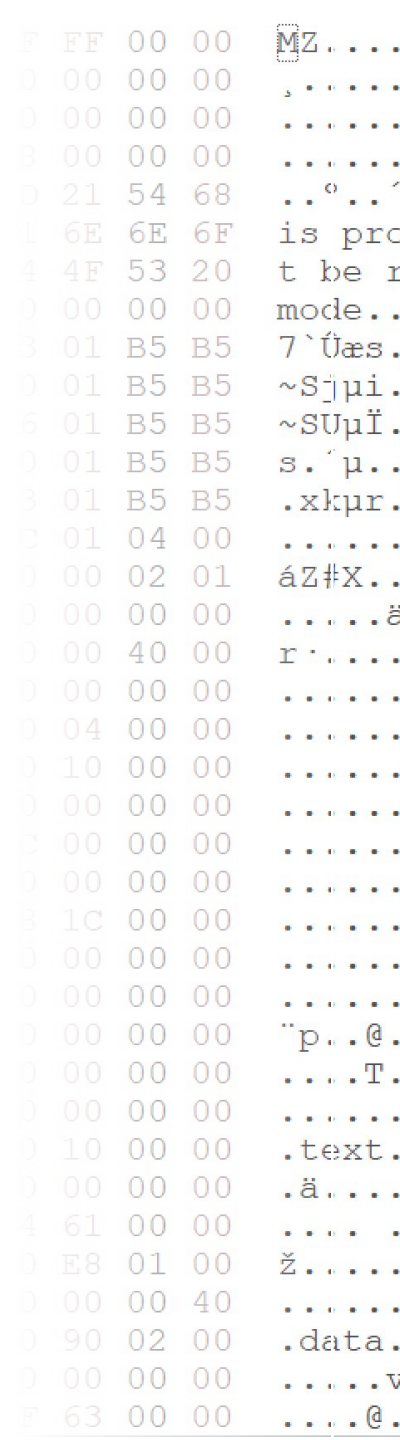
What We're About to Do

- 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

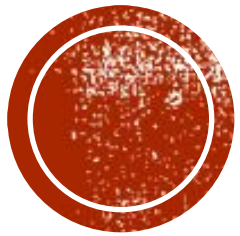


builder.exe Payload Encryption (-pass)

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



LAB TIME!



