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

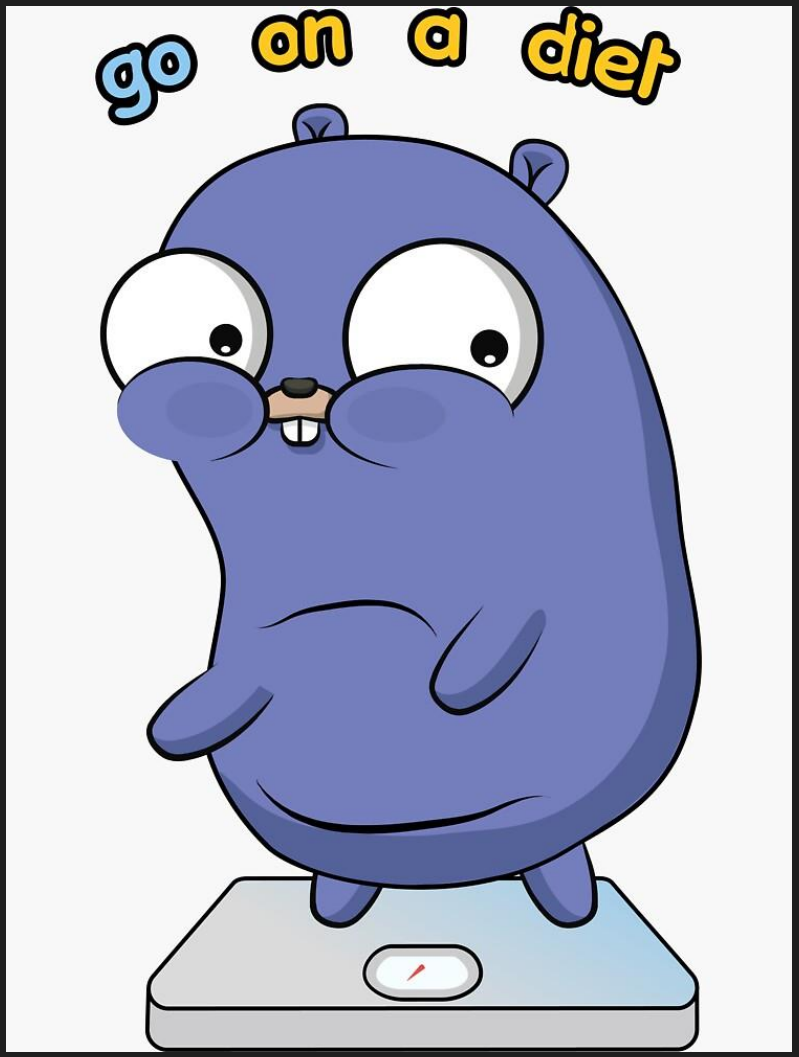
- Golang Malware
- Garble Obfuscation
- Control-flow Graphs Similarity
- GoResolver
- Conclusion

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:

