

# VOLEXITY

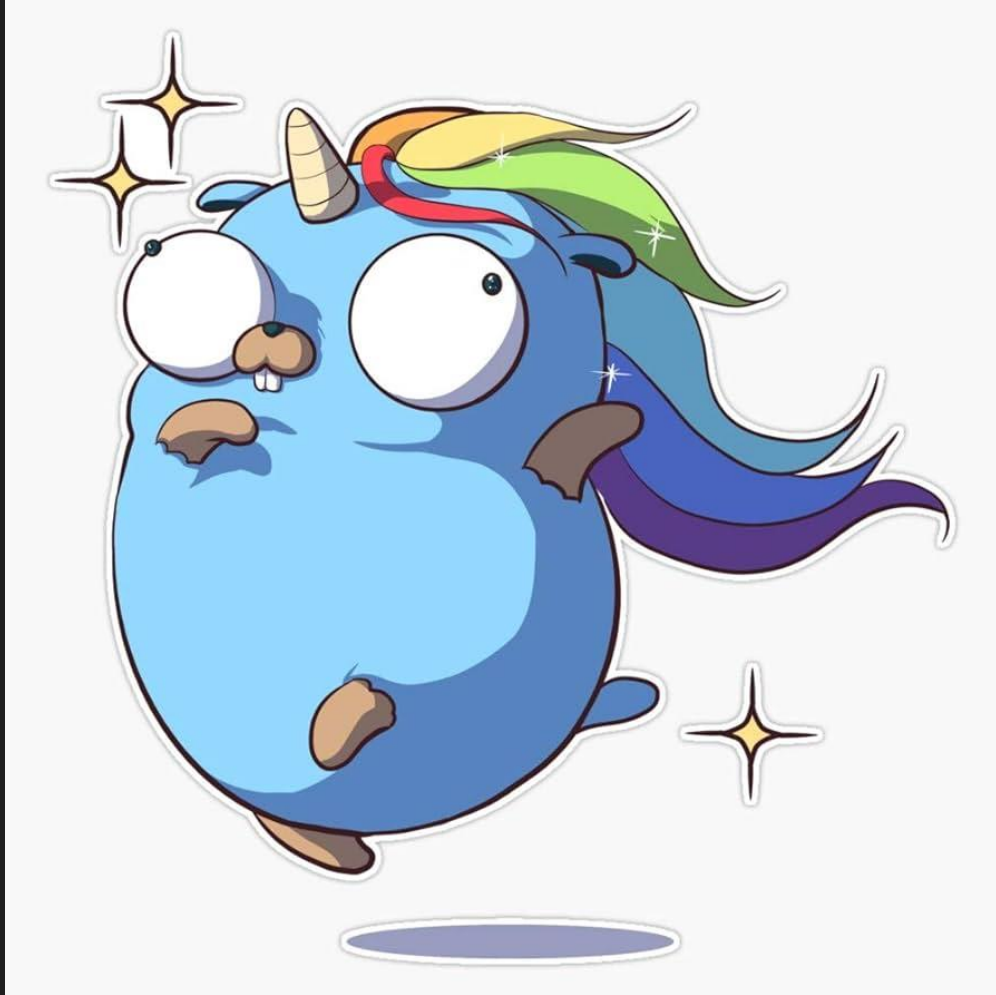
## GoResolver

Control-flow Graph Similarity  
Applied to Golang Binary  
Deobfuscation

Killian Raimbaud, Paul Rascagneres



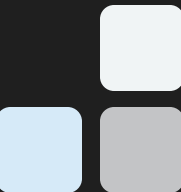
> cmd.exe /c whoami



- Paul Rascagneres  
Malware Analyst Lead @ Volexity
- Killian Raimbaud  
Malware Analyst @ Volexity
- Based in France 🇫🇷
- We hate Golang malware...

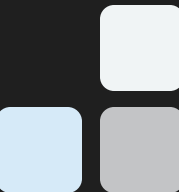
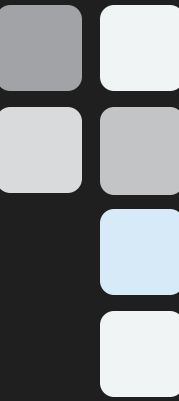
# Agenda

- Golang Malware
- Garble Obfuscation
- Control-flow Graphs Similarity
- GoResolver
- Conclusion



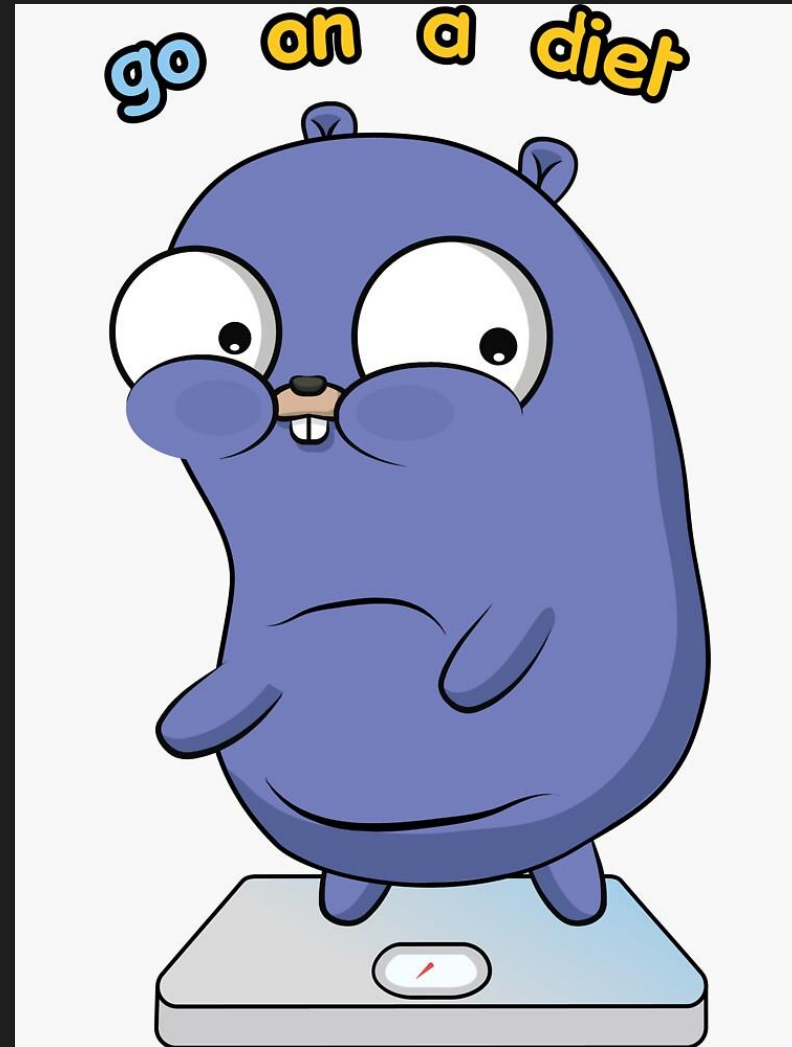
# Golang Malware

- More and more Golang samples encountered during incident response
- Cross-platform
- Used by various threat actors
- Examples of public tools
  - ❑ Sliver: <https://github.com/BishopFox/sliver>
  - ❑ iox: <https://github.com/eddieivan01/iox>
  - ❑ Rsockstun: <https://github.com/llkat/rsockstu>
- Examples of Ransomware:
  - ❑ RansomHub
  - ❑ Hellcat



