

SÄKERHET UTAN FRIKTION

SÅ GÖR DU SÄKERHET SYNLIGT OCH PRIORITERAT
VARJE DAG

OVE LINDSTRÖM

CADEC 2026.01.22 & 2026.01.28 | [CALLISTAENTERPRISE.SE](https://callistaenterprise.se)

CALLISTA

```
[2025-03-30] [CRITICAL] -SportAdmin- Data Breach Detected (1M records)
[2025-08-14] [CRITICAL] -Miljödata-  Ransomware Encryption Started
[2025-10-10] [WARNING]  -Vision-    Misusage of API NOT detected
[2025-10-10] [FATAL]    -svk.se -    Unknown Data Breach
```

```
$> YOU BEEN PWNED! TOTAL LEAK: > 2.000.000.000 RECORDS
```


HUR HAMNADE VI HÄR?



MANUELL HASTIGHET



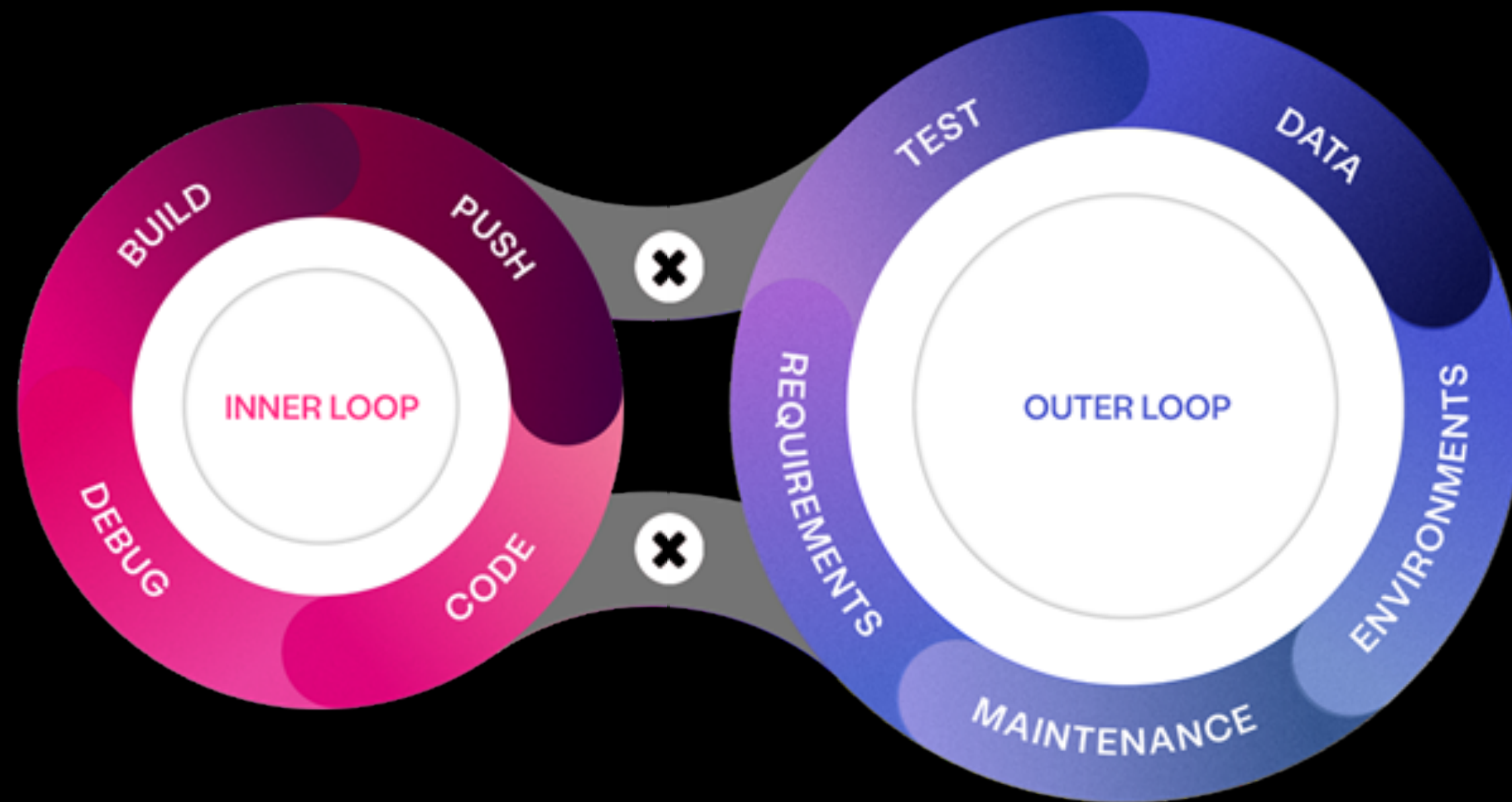
AI-HASTIGHET



AGENDA



LÅGFRIKTIONSPROCESSER



- Utvecklingsmiljön
- Repo
- Krav

Gör inte säkerhet till en stödprocess

UTVECKLARENS DAGLIGA 3

- Validera ALLT!

//  *BAD CODE: String Concatenation*

// "Lätt att skriva, lätt att hacka"

// Sårbar för SQL Injection om input är: "admin' --"

```
String query = "SELECT * FROM accounts WHERE id = " + userInput + " ";  
statement.execute(query);
```

//  *GOOD CODE: Parameterisation*

// "Säkert by default"

// Databasen hanterar input som data, aldrig som kod.

