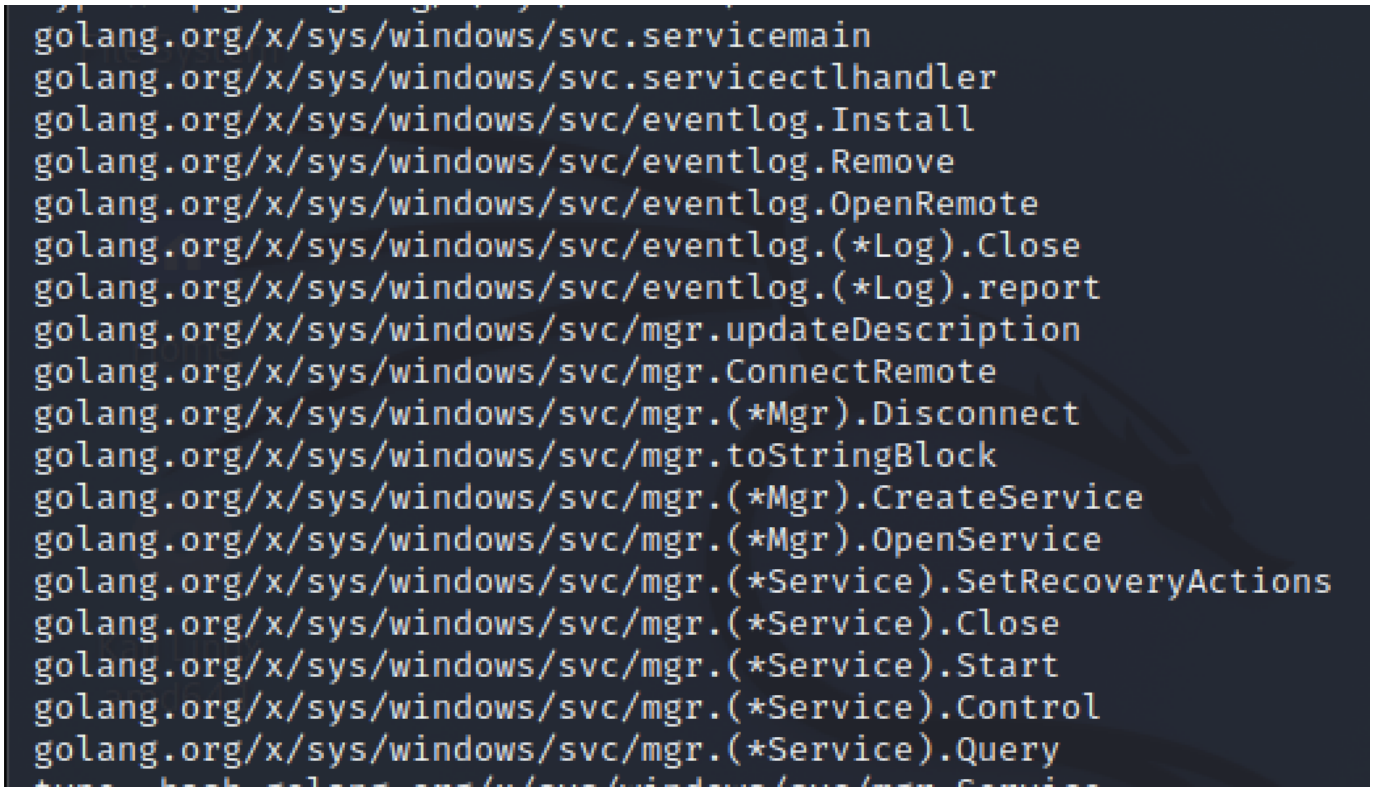
GoTime - GO Malware

> https://github.com/HuskyHacks/PMAT-labs/blob/main/labs/3-5.GoTime-GoMalware/Backdoor.srvupdat.exe.malz.7z

This binary is all about understandig and training our eyes to look for clues which help us in distinguishing the programming language in which the malware might have been written.

Beginning, if we run the strings command, it will yeild a loads of result. Cutting short, we can find golang domain in strings.



