
CROWDSTRIKE
Kraken Wrangling for Dummies
Go Modules Dependency Management at Scale

About Me

Started coding in BASIC on a Radio Shack TRS-80 in 1986

B.S. in Computer Science from LSU in 1999

Professional developer for 22 years. 7 companies, 4 different industries

Languages: COBOL, C/C++, Java, Visual Basic, C#, JS/Typescript, Python, and …

Go full-time since 2016

CrowdStrike

At CrowdStrike

We Stop Breaches

Using Go

Before Modules

Initially, all Go code lived at $GOPATH/src

Later, Go 1.5/1.6 gave us vendoring

Pros:
- All Go code was readily available in one place, your Go workspace
- Finding code was simple with tools like `find` and `grep`

Cons:
- There could only be one* version of a given package on a machine
- Builds were not reproducible by default

And now ...

Go 1.11 brought us Go Modules

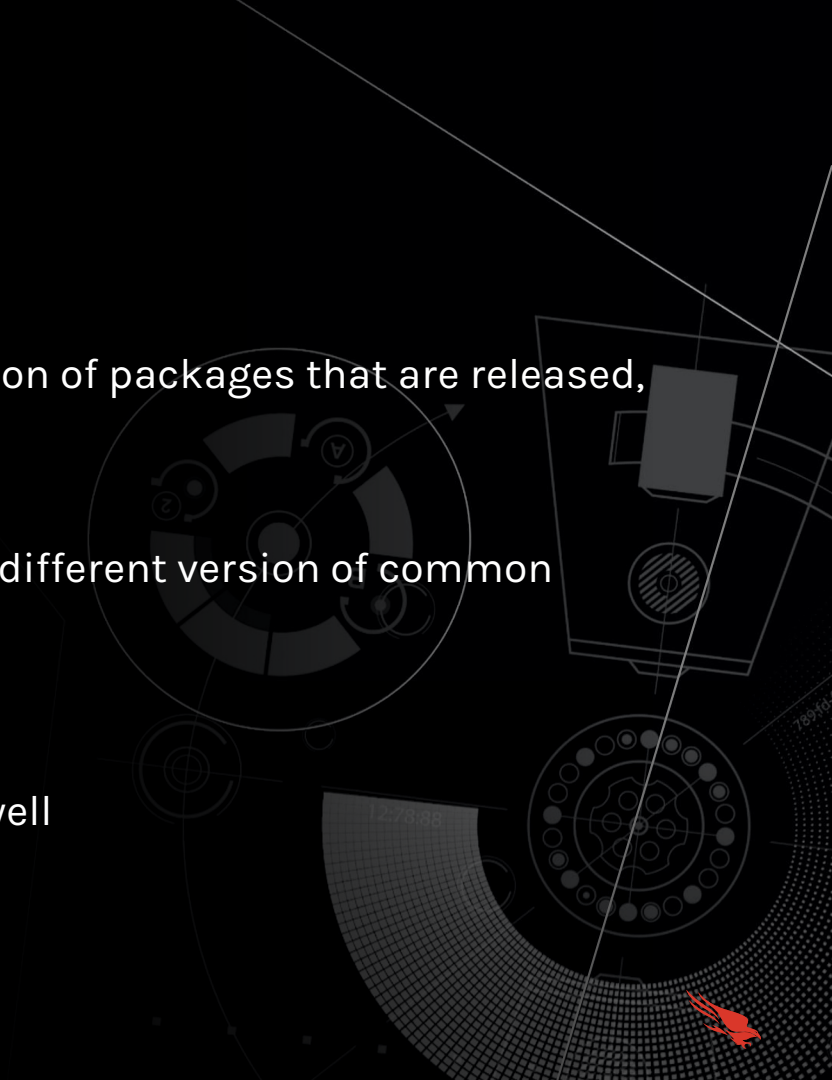
From the modules reference: “A *module* is a collection of packages that are released, versioned, and distributed together.”

Pros:

- Built-in way for different modules to reference different version of common dependencies
- Builds are nearly 100% reproducible

Cons:

- File/Folder based searches no longer work as well



Current Tooling
The go command

The go command provides sub-commands for showing module dependencies

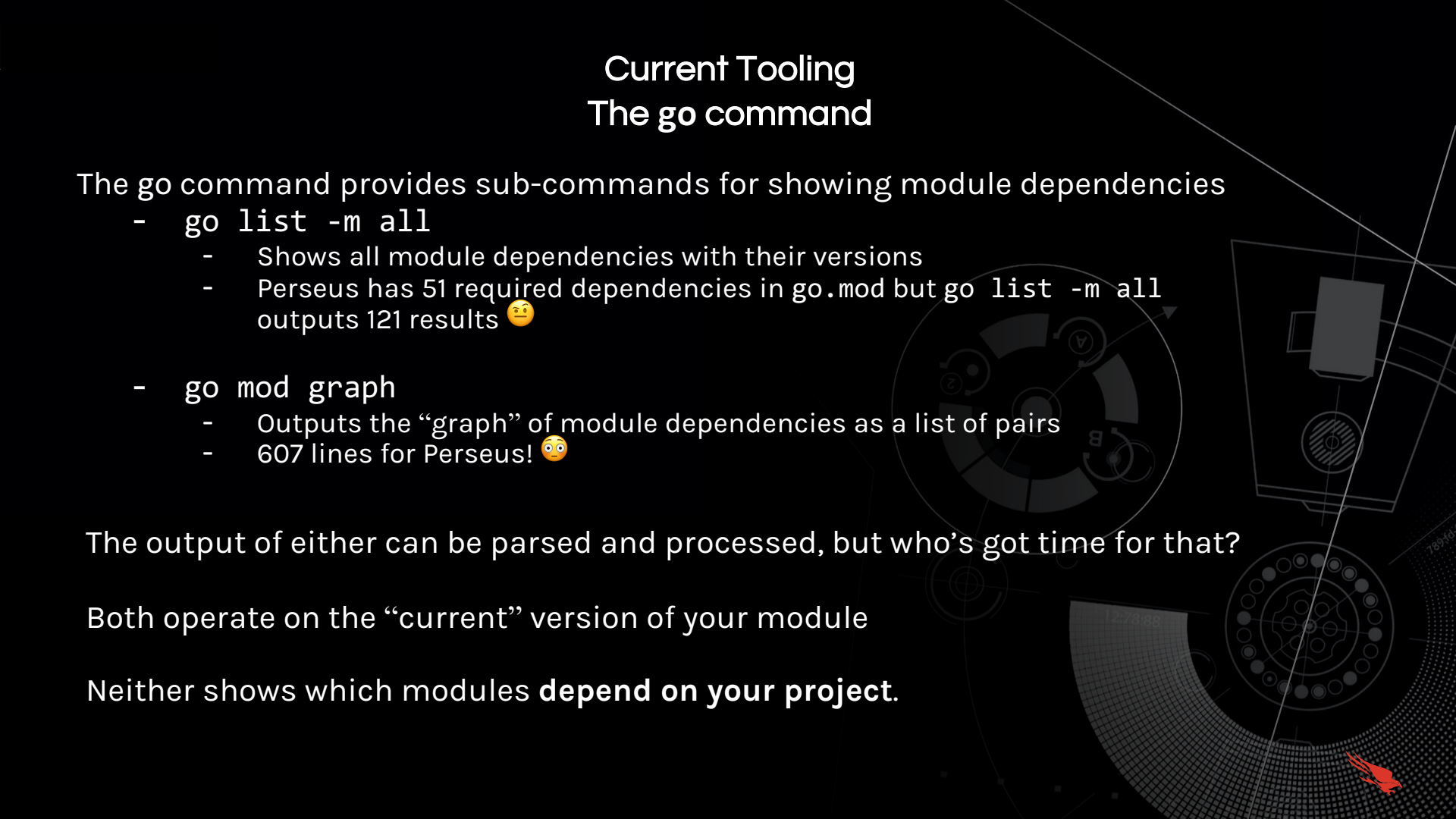
- `go list -m all`
  - Shows all module dependencies with their versions
  - Perseus has 51 required dependencies in `go.mod` but `go list -m all` outputs 121 results 🤨