# Native Golang Issues for Reversers

- Binary size
- Embedded runtimes
- Embedded dependencies
- Strings separator
- Objects



# New Layer of Issues: Garble

- <https://github.com/burrowers/garble>

## Mechanism

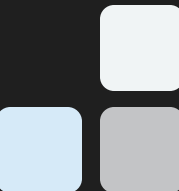
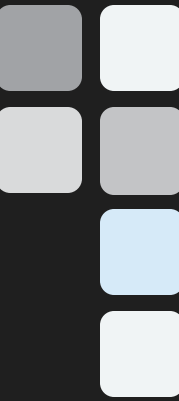
The tool wraps calls to the Go compiler and linker to transform the Go build, in order to:

- Replace as many useful identifiers as possible with short base64 hashes
- Replace package paths with short base64 hashes
- Replace filenames and position information with short base64 hashes
- Remove all [build](#) and [module](#) information
- Strip debugging information and symbol tables via `-ldflags="-w -s"`
- [Obfuscate literals](#), if the `-literals` flag is given
- Remove [extra information](#), if the `-tiny` flag is given

By default, the tool obfuscates all the packages being built. You can manually specify which packages to obfuscate via `GOGARBLE`, a comma-separated list of glob patterns matching package path prefixes. This format is borrowed from `GOPRIVATE`; see `go help private`.

Note that commands like `garble build` will use the `go` version found in your `$PATH`. To use different versions of Go, you can [install them](#) and set up `$PATH` with them. For example, for Go 1.17.1:

```
$ go install golang.org/dl/go1.17.1@latest
$ go1.17.1 download
$ PATH=$(go1.17.1 env GOROOT)/bin:${PATH} garble build
```



# Before – After Garble

Function name	Segment
os_runtime_beforeExit	.text
os_init	.text
os_init_func1	.text
os_Getenv	.text
os_ptr_SyscallError_Error	.text
os_ptr_SyscallError_Unwrap	.text
os_ptr_SyscallError_Timeout	.text
os_newHandleProcess	.text
os_ptr_Process_handleTransientRelease	.text
os_ptr_Process_handlePersistentRelease	.text
os_StartProcess	.text
os_ptr_Process_Release	.text
os_startProcess	.text
os_ptr_ProcessState_String	.text
os_ptr_Process_wait	.text
os_ptr_Process_wait_deferwrap2	.text
os_ptr_Process_wait_deferwrap1	.text
os_ptr_Process_signal	.text
os_ptr_Process_signal_deferwrap2	.text
os_ptr_Process_signal_deferwrap1	.text
os_ptr_Process_release	.text
os_init_0	.text
os_readNextArg	.text
os_commandLineToArgv	.text
os_Executable	.text
os_getModuleFileName	.text
os_ptr_File_Name	.text
os_ptr_File_Read	.text
os_ptr_File_ReadFrom	.text
os_noReadFrom_ReadFrom	.text
os_genericReadFrom	.text
os_ptr_File_Write	.text
os_ptr_File_WriteTo	.text
os_noWriteTo_WriteTo	.text
os_genericWriteTo	.text
os_ptr_File_WriteString	.text
os_OpenFile	.text
os_ptr_File_wrapErr	.text
os_ptr_File_SetDeadline	.text
os_ptr_File_SetReadDeadline	.text
os_ptr_File_SetWriteDeadline	.text
os_ptr_File_Close	.text
os_ptr_File_setDeadline	.text
os_ptr_File_setReadDeadline	.text
os_ptr_File_setWriteDeadline	.text
os_newFile	.text

Line 90 of 550, /os/os.init.func1

```
os_init_func1 proc near
var_8= qword ptr -8
cmp     rsp, [r14+10h]
jbe    short loc_4C4910

push   rbp
mov    rbp, rsp
sub    rsp, 20h
lea   rax, RTYPE_slice_uint8
call  runtime_newobject
mov   [rsp+20h+var_8], rax
mov   ebx, 10000h
mov   rcx, rbx
mov   rax, RTYPE_uint8
lea   rax, runtime_makeslice
call  runtime_makeslice
mov   rbx, [rsp+20h+var_8]
mov   qword ptr [rbx+8], 10000h
mov   qword ptr [rbx+10h], 10000h
cmp   cs:dword_9ABF00, 0
jz    short loc_4C4900

call  runtime_gcWriteBarrier2
mov   [r11], rax
mov   rcx, [rbx]
mov   [r11+8], rcx

loc_4C4900:
mov   [rbx], rax
lea   rax, RTYPE_ptr_slice_uint8
add   rsp, 20h
pop   rbp
retn
```

Function name	Segment
os_init	.text
unknown_libname_120	.text
sub_4C4805	.text
unknown_libname_122	.text
sub_4C487C	.text
unknown_libname_123	.text
os_ptr_SyscallError_Timeout	.text
sub_4C4953	.text
sub_4C4960	.text
sub_4C49E5	.text
sub_4C4A00	.text
sub_4C4B00	.text
sub_4C4BB7	.text
os_StartProcess	.text
sub_4C4CA0	.text
sub_4C4CC5	.text
sub_4C4CE0	.text
sub_4C508D	.text
sub_4C50C0	.text
sub_4C521E	.text
sub_4C5240	.text
sub_4C571E	.text
sub_4C5740	.text
sub_4C57A0	.text
sub_4C5800	.text
sub_4C5B79	.text
sub_4C5BA0	.text
sub_4C5C00	.text
sub_4C5C60	.text
sub_4C5CD1	.text
os_init_0	.text
sub_4C5DE0	.text
sub_4C6178	.text
os_commandLineToArgv	.text
sub_4C62E5	.text
sub_4C6300	.text
sub_4C6325	.text
os_getModuleFileName	.text
sub_4C6400	.text
os_ptr_File_Read	.text
sub_4C648D	.text
sub_4C64E0	.text
sub_4C6545	.text
unknown_libname_124	.text