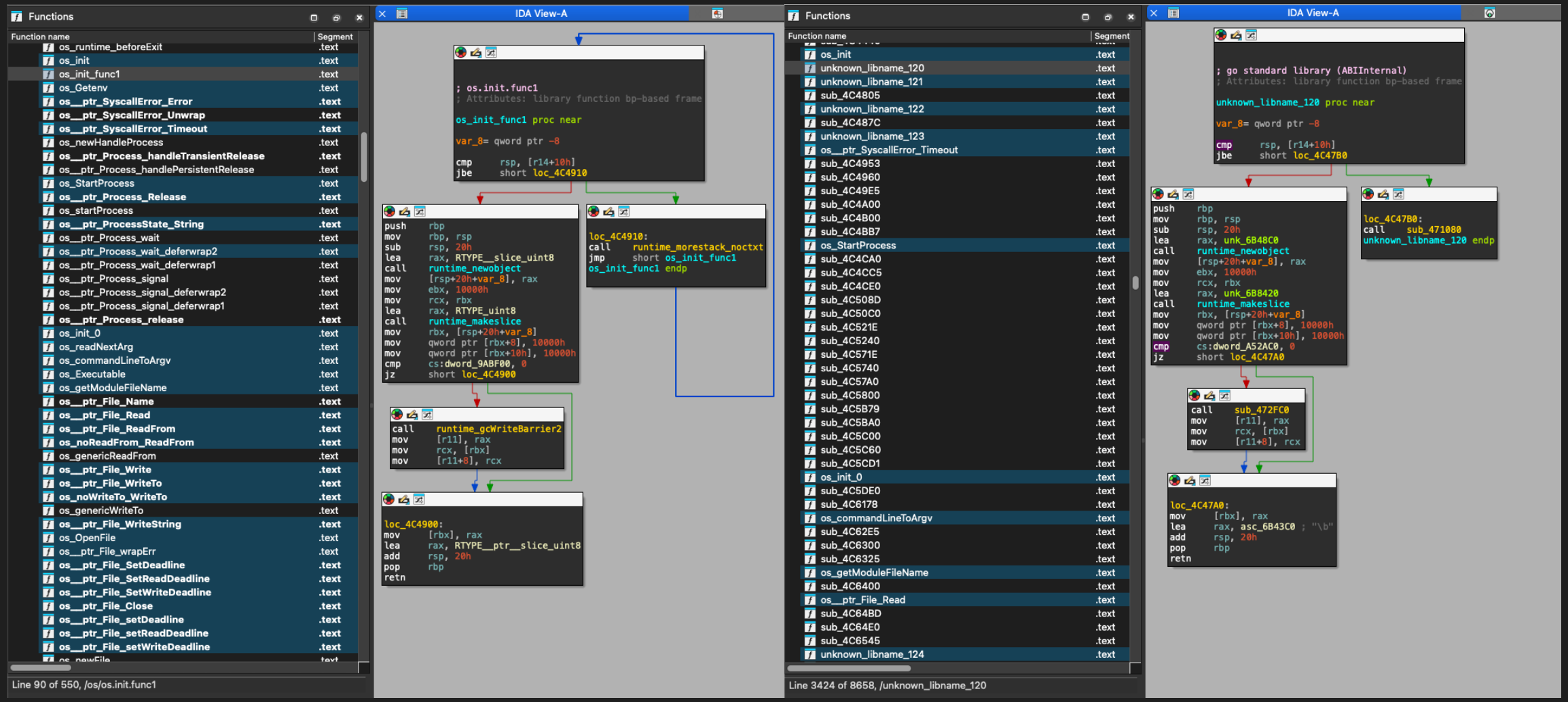
- Replace as many useful identifiers as possible with short base64 hashes
- Replace package paths with short base64 hashes
- Replace filenames and position information with short base64 hashes
- Remove all build and module information
- Strip debugging information and symbol tables via `-ldflags="-w -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



