FORM3

# The shimmy to the left: why security is coming for engineers

Adelina Simion & Artur Kondas

Join us to learn the essentials for writing secure application code in Go! If we can do it, YOU can do it.

Oh Hai!

# 👋 Adelina Simion



🎙️ Tech Evangelist

🇬🇧 Based in London, UK

📙 Shares her Go knowledge and organises Women Who Go

adelina-simion

addetz

Oh Hai!

# 👋 Artur Kondas

💻 Lead Engineer @ Form3

🇵🇱 Based in Krakow, Poland

🛠️ Go engineer focusing on SEPA payments

arturkondas

youshy

FORM3

Oh Hai!

# 💼 About Form3

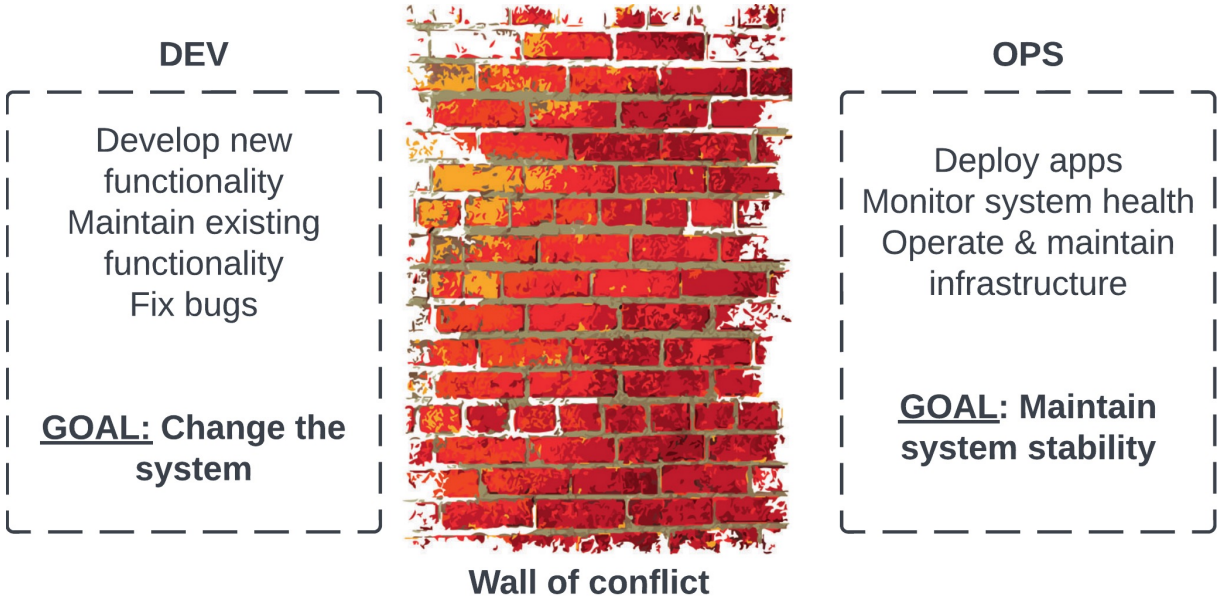
- Real-time payments processing platform
- Multi-cloud (AWS, GCP, Azure)
- Go, IaC (Terraform)
- SecDevOps culture
- Fully remote


FORM3

# 🛠️ Brief history of (Sec)DevOps

# 📜 Ancient history cc. early 2000s

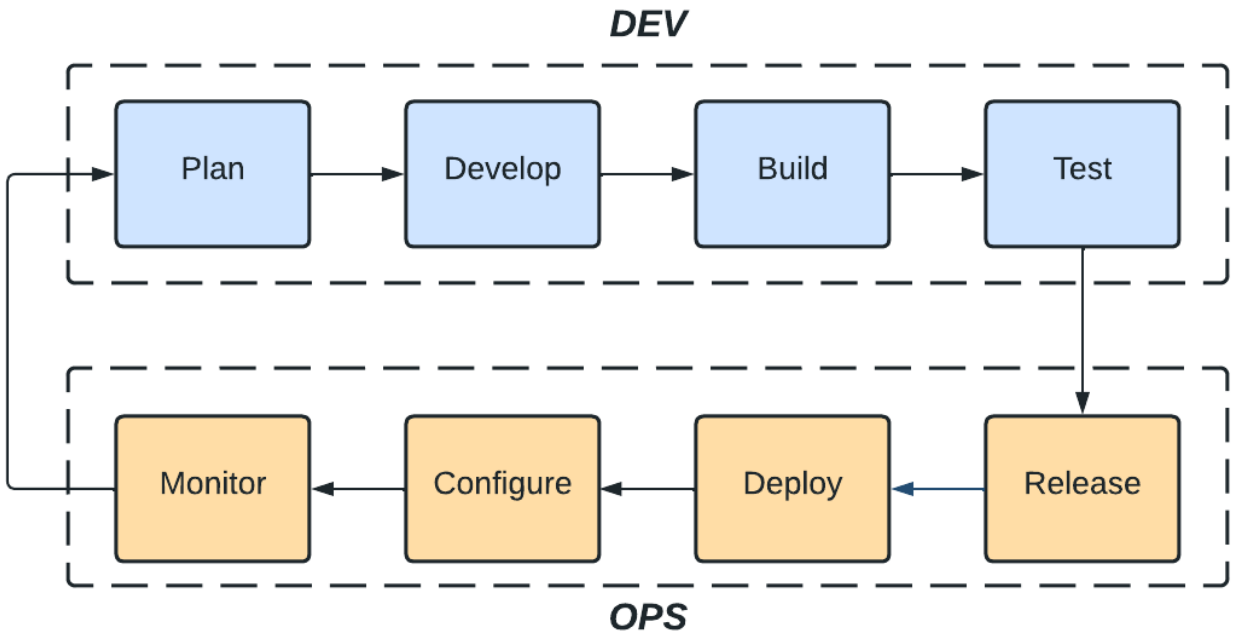
- Despite Agile adoption, teams remained siloed
- Months-long releases
- IT professionals & engineers have different goals and priorities
- Those who **wrote** the code were separate from those who **support** it, resulting in reduced ownership and responsibility.