Line 3424 of 8658, /unknown\_libname\_120

```
; go standard library (ABIInternal)
; Attributes: Library function bp-based frame
unknown_libname_120 proc near
var_8= qword ptr -8
cmp     rsp, [r14+10h]
jbe    short loc_4C4780

push   rbp
mov    rbp, rsp
sub    rsp, 20h
lea   rax, unk_6B48C0
call  runtime_newobject
mov   [rsp+20h+var_8], rax
mov   ebx, 10000h
mov   rcx, rbx
lea   rax, unk_6B8420
call  runtime_makeslice
mov   rbx, [rsp+20h+var_8]
mov   qword ptr [rbx+8], 10000h
mov   qword ptr [rbx+10h], 10000h
cmp   cs:dword_A52AC0, 0
jz    short loc_4C47A0

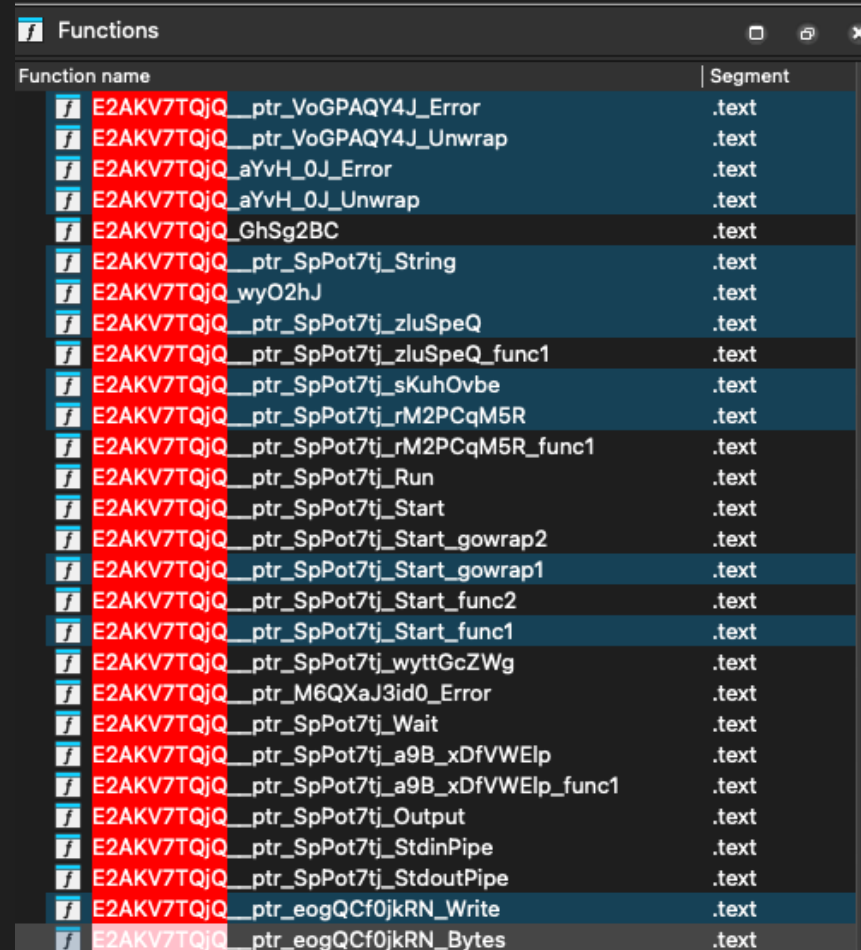
call  sub_472FC0
mov   [r11], rax
mov   rcx, [rbx]
mov   [r11+8], rcx

loc_4C4780:
call  sub_471080
unknown_libname_120 endp

loc_4C47A0:
mov   [rbx], rax
lea   rax, asc_6B43C0 ; "\b"
add   rsp, 20h
pop   rbp
retn
```

# Garble Limitations

- Randomized names stay consistent across all functions



The screenshot shows a debugger window titled 'Functions' with a list of function names and their corresponding segments. The function names are randomized but follow a consistent pattern, such as 'E2AKV7TQjQ\_ptr\_VoGPAQY4J\_Error'. The segments are all '.text'.

Function name	Segment
E2AKV7TQjQ_ptr_VoGPAQY4J_Error	.text
E2AKV7TQjQ_ptr_VoGPAQY4J_Unwrap	.text
E2AKV7TQjQ_aYvH_0J_Error	.text
E2AKV7TQjQ_aYvH_0J_Unwrap	.text
E2AKV7TQjQ_GhSg2BC	.text
E2AKV7TQjQ_ptr_SpPot7tj_String	.text
E2AKV7TQjQ_wyO2hJ	.text
E2AKV7TQjQ_ptr_SpPot7tj_zluSpeQ	.text
E2AKV7TQjQ_ptr_SpPot7tj_zluSpeQ_func1	.text
E2AKV7TQjQ_ptr_SpPot7tj_sKuhOvbe	.text
E2AKV7TQjQ_ptr_SpPot7tj_rM2PCqM5R	.text
E2AKV7TQjQ_ptr_SpPot7tj_rM2PCqM5R_func1	.text
E2AKV7TQjQ_ptr_SpPot7tj_Run	.text
E2AKV7TQjQ_ptr_SpPot7tj_Start	.text
E2AKV7TQjQ_ptr_SpPot7tj_Start_gowrap2	.text
E2AKV7TQjQ_ptr_SpPot7tj_Start_gowrap1	.text
E2AKV7TQjQ_ptr_SpPot7tj_Start_func2	.text
E2AKV7TQjQ_ptr_SpPot7tj_Start_func1	.text
E2AKV7TQjQ_ptr_SpPot7tj_wyttGcZWg	.text
E2AKV7TQjQ_ptr_M6QXaJ3id0_Error	.text
E2AKV7TQjQ_ptr_SpPot7tj_Wait	.text
E2AKV7TQjQ_ptr_SpPot7tj_a9B_xDfVWElp	.text
E2AKV7TQjQ_ptr_SpPot7tj_a9B_xDfVWElp_func1	.text
E2AKV7TQjQ_ptr_SpPot7tj_Output	.text
E2AKV7TQjQ_ptr_SpPot7tj_StdinPipe	.text
E2AKV7TQjQ_ptr_SpPot7tj_StdoutPipe	.text
E2AKV7TQjQ_ptr_eogQCf0jkRN_Write	.text
E2AKV7TQjQ_ptr_eogQCf0jkRN_Bytes	.text



# Existing Tools

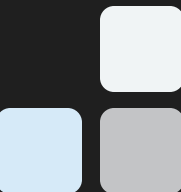
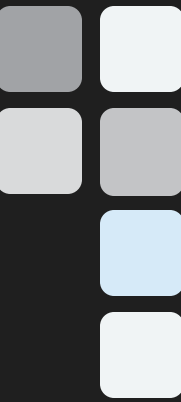
- GoReSym from Mandiant:  
<https://github.com/mandiant/GoReSym>

## GoReSym

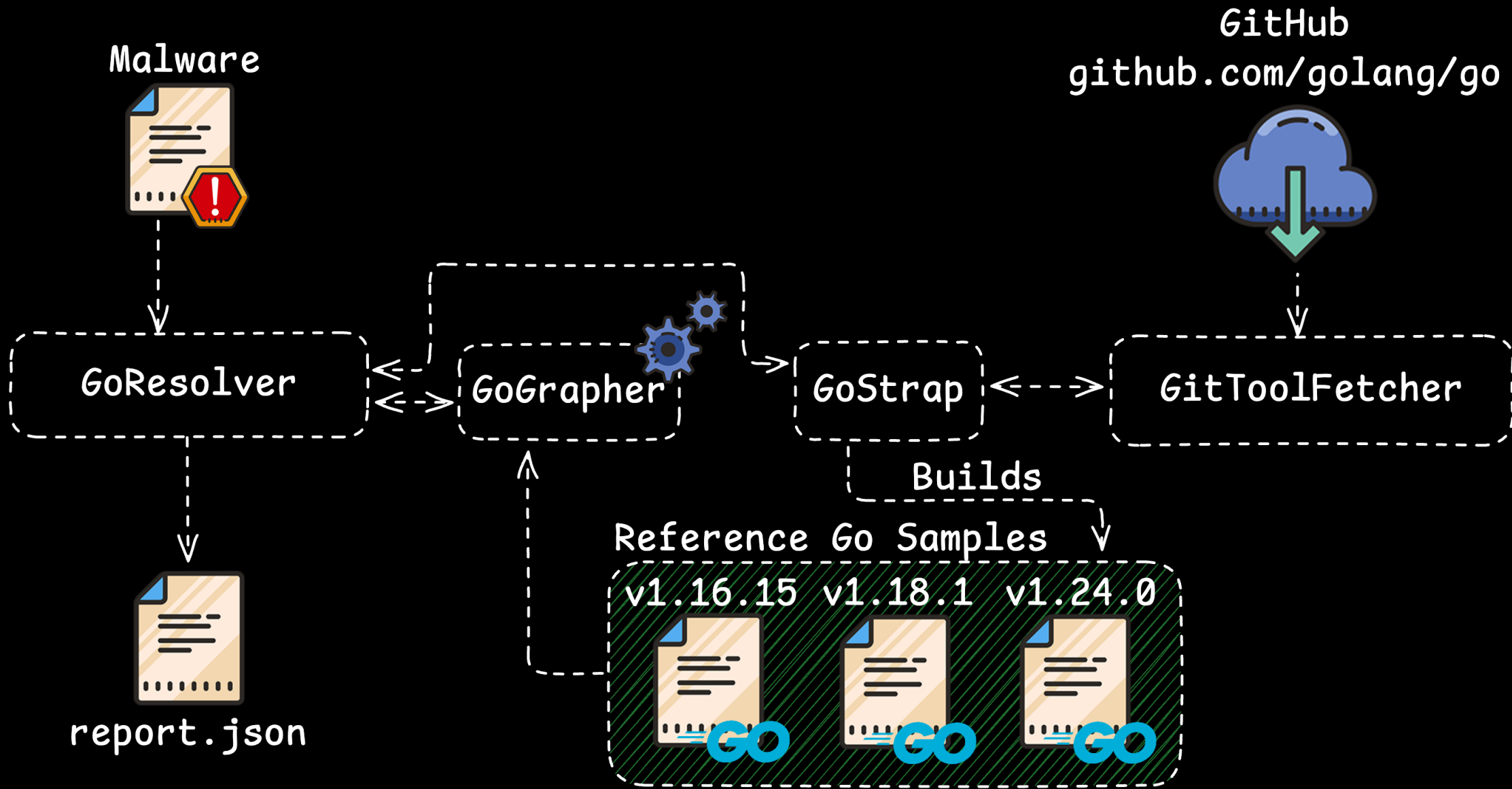
GoReSym is a Go symbol parser that extracts program metadata (such as CPU architecture, OS, endianness, compiler version, etc), function metadata (start & end addresses, names, sources), filename and line number metadata, and embedded structures and types. This cross platform program is based directly on the [open source Go compiler](#) and runtime code.