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
```

```
nop     dword ptr [eax+00h]
call    sub_40639C
-------------------------------------------------
db 66h, 90h
db 0E8h, 2, 5
dd 1F0FFFFFh, 0C1E80040h, 66FFFFDFh, 441F0Fh,
dd 4255C815h, 66666600h, 841F0Fh, 0
```

Packed

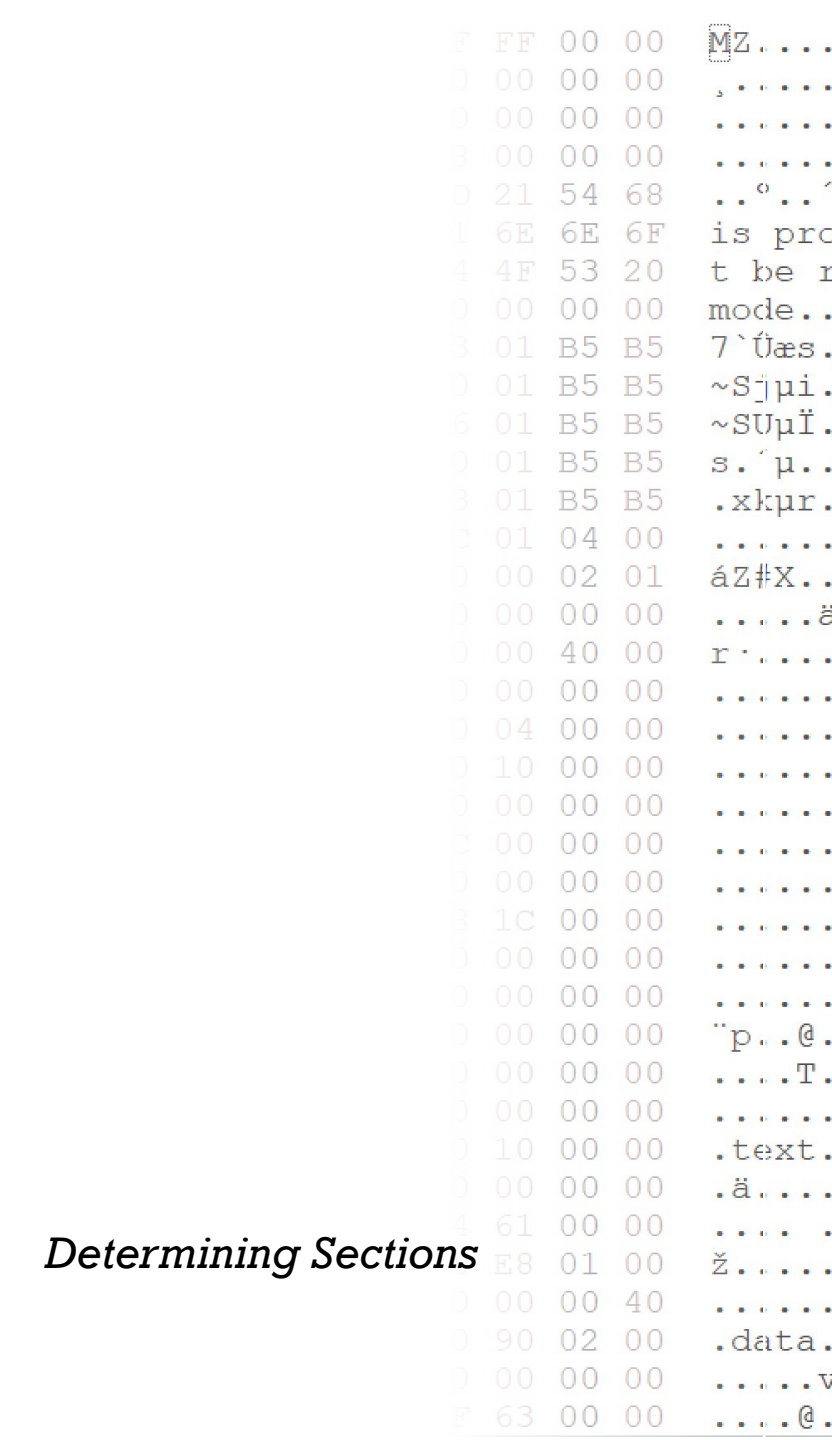
Mash That Easy Button!

- 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

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 *LdrGetProcessAddress*
- 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!

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

```
.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	B0 37 6A 02	F8 37 6A 02	A8 36 6A 02
00BD54A8	D0 3C 6A 02	C8 3E 6A 02	00 3D 6A 02	98 3E 6A 02
00BD54B8	A0 3F 6A 02	F8 3E 6A 02	B0 3E 6A 02	08 3E 6A 02