```
String query = "SELECT * FROM accounts WHERE id = ?";  
PreparedStatement stmt = conn.prepareStatement(query);  
stmt.setString(1, userInput);  
stmt.executeQuery();
```


UTVECKLARENS DAGLIGA 3

- Minsta Möjliga Privilegium

 *BAD CONFIG: Root Access*
Applikationen kör som root/sa.
Om kod injiceras kan angriparen droppa
tabeller eller läsa andras data.

datasource:
 url: jdbc:mysql://db-prod:3306/customers
 username: root
 password: superSecretPassword!

 *GOOD CONFIG: Service Account*
Applikationen får BARA göra INSERT och SELECT
på specifika tabeller.
Den kan inte göra DROP TABLE eller läsa admin-tabeller.

datasource:
 url: jdbc:mysql://db-prod:3306/customers
 # Har endast GRANT SELECT, INSERT, UPDATE
 username: app_service_user
 password: limitedUserPassword!

UTVECKLARENS DAGLIGA 3

- Utdatakontroll och Spårbarhet

//  *BAD CODE: Blind Logging*

// "Bekvämt vid debug, katastrof vid intrång"

// User-objektets `.toString()` läcker personnummer och email i klartext till loggen.

```
catch (Exception e) {  
    log.error("Failed to update user: " + userObj, e);  
}
```

