T.I.S.P./T.P.S.S.E. Community Meeting 2025

Berlin, 04. - 05.11.2025

Threat Modeling in der (DevOps-) Softwareentwicklung und Secure Design

Dr. Christoph Niehoff

(TNG Technology Consulting)

TNG TECHNOLOGY CONSULTING

Threat Modeling in der (DevOps-)Softwareentwicklung und Secure Design

Christoph Niehoff

11/4/2025

T.I.S.P./T.P.S.S.E. Community Meeting 2025

Über mich

TNG TECHNOLOGY CONSULTING

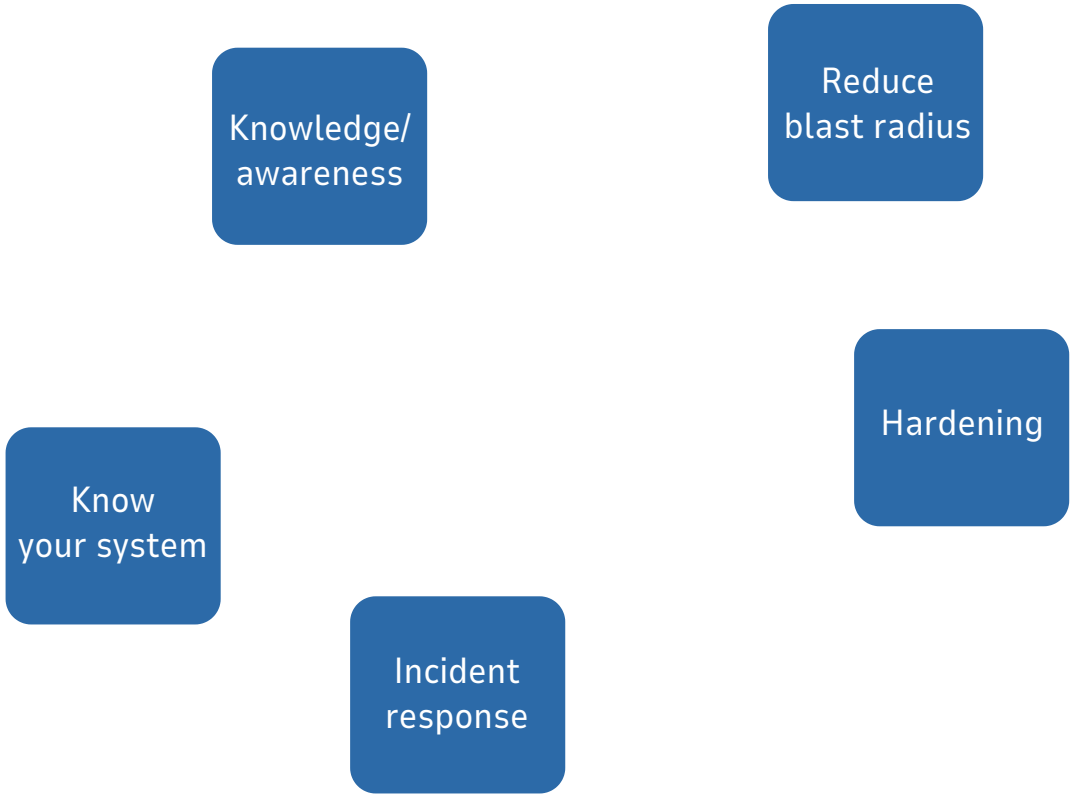
Christoph Niehoff

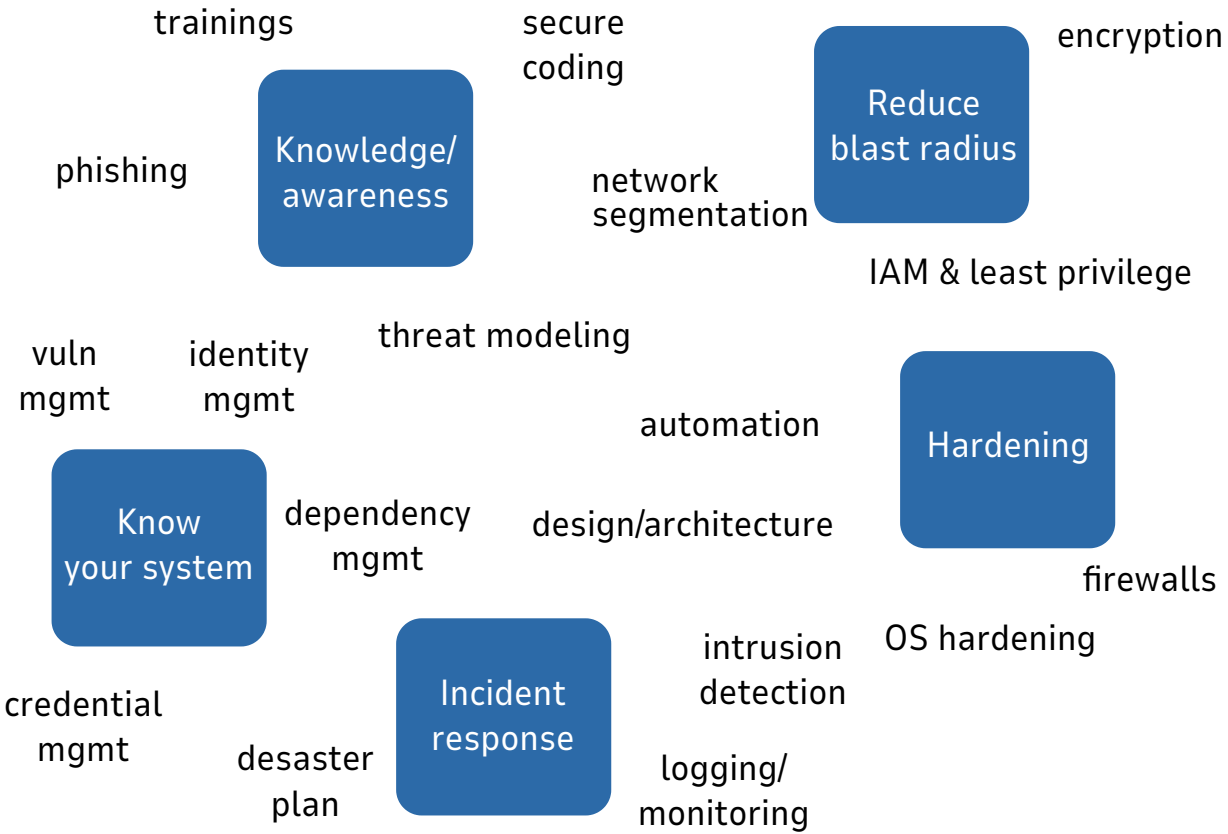
christoph.niehoff@tngtech.com

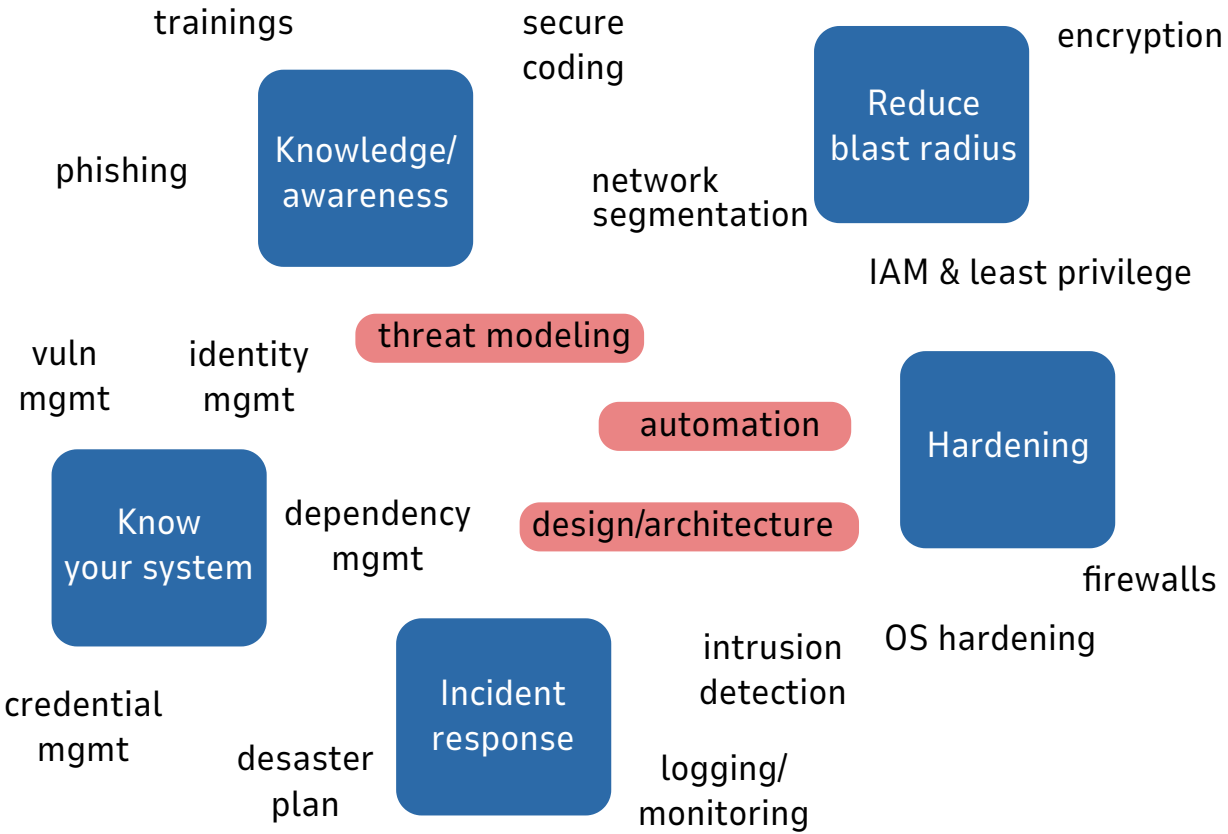
- Früher einmal Physiker
- Full-stack Entwickler
- DevOps engineer
- Ziel: Sichere Produkte bauen
- Project Lead von OWASP Cumulus