Garble Limitations

- Randomized names stay consistent across all functions

Functions

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

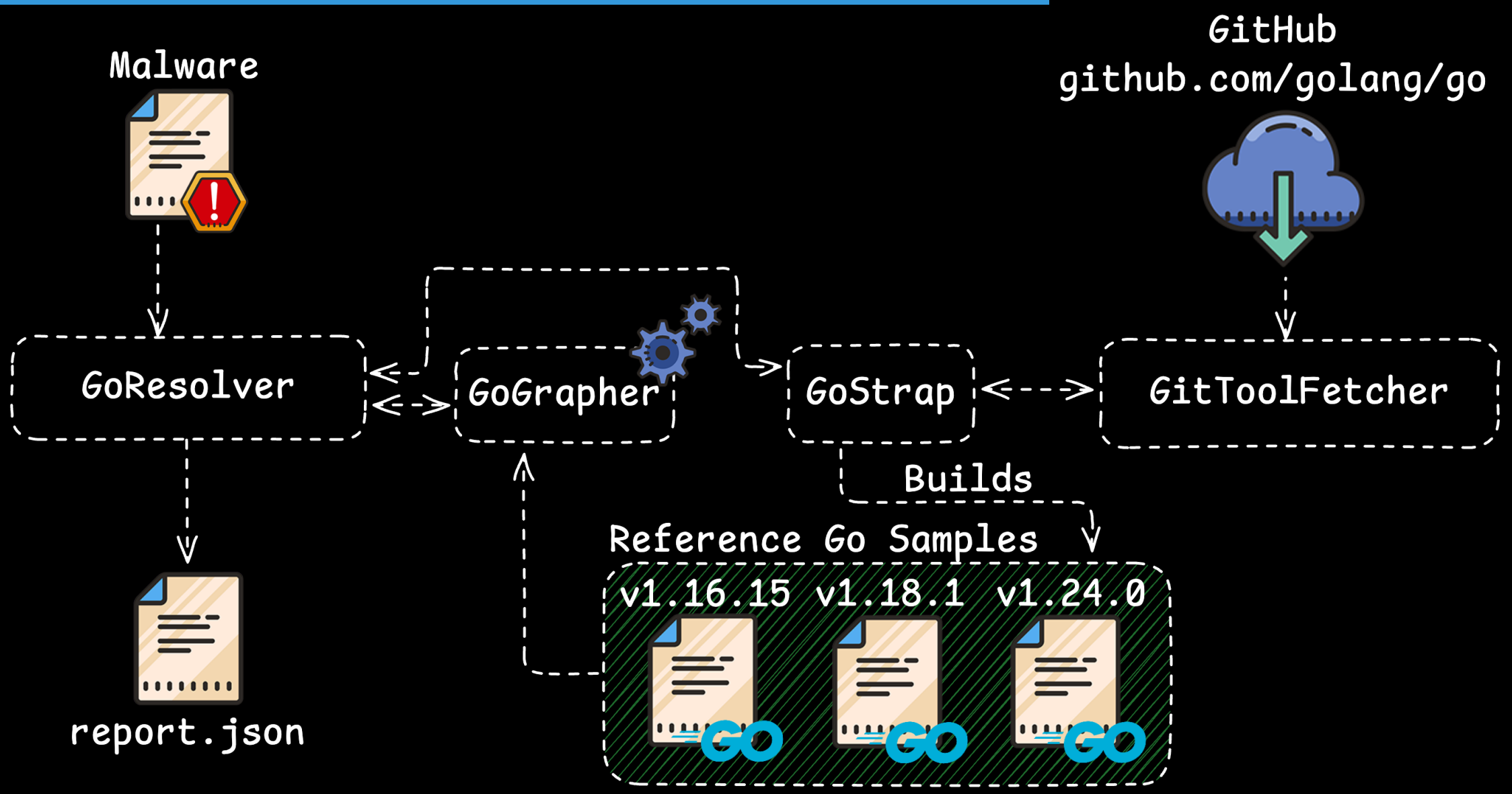
GoReSym

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.