The upstream Go runtime code is extended to handle:

- stripped binaries
- malformed unpacked binaries, such as from UPX
- binaries that split single data ranges across multiple sections
- the location of the `moduledata` structure

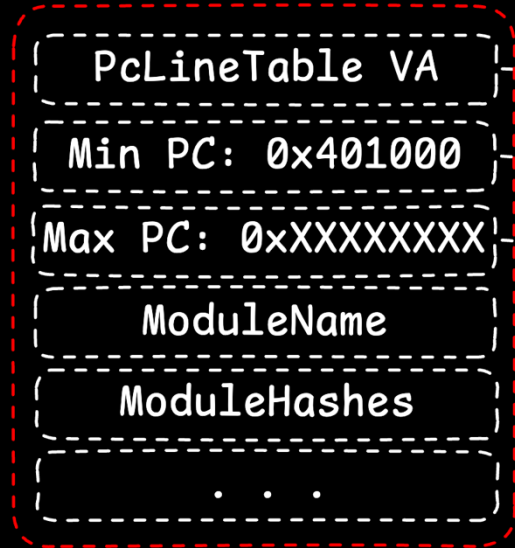


# GoResolver: Toolchain Overview

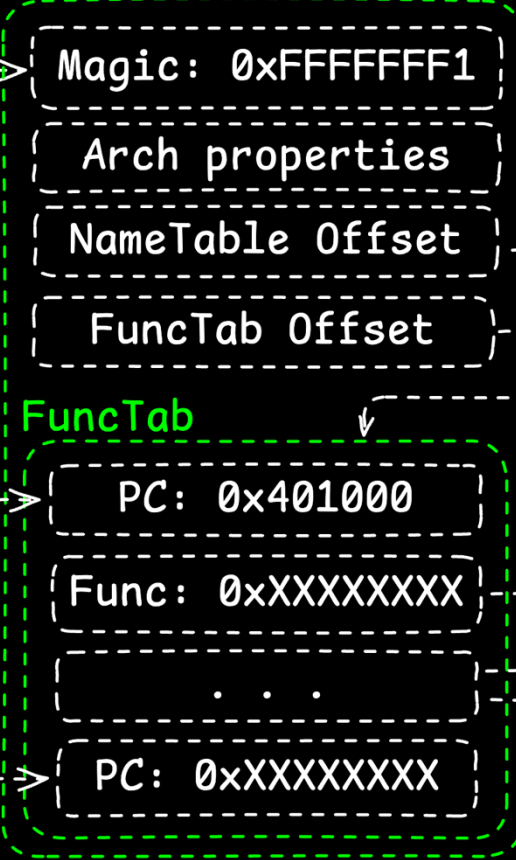