00BD54C8	E0 3E 6A 02	78 3D 6A 02	10 3F 6A 02	40 3F 6A 02
00BD54D8	28 3F 6A 02	D8 3D 6A 02	58 3F 6A 02	70 3F 6A 02
00BD54E8	88 3F 6A 02	B8 3C 6A 02	90 3D 6A 02	E8 3C 6A 02
00BD54F8	F0 3D 6A 02	00 00 00 00	A8 3D 6A 02	50 3E 6A 02
00BD5508	20 3E 6A 02	38 3E 6A 02	18 3D 6A 02	C0 3D 6A 02
00BD5518	30 3D 6A 02	48 3D 6A 02	60 3D 6A 02	68 3E 6A 02
00BD5528	80 3E 6A 02	30 40 6A 02	18 40 6A 02	78 40 6A 02
00BD5538	B8 3F 6A 02	D0 3F 6A 02	E8 3F 6A 02	00 40 6A 02
00BD5548	48 40 6A 02	60 40 6A 02	40 42 6A 02	58 42 6A 02
00BD5558	A0 42 6A 02	10 42 6A 02	90 43 6A 02	50 41 6A 02
00BD5568	A8 43 6A 02	30 43 6A 02	48 43 6A 02	70 42 6A 02
00BD5578	B0 41 6A 02	78 43 6A 02	C8 41 6A 02	F8 41 6A 02
00BD5588	00 43 6A 02	D0 42 6A 02	28 42 6A 02	C0 40 6A 02
00BD5598	E8 42 6A 02	88 42 6A 02	98 41 6A 02	E0 41 6A 02
00BD55A8	B8 42 6A 02	08 41 6A 02	18 43 6A 02	60 43 6A 02