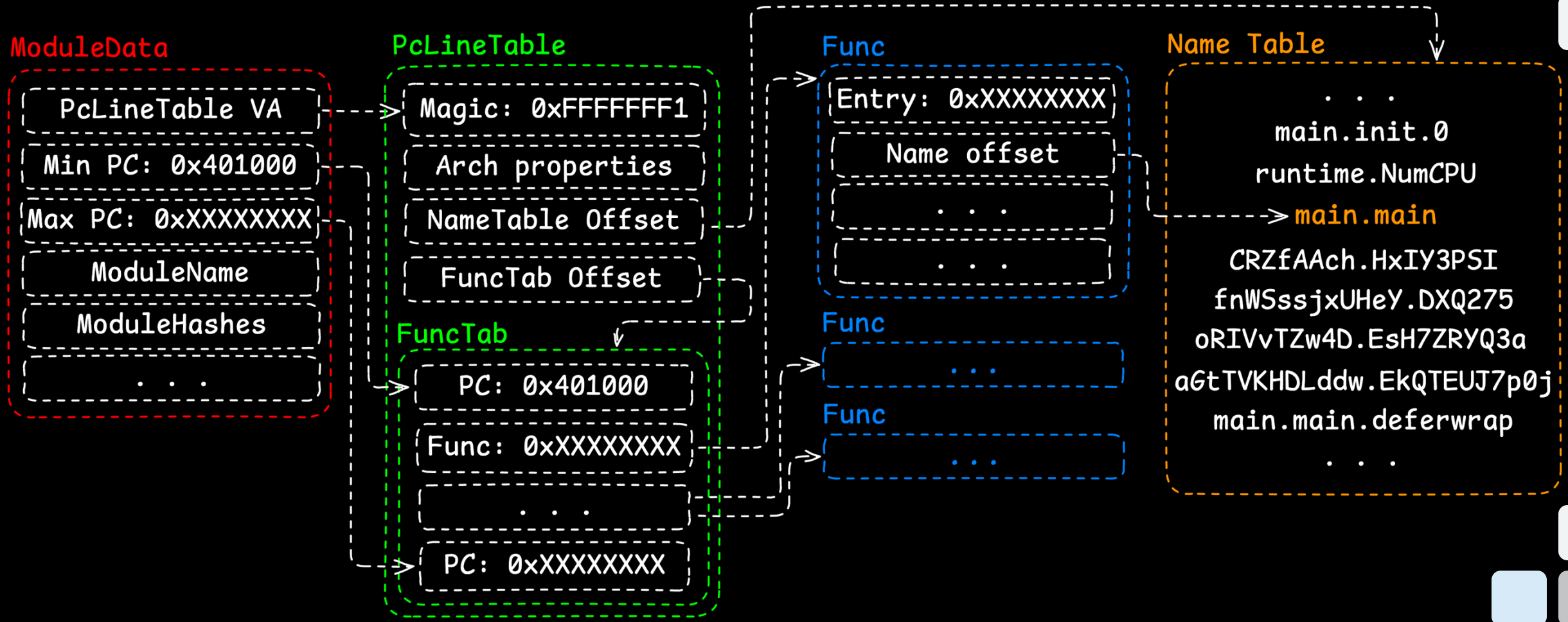
The upstream Go runtime code is extended to handle:

