DISCOVER documentation v1.0.2

Document Version

Version	Date	Author	Description
1.0.0	23.06.2021	mpetavy	Initial release
1.0.2	23.06.2010	mpetavy	some typos

Description

DISCOVER is a service discovery tool by nwtwork broadcasting

Connection roles and data roles

With two DISCOVER there must always be a "client" and a "server" to communicate.

DISCOVER can be starter in "client" (-c) or "server" (-s) connection role.

- As a "server" DISCOVER waits for incoming network broadcast messages
- As a "client" DISCOVER sends network broadcast messages and waits fro responses of running servers

Architecture

Network boradcasts is a very commonway to to services recovers by using the UDP protocol.

Please keep in mind that such broadcasts are not transfered outside the current network or routers.

A service discovery is setup and find by an UID and a broadcast network port.

The DISOVER server "-s" waits for incoming network broadcasts via the UDP protocol. Only if the same UDP port for broadcast sending and receiving of DISOVER server "-s" and client "-c" is used and the same UID "-uid" is defined on both endpoints a successfull service discovery happens. The DISCOVER server then reports back to the DISCOVER client its "-info" information.

Sample:

```
discover -s :5000 -uid hydra -info "myinfo"
discover -c :5000 -uid hydra
```

GO development

DISCOVER is developed with the Google GO tooling.

Current used version 1.16.5

By using the GO programming language (https://golang.org) multiple architectures and OS are supported. You can find a complete list of all supported architectures and OS at https://github.com/golang/go/blob/master/src/go/build/syslist.go

Currently these environments are tested in development

- Linux (x86_64)
  - Manjaro on x86_64 based PC https://www.manjaro.org
  - Raspian on ARM7 based Raspberry Pi Model 3 Model B+ https://www.raspberrypi.org/downloads/raspbian
- Windows 10 (x86_64)

Compiling DISCOVER

Before DISCOVER can be compiled please make sure the following tasks in this order.

1. i18n and opensource license files. To generate those files please execute inside the DISCOVER source code folder the following command "go run . -codegen".
    i. This generates an updated "static/DISCOVER.i18n" file with all i18n strings inside the DISCOVER source code files.
    ii. This generates an updated "static/discover-opensource.json" file with an listing if all used opensource modules along with their license infos.
2. BINPACK resources. All resources of DISCOVER must be transpiled to "binpack" source code files in order to be compiled statically to the DISCOVER executable. For that please use the BINPACK executable (https://github.com/mpetavy/binpack). Execute the transpile with the command "binoack -i static" inside the DISCOVER source code folder. After successfull execution an updated GO soource code file "binpack.go" is generated.
3. "vendor.tar.gz" file. When DISCOVER is compiled with Docker the compilation process uses GO's feature of "vendor"ing, That means the GO compiler in the Docker build does not use the standard GOPATH directory for 3d party modules source code files but uses the "vendor" directory in the DISCOVER source code folder. The "vendor" is generated automatically in the Docker build by untaring the TAR file "vendor.tag.gz". To update the "vendor.tar.gz" file to match the latest GO modules in the GOPATH of the development environment the batch job "update-vendor.bat" can be used.

Build with Docker

DISCOVER can be built either by the BUILD.SH (Linux) or BUILD.BAT (Windows) batch jobs. The Build uses Docker to generate an Image in which the complete packages for Windows and Linux are generated.

This is done by using GO's built-in feature to do cross-compiling to any supported plattform inside the Docker images.

After the docker image creatiion a temporaty docker container is built from which the following 3 software packages are extracted:

Sample for Version "1.0.0" and Build number "1234":

- discover-1.0.0-1234-linux-amd64.tar.gz
- discover-1.0.0-1234-linux-arm6.tar.gz
- discover-1.0.0-1234-windows-amd64.tar.gz

Those software packages containes everything for running DISCOVER on the defined platform.

Build manual

To build a binary executable for your preferred OS please do the following:

1. Install the GO programming language support (http://golang.org)
2. configure your OS env environment with the mandatory GO variables
    i. GOROOT (points to your)
    ii. GOPATH (points to your)
    iii. OS PATH (points to your /bin)
3. Open a terminal
4. CD into your GOPATH root directory
5. Create a "src" subdirectory
6. CD into the "src" subdirectory
7. Clone the DISCOVER repository
8. CD into the "DISCOVER" directory
9. Build:
    i. If you would like to cross compile to an other OS/architecture define the env variable GOOS and GOARCH along to the values defined here https://github.com/golang/go/blob/master/src/go/build/syslist.go
    ii. Build DISCOVER by "go install". Multiple dependent modules will be downloaded during the build
    iii. After a successful build you will find the DISCOVER executable in the "GOPATH\bin" directory

Installation as application

Like all other GO based application there is only the file `DISCOVER.exe` or `DISCOVER` which contains the complete application.