FORM3

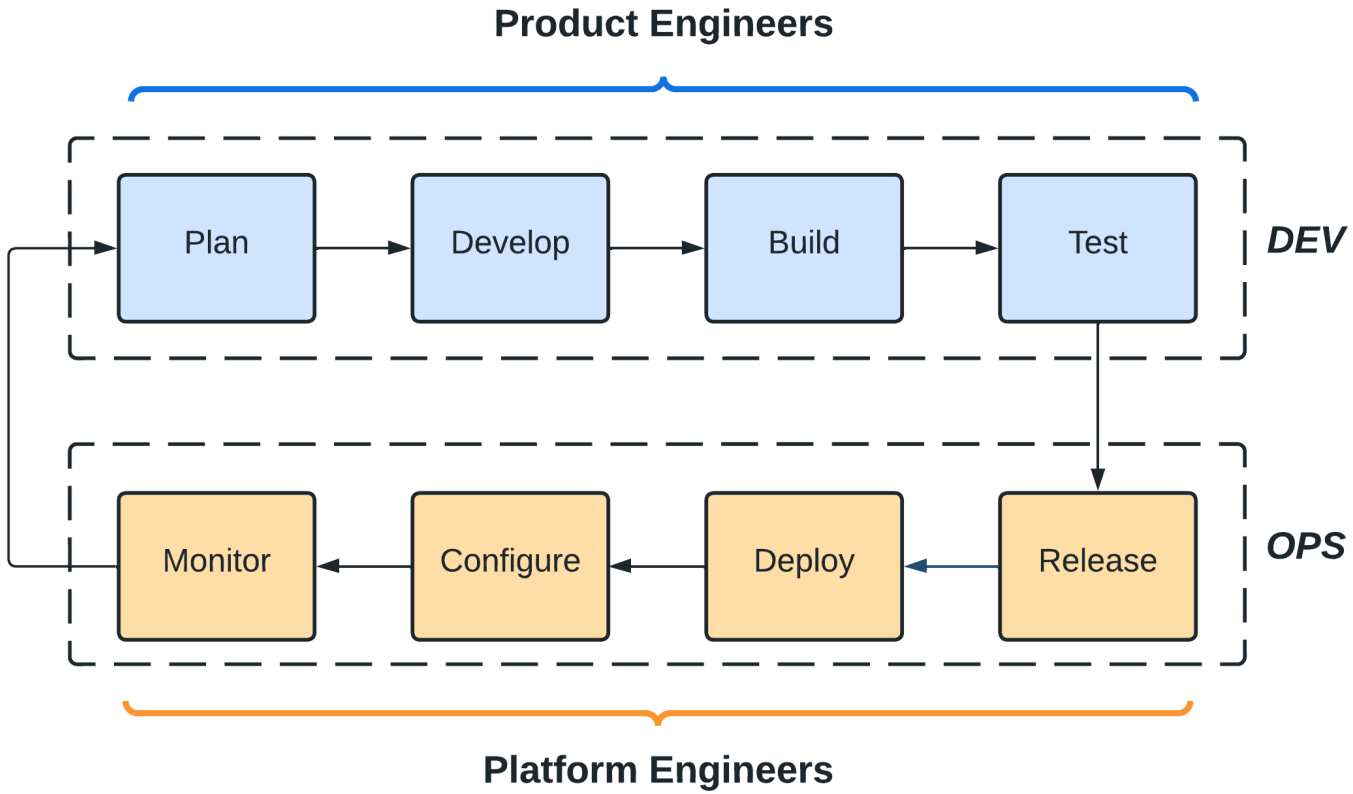
# 🌎 Hello, ~~world~~ DevOps!

- Methodology emerged in 2007
- Touches every phase of the SDLC
- Promote communication and collaboration between operation and development
- Shorter lead times
- Lower failure rates



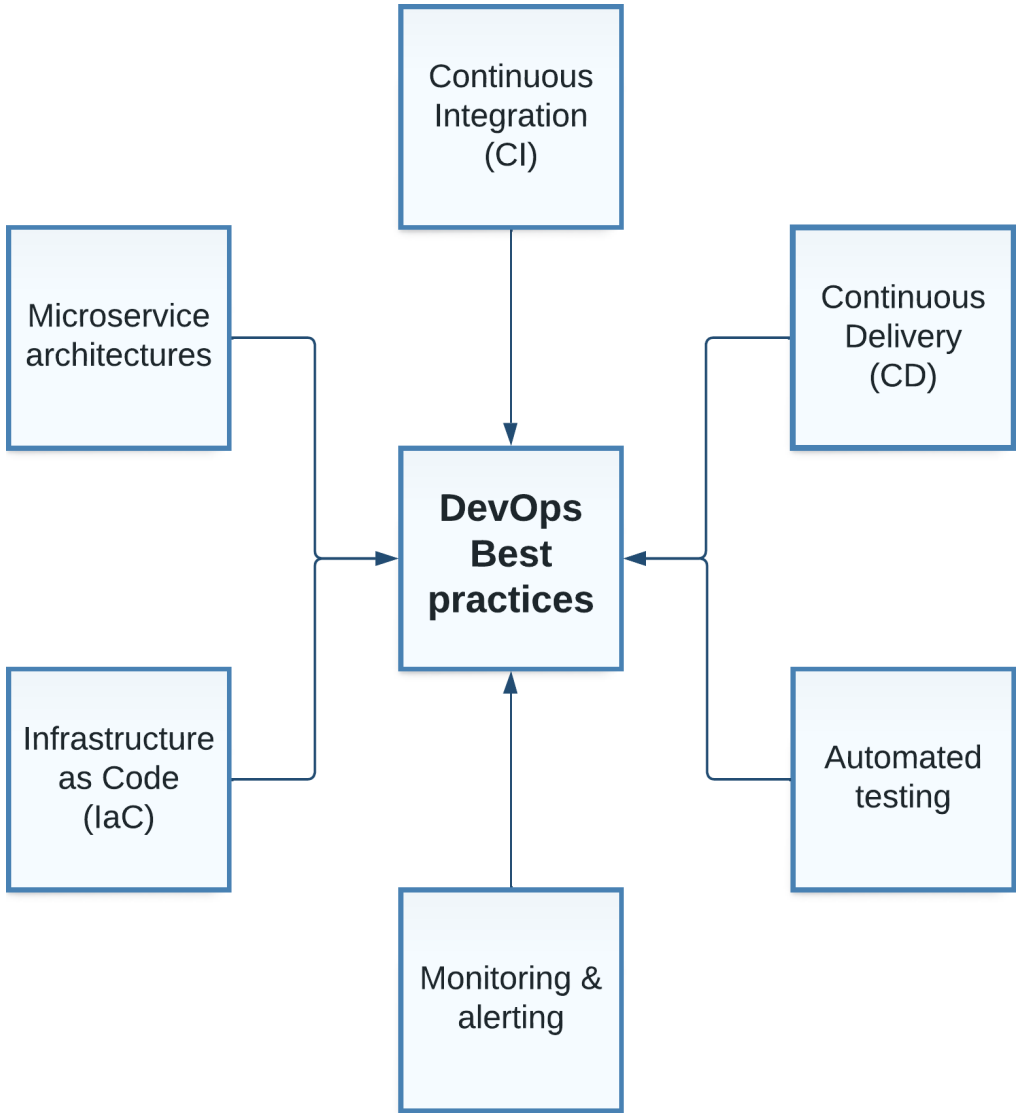
# 🌎 Hello, ~~world~~ DevOps!

- Methodology emerged in 2007
- Touches every phase of the SDLC
- Promote communication and collaboration between operation and development
- Shorter lead times
- Lower failure rates



# 🛠️~~Brave~~ DevOps new world

- How many of you work in DevOps organisations? (Show of hands 🖐)
- Agile practices bring developers closer to the business stakeholders
- DevOps practices bring developers closer to operations
- Varying definitions of DevOps, but the community agrees on best practices