- `go mod graph`
  - Outputs the “graph” of module dependencies as a list of pairs
  - 607 lines for Perseus! 😳

The output of either can be parsed and processed, but who’s got time for that?

Both operate on the “current” version of your module

Neither shows which modules **depend on your project**.



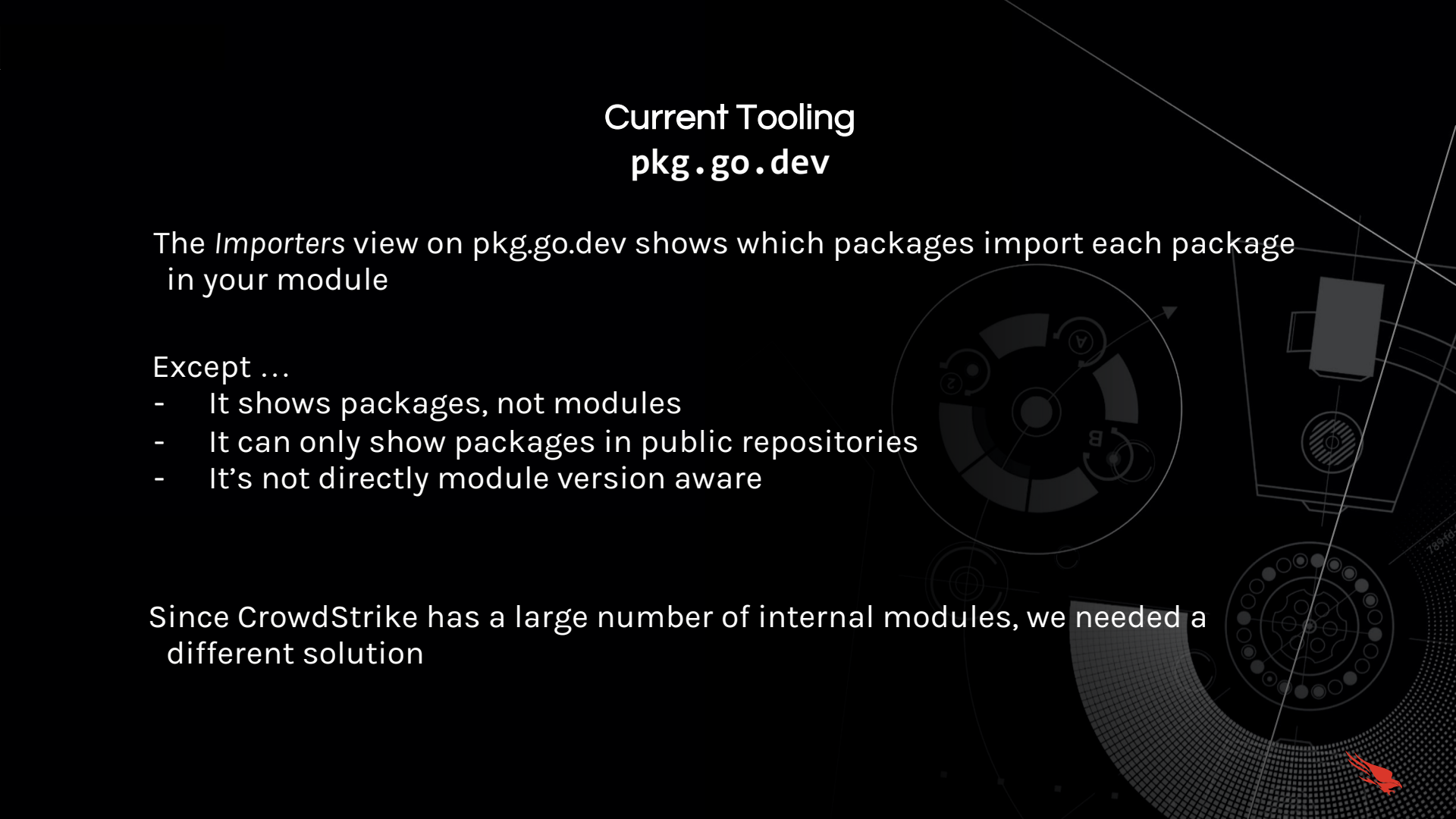
Current Tooling
pkg.go.dev

The *Importers* view on pkg.go.dev shows which packages import each package in your module

Except …

- It shows packages, not modules
- It can only show packages in public repositories
- It's not directly module version aware