Just copy this executable into any installation directory you would like. Start the application by calling the executable `DISCOVER.exe` or `./DISCOVER`

Installation as OS service

Follow the instructions "Installation as application". To register DISCOVER as an OS service do the following steps.

1. Open a terminal
2. Switch to root/administrative rights
3. CD into your installation directory
4. Installation DISCOVER as an OS service:
   i. Windows: `DISCOVER -service install`
   ii. Linux: `./DISCOVER -service install`
5. Uninstallation DISCOVER as an OS service:
   i. Windows: `DISCOVER -service uninstall`
   ii. Linux: `./DISCOVER -service uninstall`

Running DISCOVER with Docker

The Linux amd64 package containes everything for running DISCOVER with Docker. Just use the provided " docker-compose-up.bat" and "docker-compose-down.bat". Here a sample Dockerfile:

```
FROM alpine:latest
RUN mkdir /app
WORKDIR /app
COPY ./DISCOVER .
EXPOSE 8443 15000-15050
EXPOSE 15000/udp
RUN apk --no-cache update \
    && apk --no-cache upgrade \
    && apk --no-cache add ca-certificates \
    && apk --no-cache add dbus \
    && apk --no-cache add tzdata \
    && cp /usr/share/zoneinfo/Europe/Berlin /etc/localtime \
    && echo "Europe/Berlin" >  /etc/timezone \
    && dbus-uuidgen > /var/lib/dbus/machine-id
ENTRYPOINT /app/DISCOVER
```

Running DISCOVER with Linux Container (LXC)

The Linux amd64 package containes everything for running DISCOVER with Linux container (LXC). Here a sample script to setup and install DISCOVER inside a Linux container based on Debian. Finally the LXC is exported to a tar.gz file.

- Used LXC version is 4.0.0 (compatible 2.x+)
- LXC container name is 'DISCOVER'
- DISCOVER is installed as service 'DISCOVER.service'
- DISCOVER service is enabled and started
- Assuming that the executable file "DISCOVER" ist in the current path.

Content of the file 'lxc.sh':

```
#/bin/sh
# lxc delete debian-discover --force
lxc launch images:debian/10 debian-discover
lxc exec debian-discover -- mkdir /opt/DISCOVER
lxc file push DISCOVER debian-discover/opt/DISCOVER/
lxc exec debian-discover -- /opt/DISCOVER/DISCOVER -log.verbose -service install
lxc exec debian-discover -- systemctl enable DISCOVER.service
lxc exec debian-discover -- systemctl start DISCOVER.service
lxc export debian-discover lxc-debian-discover.tar.gz --instance-only
```

The ending line 'lxc list debian-discover' prints out the IP address on which you can connect to the DISCOVER interface.

Runtime parameter

Parameter	Default value	Only CmdLine	Description

Parameter	Default value	Only CmdLine	Description
app.language	en		language for messages
app.product			app product
c			discover client
cfg.create	false	*	Reset configuration file and exit
cfg.file	./discover.json	*	Configuration file
cfg.reset	false	*	Reset configuration file
h	false	*	show flags description and usage
hmd	false	*	show flags description and usage in markdown format
info			discover info
io.file.backups	3		amount of file backups
io.network.timeout	3000		network server and client dial timeout
log.file			filename to log logFile (use "." for /home/ransom/go/src/discover/discover.log)
log.filesize	5242880		max log file size
log.io	false		trace logging
log.json	false		JSON output
log.sys	false		Use OS system logger
log.verbose	false		verbose logging
nb	false		no copyright banner
s			discover server
service			Service operation (simulate,start,stop,restart,install,uninstall)
service.password			Service password
service.timeout	1000		Service timeout
service.username			Service user
t	1000		discover timeout
tls.certificate			Server TLS PKCS12 certificates & privkey container file or buffer
tls.ciphers			TLS ciphers zo use
tls.insecure	false		Use insecure TLS versions and cipher suites
tls.maxversion	TLS1.3		TLS max version
tls.minversion	TLS1.2		TLS min version
tls.mutual			Mutual TLS PKCS12 certificates & privkey container file or buffer
tls.password	changeit		TLS PKCS12 certificates & privkey container file (P12 format)
tls.servername	.*		TLS expected servername
tls.verify	false		Verify TLS certificates and server name
uid			discover uid