

Is Your YAML Hiding Something?

Unicode · Invisible Characters · Silent Failures

Harsha Dixit · github.com/harshadixit12 · harshadixit.com
Engineering @ Kong

YAML is everywhere

- ▶ Kubernetes manifests & Helm charts
- ▶ GitHub Actions workflows
- ▶ CI/CD pipelines (Argo, Tekton, Drone)
- ▶ OpenAPI / Swagger specs
- ▶ Kong Gateway Configuration

If YAML breaks, infrastructure breaks.

The discovery

A customer reached out with a problem - their configuration was not being loaded correctly - no error, just missing configuration.

Support team was able to reproduce the issue, but Engineering team was not.

Our first thought - platform specific bug in a parser.

The YAML looked perfectly fine.

Or did it?

Spot the problem

This YAML looks correct. What's wrong?

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  ENV: production
```

↑ *Something is hidden in that first key.*

⚡ LIVE
DEMO

Demo 1: The invisible character

The Go parser sees something else

```
type Config struct {  
    APIVersion string  
    `yaml:"apiVersion"`  
}  
  
var c Config  
yaml.Unmarshal(data, &c)  
  
fmt.Printf("%+v\n", c)  
// Config{APIVersion:""}
```

APIVersion is empty!

No error.

No warning.

Silent failure.

The parser matched
"apiVersion\u200b"
not "apiVersion"

How to see the invisible characters?

UTF-8 encoding
revealed

```
$ xxd bad.yaml
00000000: 6170 6956 6572 7369 6f6e e280 8b3a 2076
apiVersion...: v
00000010: 310a 6b69 6e64 3a20 436f 6e66 6967 4d61
1.kind: ConfigMa
00000020: 700a
```

p.

^

^^^^^^^^^^

'n' of apiVersion E2 80 8B = U+200B

Three extra bytes hiding in plain sight.

| The culprit

U+200B
ZERO WIDTH SPACE

- ✗ Invisible to humans
- ✗ Perfectly valid UTF-8
- ✗ Silently corrupts YAML keys
- ✗ Not caught by most linters

It gets worse: homoglyphs

Two characters. One face.

A blue lowercase letter 'a' centered within a white rectangular box.

Latin a (U+0061)

A red lowercase letter 'a' centered within a white rectangular box.

Cyrillic a (U+0430)

Your eyes see the same character. The parser sees different bytes.

| Demo 2: Homoglyph attack

Why this happens

Unicode

Abstract character set

Defines code points
(e.g. U+200B)

Doesn't say how to
store them in bytes

UTF-8

Byte encoding of Unicode

U+200B → E2 80 8B
(3 bytes, all valid)

Your file is valid UTF-8,
but semantically broken

Humans review visually. Parsers compare bytes. That gap is the attack surface.

The shady characters

Character	Code point	UTF-8 bytes	Risk
Zero-width space	U+200B	E2 80 8B	Invisible key corruption
Non-breaking space	U+00A0	C2 A0	Broken indentation
Zero-width joiner	U+200D	E2 80 8D	Rendering tricks
Byte order mark	U+FEFF	EF BB BF	Parser inconsistencies
RTL override	U+202E	E2 80 AE	Visual text spoofing
Cyrillic lookalikes	U+0430...	D0 B0...	Homoglyph substitution

But it gets
MUCH worse.

YAML files aren't the only place Unicode hides.

Unicode BIDI: bidirectional text

Some languages like Hebrew, Arabic read right-to-left. Unicode handles this with 'bidirectional' control characters.

U+202E

**RIGHT-TO-LEFT
OVERRIDE**

Forces text to render RTL

U+2066

LEFT-TO-RIGHT ISOLATE

Isolates a section in LTR

U+2069

**POP DIRECTIONAL
ISOLATE**

Ends the isolate

U+202A

**LEFT-TO-RIGHT
EMBEDDING**

Embeds LTR inside RTL

BIDI makes code lie to you

```
1 package main
2
3 import "fmt"
4
5 ✓ func main() {
6     var accessLevel = "user"
7     if accessLevel != "user" // Check if admin {
8         fmt.Println("You are an admin.")
9     }
10 }
```

```
1 package main
2
3 import "fmt"
4
5 ✓ func main() {
6     var accessLevel = "user"
7     if accessLevel != "user"U+202E U+2066// Check if adminU+2069 U+2066" {
8         fmt.Println("You are an admin.")
9     }
10 }
```

| Demo 3: BIDI in Go code

How is this risk handled in present day?

- ▶ Github highlights the BIDI characters - <https://github.com/nickboucher/trojan-source/blob/main/Go/string/etcd-string.go#L7>
- ▶ Linters like golangci-lint already support flagging BIDI

Recommended reading

"Trojan Source: Invisible Vulnerabilities"

Brady, Laurie, Cadar, Anderson — University of Cambridge, 2021

- ▶ BIDI control chars can make comments look like code — and vice versa
- ▶ Affects C, C++, Go, Rust, Python, Java, JavaScript, Ruby, and more
- ▶ Works on GitHub's code review UI — attackers hide exploits in PRs
- ▶ CVE-2021-42574 — critical severity

Defending your CI/CD pipelines

1. Grep for non-ASCII

```
grep -rPn '[^\x00-\x7F]' .  
--include='*.yaml'
```

*Catches zero-width chars, Cyrillic, BIDI
— everything*

2. yamllint

```
extends: default  
rules:  
  key-duplicates: enable
```

Structural lint, catches duplicate keys

3. Strict unmarshaling

(Go)

```
decoder :=  
yaml.NewDecoder(f)  
decoder.KnownFields(true)
```

*Unknown fields become errors instead
of silent drops*

4. Schema validation

OpenAPI / JSON Schema

Type-check values, enforce key naming

yuc — YAML Unicode Checker

One tool that catches all of the above, with configurable severity rules yuc

```
# Install
curl -fsSL
https://raw.githubusercontent.com/harshadixit12/yuc/refs
/heads/main/install.sh | bash

# Run against your YAML
yuc bad.yaml

# Run against a whole repo
yuc ./k8s/
```

A reasonable Unicode policy

Not all Unicode is bad — you need it for real i18n content.

YAML area	Policy	Why
Keys	ASCII-only	<i>No surprises in struct tags</i>
Identifiers	ASCII-only	<i>No homoglyph confusion</i>
Values	Unicode allowed	<i>Real content: 接口-api is fine</i>
Comments	Restrict BIDI	<i>Prevent Trojan Source in configs</i>

Key takeaways

Valid UTF-8 $\neq$ safe YAML.

Zero-width chars silently corrupt keys.

Homoglyphs fool human reviewers.

BIDI control chars can make code lie to the compiler.

Defense is layered: lint $\rightarrow$ grep $\rightarrow$ strict parsing $\rightarrow$ schema.

Check out the Trojan Source paper. It will change how you read

Resources

Trojan Source Paper

trojansource.codes | arxiv.org/abs/2111.11997

CVE

[CVE-2021-42574](#) (critical – affects most compilers and editors)

yuc — YAML Unicode Checker

github.com/harshadixit12/yuc

Blog post on this topic

harshadixit.com/articles/is_your_yaml_hiding_something/

Thank you

github.com/harshadixit12 · harshadixit.com