Secondly, we can also search for the main function which is GoLang case is "main.main".

```
└─PS> strings ./Backdoor.srvupdat.exe.malz | grep main.main
main.main
main.main.stkobj
main.main
```

Moreover, ".symtab" and Go Build ID can also be found in the Go Binaries.

```
B/124
B.idata
.symtab
 Go build ID: "3RKTvI30Uc68A_c6-sMB/uAW0VxD6KHkdILq2izWY/BG7xKfK9Q2QEBRq4rzpy/NCZHQC8
D$ H
D$(H
D$(H
```

As the code is not obfuscated, we can also hunt down the URLs which might not be obfuscated in the build.

```
strings ./Backdoor.srvupdat.exe.malz | grep "http://"
```

```
internal error: missing cancel errorexitsyscall: sy
ith non-pointer typehttp: no Client.Transport or De
tatehttp://ec2-3-109-20-24-srv3.local/favicon.icoin
ternal error: connCount underflowparsing/packing of
alid method indexreflect: nil type passed to Type.A
=tls: internal error: failed to update binderstls:
```

Moreover, if the Go Malware makes any kind of internet connection and the Author has not changed the User-Agent value then you'll be able to find the below User-Agents within the binary or via WireShark/Network Monitoring Tools.

```
strings ./Backdoor.srvupdat.exe.malz | grep "Go-http-client/1.1" --color=always
```

```
┌──(froggy㉿kali)-[/home/froggy/Desktop/PMAT]
└─PS> strings ./Backdoor.srvupdat.exe.malz | grep "Go-http-client/1.1" --color=always
wrong medium type  but memory size  because dotdotdot to non-Go memory , locked to thread2980232238769531 25: day out of rangeArab Standard TimeCaucasian_AlbanianCertGetN
ameStringWCloseServiceHandleCommandLineToArgvWCreateFileMappingWCreateWellKnownSidCryptUnprotectDataCuba Standard TimeEnumProcessModulesExpectation FailedFLOW_CONTROL_ER
RORFiji Standard TimeGetComputerNameExWGetCurrentThreadIdGetExitCodeProcessGetFileAttributesWGetModuleBaseNameWGetModuleFileNameWGetModuleHandleExWGetSidSubAuthorityGetV
olumePathNameWGo-http-client/1.1Go-http-client/2.0Iran Standard TimeLookupAccountNameWMakeSelfRelativeSDMethod Not AllowedOmsk Standard TimePFXImportCertStorePermanent R
edirectProxy-AuthenticateQueryServiceStatusRCodeServerFailureRFS specific errorRegional_IndicatorRussia Time Zone 3SetFileAttributesWSystemFunction036
```

> strings ./Backdoor.srvupdat.exe.malz | grep "Go-http-client/2.0" --color=always

```
┌──(froggy㉿kali)-[/home/froggy/Desktop/PMAT]
└─PS> strings ./Backdoor.srvupdat.exe.malz | grep "Go-http-client/2.0" --color=always
wrong medium type  but memory size  because dotdotdot to non-Go memory , locked to thread2980232238769531 25: day out of rangeArab Standard TimeCaucasian_AlbanianCertGetN
ameStringWCloseServiceHandleCommandLineToArgvWCreateFileMappingWCreateWellKnownSidCryptUnprotectDataCuba Standard TimeEnumProcessModulesExpectation FailedFLOW_CONTROL_ER
RORFiji Standard TimeGetComputerNameExWGetCurrentThreadIdGetExitCodeProcessGetFileAttributesWGetModuleBaseNameWGetModuleFileNameWGetModuleHandleExWGetSidSubAuthorityGetV
olumePathNameWGo-http-client/1.1Go-http-client/2.0Iran Standard TimeLookupAccountNameWMakeSelfRelativeSDMethod Not AllowedOmsk Standard TimePFXImportCertStorePermanent R
edirectProxy-AuthenticateQueryServiceStatusRCodeServerFailureRFS specific errorRegional_IndicatorRussia Time Zone 3SetFileAttributesWSystemFunction036
```

The following key-points are required to be noted during GO-Malware Analysis.