"Secure by Design" bedeutet Proaktivität

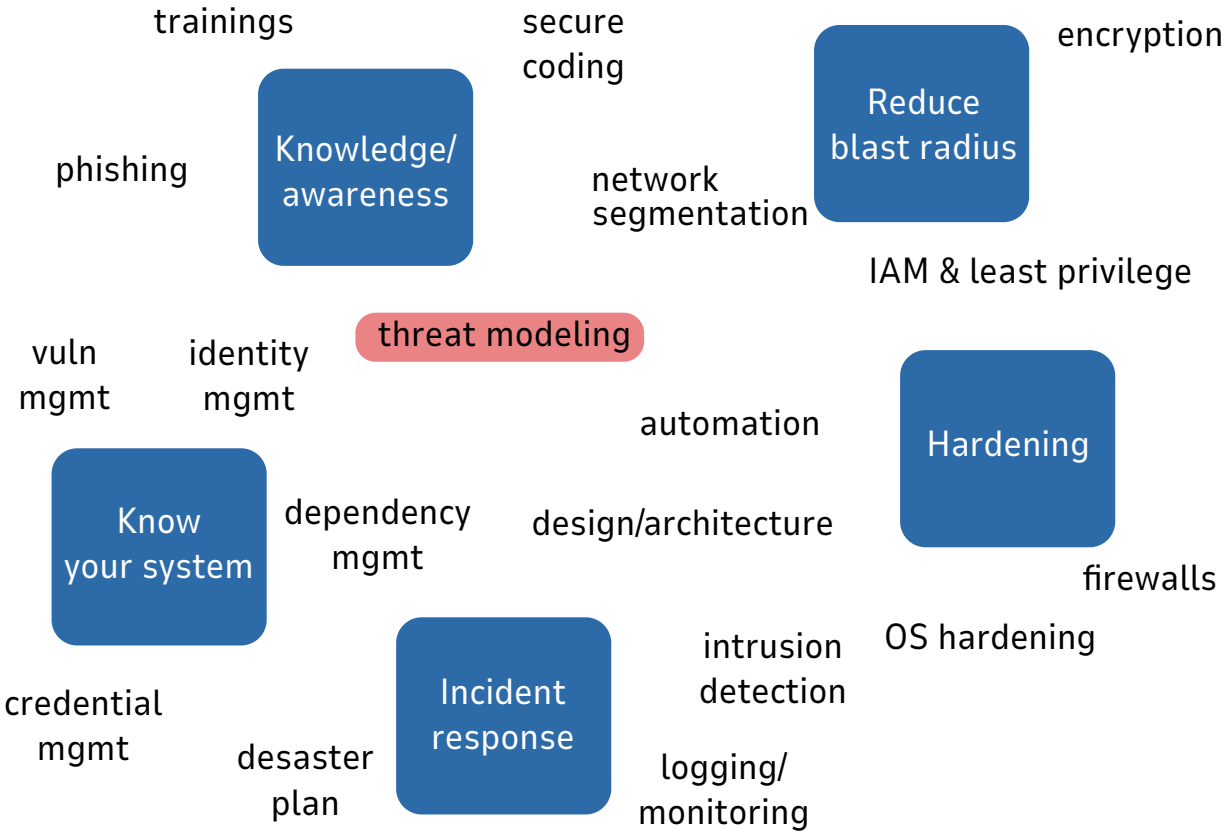
DevOps bedeutet Eigenverantwortung


Knowledge/
awareness
Reduce
blast radius
Hardening
Know
your system
Incident
response


trainings
secure coding
encryption
Knowledge/ awareness
Reduce blast radius
phishing
network segmentation
IAM & least privilege
threat modeling
vuln mgmt
identity mgmt
automation
Hardening
Know your system
dependency mgmt
design/architecture
firewalls
OS hardening
intrusion detection
Incident response
credential mgmt
desaster plan
logging/ monitoring


trainings
secure coding
encryption
Knowledge/ awareness
Reduce blast radius
phishing
network segmentation