Since CrowdStrike has a large number of internal modules, we needed a different solution



Enter: Perseus

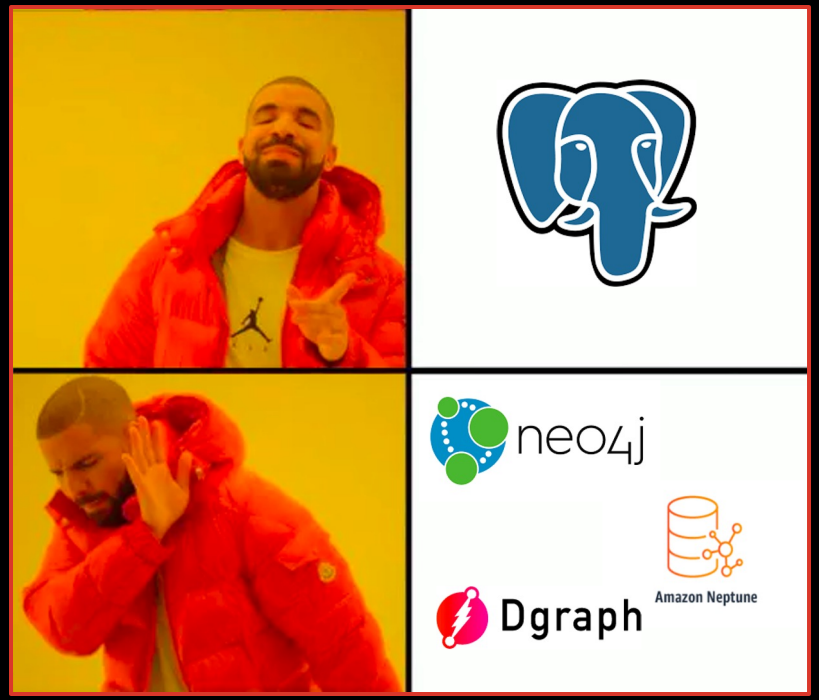
[horns blowing triumphantly]

Stores links between module versions in a graph database

Provides an API to update and query the graph

Also provides a CLI client for the API

The Database



PostgreSQL instead of an actual Graph DB because …

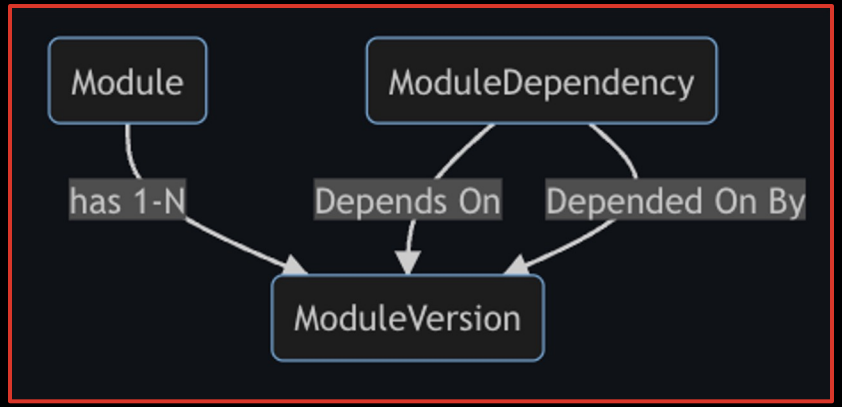
We had existing infra for running PostgreSQL

The data model is very simple, really just 2 vertices and 2 edges

We have several teams that have built on top of "real" graph databases and their opinions were YAGNI

"You don't use a bazooka to swat flies"

Perseus Data Model



The `Module` table is a reference to a module, identified by its path, like `github.com/CrowdStrike/perseus`

`ModuleVersion` is a reference to Module and a semantic version

`ModuleDependency` is many-to-many join table, effectively an edge, between two `ModuleVersion` records

Perseus Data Model (cont.)

Comparing and sorting by semantic version requires special logic

Perseus needs to do that … a lot

We used the `pg-semver` PostgreSQL extension so that things "just work"

Also created a secondary index on module ID + version for query performance

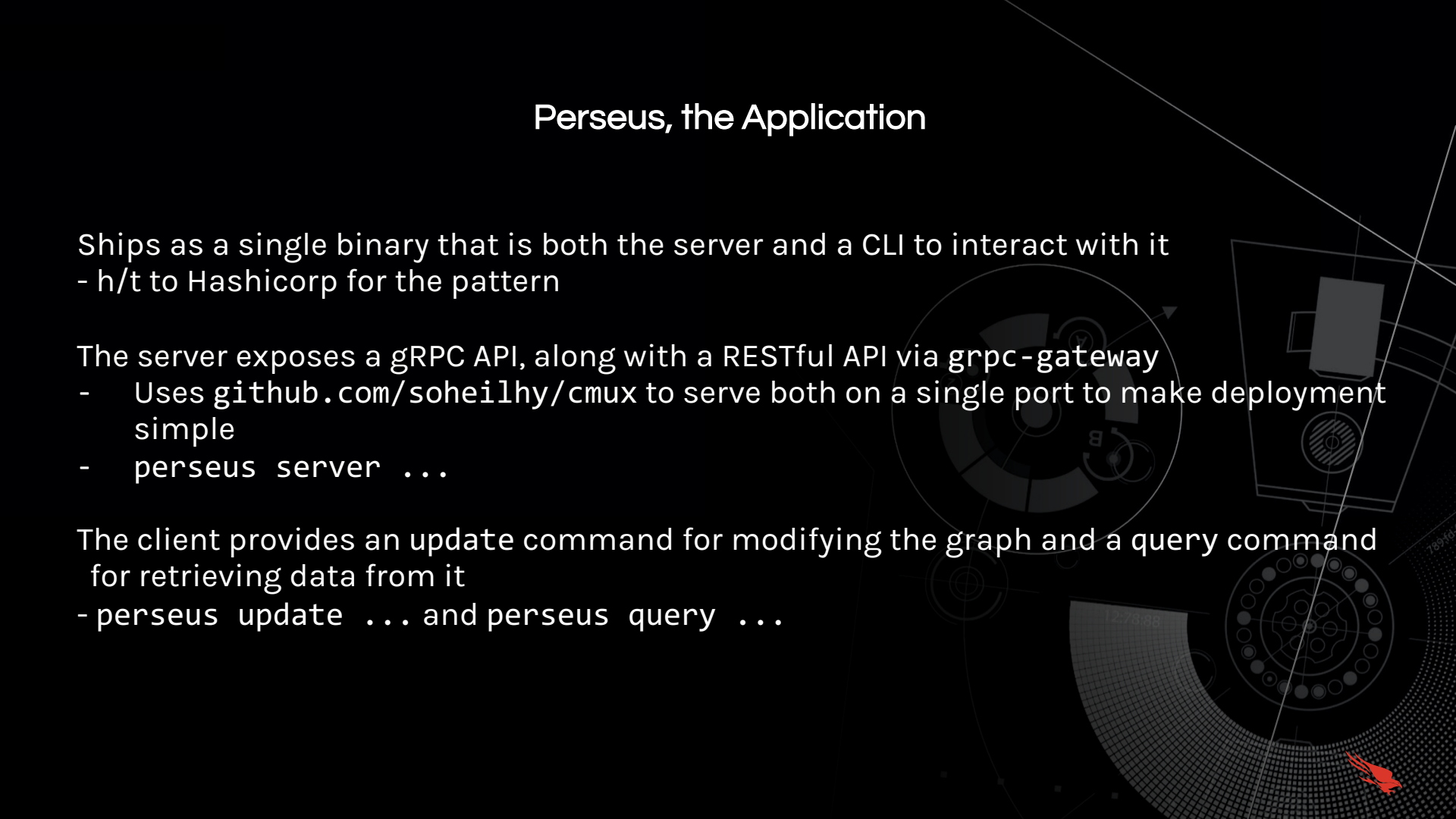
```
CREATE EXTENSION semver;
…
CREATE TABLE module_version (
    id         SERIAL  NOT NULL,
    module_id  INTEGER NOT NULL,
    version    SEMVER  NOT NULL
);
…
CREATE INDEX idx_module_version
ON module_version USING btree (
    module_id ASC  NULLS LAST,
    version   DESC NULLS FIRST
);
```