//  *GOOD CODE: Reference Logging*

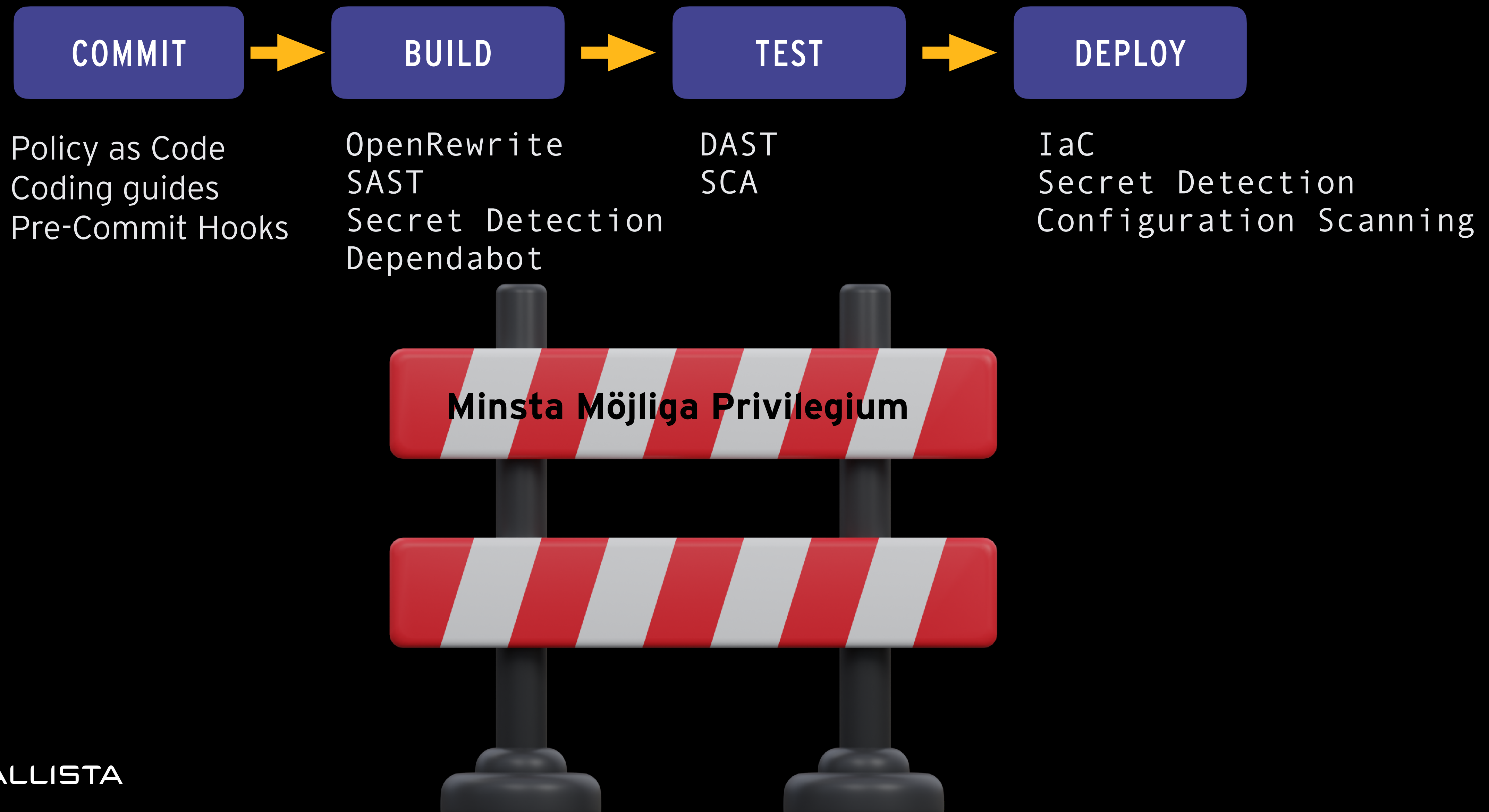
// "Spårbart men privat"

// Logga ID för att kunna felsöka, men håll känslig data borta.

```
catch (Exception e) {  
    log.error("Failed to update user with ID: {}", userObj.getId(), e);  
}
```