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

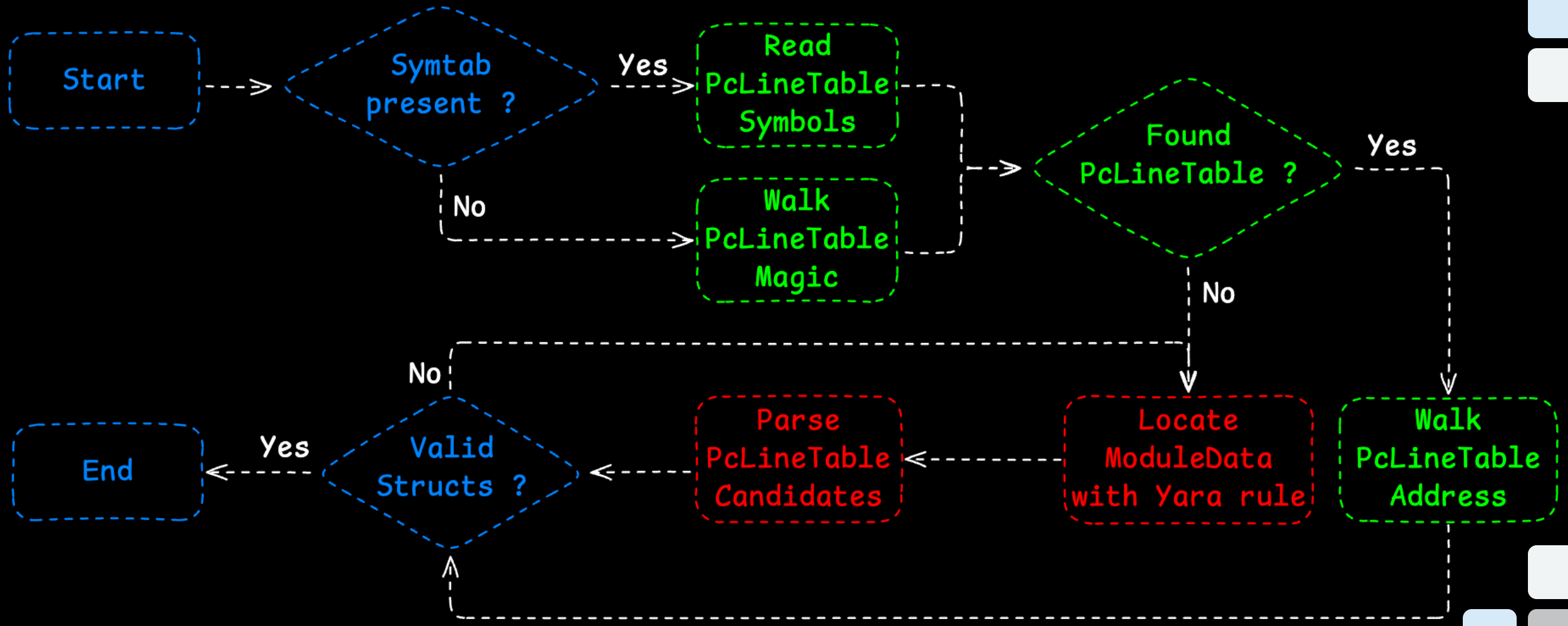
GoResolver: Toolchain Overview



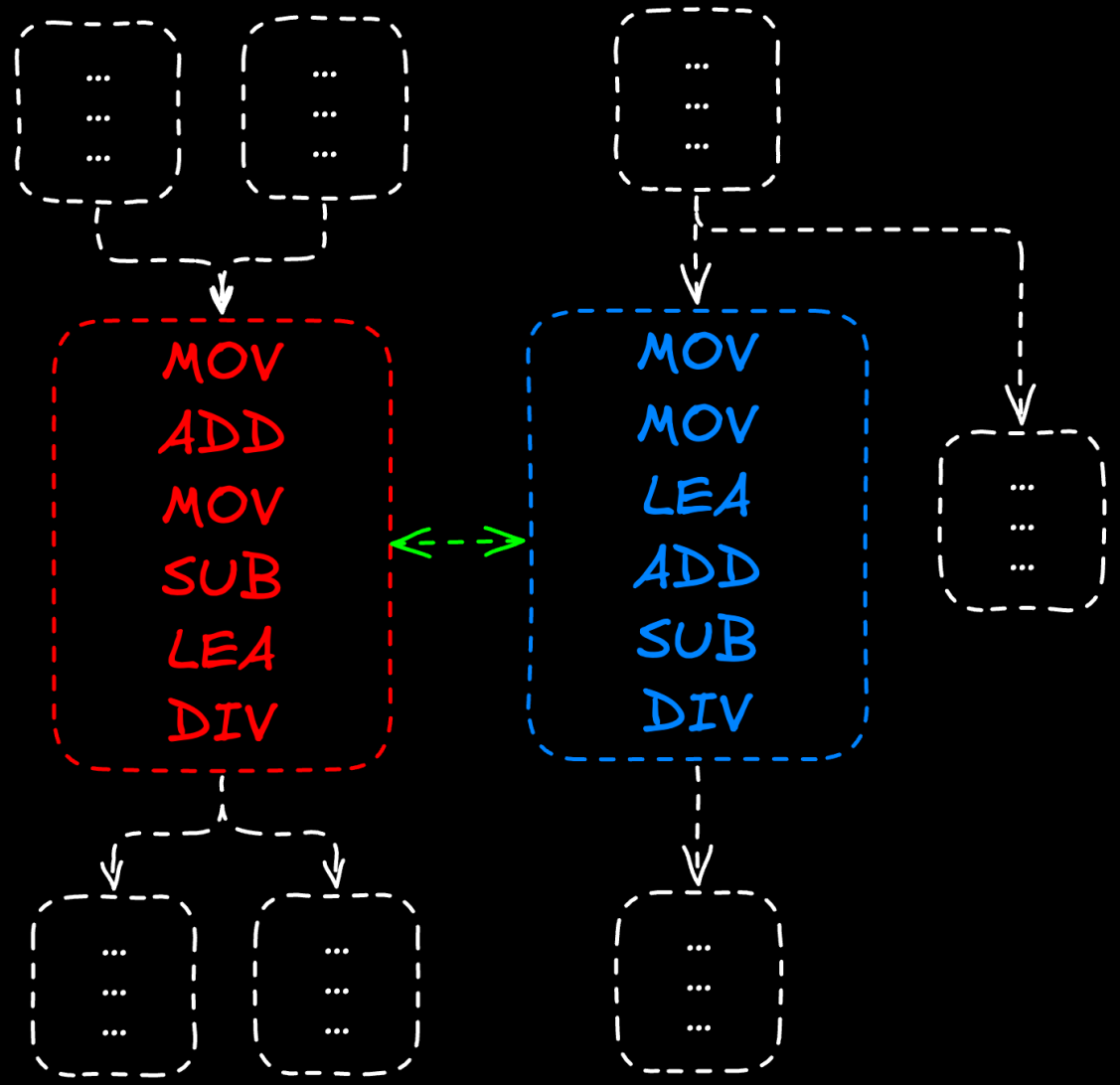
GoResolver: Golang Symbol Extraction