Perseus, the Application

Ships as a single binary that is both the server and a CLI to interact with it
- h/t to Hashicorp for the pattern

The server exposes a gRPC API, along with a RESTful API via `grpc-gateway`
- Uses `github.com/soheilhy/cmux` to serve both on a single port to make deployment simple
- `perseus server ...`

The client provides an `update` command for modifying the graph and a `query` command for retrieving data from it
- `perseus update ...` and `perseus query ...`



Perseus, the Server
✅ What It Does ✅

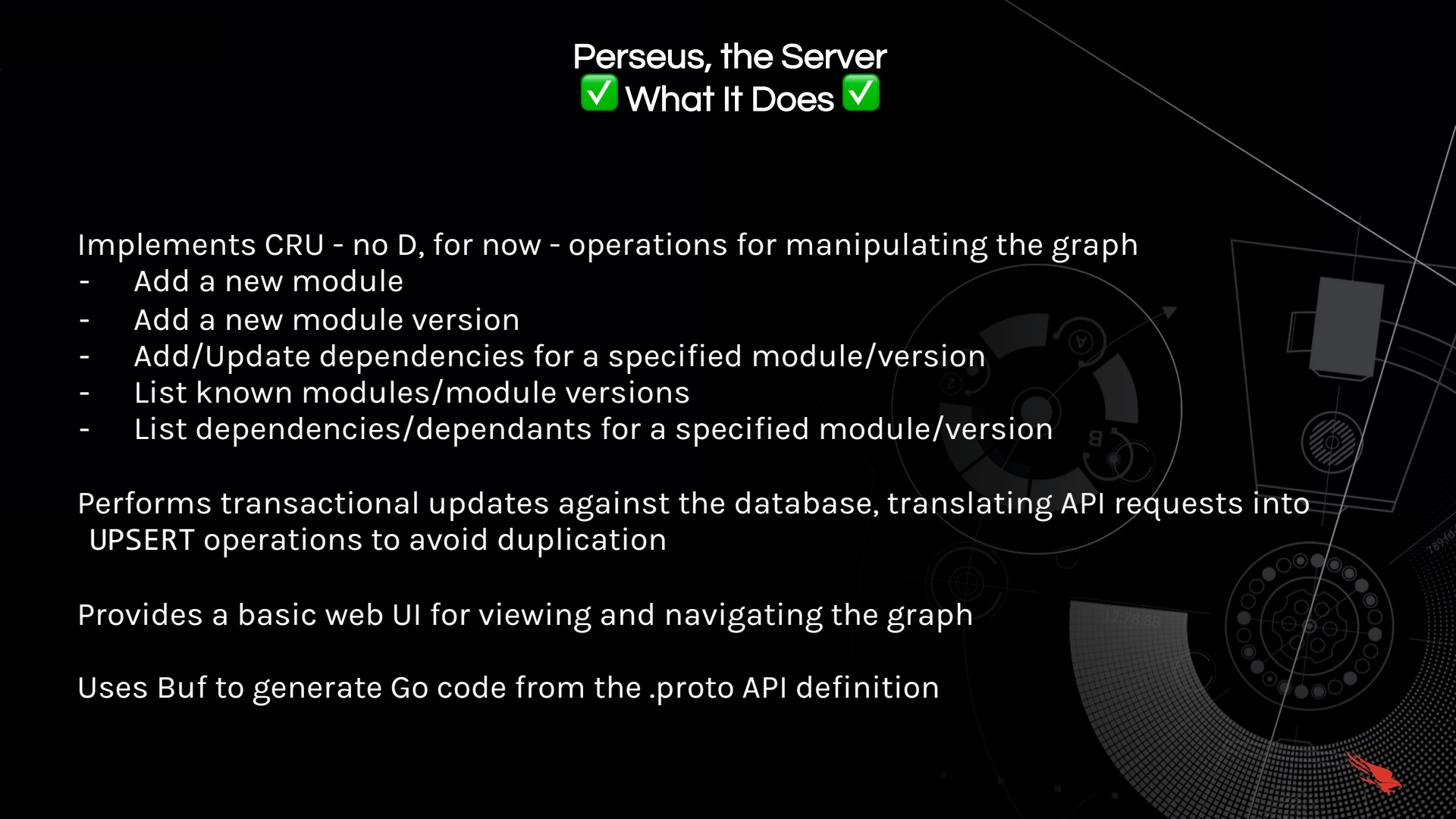
Implements CRU - no D, for now - operations for manipulating the graph

