

# Detect Trojan Source Attack

Creation of `bidichk` and  
integration in `golangci-lint`

Bärner Go – 07.12.2021

# About myself – Lucas Bremgartner

I work in IT since more than 20 years in several different roles.

Currently I work as contractor for different clients, usually in roles like architect, coach or software engineer. My fields of expertise are cloud native applications (backend), cloud infrastructure and automation as well as the Elastic Stack. My programming language of choice is Go. I am an active part of the Go community since ~7 years.

Maintainer of:

[rootcerts](#), [logstash-config](#), [errchkjson](#), [mna/pigeon](#), [magnusbaeck/logstash-filter-verifier](#)

You find me online:

[github.com/breml](#) | [@\\_breml](#) | [linkedin.com/in/lucas-bremgartner/](#)

# Pop Quiz: What does this program print? We will find out.

```
package main

import "fmt"

func main() {

    var accessLevel = "user"

    if accessLevel != "user" { // Check if admin

        fmt.Println("You are an admin.")

    } else {

        fmt.Println("You are a user.")

    }

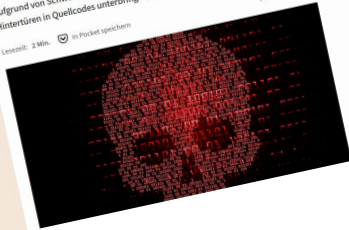
}
```

Angreifer könnten Source Code trojanisieren der trotzdem legitim aussieht

Programmierersprachen lassen sich per Unicode trojanisieren

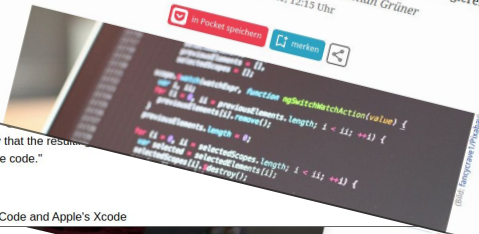
## Angreifer könnten Source Code trojanisieren, der trotzdem legitim aussieht

Aufgrund von Schwachstellen im Unicode-Standard könnten Angreifer etwa Hintertüren in Quellcodes unterbringen, die Sicherheitsforschern nicht auffallen.



## Programmierersprachen lassen sich per Unicode trojanisieren

Ein Forschungsteam zeigt systematisch, wie sich mit Unicode-Tricks Code manipulieren lässt. [Open-Source-Communitys](#) und die IT-Industrie reagieren.  
Ein Bericht von Moritz Tremmel und Sebastian Grüner  
2. November 2021, 12:13 Uhr



"Embedding multiple layers of LRI and near-arbitrary reordering of strings," says their paper, "we can reorder source code characters in such a way that the resulting display order also represents syntactically valid source code."

"In effect, we anagram program A into program B."

Concerningly, the academics say that Microsoft's VS Code and Apple's Xcode

Trojan Source

Dein Job in der IT!

Jetzt bewerben!  
[www.bwi.de/professionals](https://www.bwi.de/professionals)

# Still sure about your previous answer?

```
package main

import "fmt"

func main() {

    var accessLevel = "user"

    if accessLevel != "user" { // Check if admin

        fmt.Println("You are an admin.")

    } else {

        fmt.Println("You are a user.")

    }

}
```

Let's find out:

[https://play.golang.org/p/LuncON17\\_2g](https://play.golang.org/p/LuncON17_2g)

# What the compiler sees

```
package main

import "fmt"

func main() {

    var accessLevel = "user"

    if accessLevel != "user[U+202E][U+2066]// Check if admin[U+2069][U+2066]" {

        fmt.Println("You are an admin.")

    } else {

        fmt.Println("You are a user.")

    }

}
```

# Bidirectional text / Unicode Bidi Algorithm

- Bidirectional text contains two text directionalities, right-to-left and left-to-right
- The Bidi algorithm translates the logical order (in memory, always from left-to-right) into the visual order.
- Each Unicode character has a type: strong, weak, neutral and explicit formatting.
- Explicit formatting characters are special Unicode sequences, that direct the Bidi algorithm to modify its default behavior. These are subdivided into “marks”, “embeddings”, “isolates”, and “overrides”.
  - In the previous example:  
[U+202E] = RIGHT-TO-LEFT-OVERRIDE  
[U+2066] = LEFT-TO-RIGHT-ISOLATE  
[U+2069] = POP-DIRECTIONAL-ISOLATE

See also: [https://en.wikipedia.org/wiki/Bidirectional\\_text](https://en.wikipedia.org/wiki/Bidirectional_text)