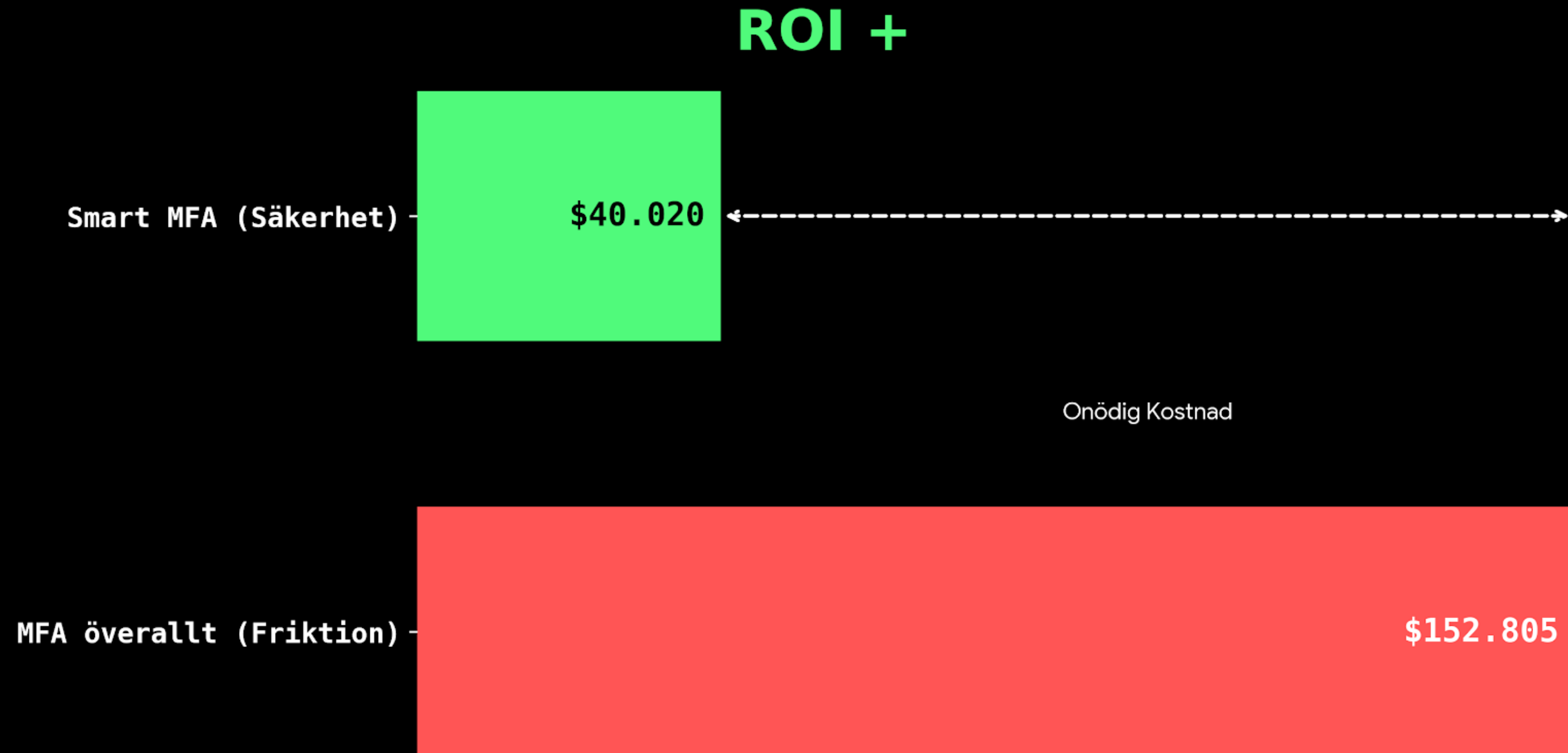

//  *GOOD CODE - NO CODE*

| CI/CD: AUTOMATISERADE SKYDDSRÄCKEN



| KVANTIFIERA SÄKERHET SOM EN INVESTERING

- Bluffa - Bottom Line Up Front (BLUF)