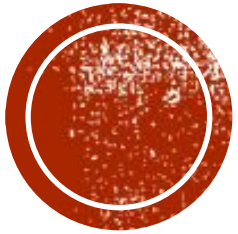
```
026A3798   B8 189E3B58    mov eax,583B9E18
026A379D   C1C0 09        rol eax,9
026A37A0   - FFE0         jmp eax
```

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

Default (stdcall) 5

```
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
```

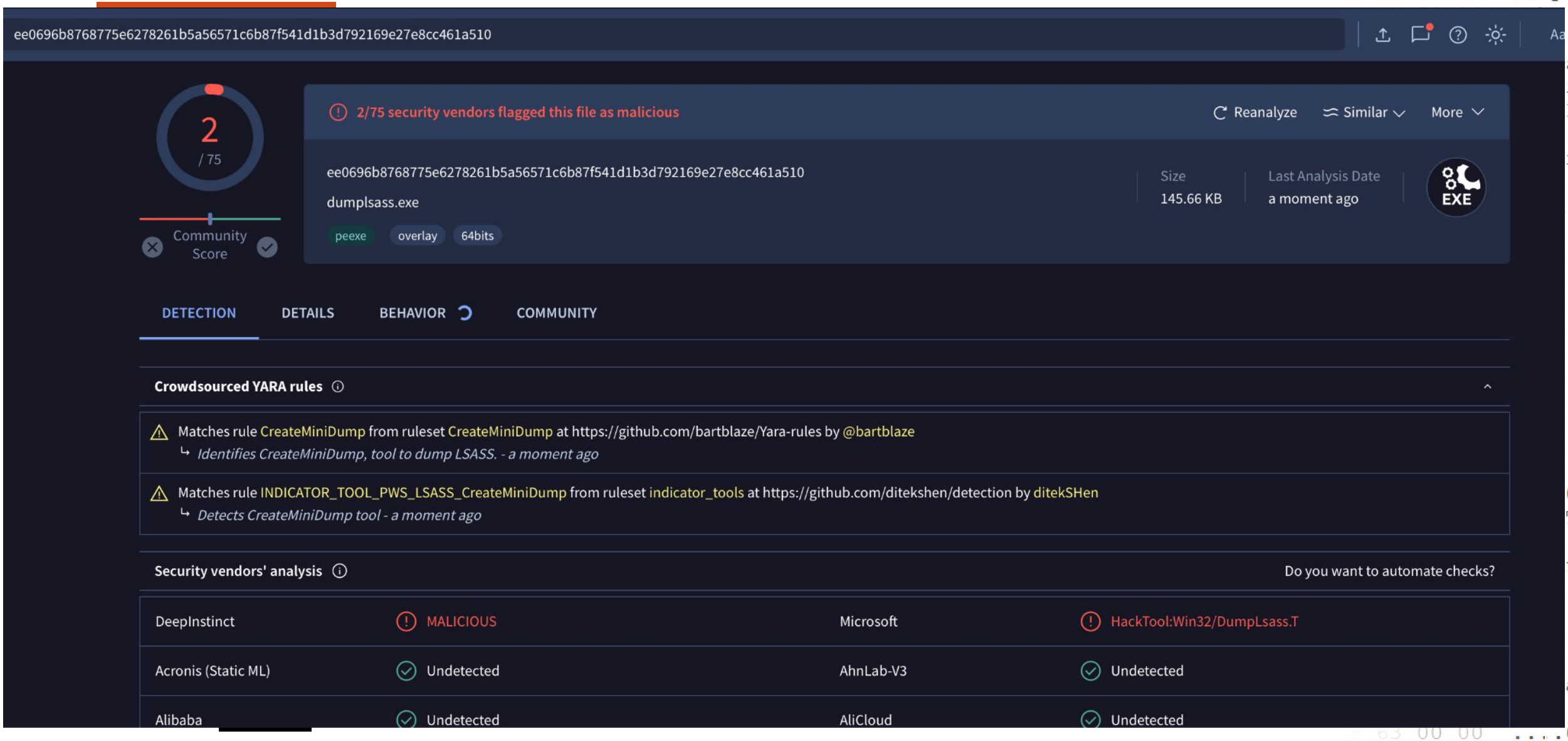
What Else Can We Find?!