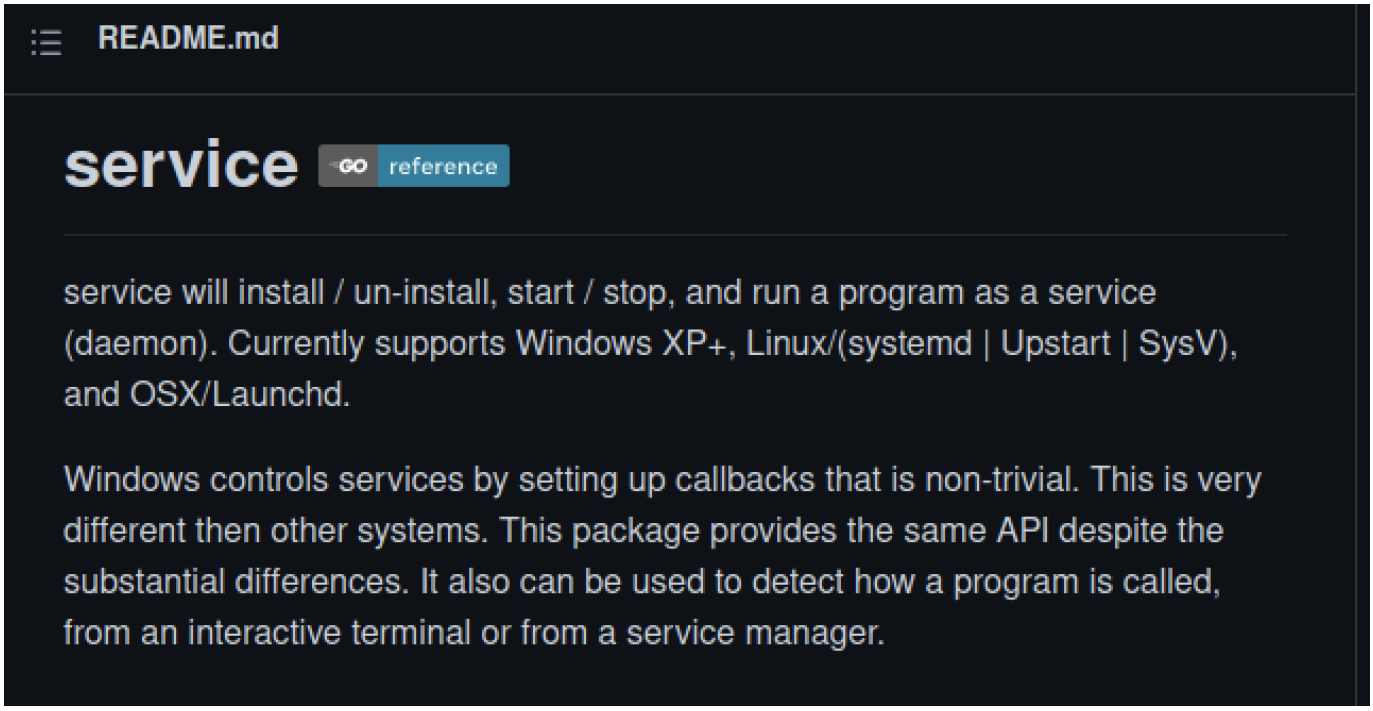
- OSX or Linux samples with embedded URLs referencing ‘Go.org’
- Samples using the ‘Go-http-client/1.1’ user-agent
- Samples using the ‘GRequests’ user-agent
- PE samples containing the ‘.symtab’ section name
- PE samples using a series of identified import hashes
- OSX samples referencing Google’s gopacket github repository
- OSX samples referencing gopkg.in
- Samples matching YARA rules

While analyzing the GO Sample, a GitHub repo comes up!

```
vendor/golang.org/x/net/http2/hpack.appendIndexedNa
vendor/golang.org/x/net/http2/hpack.(*Decoder).pars
github.com/kardianos/service.(*Config).execPath
```

On checking the GitHub Repo, we can observe that it is using

service repo code which will enable this program to run as a service on the OS.



During the analysis of this particular binary, I came across the OWNER name who actually compiled the GO Binary.

> /home/husky

Resources

- https://unit42.paloaltonetworks.com/the-gopher-in-the-room-analysis-of-golang-malware-in-the-wild/