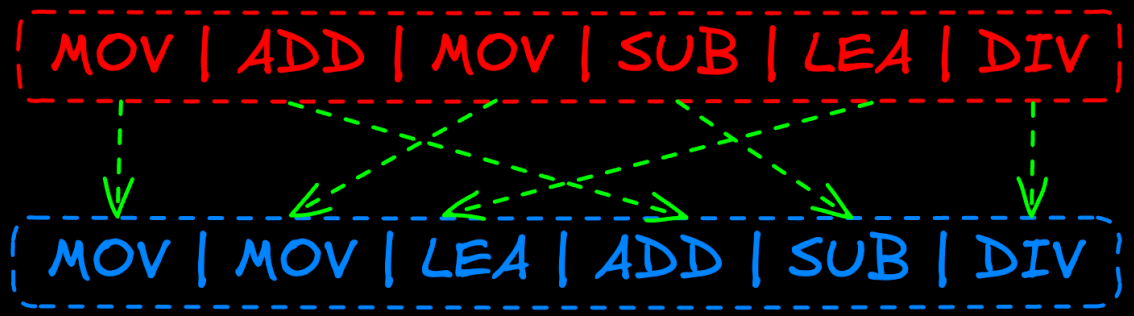


GoResolver: Golang Symbol Extraction



GoGrapher: Control-flow Graph Similarity





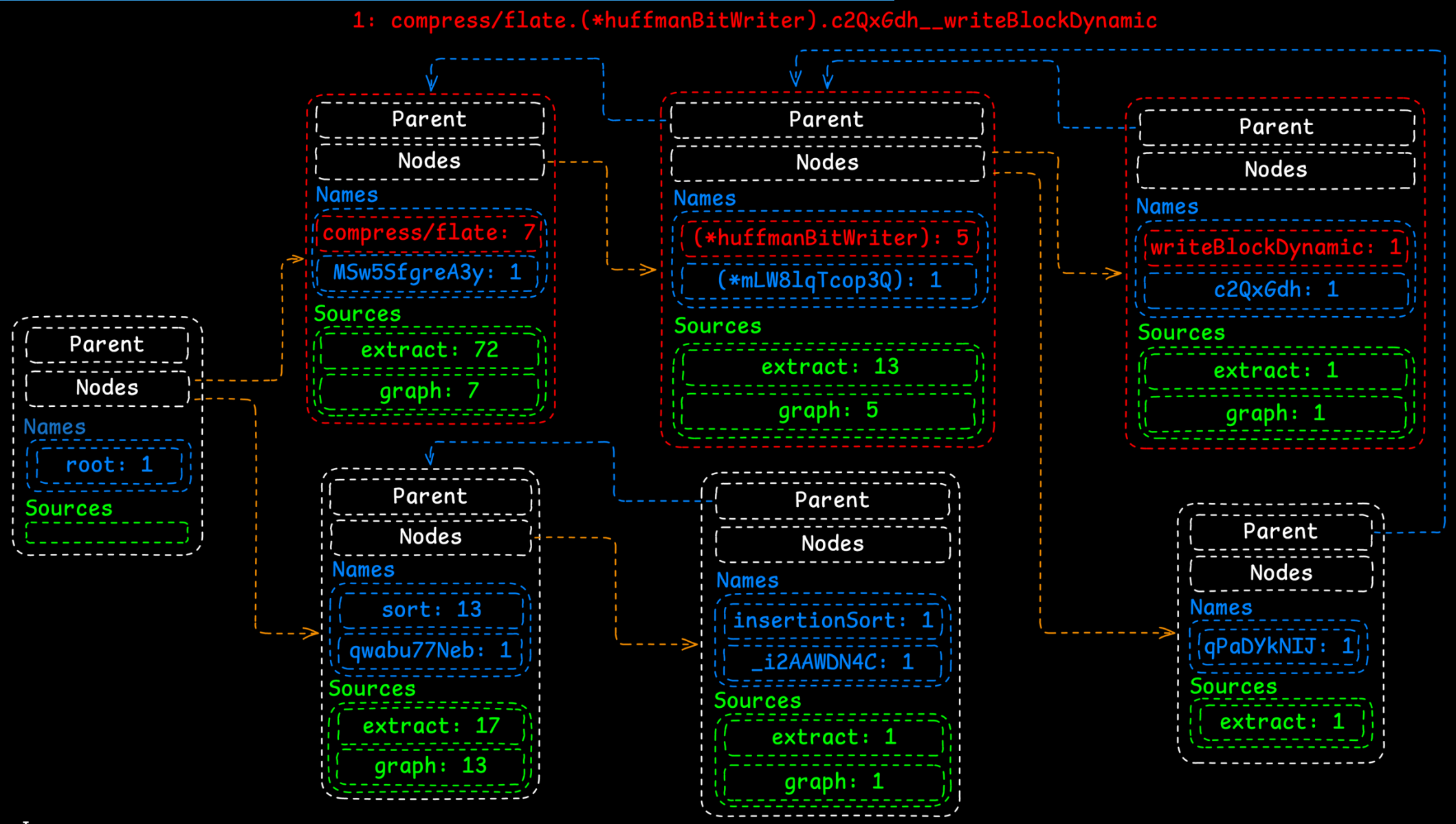