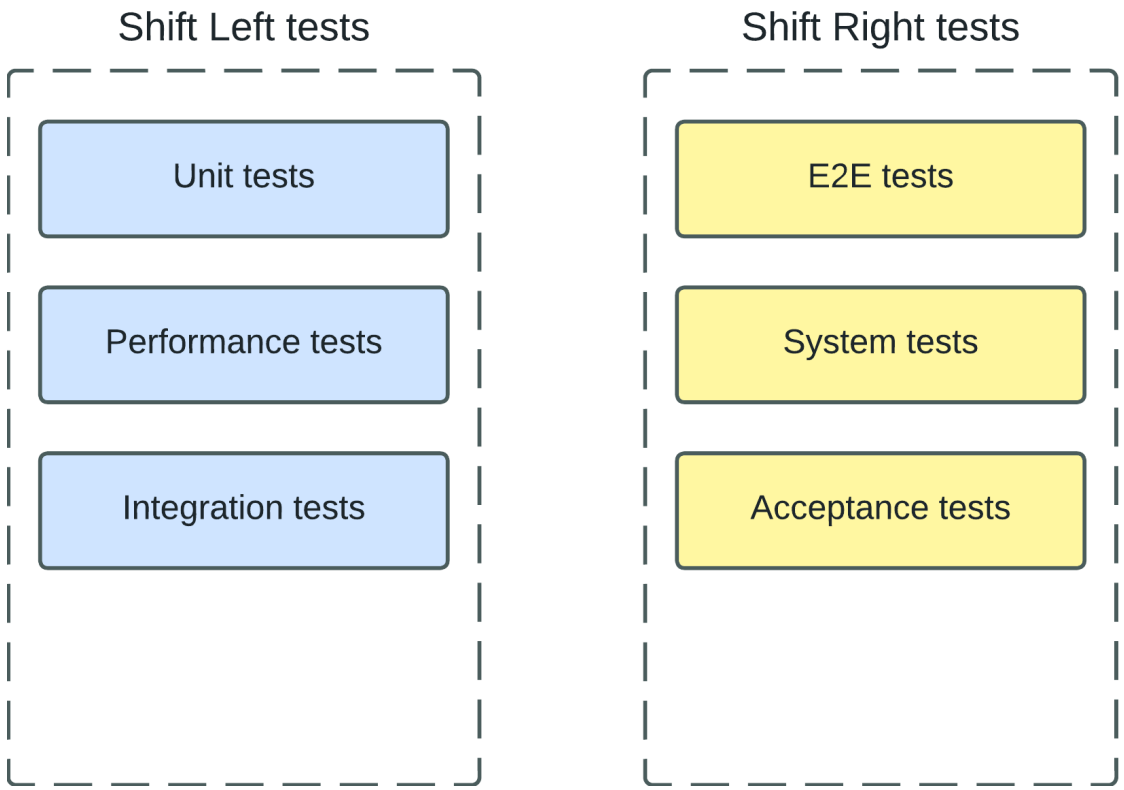
# Can we get some directions? 👈🤔👉

**Shift Left**

- Move testing, quality and performance evaluation early in the development process.
- Tests the system in simulated environments

**Shift Right**

- Perform testing at or near deployment, later in the development process.
- Verifies the system under real conditions



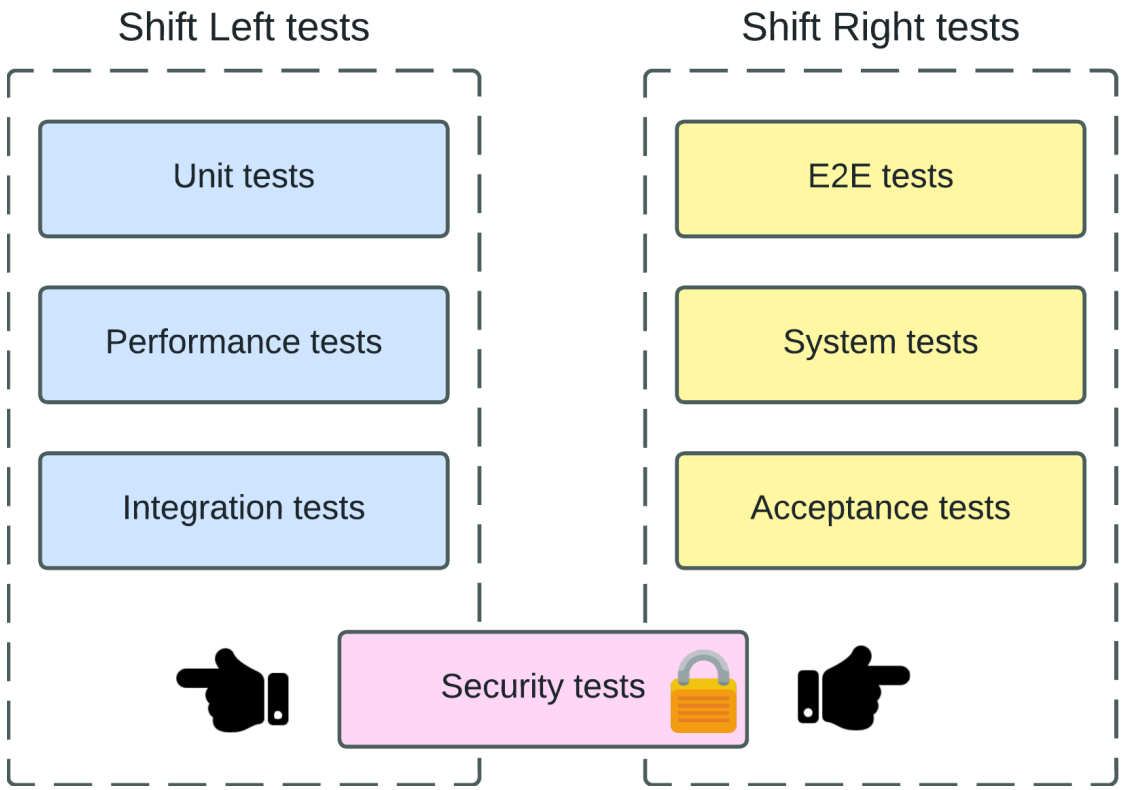
# Can we get some directions? 👈🤔👉

**Shift Left**

- Move testing, quality and performance evaluation early in the development process.
- Tests the system in simulated environments

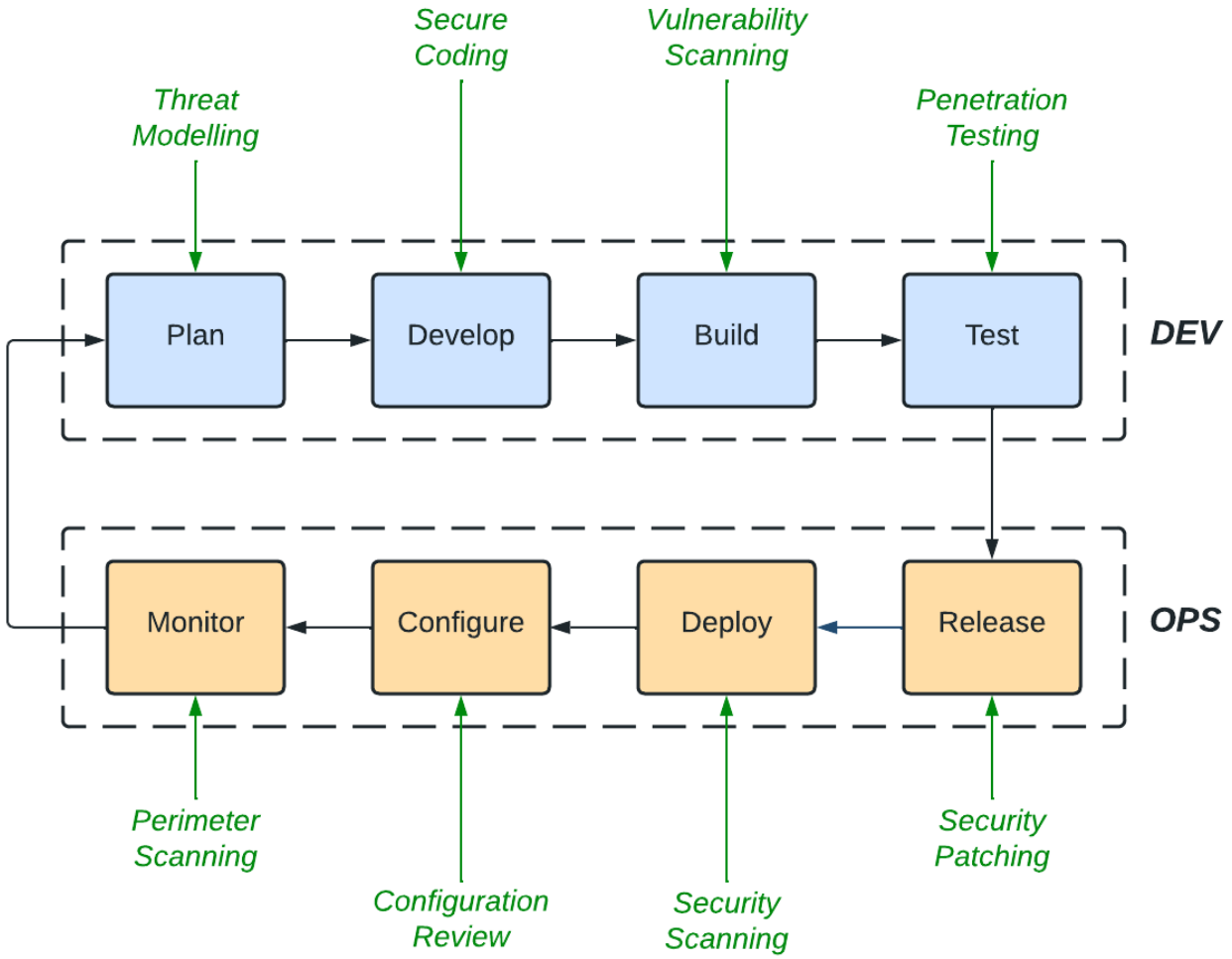
**Shift Right**

- Perform testing at or near deployment, later in the development process.
- Verifies the system under real conditions