- Add a new module
- Add a new module version
- Add/Update dependencies for a specified module/version
- List known modules/module versions
- List dependencies/dependants for a specified module/version

Performs transactional updates against the database, translating API requests into UPSERT operations to avoid duplication

Provides a basic web UI for viewing and navigating the graph

Uses Buf to generate Go code from the .proto API definition



Perseus, the Server
🚫 What It Doesn't Do 🚫

Walk the graph

Render the graph

Not every consumer will want/need the same behavior and visualizations

K.I.S.S. lets the service avoid code churn

Perseus, the CLI
Updating the Graph

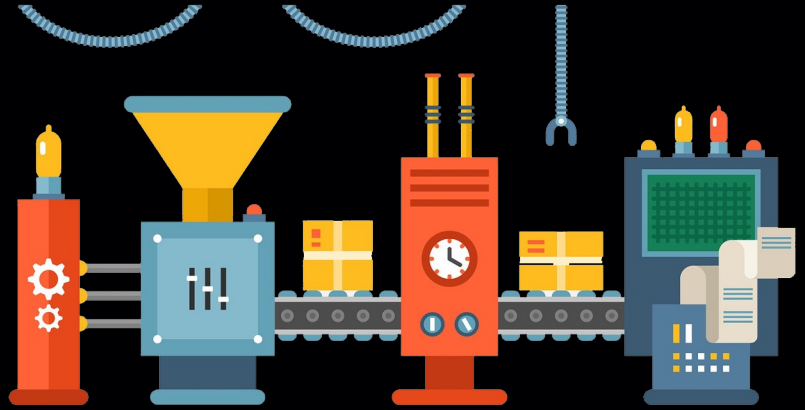
```
# process the module in $PWD, inferring the version
perseus update .

# process the foo module using v0.11.38
perseus update $HOME/go/foo --version v0.11.38
```

Parses `go.mod` for the specified module, extracting the list of direct dependencies and calls the Perseus server to update the graph with the results

Requires that the module repo exists locally and that there is a Go module SemVer tag at the current commit or that the version is specified explicitly

Like before, not processing dependencies recursively keeps things simple

Automating Graph Updates



Write a script

- The Perseus GitHub repo includes a Bash script, `processmod.sh`, that enumerates all of the SemVer tags on a Git repository and executes `perseus update` for each one

Integrate Perseus into CI/CD

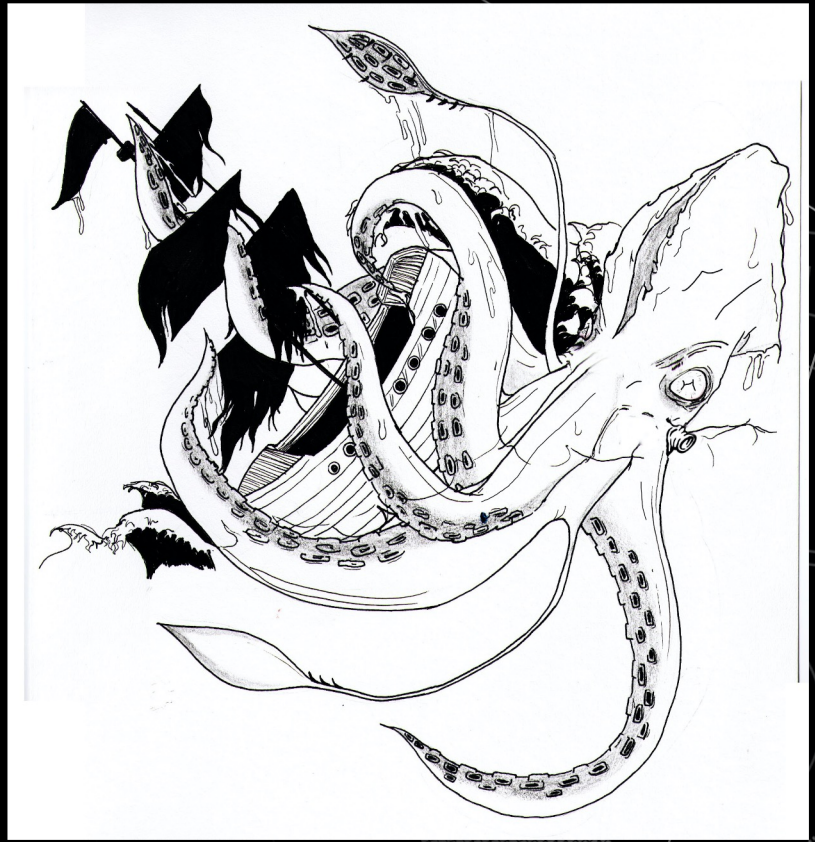
- We run `perseus update` as part of our internal build process when a new stable module version is tagged

How big is this kraken, really?

Short answer: It's probably bigger than you think

For a sampling of 127 internal modules

- 3,944 versions of 347 unique modules
- 95,118 dependency edges



Querying the Graph

```
$ perseus help query
Executes a query against the Perseus graph

Usage:
  perseus query [command]

Aliases:
  query, q

Available Commands:
  ancestors    Outputs the list of modules that the specified module depends on
  descendants  Outputs the list of modules that depend on the specified module
  list-modules Outputs the list of modules, along with the highest version, that match a provided glob pattern

Flags:
      --dot                  specifies that the output should be a DOT directed graph (not supported for list-modules)
  -f, --format string        provides a Go text template to format the output.
                             Each result is an instance of the following struct:
                              type Item struct {
                                        // the module name and version, ex: github.com/CrowdStrike/perseus and v0.11.38
                                        Path, Version string
                                        // true if this module is a direct dependency of the "root" module, false if not
                                        IsDirect bool
                                        // the number of dependency links between this module and the "root" module
                                        // - direct dependencies have a degree of 1, dependencies of direct dependencies
                                        //   have a degree of 2, etc.
                                        Degree int
                              }
                             The Name() method also returns a string containing "[Path]@[Version]".
  -h, --help                 help for query
      --json                 specifies that the output should be formatted as JSON
      --list                 specifies that the output should be formatted as a tabular list
      --max-depth int        specifies the maximum number of levels to be returned (default 4)
      --server-addr string   the TCP host and port of the Perseus server

Global Flags:
  -x, --debug   enable verbose logging
```