Similarity = 1.0

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

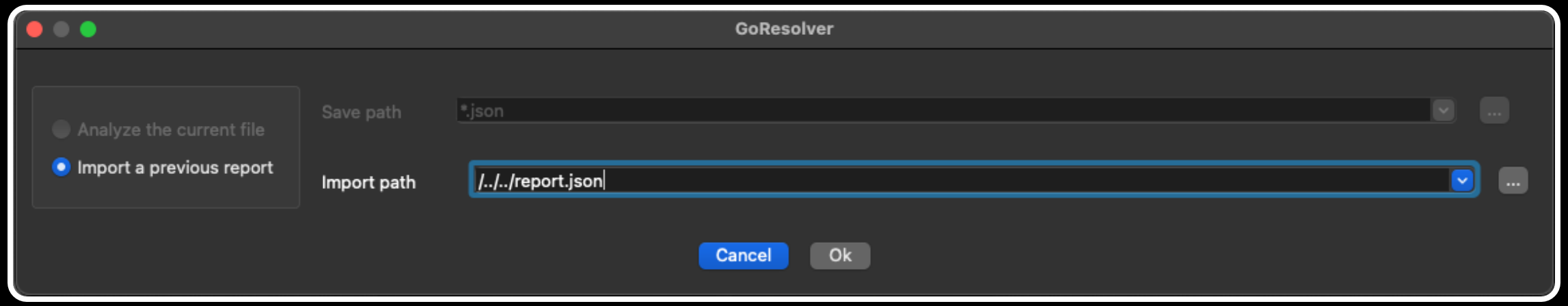
$$S(P, Q) = \frac{2 * BBlock(P, Q) + BBlock(P, Q)_{out} + BBlock(P, Q)_{in}}{4}$$