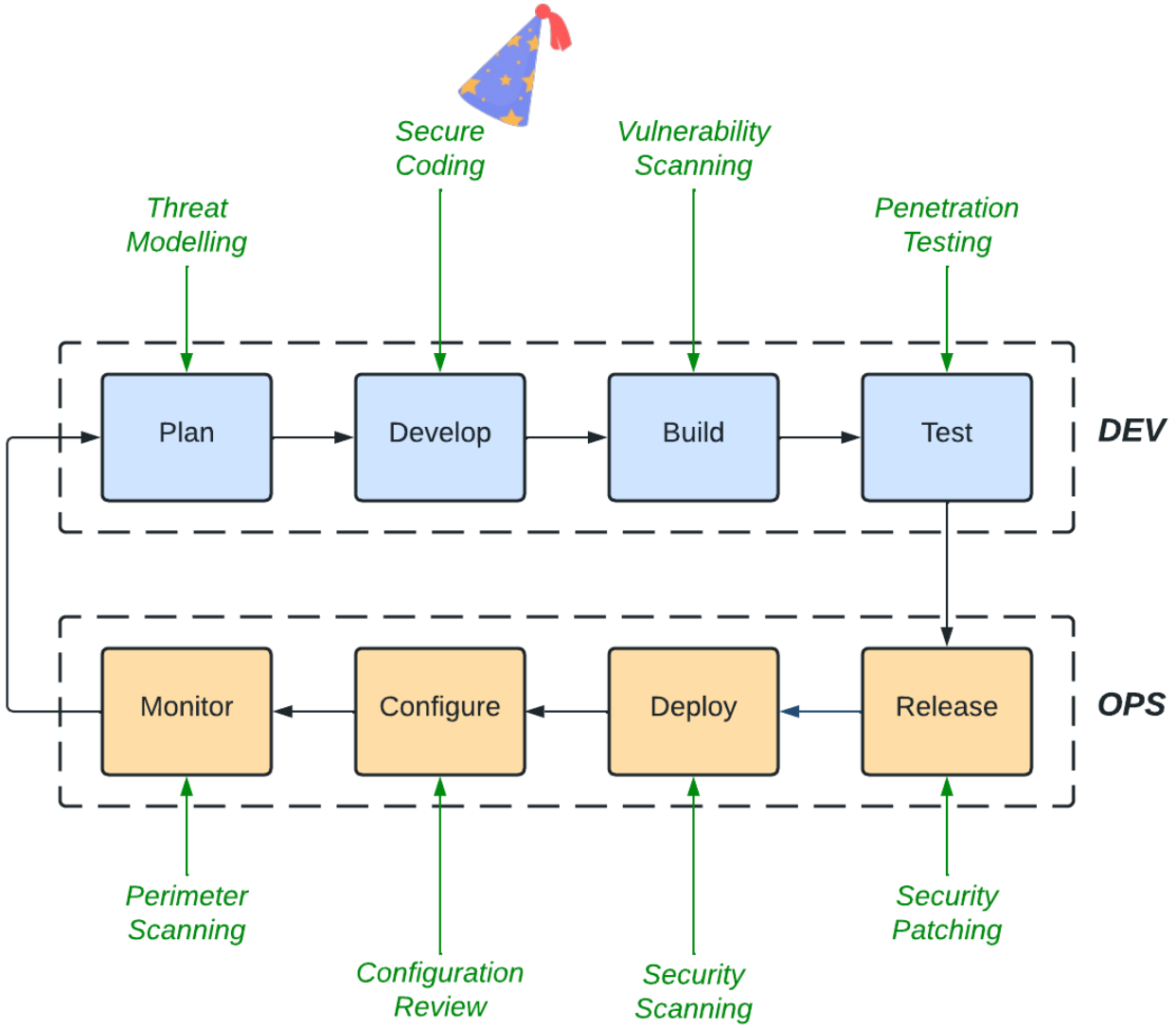
# 🔒 Here comes SecDevOps

- Brings security teams together with development and IT Ops teams.
- Extension of DevOps that includes the automation of security testing.
- Security practices should be included in every stage of development.
- Security teams focus on developer education.

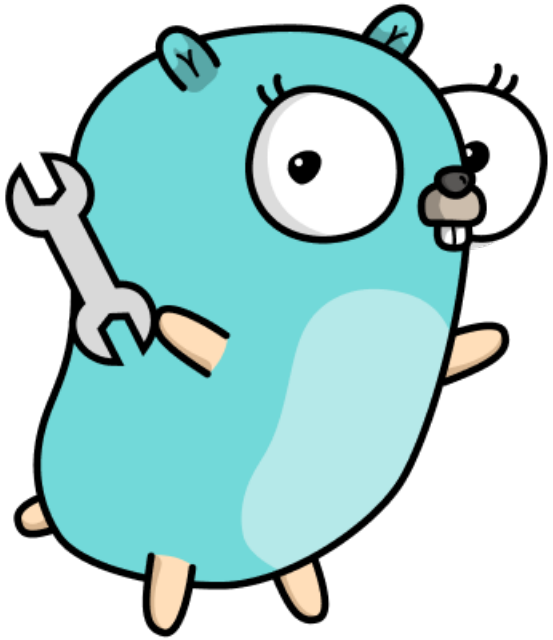


# 🔒 Here comes SecDevOps

- Brings security teams together with development and IT Ops teams.
- Extension of DevOps that includes the automation of security testing.
- Security practices should be included in every stage of development.
- Security teams focus on developer education.



# Mastering secure coding practices makes us better, more well-rounded developers and future-proofs your career.



# New phone, who dis OWASP? 🐝

# 🐝 What is OWASP?

- The **O**pen **W**orldwide **A**pplication **S**ecurity **P**roject (**OWASP**) foundation was established in 2001.
- Non-profit that works to improve the security of software worldwide.
- Community initiative with worldwide chapters with the purpose of **educating developers** and **promoting security best practices**.
- owasp.org

# 📈 The Top 10

- Standard awareness document based on community consensus.
- Vulnerabilities move according to how commonly they appear in tested applications.
- The latest Top 10 for web applications was compiled in 2021, the list to the right is from 2023.

| # | Name | Description |
|---|---|---|
| **1** | Broken Object Level Auth | APIs expose object IDs. Improper auth checks for operations using a user provided ID. |
| **2** | Broken Auth | Faulty auth mechanisms allow attackers to assume other user identities. |
| **3** | Broken Object Property Level Auth | APIs expose sensitive object data. Improper auth at object property level. |
| **4** | Unrestricted Resource Consumption | API requests consume resources. Unbounded consumption can lead to DoS or increased costs. |
| **5** | Broken Function Level Auth | Improper auth mechanisms allows attackers access to admin functionality. |

# 📈 The Top 10

- Standard awareness document based on community consensus.
- Vulnerabilities move according to how commonly they appear in tested applications.
- The latest Top 10 for web applications was compiled in 2021, the list to the right is from 2023.