# GoResolver: Golang Symbol Extraction

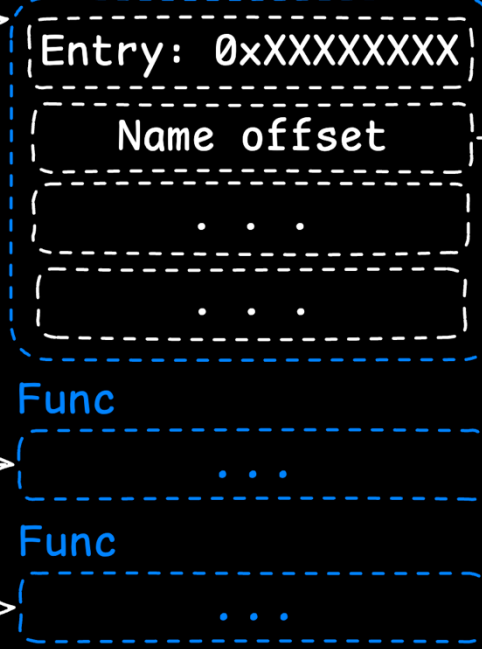
## ModuleData



## PcLineTable



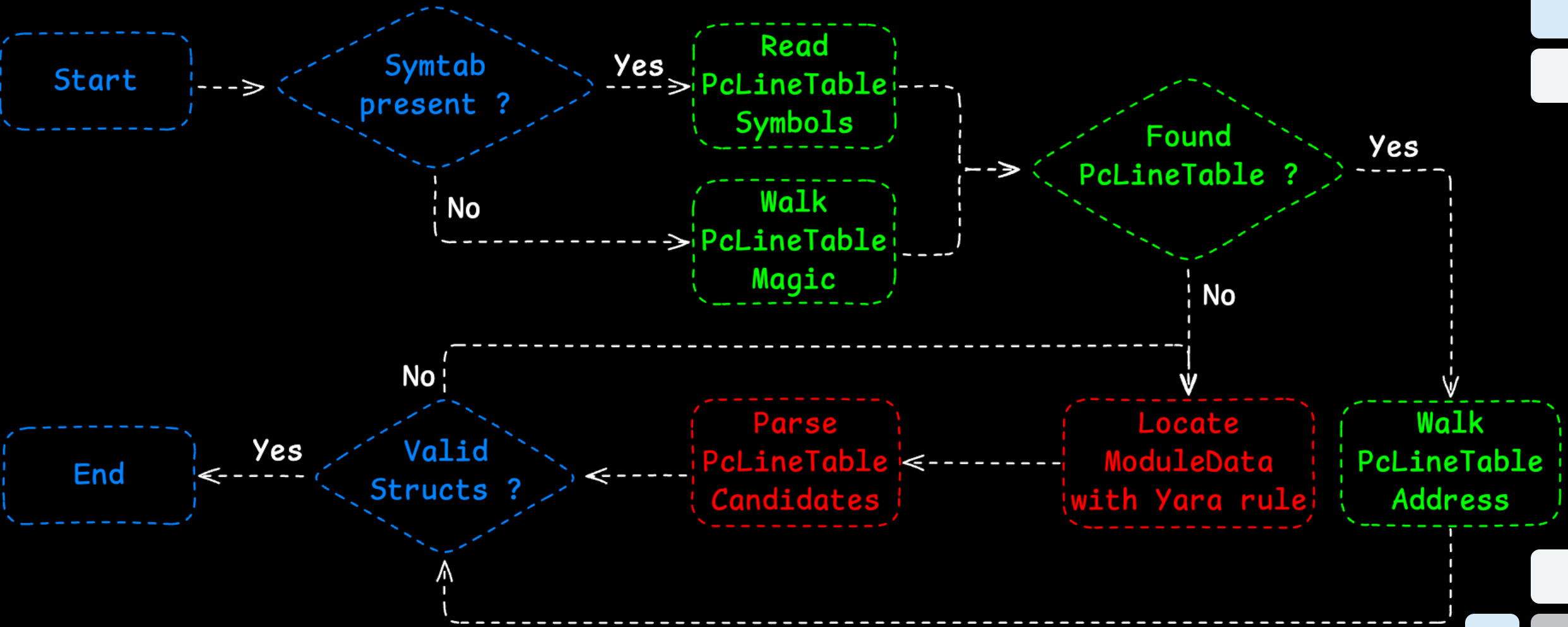
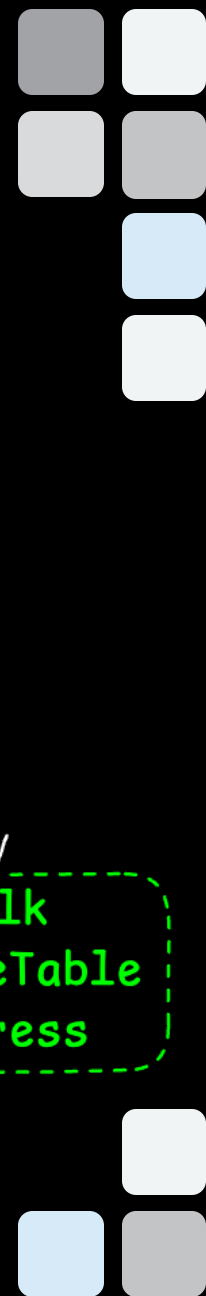
## Func



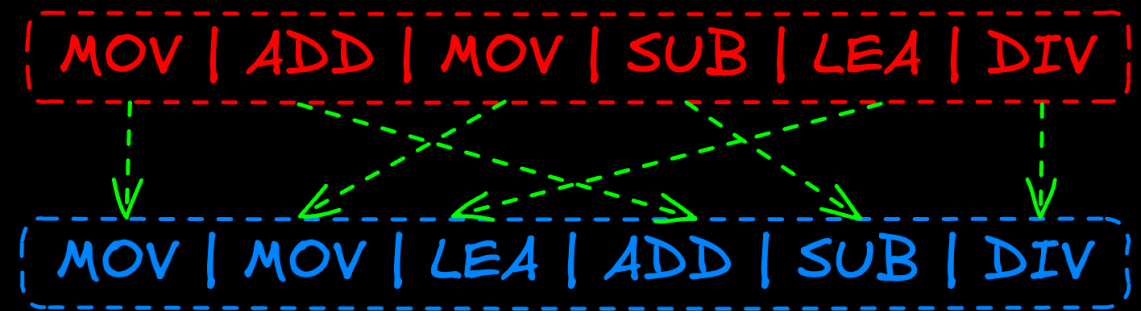
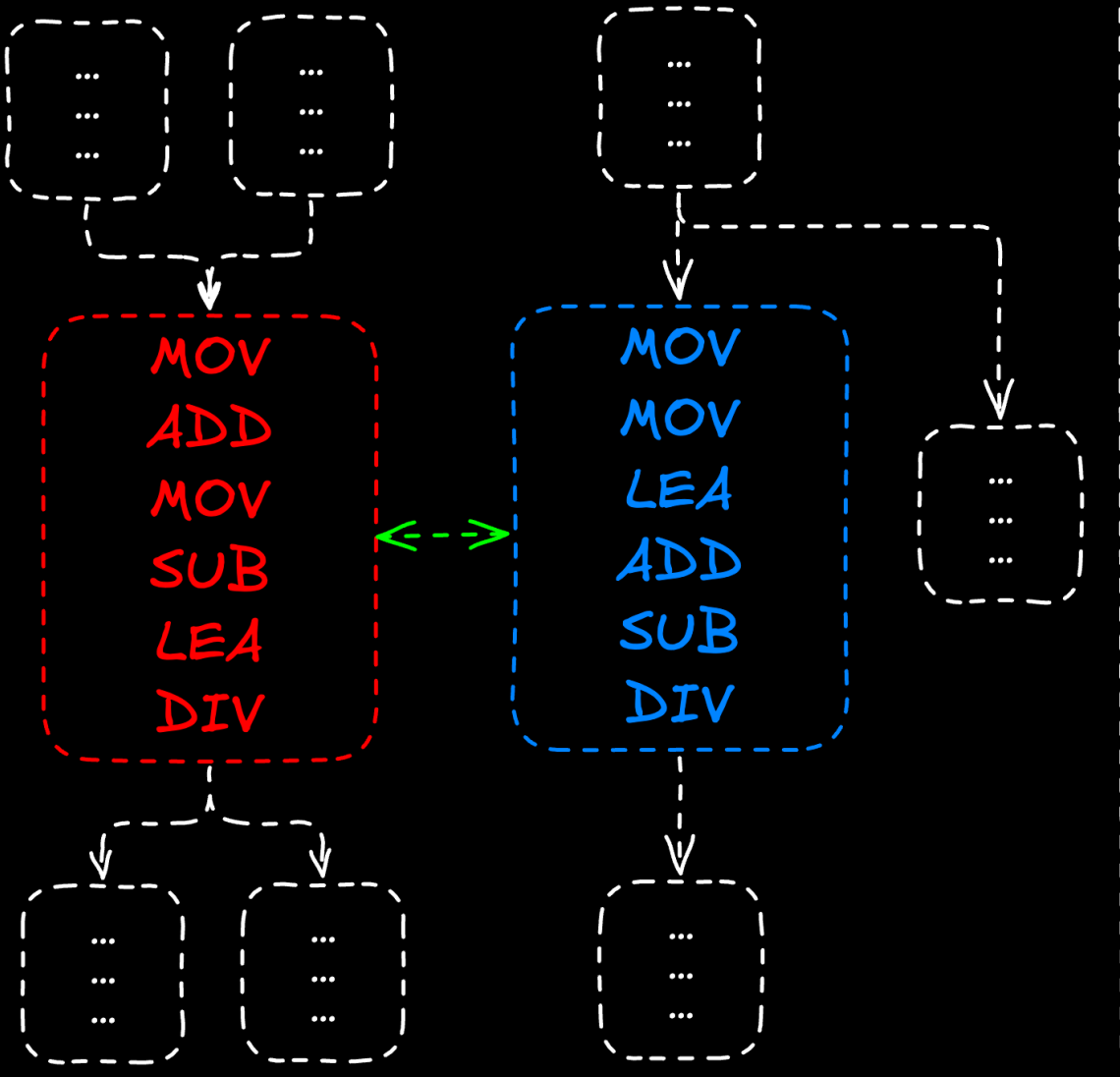
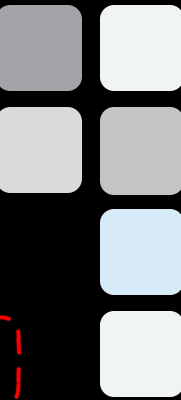
## Name Table