WRITING CUSTOM EVASION PROCEDURES

Aaron Rosenmund

Dumplsass.exe Signatures



Dumplsass Yara Ruleset 1

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

Notice the condition.

Ruleset: CreateMiniDump
https://github.com/bartblaze/Yara-rules

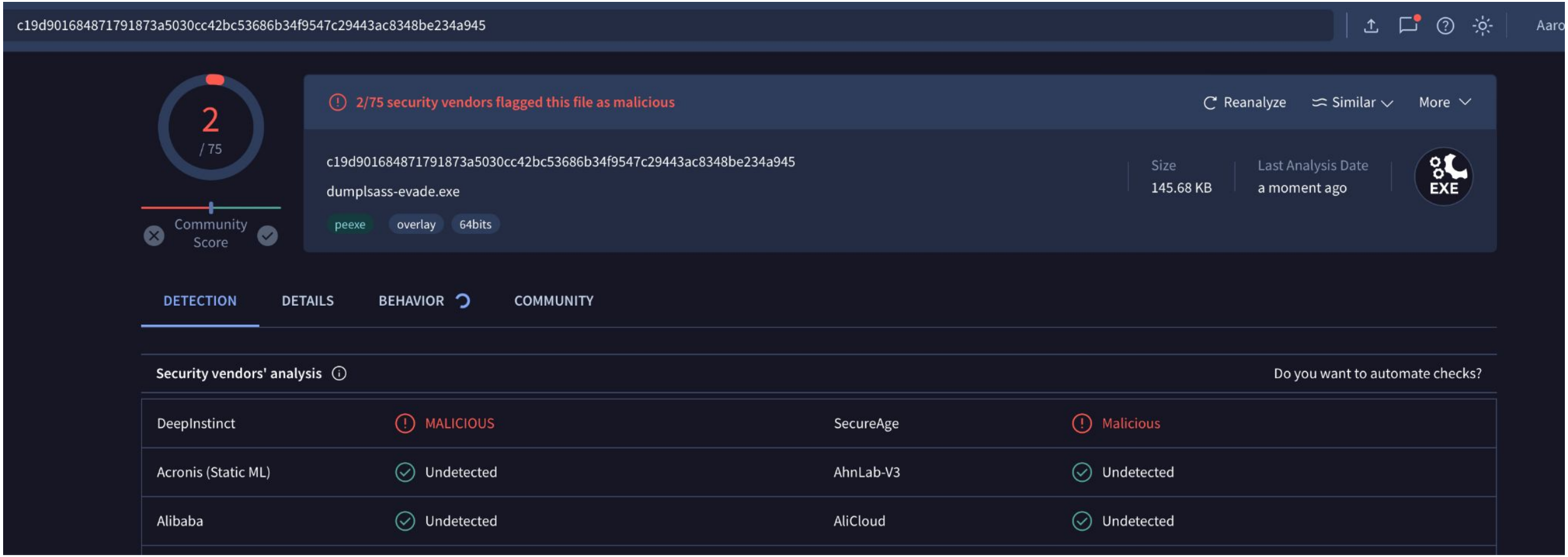
Go to matched rule | Copy ruleset | Download

```
rule CreateMiniDump
{
    meta:
        id = "kMNDXhwJQURe8ehDOueqk"
        fingerprint = "b391a564b4730559271e11de0b80dce1562a9038c230a2be729a896913c7
        version = "1.0"
        creation_date = "2021-03-01"
        first_imported = "2021-12-30"
        last_modified = "2021-12-30"
        status = "RELEASED"
        sharing = "TLP:WHITE"
        source = "BARTBLAZE"
        author = "@bartblaze"
        description = "Identifies CreateMiniDump, tool to dump LSASS."
        category = "HACKTOOL"
        tool = "CREATEMINIDUMP"
        reference = "https://www.ired.team/offensive-security/credential-access-and

    strings:
        $ = "[+] Got lsass.exe PID:" ascii wide
        $ = "[+] lsass dumped successfully!" ascii wide
        $ = { 40 55 57 4? 81 ec e8 04 00 00 4? 8d ?? ?4 40 4? 8b fc b9 3a 01 00 00
  8b 05 ?? ?? ?? ?? 4? 33 c5 4? 89 8? ?? ?? ?? ?? c7 4? ?? 00 00 00 00 4? c7 4? ??
  c7 44 ?? ?? 00 00 00 00 c7 44 ?? ?? 80 00 00 00 c7 44 ?? ?? 02 00 00 00 45 33 c9
  00 00 10 4? 8d 0d ?? ?? ?? ?? ff 15 ?? ?? ?? ?? 4? 89 4? ?? 33 d2 b9 02 00 00 00
  4? 89 4? ?? 4? 8d ?? 90 00 00 00 4? 8b f8 33 c0 b9 38 02 00 00 f3 aa c7 8? ?? ??
  00 4? 8d 05 ?? ?? ?? ?? 4? 89 ?? ?? ?? ?? ?? 4? 8d ?? 90 00 00 00 4? 8b 4? ?? e8
  c0 74 ?? 4? 8d 15 ?? ?? ?? ?? 4? 8b ?? ?? ?? ?? ?? ff 15 ?? ?? ?? ?? 85 c0 74 ??
  00 00 4? 8b 4? ?? e8 ?? ?? ?? ?? 4? 8d ?? bc 00 00 00 4? 89 8? ?? ?? ?? ?? 8b 8?

    condition:
        any of them
}
```