| # | Name | Description |
|---|---|---|
| **1** | Broken Object Level Auth | APIs expose object IDs. Improper auth checks for operations using a user provided ID. |
| **2** | Broken Auth | Faulty auth mechanisms allow attackers to assume other user identities. |
| **3** | Broken Object Property Level Auth | APIs expose sensitive object data. Improper auth at object property level. |
| **4** | Unrestricted Resource Consumption | API requests consume resources. Unbounded consumption can lead to DoS or increased costs. |
| **5** | Broken Function Level Auth | Improper auth mechanisms allows attackers access to admin functionality. |

**Remember to authenticate all sensitive operations on the backend!**

# 📈 The Top 10

- Standard awareness document based on community consensus.
- Vulnerabilities move according to how commonly they appear in tested applications.
- The latest Top 10 for web applications was compiled in 2021, the list to the right is from 2023.

| # | Name | Description |
|---|---|---|
| **6** | Unrestricted Access to Sensitive Business Flows | The business exposes functionality (e.g. promotional campaign) without considering how it can be abused if used excessively. |
| **7** | Server Side Request Forgery | Requests are sent to user supplied URLs even when protected by a firewall. |
| **8** | Security Misconfiguration | Incorrectly configured customizable APIs. |
| **9** | Improper Inventory Management | API exposed endpoints are not properly documented. Versions are are not properly maintained. |
| **10** | Unsafe Consumption of APIs | Integrated third-party API supplied input is not properly validated and secured. |

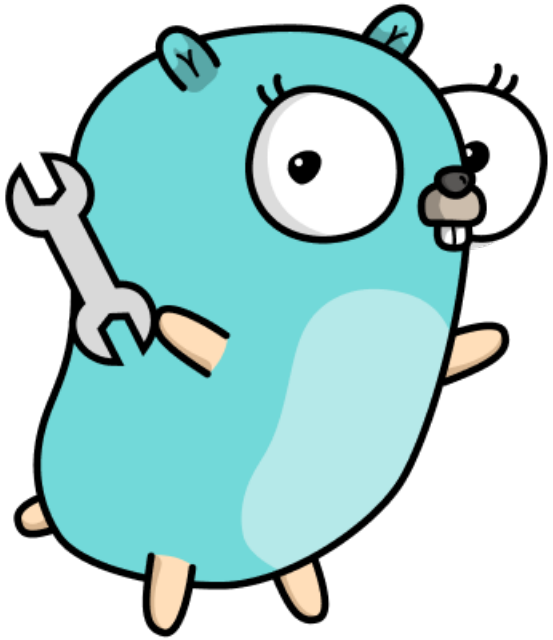
# 📈 The Top 10

- Standard awareness document based on community consensus.
- Vulnerabilities move according to how commonly they appear in tested applications.
- The latest Top 10 for web applications was compiled in 2021, the list to the right is from 2023.

| # | Name | Description |
|---|---|---|
| **6** | Unrestricted Access to Sensitive Business Flows | The business exposes functionality (e.g. promotional campaign) without considering how it can be abused if used excessively. |
| **7** | Server Side Request Forgery | Requests are sent to user supplied URLs even when protected by a firewall. |
| **8** | Security Misconfiguration | Incorrectly configured customizable APIs. |
| **9** | Improper Inventory Management | API exposed endpoints are not properly documented. Versions are are not properly maintained. |
| **10** | Unsafe Consumption of APIs | Integrated third-party API supplied input is not properly validated and secured. |

**All external supplied input needs to be validated and limited on the backend!**

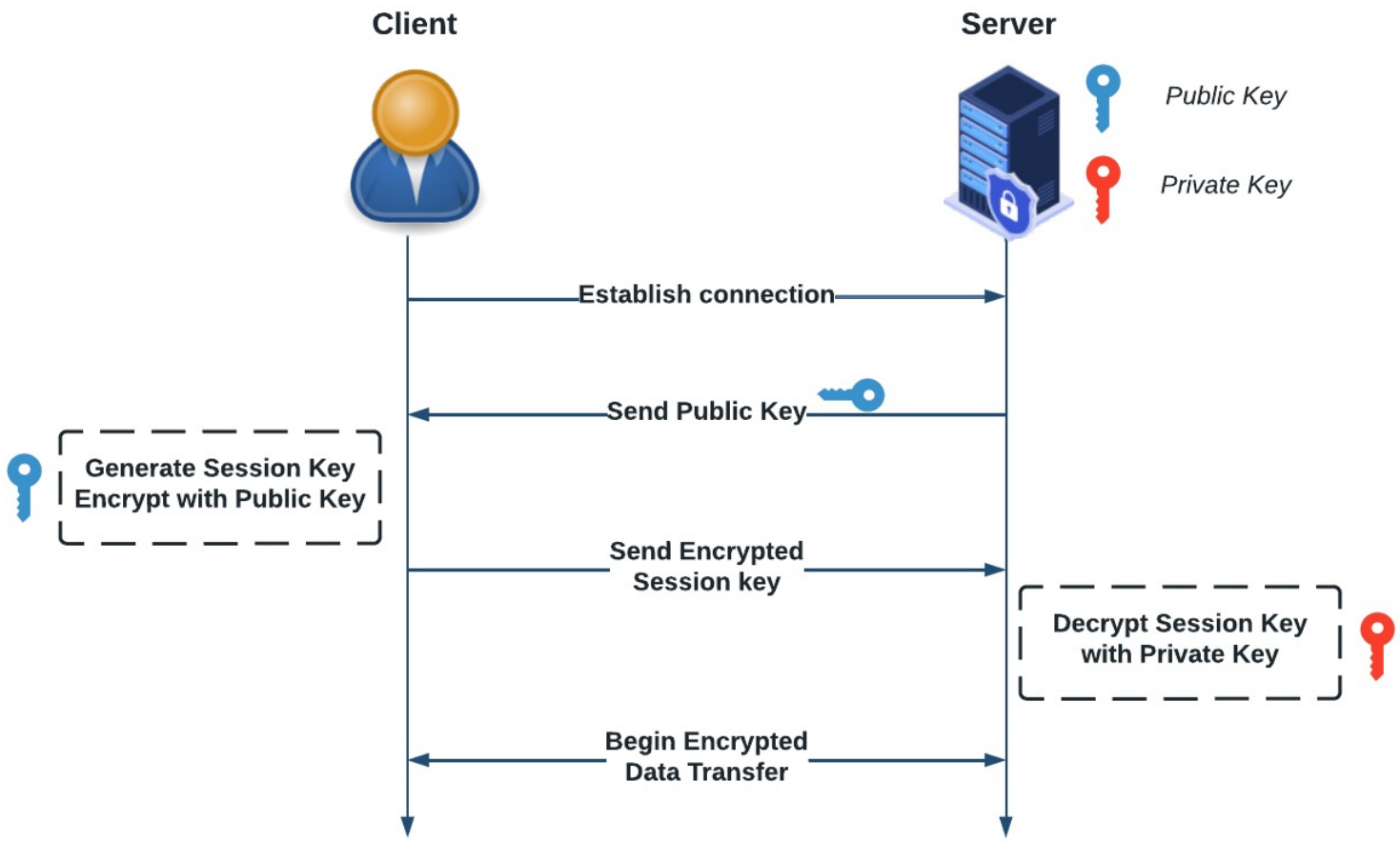
The OWASP Top 10 gives us a series of best practices. It is easier to write secure code from the beginning as opposed to implementing it before release.