IAM & least privilege
threat modeling
vuln mgmt
identity mgmt
automation
Hardening
Know your system
dependency mgmt
design/architecture
firewalls
OS hardening
intrusion detection
credential mgmt
Incident response
desaster plan
logging/ monitoring

TNG TECHNOLOGY CONSULTING



Niemand baut unsichere Software!

Niemand baut bewusst unsichere Software!

Niemand baut bewusst unsichere Software!

Erwarte das Unerwartete!

Threat Modeling

Die 4 Fragen von Adam Shostack

- Woran arbeiten wir?
- Was kann schiefgehen?
- Was können wir dagegen tun?
- Haben wir ausreichende Arbeit geleistet?

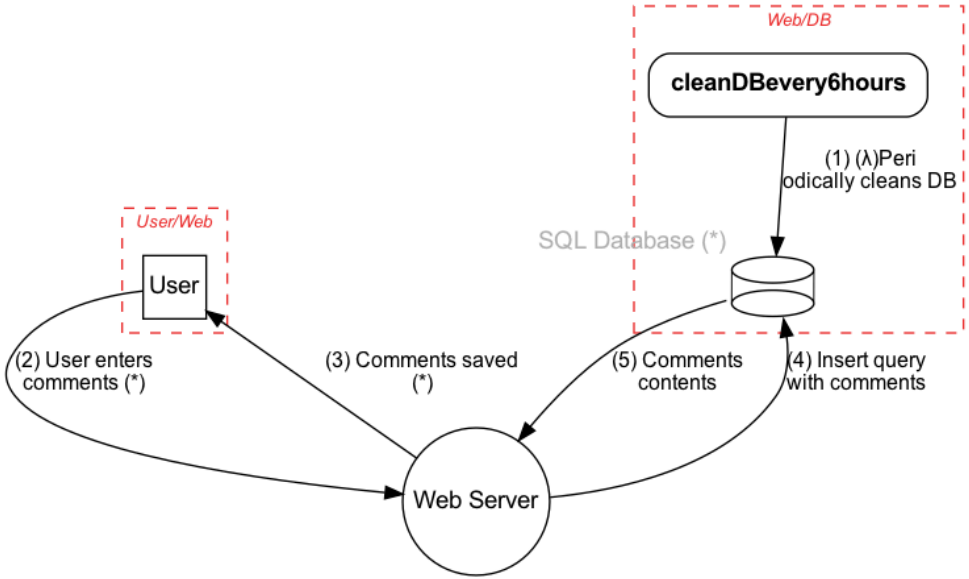
Woran arbeiten wir?