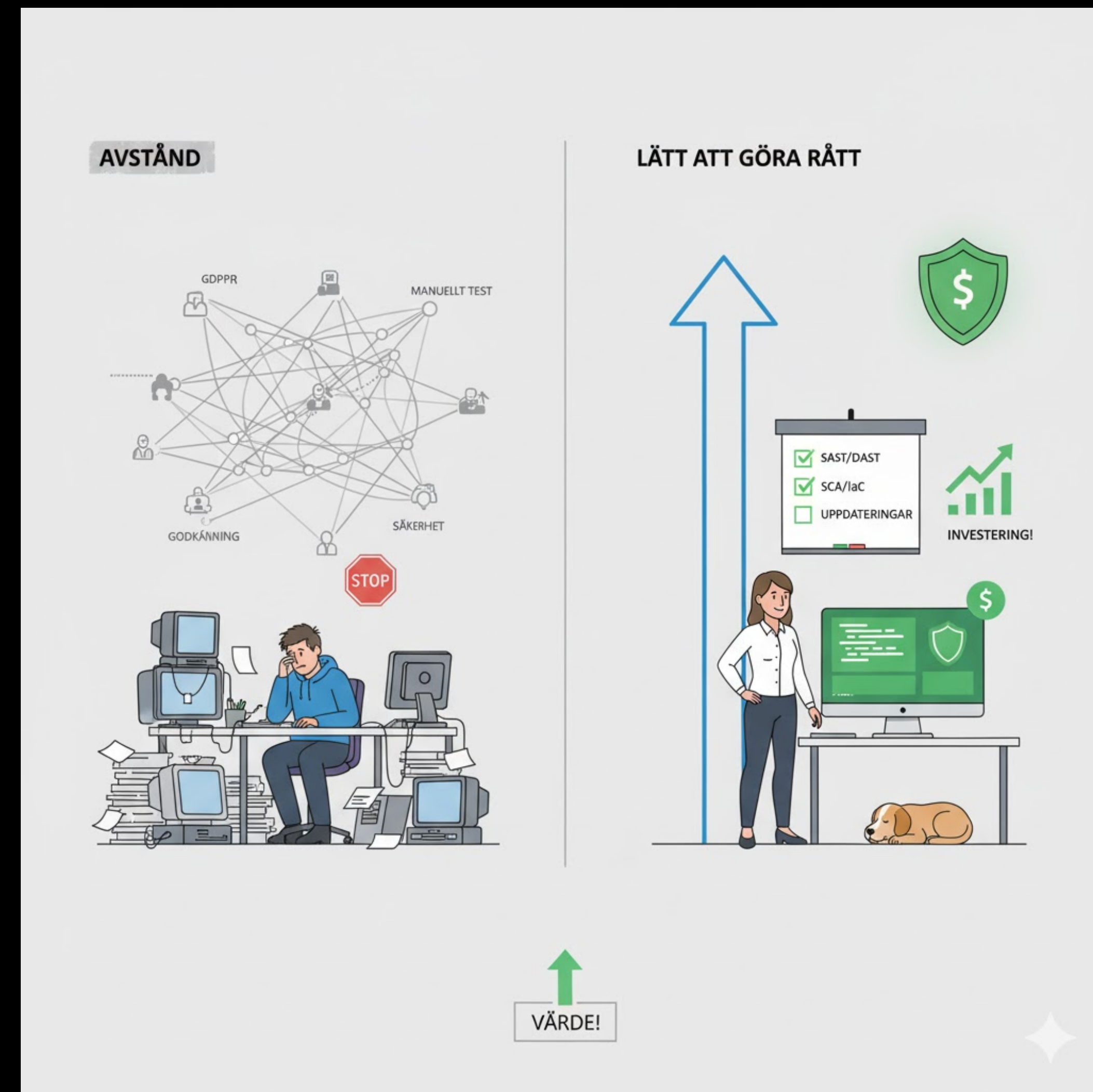
| HÄVSTÅNGSEFFEKTER

- Minskad Mean Time To Remediate - MTTR
 - Från 200+ till 60 dagar
- Förväntat kostnad för läckaget
- Kostnad för eskalering
- Minskad friktion mellan Dev-Sec-Ops



| SLUTSATS: INTEGRERA SÄKERHET SOM SERVICE

- Kostnad och friktion ökar med avståndet
 - Säkerhet tillhör kravställningen
- Säkra den Inre Loopen
 - Det ska vara lätt att göra rätt
- Inför Utvecklarens Dagliga 3
- Automatisera:
 - SAST/DAST
 - SCA, IaC
 - Uppdateringar, Dependabot
- Investering, inte kostad.





```
$> git commit -m "feat: install F1 brakes to ensure slow driving"
$> git amend -m "feat: brakes are to dare to drive fast"
```

```
$> git commit -m "security: calculating cost vs revenue"
$> git commit -m "feat: the reason for high revenue"
```

```
$> git commit -m "chore: invest in security (since 'git revert data-
breach' won't work)"
```



```
$> git switch q-and-a
```