go-ing for an evening stroll

golang beasts & where to find them

\$ who

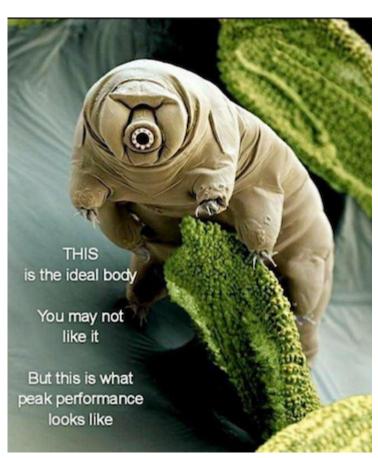
- Stefan Edwards, Practice Lead ComServ
 - GitHub/Twitter/Lobste.rs: lojikil
- Bobby Tonic, Senior AppSec Engineer
 - Twitter: b0bbytabl3s
 - GitHub: btonic
- Trail of Bits

\$ what presentation.md

- Common Go vulnerabilities we see
 - Integers/Numerics
 - Standard Library Issues
 - Error checking & type assertions
 - Defer semantics
- How to find them
- How to fix them

\$ what takeaway.md

- Go has some edge cases
- Luckily, most are easily uncovered
- Tooling can help... if you know how to use them



Go's integer semantics are interesting...

```
v, err := strconv.Atoi("4294967377")
g := int32(v)
fmt.Printf("v: %v, g: %v\n", v, g);
```

- on 32bit: v is an int, 32bits wide, and truncated before conversion in g
- on 64bit: v is an int , 64bits wide, and only g is truncated:
 v: 4294967377, g: 81

- that may seem obvious, but we see it often
 - CNCF projects, blockchains, &c
- Finding it is mostly manual:
 - Any uses of strconv.Atoi is likely wrong
 - Flows with ParseInt/ParseUint and down casts
 - Nico has done some great hunts with CodeQL

an aside

- always check for things that return 'naked' int
- ensure what build platform you're using (amd64, &c)
- check dataflows from strconv.ParseInt and assume strconv.Atoi is wrong
- Tooling? CodeQL, two fists

Standard Library

- Standard libraries: the building blocks of most software.
- In theory, libraries should avoid exposing abusable behavior.
- This doesn't always happen, even in standard libraries.

Standard Library

An example:

```
os.MkdirAll("/some/path/i/want/to/make", 0600)
ioutil.WriteFile("/some/file/i/want/to/make/here.txt",
":)", 0600)
```

- os.MkdirAll will populate the directories specified if they do not already exist, with the specified permissions.
- ioutil.WriteFile will create a file at a specified path (if it doesn't already exist) with the specified permissions, and write to it.

Standard Library

- enhance: ...populate the directories specified if they do not already exist...
 - What happens if some directories in the path exist, but with different permissions?
- enhance: ...create a file at a specified path (if it doesn't already exist)...
 - What happens if the file exists, but with different permissions?
- Both cases: No error or warning. Existing directories/files will be used and retain original permissions.
- Attacker: Pre-population attacks. Create a directory/file path with open permissions before the program does, then let the program write sensitive information there!

Errors & Assertions

- Go errors are great...
 - assuming you pay attention
- Type assertions fit in here too
- Four major issues:

```
v, err := SomeFunc(...)
g, err := SomeFunc(...)
h := someval.(int)
err == nil || err != nil
```

Errors & Assertions

- We see a lot of type assertion panics & missed error checks
- Luckily easier to catch
 - use errcheck –asserts for both error checks & type assertions
 - ∘ use ineffassign
 - maybe use errcheck -assert -blank
- Fuzz... everything

Errors & Assertions

- Check your error results rigorously
- Understand your flows as much as possible
 - Compiler won't save you
 - Can continue on... but is that ok?
- Avoid type assertions, and always check ok
 - o if h, ok := someval.(int); ok { ...
 - o always check ok is true

Defer Semantics

- defer is central to error handling, and often used for finalizing operations. You might:
 - defer a function that handles a panic through recover,
 which might be raised during a function's execution.
 - o defer a resource.Close() or resource.Finish() to finalize a resource.

- Officialities, using defer carrie difficultive.
- When defer ing a function, the values returned by the defer ed function are ignored, including error values.
 - When opening a file for writing and using defer
 file Close(), errors could be produced but execution will continue.
- When defer ing a function which handles a panic, the defer FILO stack order matters.
 - A panic in a defer ed function can be recover ed by deferred function earlier in the defer stack.
 - An un- recover ed panic in a defer ed function will bubble up to the function's caller.
- tangentially related: In what cases would the following fail to process a panic?

16

Defer Semantics

- Avoid defer ing a function that returns an error value.
- Enumerate the values passed to panic.
 - panic will pass nil to recover! (Hint: on the last toy example)
- Ensure you are defer ing functions that recover from a panic in the proper order.
- Review the specification to understand scoping rules for
 defer ed inline functions, pass-by-copy vs pass-by-reference,
 and when a defer executes in the context of a function's body
 and the return of a function.

Thanks!

https://github.com/lojikil/kyoto-go-nihilism