TNG TECHNOLOGY CONSULTING

Kenne das System

- Zeichne Architekturdiagramme
- Erfasse Abhängigkeiten
- Erfasse Zugänge und Zugriffe
- Kläre Verantwortlichkeiten
- Dokumentiere Requirements

Das allein ist schon sehr wertvolle Information, die zumeist implizit ist!



Quelle: https://github.com/izar/pytm

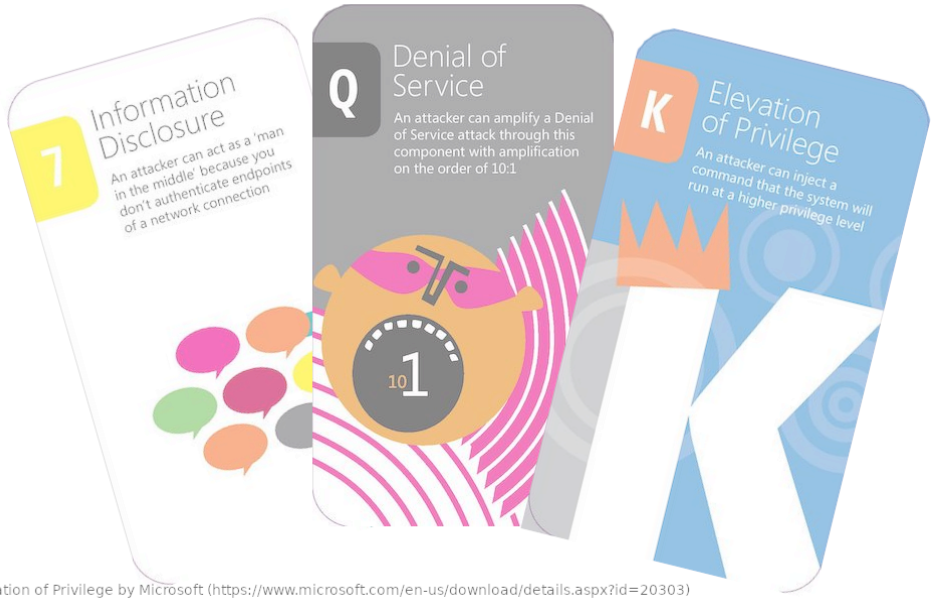
Was kann schiefgehen?

- 🎯 Herzstück vom Threat Modeling
- **Jeder** sollte Anteil haben an der Sicherheit (auch die Newbies)
  - => braucht Niederschwelligkeit
  - => muss mitreißend sein
  - => braucht Struktur
- 💡 Gamification als Kartenspiel (*großartige Idee von Adam Shostack*)

Was kann schiefgehen?

TNG TECHNOLOGY CONSULTING

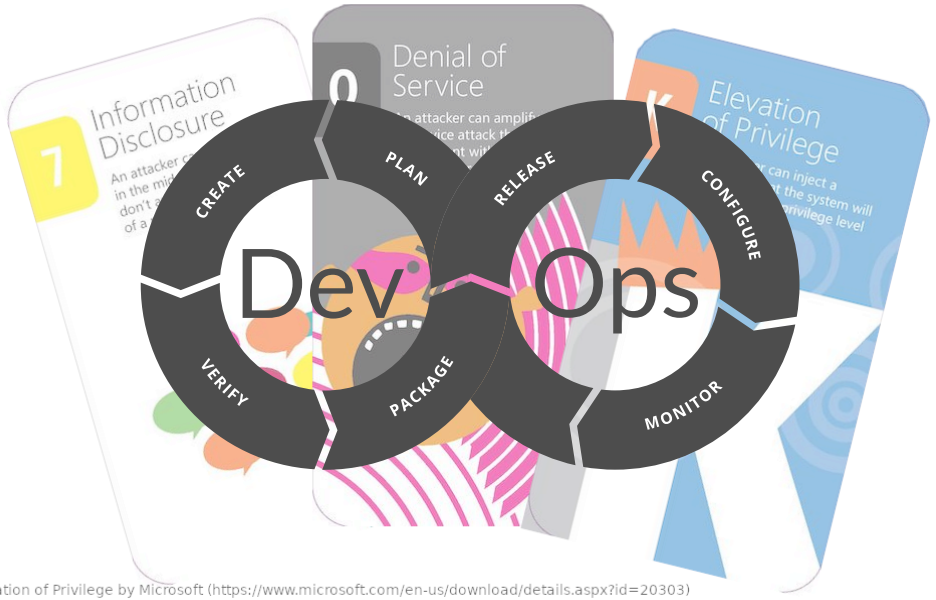
- 🎯 Herzstück vom Threat Modeling
- **Jeder** sollte Anteil haben an der Sicherheit (auch die Newbies)
  - => braucht Niederschwelligkeit
  - => muss mitreißend sein
  - => braucht Struktur
- 💡 Gamification als Kartenspiel (*großartige Idee von Adam Shostack*)