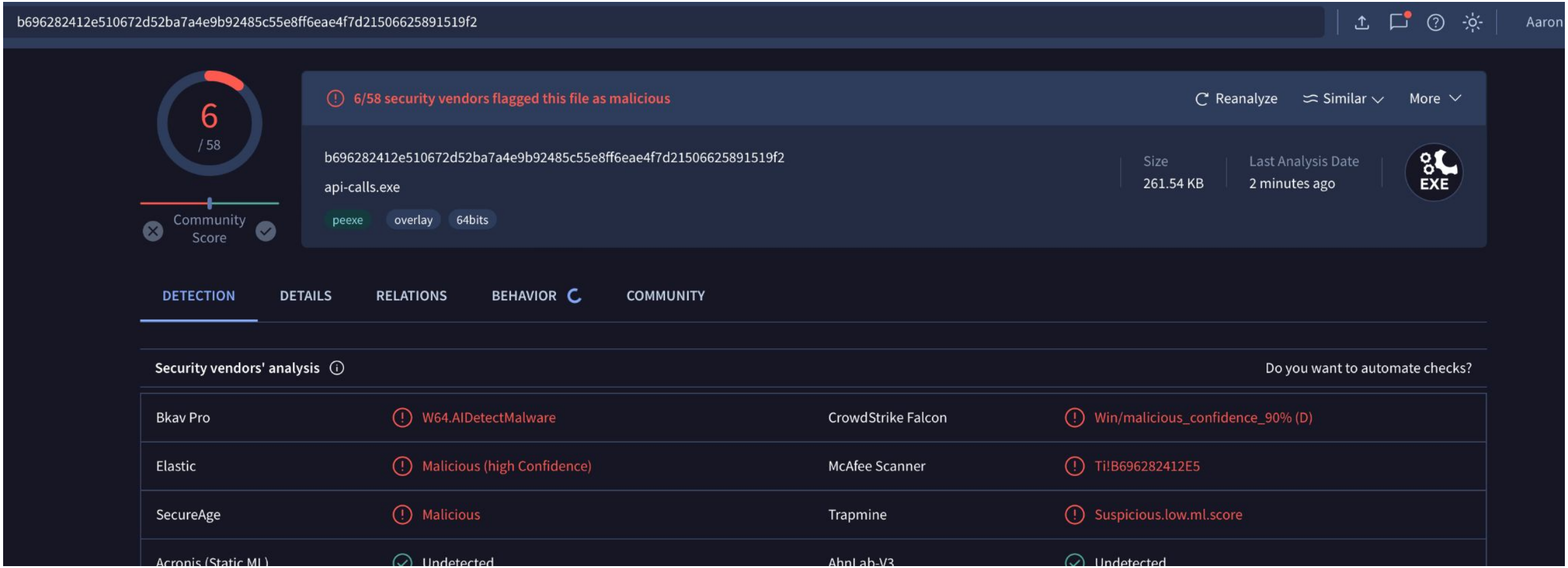
Dumplsass Yara Ruleset 2

```
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



“Malicious” API Calls



Top WinAPIs Used in Malware

isDebuggerPresent

CreateThread

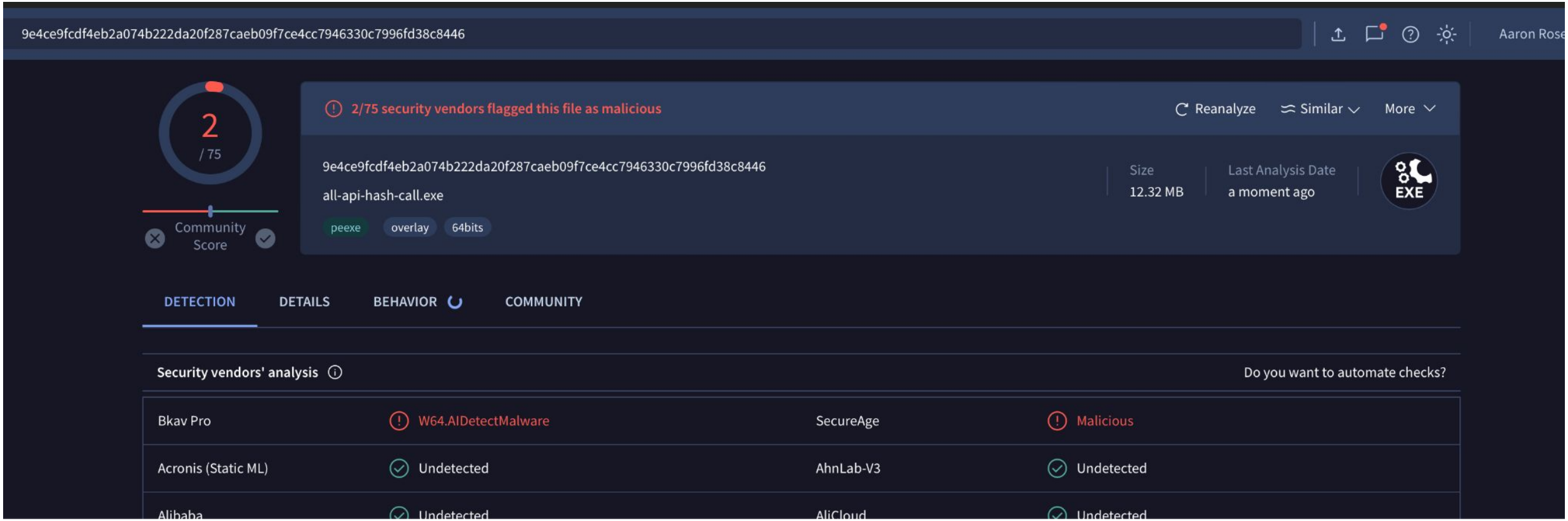
CreateRemoteThread

VirtualAlloc

ShellExecute

Others:

WinAPI Hashing



LAB TIME!

THANK YOU!