Querying the Graph
Ancestors/Dependencies

```
# list the path and version of the direct dependencies of Perseus
$ perseus query ancestors github.com/CrowdStrike/perseus --format '{{.Name}}' --max-depth 1

github.com/Masterminds/squirrel@v1.5.3
github.com/go-git/go-git/v5@v5.4.2
github.com/grpc-ecosystem/grpc-gateway/v2@v2.11.3
github.com/jackc/pgx/v4@v4.17.2
github.com/jmoiron/sqlx@v1.3.5
github.com/mattn/go-isatty@v0.0.16
github.com/soheilhy/cmux@v0.1.5
github.com/spf13/cobra@v1.5.0
github.com/spf13/pflag@v1.0.5
github.com/theckman/yacspin@v0.13.12
golang.org/x/mod@v0.1.1-0.20191105210325-c90efee705ee
golang.org/x/sync@v0.0.0-20220601150217-0de741cfad7f
google.golang.org/genproto@v0.0.0-20220822174746-9e6da59bd2fc
google.golang.org/grpc@v1.49.0
google.golang.org/grpc/cmd/protoc-gen-go-grpc@v1.2.0
google.golang.org/protobuf@v1.28.1
```

Querying the Graph
Ancestors/Dependencies

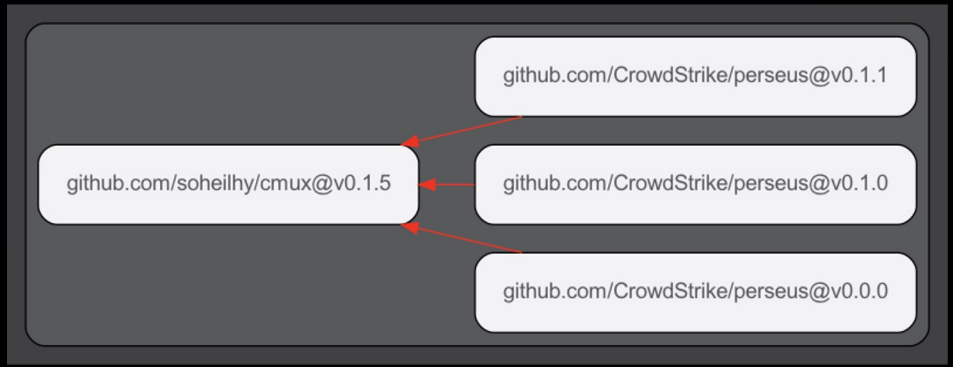
```
# show a table of the direct and indirect dependencies of Perseus
# within 4 degrees of separation

$ perseus query ancestors github.com/CrowdStrike/perseus --list

Dependency                                                  Direct
github.com/Masterminds/squirrel@v1.5.3                      true
github.com/go-git/go-git/v5@v5.4.2                          true
github.com/grpc-ecosystem/grpc-gateway/v2@v2.11.3           true
github.com/jackc/pgx/v4@v4.17.2                             true
github.com/jmoiron/sqlx@v1.3.5                              true
github.com/mattn/go-isatty@v0.0.16                          true
github.com/soheilhy/cmux@v0.1.5                             true
github.com/spf13/cobra@v1.5.0                               true
github.com/spf13/pflag@v1.0.5                               true
github.com/theckman/yacspin@v0.13.12                        true
golang.org/x/mod@v0.1.1-0.20191105210325-c90efee705ee       true
golang.org/x/sync@v0.0.0-20220601150217-0de741cfad7f        true
golang.org/x/sys@v0.0.0-20220811171246-fbc7d0a398ab         false
google.golang.org/genproto@v0.0.0-20220822174746-9e6da59bd2fc  true
google.golang.org/grpc@v1.49.0                              true
google.golang.org/grpc/cmd/protoc-gen-go-grpc@v1.2.0        true
google.golang.org/protobuf@v1.28.1                          true
```

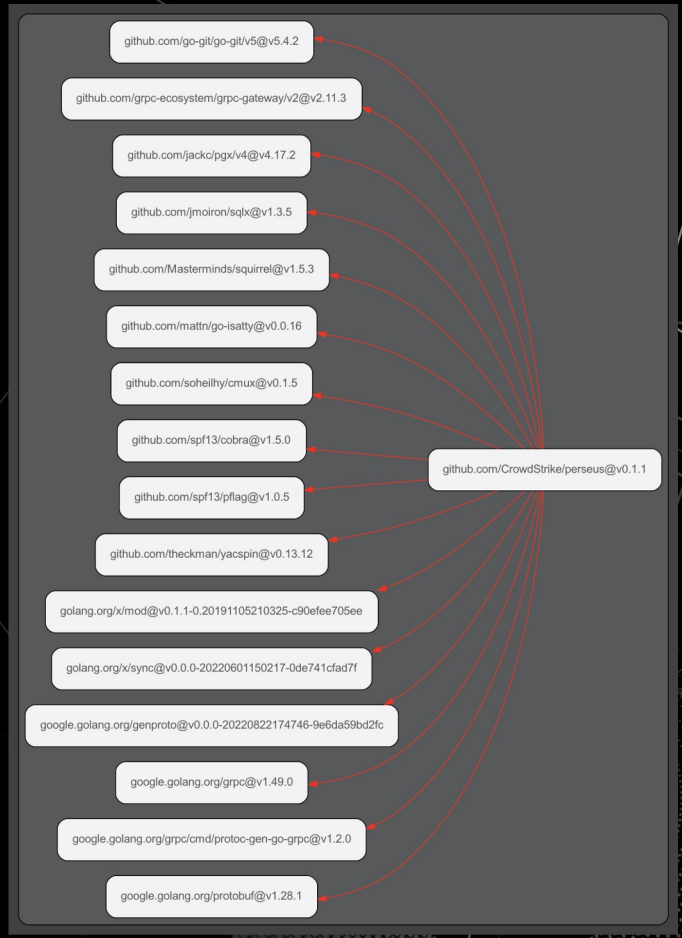
Querying the Graph
Descendants/Dependants