# 💥 Common application level mistakes

# 🌎 Client-server security

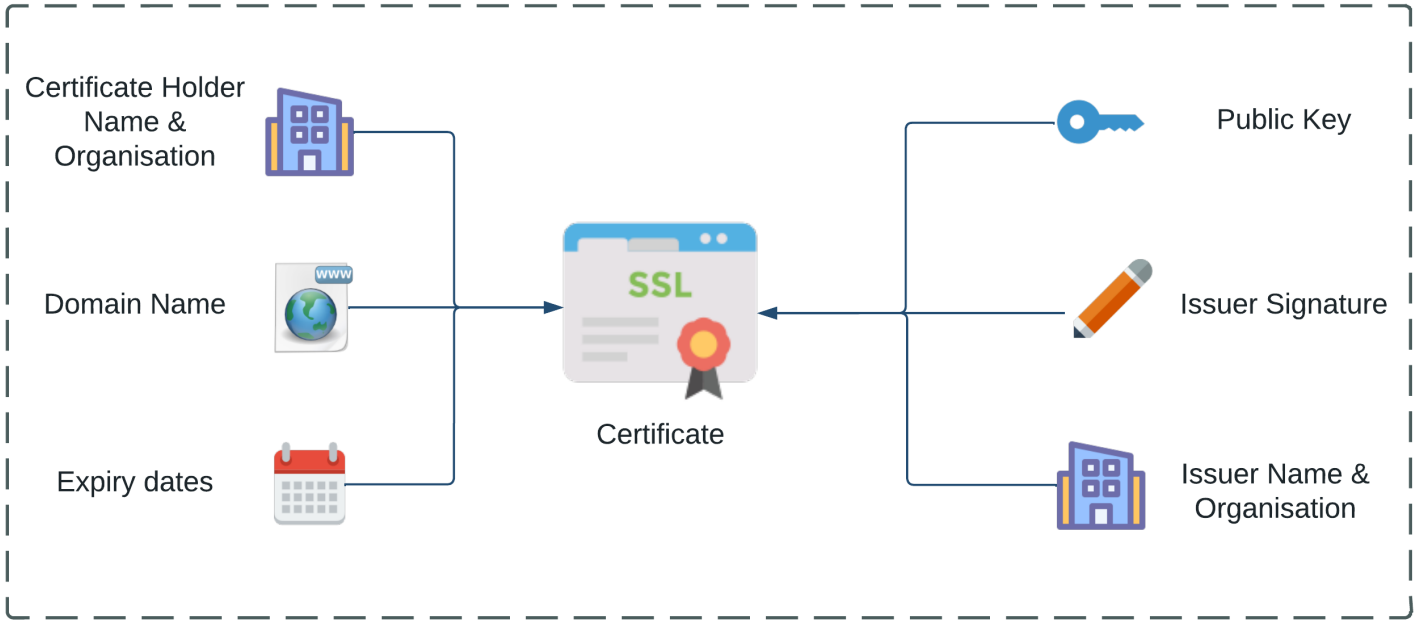
- Enable HTTPS on servers
- Encrypt data using a public key and session key. The server decrypts using the private key, which only it knows
- Set timeouts to prevent DDOS attacks




FORM3

# What about certificates?

- Authenticates a website's identity
- Enables an encrypted connection through SSL, which runs at a lower level than HTTPS
- Trusted Certificate Authorities (CAs) issue certificates which contain the Public Key
- Self-signed certificates should be avoided as they cannot be vetted



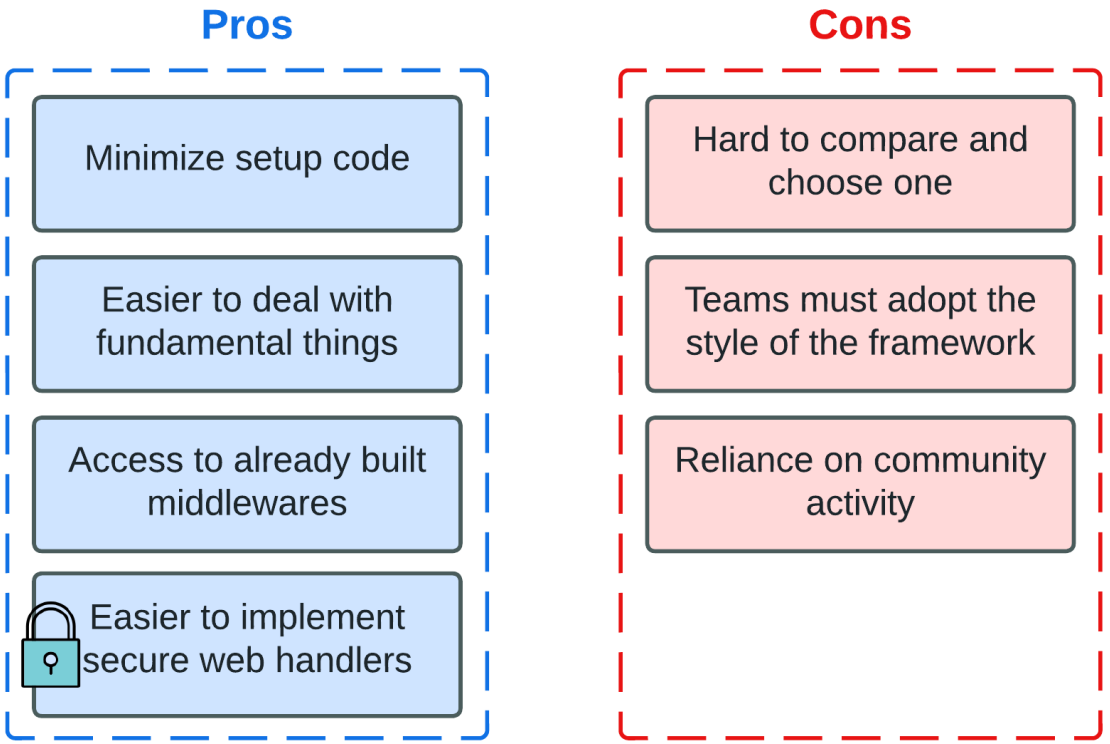

FORM3

# 🛠️ Web frameworks

The Go community has a love-hate relationship with web frameworks

- github.com/gin-gonic/gin
- github.com/labstack/echo
- github.com/valyala/fasthttp
- github.com/gorilla
- github.com/gofiber/fiber

**... AND MORE!**



# 🥳 Introduction to echo

- We will be using **labstack/echo** in all our demo code, but what we discuss *should* be available in other frameworks too.
- Easy routing and set up
- Lots of useful middlewares

```
$ go get
  github.com/labstack/echo/v4
```


*Official docs: [echo.labstack.com](echo.labstack.com)*

FORM3

# 🥳 Show me the code!

- Certificates can be created with the awesome `FiloSottile/mkcert`
- Create a local server configured with TLS
- `echo` makes the previously discussed configuration easy-peasy!

```
$ go run demo1/server.go
```