# “Trojan Source” Vulnerability – Distilled

**Researchers:** Nicholas Boucher and Ross Anderson, **Official Website:** <https://www.trojansource.codes/>

[CVE-2021-42574](#) | [nist.gov](https://nvd.nist.gov): CVSS v3.x: **9.8 (Critical)** | CVSS v2.x: **7.5 (High)**

**Affected Language:** C, C++, C#, JavaScript, Java, Rust, Go, and Python (+ suspect, that it will work in most modern languages)

**Timeline** (from Rust Security Team):

- 2021-07-25: we received the report and started working on a fix.
- 2021-09-14: the date for the embargo lift (2021-11-01) is communicated to us.
- 2021-10-17: performed an analysis of all the source code ever published to crates.io to check for the presence of this attack.
- 2021-11-01: embargo lifts, the vulnerability is disclosed and Rust 1.56.1 is released.

**Is it new?** Nah, Go issue [#20209](#) from 05.05.2017 already mentions this kind of attack.

**But:** Give a vulnerability a good name and you can create quite some fuss in the news.

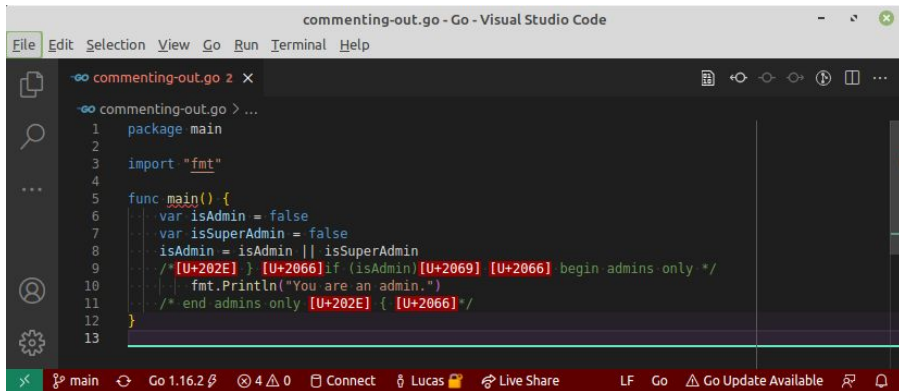


# The attack vector

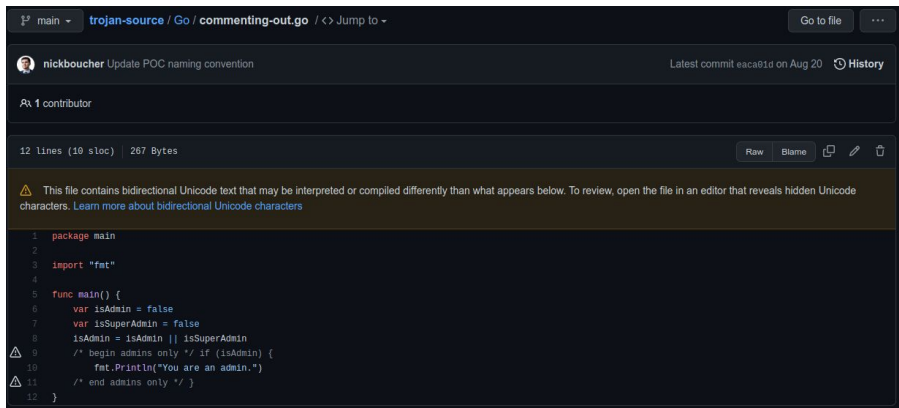
- Use Unicode control characters to reorder tokens in source code.
- These visually reordered tokens can be used to display logic that, while semantically correct, diverges from the logic presented by the logical ordering of source code tokens.
- Compilers and interpreters adhere to the logical ordering of source code, not the visual order.
- The attack is to use control characters embedded in comments and strings to reorder source code characters in a way that changes its logic.
- Examples of interesting Unicode characters:  
RIGHT-TO-LEFT-OVERRIDE (U+202E), LEFT-TO-RIGHT-ISOLATE (U+2066), POP-DIRECTIONAL-ISOLATE (U+2069)