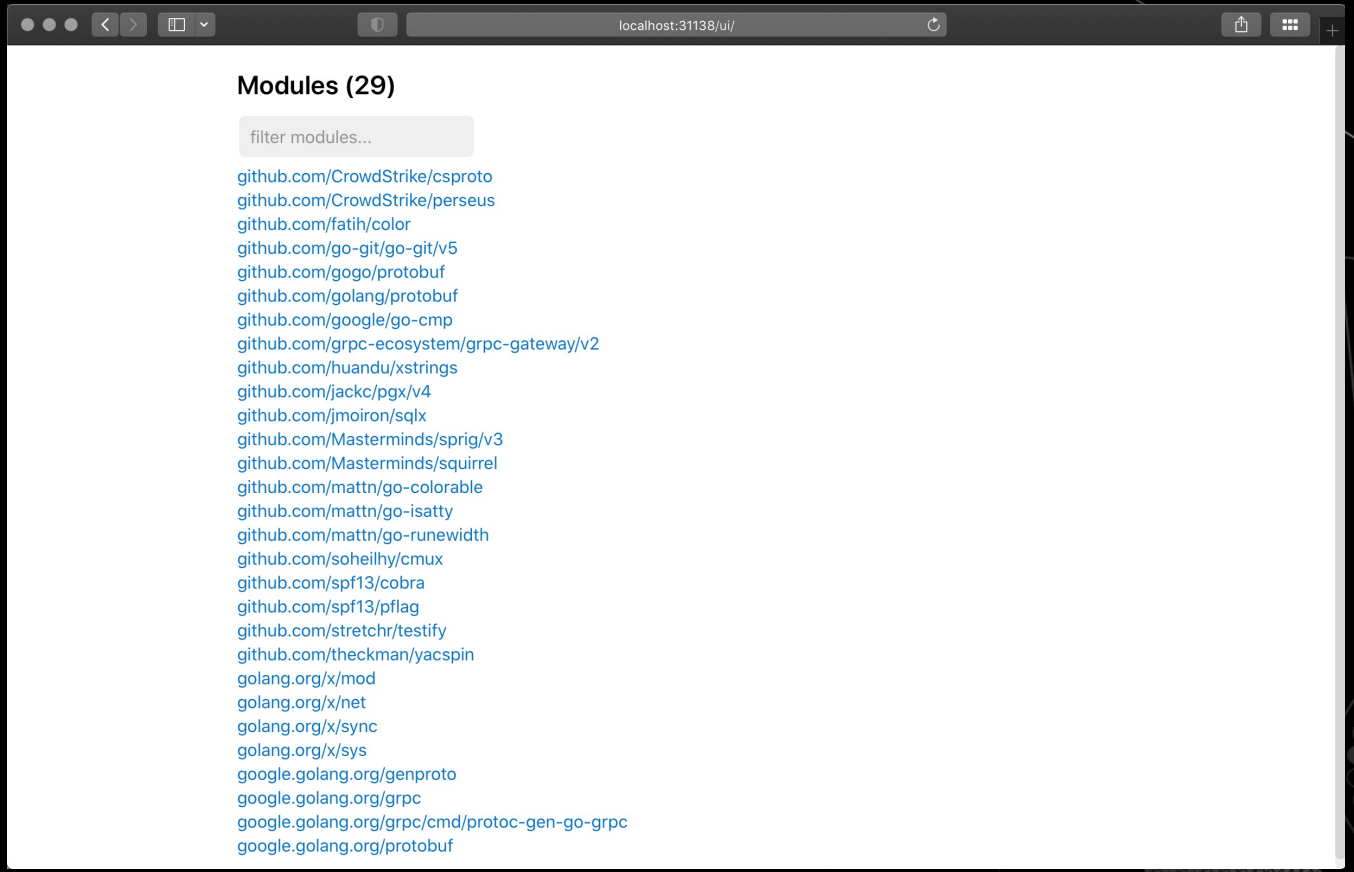
```
# list the path and version of all modules that directly depend on cmux@v0.1.5
$ perseus query descendants github.com/soheilhy/cmux@v0.1.5 --max-depth 1 --format '{{.Name}}'
github.com/CrowdStrike/perseus@v0.1.1
github.com/CrowdStrike/perseus@v0.1.0
github.com/CrowdStrike/perseus@v0.0.0


# same thing, but output as a table
$ perseus q d github.com/soheilhy/cmux@v0.1.5 --max-depth 1 --list
Dependent                               Direct
github.com/CrowdStrike/perseus@v0.1.1   true
github.com/CrowdStrike/perseus@v0.1.0   true
github.com/CrowdStrike/perseus@v0.0.0   true


# again, but de-duplicate by module path
$ perseus q d github.com/soheilhy/cmux@v0.1.5 --format '{{.Path}}' | uniq
github.com/CrowdStrike/perseus
```