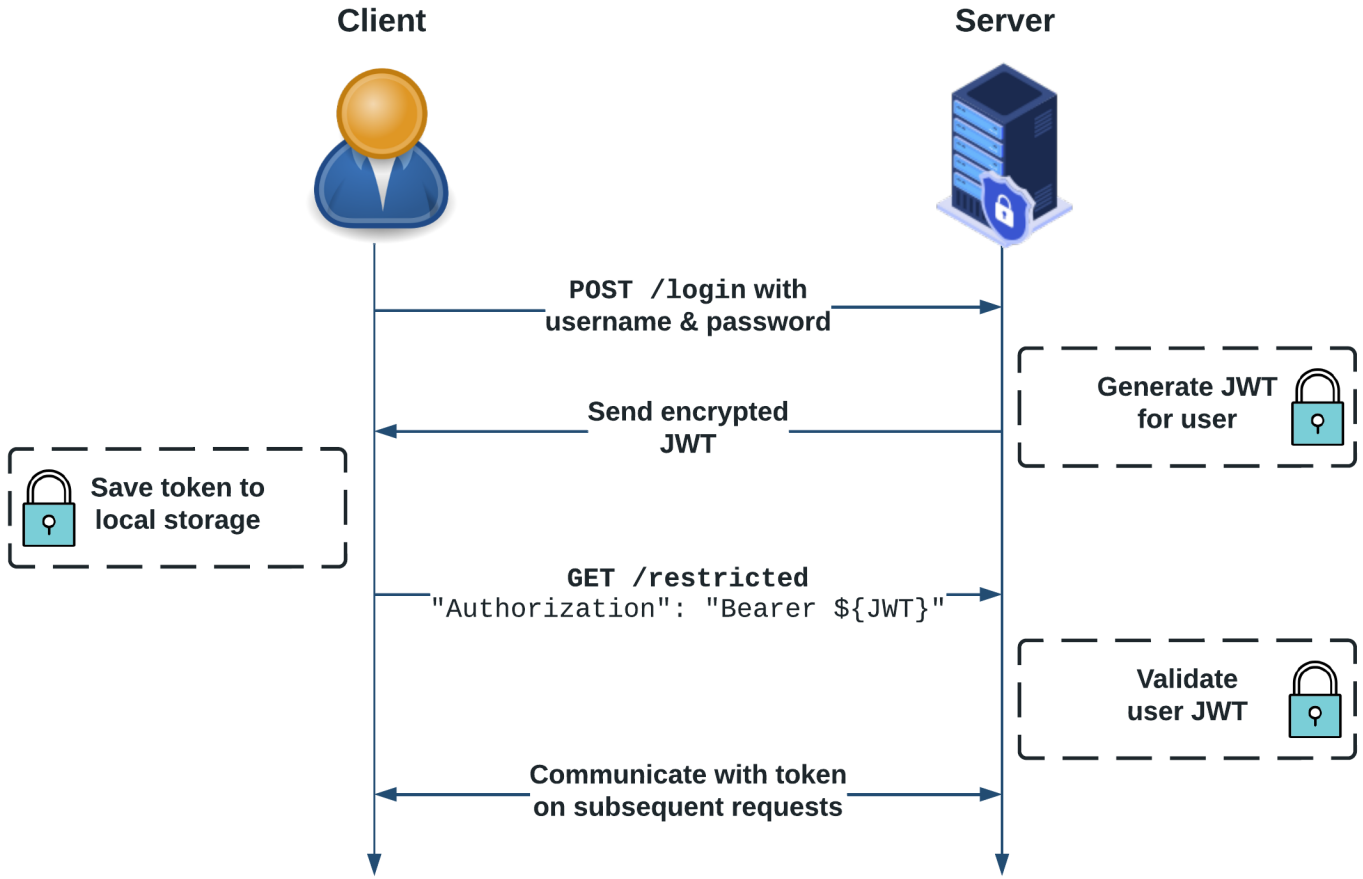
See it on GitHub:

github.com/addetz/secure-code-go

FORM3

# ✅ Handling authentication

- Implement session management for your backends
- Don’t permit weak passwords
- Leave no default or easily guessable passwords for your admin pages.



# 🥳 Show me the code!

- `echo` provides a JWT middleware.
- `wagslane/go-password-validator` is a simple password validator.
- `golang.org/x/crypto/bcrypt` hashes passwords for storage

```
$ go run demo2/server.go
```

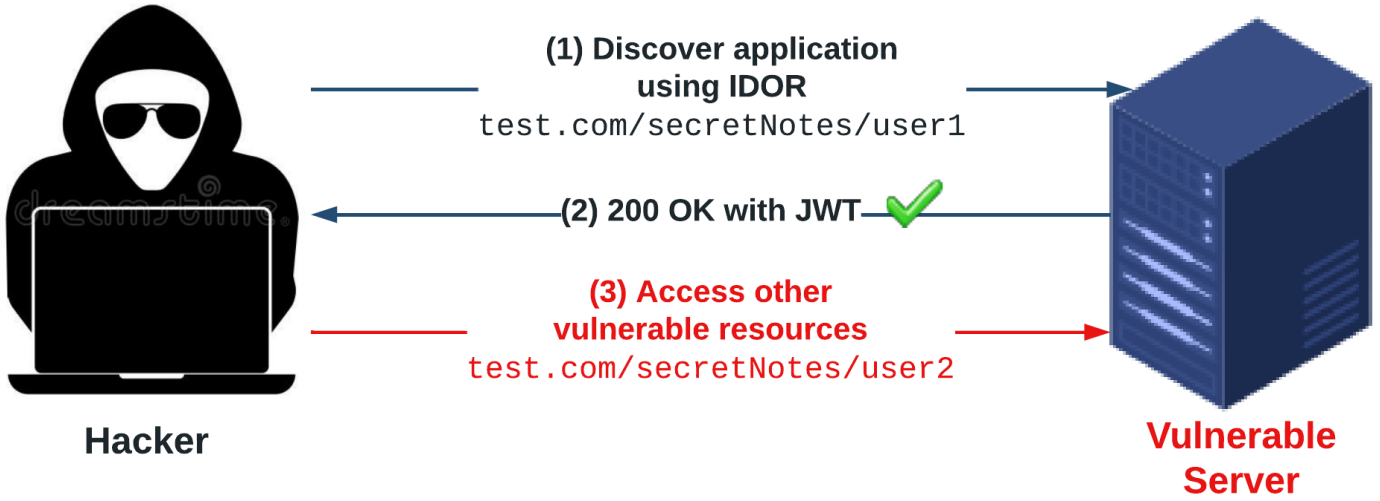
See it on GitHub:

github.com/addetz/secure-code-go

FORM3

# ✍️ Modifying resources

- Don’t leak Insecure Direct Object References (IDOR).
- Enforce access control policies in the backend.
- Ensure that users cannot elevate to administrative privileges.




FORM3

# 🥳 Show me the code!

- Enforce ownership checks and role validations on operation level.
- If a resource ID leaks, the server will prevent disallowed operations.

```
$ go run demo3/server.go
```

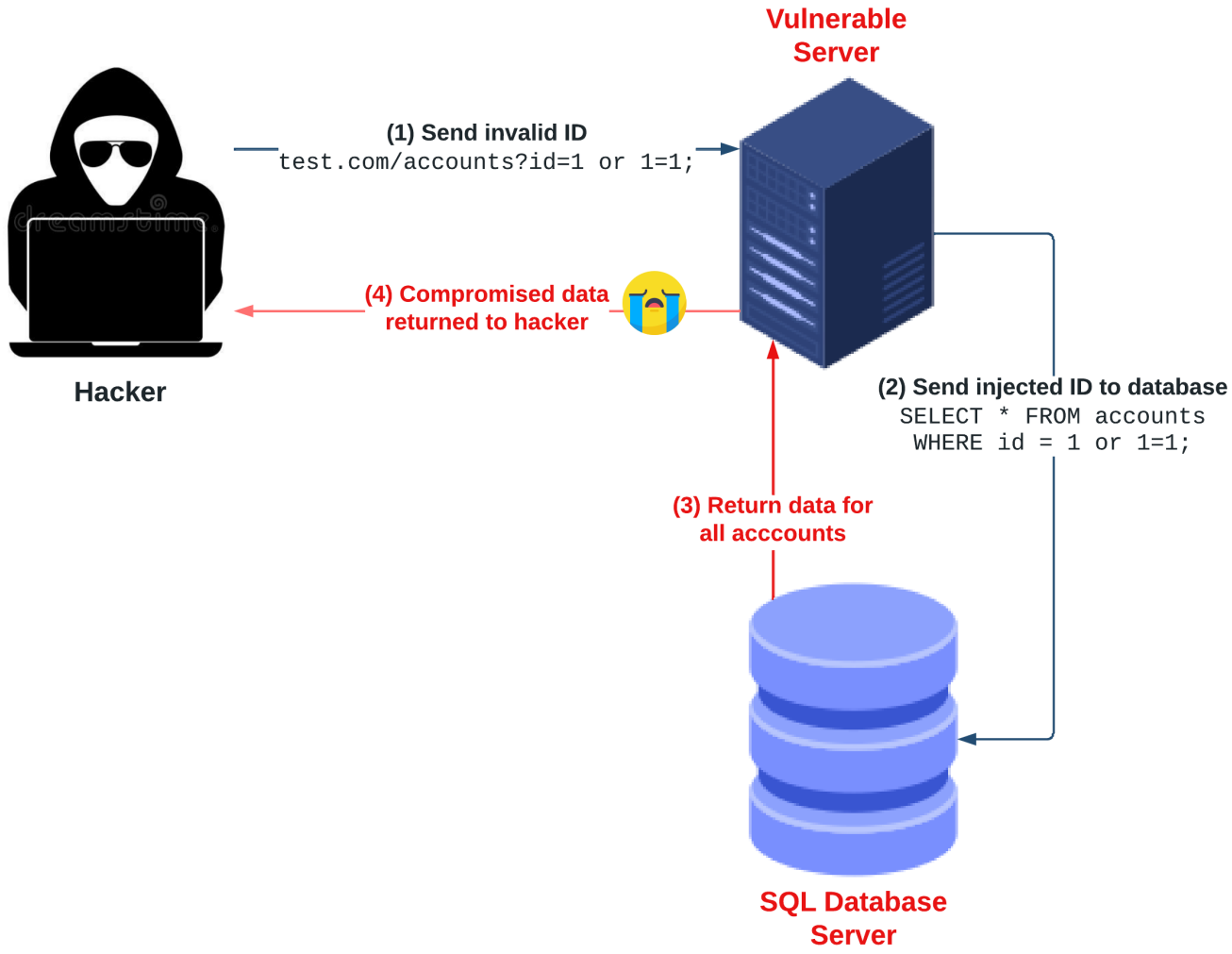
See it on GitHub:

github.com/addetz/secure-code-go

FORM3

# 💾 Handling SQL operations

- How many of you out use a SQL database? (Show of hands ✋)
- ORMs are not guaranteed to be immune to SQL injection
- Prepared statements allow us to distinguish between code and supplied values




FORM3

# 🥳 Show me the code!

- We will add a PostgreSQL database to our app to save our users and notes.
- The `database/sql` has prepared statements and transactions functionality for us to use

```
$ go run demo4/server.go
```

See it on GitHub:

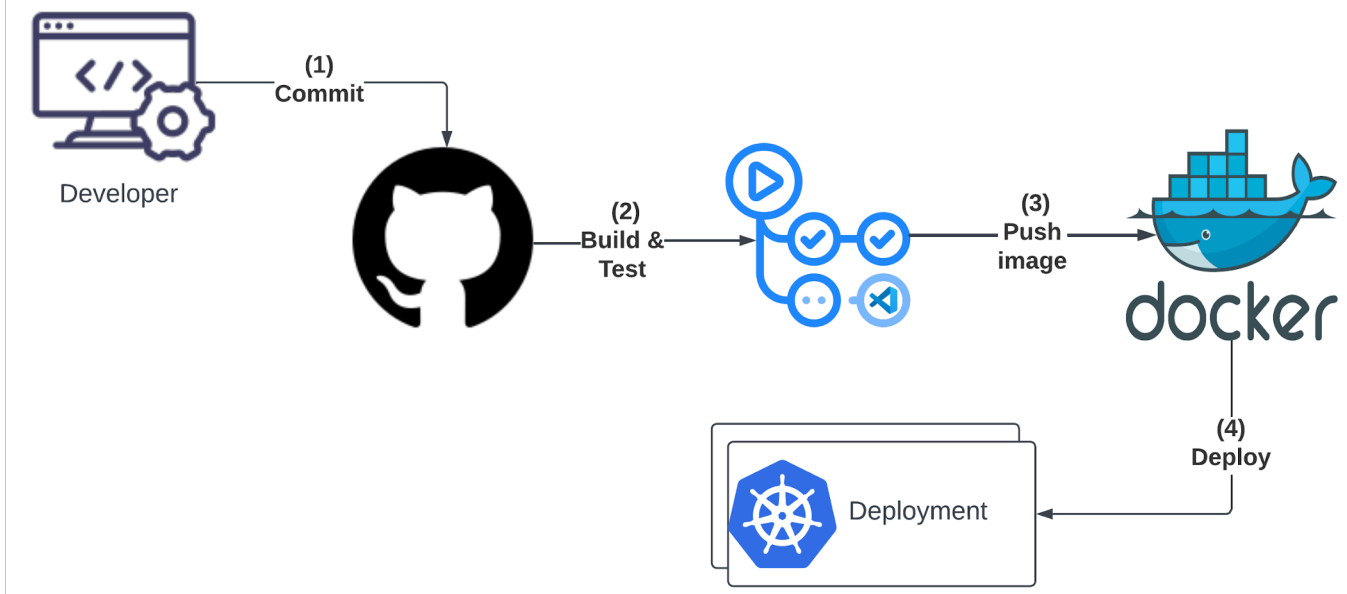
github.com/addetz/secure-code-go

FORM3

# Automated tooling to make life easy breezy 🍹

# 🚦 Pipelines

- Security tooling should be run as part of the CI/CD pipelines for security scanning and vulnerability detection.
- **GitHub Actions** are a popular solution for creating custom CI/CD pipelines.




FORM3

# 🧐 Fuzz testing

- Automated tests that are run using the same testing commands.
- Fuzzed tests generate values, making it easy to write many unit tests without having to manually write test cases.
- Help us to ensure **code robustness** and detect **dangerous input**

```
func FuzzSayHello(f *testing.F) {
  f.Add("Anna")
  f.Add("Belle")
  f.Fuzz(func(t *testing.T, n string) {
    want := "Hello, " + n + "!"
    got := SayHello(n)
    if got != want {
      t.Fatalf("got: %s; want: %s",
        got, want)
    }
  })
}
```

FORM3

# 🥳 Show me the code!

- We will add a GitHub Actions pipeline for linting & security checking
- We will also fuzz test & database using `DATA-DOG/go-sqlmock`

```
$ go run demo5/server.go
$ go run demo6/server.go
$ go run demo7/server.go
```

See it on GitHub:
github.com/addetz/secure-code-go

FORM3

# Conclusions

# Secure coding practices should be second nature

Despite popular belief, incorporating secure coding practices doesn't have to be hard. If these practices are top of mind for engineers, they can be implemented easily with the help of Go web frameworks.

# Secure coding practices should be second nature

Despite popular belief, incorporating secure coding practices doesn't have to be hard. If these practices are top of mind for engineers, they can be implemented easily with the help of Go web frameworks.

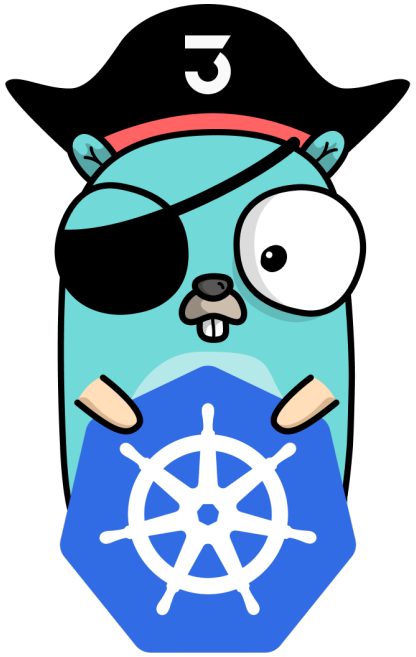
Security tooling monitors and detects vulnerabilities, ensuring that your code stays secure. You should remember to add it to your CI/CD pipelines alongside your functional tests/verifications.

# Secure coding practices should be second nature

Despite popular belief, incorporating secure coding practices doesn't have to be hard. If these practices are top of mind for engineers, they can be implemented easily with the help of Go web frameworks.

Security tooling monitors and detects vulnerabilities, ensuring that your code stays secure. You should remember to add it to your CI/CD pipelines alongside your functional tests/verifications.

In order to truly embrace SecDevOps, you should embrace a culture of security. It is OK to ask questions and upskill together.



# Thank you!

🎧 Podcast: techpodcast.form3.tech

📖 Blog: form3.tech/engineering/content

FORM3