# GoResolver: Golang Symbol Extraction



# GoGrapher: Control-flow Graph Similarity

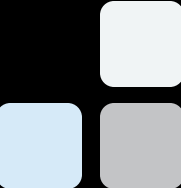


Similarity = 1.0

$$BBlock(P, Q) = \frac{|P \cap Q|}{|P \cup Q|}$$

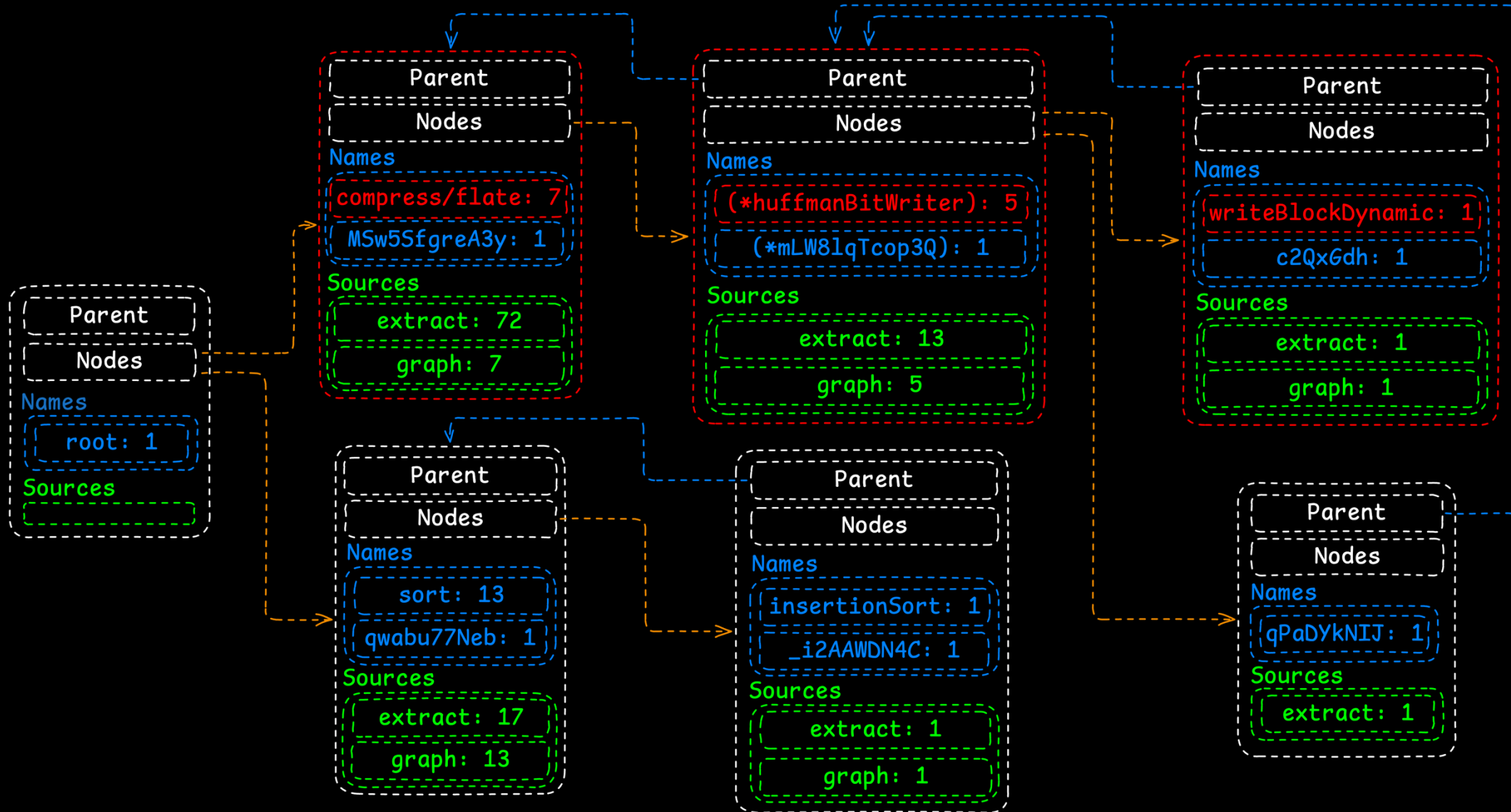
$$S(P, Q) = \frac{2 * BBlock(P, Q) + BBlock(P, Q)_{out} + BBlock(P, Q)_{in}}{4}$$

$$Sim(A, B) = \frac{\sum_{(P_i, Q_j) \in Match(A, B)} S(P_i, Q_j)}{min(m, n)}$$

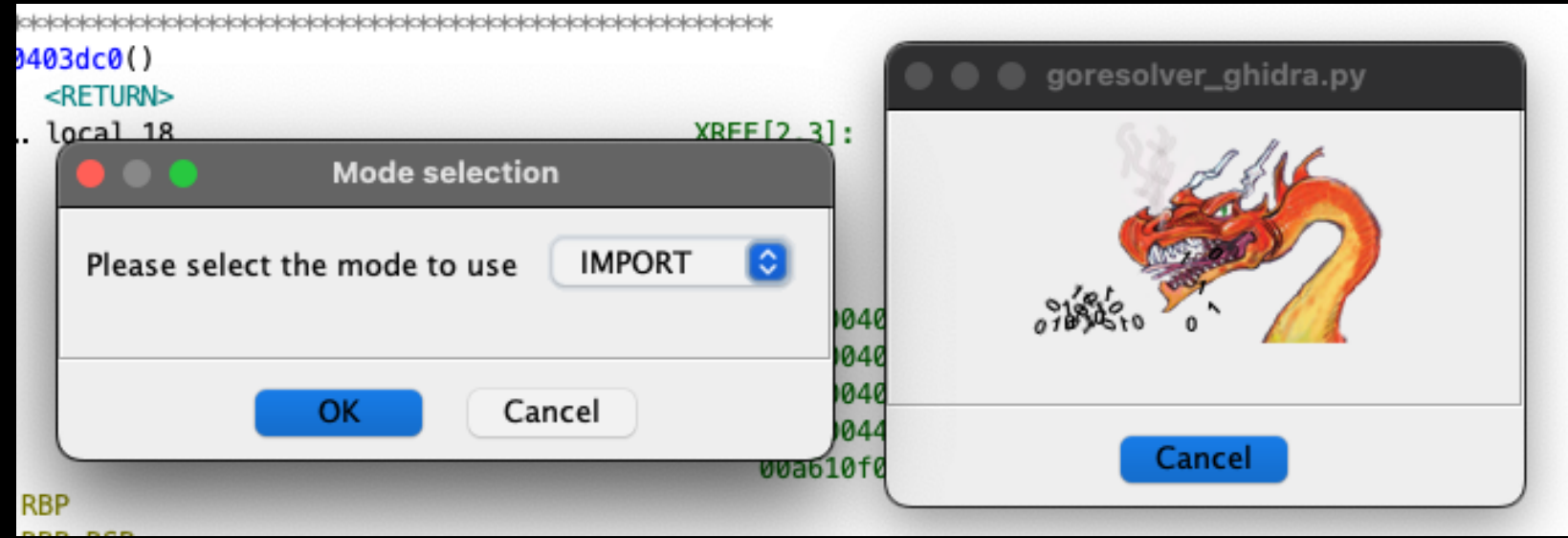
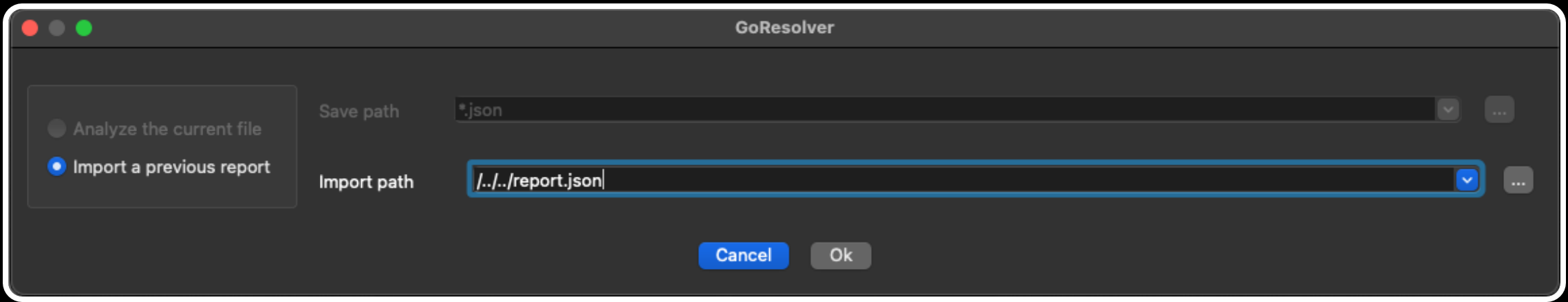