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

Symbol Sources Aggregation



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

Functions

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

Functions

Function name	Segment
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_gdLeIzJLE	.text
os__ptr_H5BJyF_hVk6yqafz	.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_wPl9mm2jojY__commandLineToArgv	.text
sub_4C62E5	.text
os_UFs5o6	.text
sub_4C6325	.text
os_IS8A0REQ	.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_mhjxiqG4	.text
sub_635BBE	.text
MSw5SfgreA3y__ptr_mLW8lqTcop3Q_nLHubp3h	.text
sub_635DB5	.text
MSw5SfgreA3y__ptr_mLW8lqTcop3Q__HLmlwmxH	.text
MSw5SfgreA3y__ptr_mLW8lqTcop3Q_rbHiOZFnWfVx	.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_Z4tFf4_Error	.text
sub_638985	.text

Line 7478 of 8658, /MSw5SfgreA3y._ptr_mLW8lqTcop3Q.c2QxGdh

Extract + Similarity

Function name	Segmen
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_mhjxiqG4__writeBits	.text
sub_635BBE	.text
compress_flate__ptr_huffmanBitWriter_nLHubp3h	.text
sub_635DB5	.text
compress_flate__ptr_huffmanBitWriter__HLmlwmxH	.text
compress_flate__ptr_huffmanBitWriter_rbHiOZFnWfVx	.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_Z4tFf4_Error	.text
sub_638985	.text

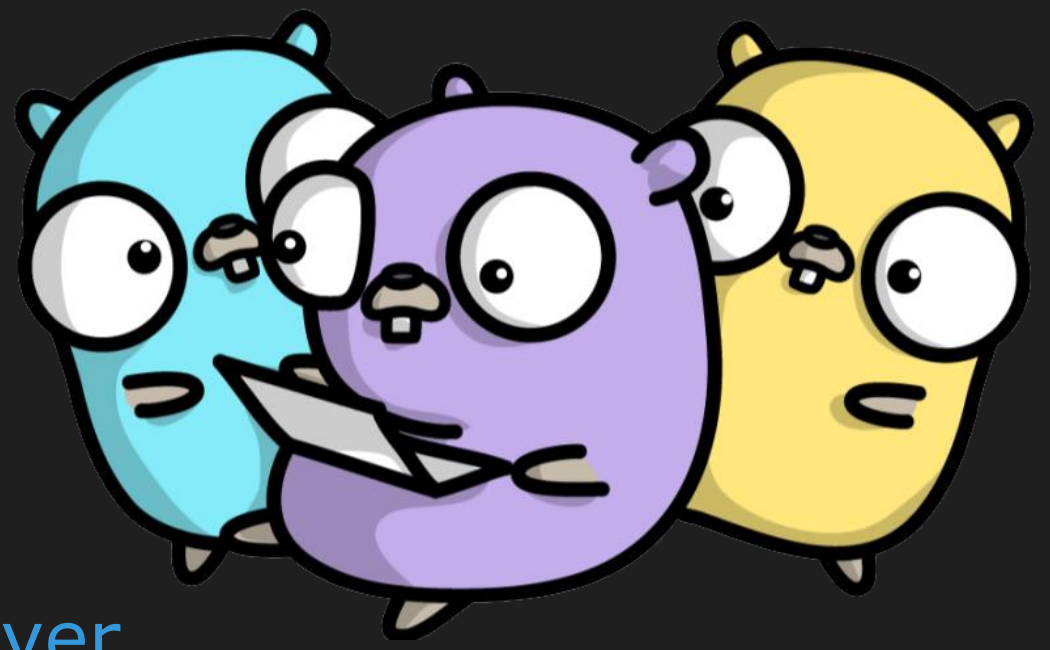
Line 7478 of 8658, /compress_flate._ptr_huffmanBitWriter.c2QxGdh__writeBlockDynam

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
hilim@kyungnam.ac.kr

Thanks!

CONTACT US:

Killian Raimbaud - kraimbaud@volexity.com

Paul Rascagneres - prascagneres@volexity.com