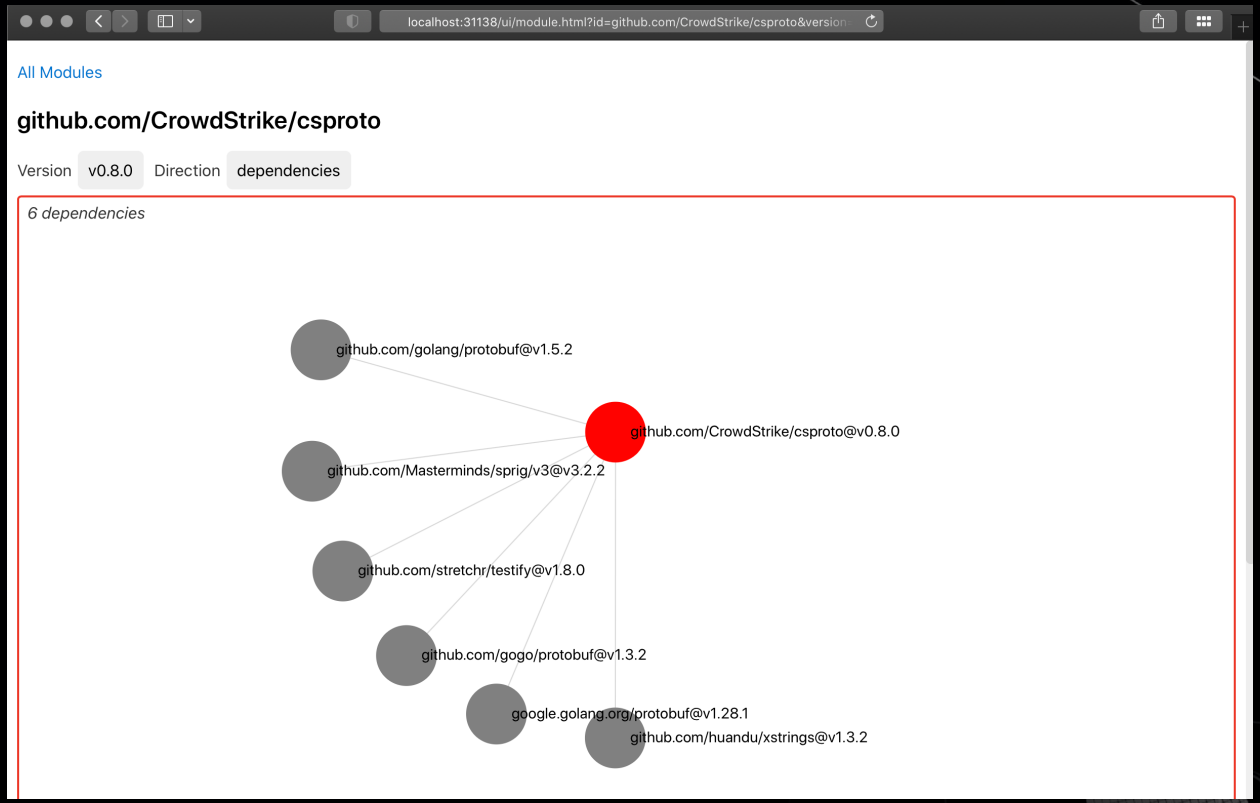
Rendering the Graph

```
# Generates the image to the right
$ perseus query ancestors github.com/CrowdStrike/perseus --dot | dot -Tpng -o perseus.png


# Generates the image below
$ perseus q d github.com/soheilhy/cmux --dot | dot -Tpng -o cmux.png
```






localhost:31138/ui/
Modules (29)
filter modules...
github.com/CrowdStrike/csproto
github.com/CrowdStrike/perseus
github.com/fatih/color
github.com/go-git/go-git/v5
github.com/gogo/protobuf
github.com/golang/protobuf
github.com/google/go-cmp
github.com/grpc-ecosystem/grpc-gateway/v2
github.com/huandu/xstrings
github.com/jackc/pgx/v4
github.com/jmoiron/sqlx
github.com/Masterminds/sprig/v3
github.com/Masterminds/squirrel
github.com/mattn/go-colorable
github.com/mattn/go-isatty
github.com/mattn/go-runewidth
github.com/soheilhy/cmux
github.com/spf13/cobra
github.com/spf13/pflag
github.com/stretchr/testify
github.com/theckman/yacspin
golang.org/x/mod
golang.org/x/net
golang.org/x/sync
golang.org/x/sys
google.golang.org/genproto
google.golang.org/grpc
google.golang.org/grpc/cmd/protoc-gen-go-grpc
google.golang.org/protobuf


localhost:31138/ui/module.html?id=github.com/CrowdStrike/csproto&version
All Modules
github.com/CrowdStrike/csproto
Version
v0.8.0
Direction
dependencies
6 dependencies
github.com/golang/protobuf@v1.5.2
github.com/CrowdStrike/csproto@v0.8.0
github.com/Masterminds/sprig/v3@v3.2.2
github.com/stretchr/testify@v1.8.0
github.com/gogo/protobuf@v1.3.2
google.golang.org/protobuf@v1.28.1
github.com/huandu/xstrings@v1.3.2

More Example Queries

```
# list all modules that depend on v1.12.1 of github.com/example/foo (within 6 degrees)
# and return the tree as JSON
$ perseus q d github.com/example/foo@v1.12.1 --max-depth 6


# emulate 'go list -m all' with a max of 10 hops
$ perseus q a github.com/example/bar --max-depth 10 --format '{{.Path}} {{.Version}}'


# 'go list -m all', but for things that *depend on* bar
$ perseus q d github.com/example/bar --max-depth 10 --format '{{.Path}} {{.Version}}' | uniq
```

🛠️ Advanced Queries 🛠️

```
# Given a list of modules, report any that depend on themselves
reportInceptionModules() {
    # $1 - space-delimited list of module/version strings
    # $2 - optional max recursion depth, default 5
    for m in $1; do
        name=$(echo $m | sed 's/@.+//')
        depth=${2:-5}
        res=$(perseus q a $m --format '{{.Path}}' --max-depth $depth | grep "$mname" | uniq)
        if [ -n "$res" ]; then
            echo $mname has a cyclic dependency on itself
        fi
    done
}

$ reportInceptionModules $(perseus q list-modules 'github.com/example/*' --format '{{.Path}}')
```

🛠️ Advanced Queries 🛠️ (cont)

```
# Report all modules that match a pattern where the highest version depends on a specified dependency
reportModulesVulnerableVia() {
	# $1 - glob pattern for modules to check
	# $2 - vulnerable dependency, path@version
	# $3 - optional max recursion depth, default 5
	for m in $(perseus q lm $1 --format '{{.Name}}'); do
		depth=${3:-5}
		res=$(perseus q a $m --format '{{.Path}}' --max-depth $depth | grep "$2" | uniq)
		if [ -n "$res" ]; then
			echo $m depends on $2 and needs remediation
		fi
	done
}
$ reportModulesVulnerableVia 'github.com/example*' github.com/shady-code/oh-noes@v1.42.13
```

Future Improvements

- Nicer UI
- GraphQL API
- New `perseus query ...` subcommands for common operations
- Ability to read module versions and `go.mod` from module proxies or VCS for public repos
- Publish pre-built Docker images for running the server

The project is on GitHub at `github.com/CrowdStrike/perseus`

PRs are certainly welcome

Credits

Perseus and Andromeda - Giuseppe Cesari - [1], Public Domain, https://commons.wikimedia.org/w/index.php?curid=31673524

Caveman image by Vecteezy - https://www.vecteezy.com/free-vector/history

OSS Projects
- pg-semver Extension - https://github.com/theory/pg-semver
- grpc-gateway - https://github.com/grpc-ecosystem/grpc-gateway
- cmux - https://github.com/soheilhy/cmux

Buf - https://buf.build