# Symbol Sources Aggregation

1: compress/flate.(\*huffmanBitWriter).c2Qx6dh\_\_writeBlockDynamic



# Plugins: GoResolver Workflow Integration



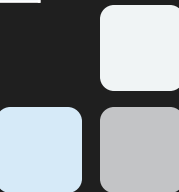
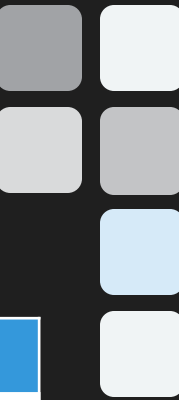
# Results

Go version autodetect [ go1.23.7 ]:

Mode	#Symbols	Ratio	#Resolved	Ratio
Extract (Only)	5730	100 %	0	0 %
Similarity (Only) – 90%	314	5,47 %	314	5,47 %
Combined	5731	100,01 %	577	10,06 %

Go version user-defined [ go1.23.4 ]:

Mode	#Symbols	Ratio	#Resolved	Ratio
Extract (Only)	5730	100 %	0	0 %
Similarity (Only) – 90%	482	8,41%	482	8,41%
Combined	5731	100,01 %	626	10,92%





# Results

Before

Function name	Segment
os_init	.text
unknown_libname_120	.text
unknown_libname_121	.text
sub_4C4805	.text
unknown_libname_122	.text
sub_4C487C	.text
unknown_libname_123	.text
os_ptr_SyscallError_Timeout	.text
sub_4C4953	.text
sub_4C4960	.text
sub_4C49E5	.text
sub_4C4A00	.text
sub_4C4B00	.text
sub_4C4BB7	.text
os_StartProcess	.text
sub_4C4CA0	.text
sub_4C4CC5	.text
sub_4C4CE0	.text
sub_4C508D	.text
sub_4C50C0	.text
sub_4C521E	.text
sub_4C5240	.text
sub_4C571E	.text
sub_4C5740	.text
sub_4C57A0	.text
sub_4C5800	.text
sub_4C5B79	.text
sub_4C5BA0	.text
sub_4C5C00	.text
sub_4C5C60	.text
sub_4C5CD1	.text
os_init_0	.text
sub_4C5DE0	.text
sub_4C6178	.text
os_commandLineToArgv	.text
sub_4C62E5	.text
sub_4C6300	.text
sub_4C6325	.text
os_getModuleFileName	.text
sub_4C6400	.text
os_ptr_File_Read	.text
sub_4C64BD	.text

Line 3423 of 8658, /os.init

After

Function name	Segment
SHZLKKDI_MjIb9g__aererwrap1	.text
os_init	.text
os_init_func1	.text
os_PvRklp	.text
sub_4C4805	.text
os_ptr_UlewAk_Error	.text
sub_4C487C	.text
os_ptr_UlewAk_Unwrap	.text
os_ptr_UlewAk_Timeout	.text
sub_4C4953	.text
os_s1nRYAVvipX8	.text
sub_4C49E5	.text
os_ptr_H5BJyF_gdLelzJLE	.text
os_ptr_H5BJyF_hVvk6yqafz	.text
sub_4C4BB7	.text
os_EP9j0Yf5YI	.text
os_ptr_H5BJyF_Release	.text
sub_4C4CC5	.text
os_ca1gEWDshe	.text
sub_4C508D	.text
os_ptr_ioUqde_String	.text
sub_4C521E	.text
os_ptr_H5BJyF_uyXYXaHBxkY	.text
sub_4C571E	.text
os_ptr_H5BJyF_uyXYXaHBxkY_deferwrap2	.text
os_ptr_H5BJyF_uyXYXaHBxkY_deferwrap1	.text
os_ptr_H5BJyF_j1PEZI_Y	.text
sub_4C5B79	.text
os_ptr_H5BJyF_j1PEZI_Y_deferwrap2	.text
os_ptr_H5BJyF_j1PEZI_Y_deferwrap1	.text
os_ptr_H5BJyF_s_tzGAC	.text
sub_4C5CD1	.text
os_init_0	.text
os_tqGjtBFe__readNextArg	.text
sub_4C6178	.text
os_wPI9mm2jojY__commandLineToArgv	.text
sub_4C62E5	.text
os_UFs5o6	.text
sub_4C6325	.text
os_ISBA0REQ	.text
os_ptr_ZMguJ7sUu5i_Name	.text
os_ptr_ZMguJ7sUu5i_Read	.text
sub_4C64BD	.text

Line 3423 of 8658, /os.init

# Results

## Extract (only)

Function name	Segment
sub_635435	.text
MSw5SfgreA3y_ptr_gDh91Q_uAZjJ5izG	.text
MSw5SfgreA3y_ptr_gDh91Q_uz5jjRzvs	.text
MSw5SfgreA3y_ptr_cctPHr_iFS5FqlMWnN	.text
sub_635907	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_ahqLlh	.text
sub_635A47	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_mhxiqG4	.text
sub_635BBE	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_nLHubp3h	.text
sub_635DB5	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_HLmlwmxH	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_rbHiOZFmWfVx	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_q6EXm9CdFb	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_zpos3R9av1	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_umt7VN	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_nopMQgOa	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_c2QxGdh	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_qPaDYkNIJ	.text
sub_6373EF	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_j3xx_vuS	.text
sub_6376EE	.text
MSw5SfgreA3y_init_0	.text
MSw5SfgreA3y_ptr_mLW8lqTcop3Q_agPNVzJKjqR	.text
MSw5SfgreA3y_i0cB0T	.text
j_compress_flate_generateFixedLiteralEncoding	.text
MSw5SfgreA3y_ptr_fB7_eAGUagjz_sapQ24Jl7b5	.text
sub_63829A	.text
MSw5SfgreA3y_ptr_fB7_eAGUagjz_aU4BpDEoWGtN	.text
sub_638485	.text
MSw5SfgreA3y_ptr_fB7_eAGUagjz_glForz0QWDeY	.text
MSw5SfgreA3y_ljD2x7MPL_Len	.text
MSw5SfgreA3y_ljD2x7MPL_Less	.text
MSw5SfgreA3y_ljD2x7MPL_Swap	.text
MSw5SfgreA3y_tu_QZa_Len	.text
MSw5SfgreA3y_tu_QZa_Less	.text
MSw5SfgreA3y_tu_QZa_Swap	.text
MSw5SfgreA3y_Z4tF4_Error	.text
sub_638985	.text