Was kann schiefgehen?

TNG TECHNOLOGY CONSULTING

- 🎯 Herzstück vom Threat Modeling
- **Jeder** sollte Anteil haben an der Sicherheit (auch die Newbies)
  - => braucht Niederschwelligkeit
  - => muss mitreißend sein
  - => braucht Struktur
- 💡 Gamification als Kartenspiel (*großartige Idee von Adam Shostack*)



Was kann schiefgehen?

- 🎯 Herzstück vom Threat Modeling
- **Jeder** sollte Anteil haben an der Sicherheit (auch die Newbies)
  - => braucht Niederschwelligkeit
  - => muss mitreißend sein
  - => braucht Struktur
- 💡 Gamification als Kartenspiel (*großartige Idee von Adam Shostack*)

Was kann schiefgehen?

- 🎯 Herzstück vom Threat Modeling
- **Jeder** sollte Anteil haben an der Sicherheit (auch die Newbies)
  - => braucht Niederschwelligkeit
  - => muss mitreißend sein
  - => braucht Struktur
- 💡 Gamification als Kartenspiel (*großartige Idee von Adam Shostack*)

Elevation of Privilege by Microsoft (https://www.microsoft.com/en-us/download/details.aspx?id=20303) licensed under CC-BY-3.0 http://creativecommons.org/licenses/by/3.0/us/

Was kann schiefgehen?

TNG TECHNOLOGY CONSULTING

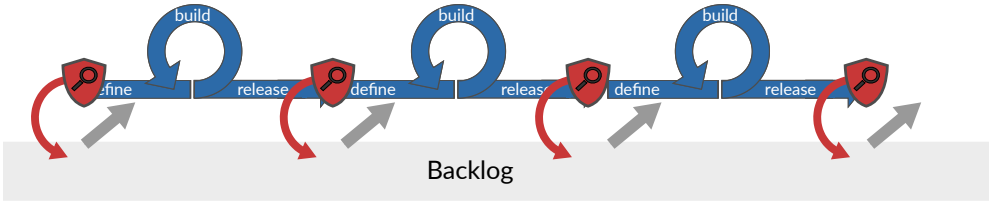
OWASP Cumulus

- Threat modeling the clouds!
- Threat model the Ops of DevOps!
- Design-Entscheidungen:
  - Vendor-Unabhängigkeit
  - Technologie-Unabhängigkeit
  - Allgemein genug, um Diskussionen anzuregen
  - Konkret genug, um zu helfen