# Attack Mitigation

- Fix the compiler:  
[Russ Cox: the compiler is the wrong place](#)
- Make visible in editors
- Make visible in review tools  
(PR for The Go Playground, anyone)
- **Have a linter**  
(challenge accepted)



```
commenting-out.go 2 X
commenting-out.go > ...
1 package main
2
3 import "fmt"
4
5 func main() {
6     var isAdmin = false
7     var isSuperAdmin = false
8     isAdmin = isAdmin || isSuperAdmin
9     /* [U+202E] [U+2066] if (isAdmin) [U+2069] [U+2066] begin admins only */
10     fmt.Println("You are an admin.")
11     /* end admins only [U+202E] { [U+2066] */
12 }
13
```



```
main - trojan-source / Go / commenting-out.go / <> Jump to -
nickboucher Update POC naming convention Latest commit eacab1d on Aug 20 History
PR 1 contributor
12 lines (10 sloc) 207 Bytes Raw Blame
⚠ This file contains bidirectional Unicode text that may be interpreted or compiled differently than what appears below. To review, open the file in an editor that reveals hidden Unicode characters. Learn more about bidirectional Unicode characters
1 package main
2
3 import "fmt"
4
5 func main() {
6     var isAdmin = false
7     var isSuperAdmin = false
8     isAdmin = isAdmin || isSuperAdmin
9     /* begin admins only */ if (isAdmin) {
10     fmt.Println("You are an admin.")
11     /* end admins only */ }
12 }
```

# How to write a linter for Go

- Use [go/analysis](#), a common interface for all linters (other linters are not accepted by golangci-lint)
- Define an analysis, which is a variable of type `*analysis.Analyzer`
- Implement the linter logic in the Analyzer's `Run` function  
`func(pass *Pass) (interface{}, error)`
- `Pass` provides information to the Analyzer's `Run` function about the package being analyzed.
- Report findings in the `Run` function with  
`pass.Reportf(pos, "message").`

Tutorial linked on [golangci-lint “new linters”](#): [Writing Useful go/analysis Linter](#)

# bidichk – Run function

```
var Analyzer = &analysis.Analyzer{
    Name: "bidichk",
    Doc:  "Checks for dangerous unicode character sequences",
    Run:  run,
}

func (b bidichk) run(pass *analysis.Pass) (interface{}, error) {
    var err error
    pass.Fset.Iterate(func(f *token.File) bool {
        if strings.HasPrefix(f.Name(), "$GOROOT") {
            return true
        }
        return b.check(f.Name(), f.Pos(0), pass) == nil
    })
    return nil, err
}
```

# bidichk – inspect a file

```
func (b bidichk) check(filename string, pos token.Pos, pass *analysis.Pass) error {
    body, err := os.ReadFile(filename)
    if err != nil { return err }

    for name, r := range b.disallowedRunes {
        start := 0
        for {
            idx := bytes.IndexRune(body[start:], r)
            if idx == -1 { break }
            start += idx
            pass.Reportf(pos+token.Pos(start),
                        "found dangerous unicode character sequence %s", name)
            start += utf8.RuneLen(r)
        }
    }
    return nil
}
```

## go/analysis based linter – Summary

- The initial version of the linter was ~60 LOC + ~10 LOC in main.go (it has grown a little bit since then).
- go/analysis does the heavy lifting and provides convenience functions for the main program of the linter as well as the unit tests.
- A proof-of-concept for a simple linter can be done in less than an hour.

# Add bidichk to golangci-lint

The necessary steps to add a new linter to golangci-lint are well documented in <https://golangci-lint.run/contributing/new-linters/>

The main steps are:

- Add some test data / test cases
- Add the linter integration
- Add the linter integration to the linter manager

Adding 3 files (+ updating go.{mod,sum}) is enough for a basic integration.

Initial PR for bidichk (~ 30 LOC including a test):

<https://github.com/golangci/golangci-lint/pull/2330>

# Questions





# Links

bidichk: <https://github.com/breml/bidichk>

PR for golangci-lint: <https://github.com/golangci/golangci-lint/pull/2330>

golangci-lint: <https://golangci-lint.run/> | <https://github.com/golangci/golangci-lint>

Trojan Source: <https://www.trojansource.codes/> | [CVE-2021-42574](https://nvd.nist.gov/vuln/detail/CVE-2021-42574) | [nist.gov](https://nvd.nist.gov)

Trojan Source & Go: [#20209](https://twitter.com/20209), [Russ Cox: the compiler is the wrong place](https://www.russcox.com/blog/2021/01/20/russ-cox-the-compiler-is-the-wrong-place)

Researchers: <https://github.com/nickboucher>, <https://www.cl.cam.ac.uk/~rja14>

Bidirectional text: [https://en.wikipedia.org/wiki/Bidirectional\\_text](https://en.wikipedia.org/wiki/Bidirectional_text)