Line 7478 of 8658, /MSw5SfgreA3y\_ptr\_mLW8lqTcop3Q.c2QxGdh

## Extract + Similarity

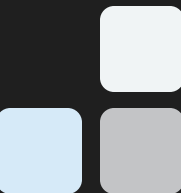
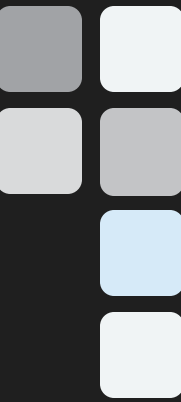
Function name	Segment
sub_635435	.text
compress_flate_ptr_gDh91Q_ptr_deflateFast_uAZjJ5izG	.text
compress_flate_ptr_gDh91Q_ptr_deflateFast_uz5jjRzvs_reset	.text
compress_flate_ptr_cctPHr_iFS5FqlMWnN	.text
sub_635907	.text
compress_flate_ptr_huffmanBitWriter_ahqLlh_flush	.text
sub_635A47	.text
compress_flate_ptr_huffmanBitWriter_mhxiqG4_writeBits	.text
sub_635BBE	.text
compress_flate_ptr_huffmanBitWriter_nLHubp3h	.text
sub_635DB5	.text
compress_flate_ptr_huffmanBitWriter_HLmlwmxH	.text
compress_flate_ptr_huffmanBitWriter_rbHiOZFmWfVx	.text
compress_flate_ptr_huffmanBitWriter_q6EXm9CdFb_writeCode	.text
compress_flate_ptr_huffmanBitWriter_zpos3R9av1	.text
compress_flate_ptr_huffmanBitWriter_umt7VN_writeStoredHeader	.text
compress_flate_ptr_huffmanBitWriter_nopMQgOa	.text
compress_flate_ptr_huffmanBitWriter_c2QxGdh_writeBlockDynamic	.text
compress_flate_ptr_huffmanBitWriter_qPaDYkNIJ	.text
sub_6373EF	.text
compress_flate_ptr_huffmanBitWriter_j3xx_vuS	.text
sub_6376EE	.text
compress_flate_init_0	.text
compress_flate_ptr_huffmanBitWriter_agPNVzJKjqR	.text
compress_flate_i0cB0T	.text
j_compress_flate_generateFixedLiteralEncoding	.text
compress_flate_ptr_fB7_eAGUagjz_sapQ24Jl7b5	.text
sub_63829A	.text
compress_flate_ptr_fB7_eAGUagjz_aU4BpDEoWGtN	.text
sub_638485	.text
compress_flate_ptr_fB7_eAGUagjz_glForz0QWDeY	.text
compress_flate_ljD2x7MPL_Len	.text
compress_flate_ljD2x7MPL_Less	.text
compress_flate_ljD2x7MPL_Swap	.text
compress_flate_tu_QZa_Len	.text
compress_flate_tu_QZa_Less	.text
compress_flate_tu_QZa_Swap	.text
compress_flate_Z4tF4_Error	.text
sub_638985	.text

Line 7478 of 8658, /compress\_flate\_ptr\_huffmanBitWriter.c2QxGdh\_writeBlockDynam



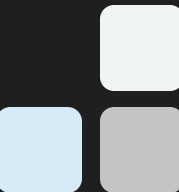
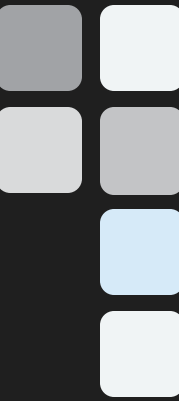
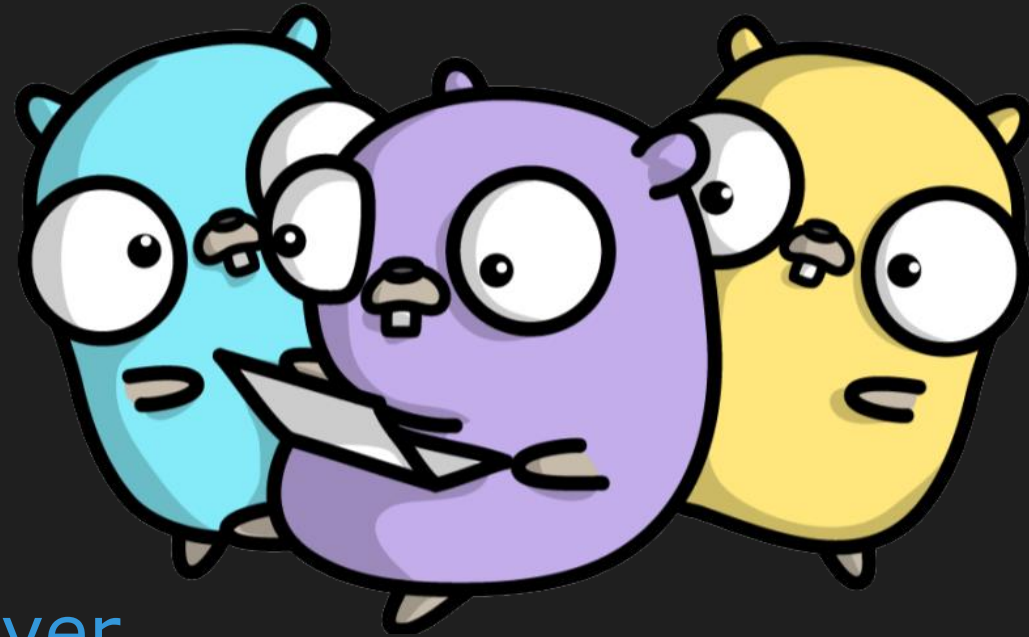
# Conclusion

- Go language (Golang) is increasing in popularity with both legitimate developers *and* malicious actors.
- Volexity frequently encounters malware written in Golang, often obfuscated to hinder analysis.
- Obfuscated Golang malware samples are significantly harder to statically analyze for reverse engineers.
- GoResolver's control-flow graph similarity techniques offer a significant advantage in recovering garbled symbol information.



# Conclusion

- The GoResolver toolchain is being actively developed by Volexity
- New upcoming features further expand Go binary analysis !
- Fully Open-Source and available at <https://github.com/volexity/GoResolver>



# Special Thanks



Mandiant

<https://github.com/mandiant/GoReSym>



[www.ijcse.com/docs/INDJCSE20-11-03-237.pdf](http://www.ijcse.com/docs/INDJCSE20-11-03-237.pdf)  
[hilim@kyungnam.ac.kr](mailto:hilim@kyungnam.ac.kr)



Thanks!

**CONTACT US:**

Killian Raimbaud - [kraimbaud@volexity.com](mailto:kraimbaud@volexity.com)

Paul Rascagneres - [prascagneres@volexity.com](mailto:prascagneres@volexity.com)