Einbettung ins Agile

Software ist ein bewegliches Ziel

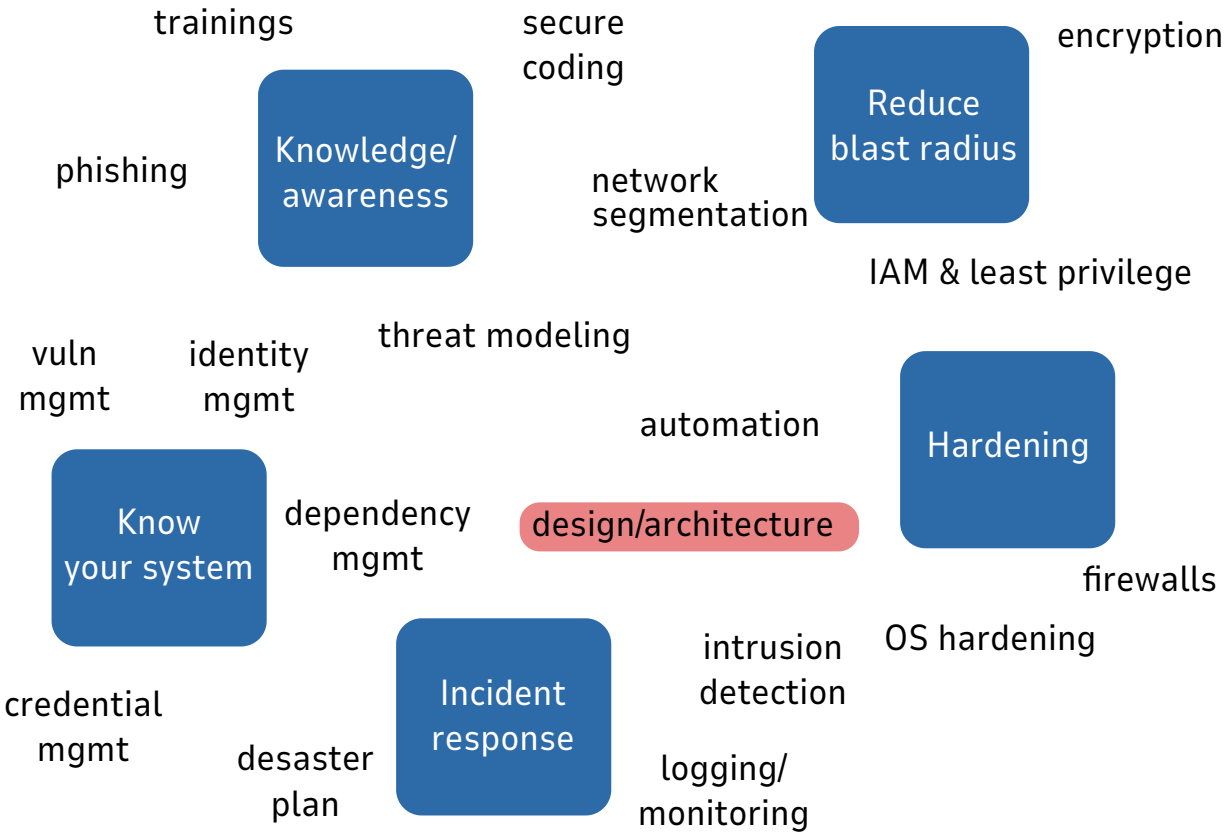
- Sicherheitsprozess muss Schritt halten mit der Entwicklung
- Threat model everything
  - Frage andauernd:
    "Was kann schiefgehen?"
  - Und rede darüber!
  - Iterativ die Sicherheit erhöhen



Continuous Threat Modeling

Warum Threat Modeling?

- **Denke nach über Security**
  - Vermeide Probleme, bevor sie implementiert werden
  - Verbessere Design und Resilienz
  - Eigenverantwortliche Entwicklungsteams
- **Rede über Security**
  - Verteile Sicherheitswissen
  - Schaffe ein Sicherheitsbewusstsein
- Eine der günstigeren Sicherheitsmaßnahmen!


trainings
secure coding
encryption
Knowledge/ awareness
Reduce blast radius
phishing
network segmentation
IAM & least privilege
threat modeling
vuln mgmt
identity mgmt
automation
Hardening
Know your system
dependency mgmt
design/architecture
firewalls
OS hardening
intrusion detection
Incident response
credential mgmt
desaster plan
logging/ monitoring

Secure by Design

> Management

> Businesslogik

> Architektur

> Code Design

Saltzer/Schroeder, 1975

- Least Privilege
- Fail-Safe Defaults
- Economy of Mechanism
- Open Design
- Separation of Privilege
- Least Common Mechanism
- Psychological Acceptability
- Complete Mediation
- Work Factor
- Compromise Recording

Berechtigungen

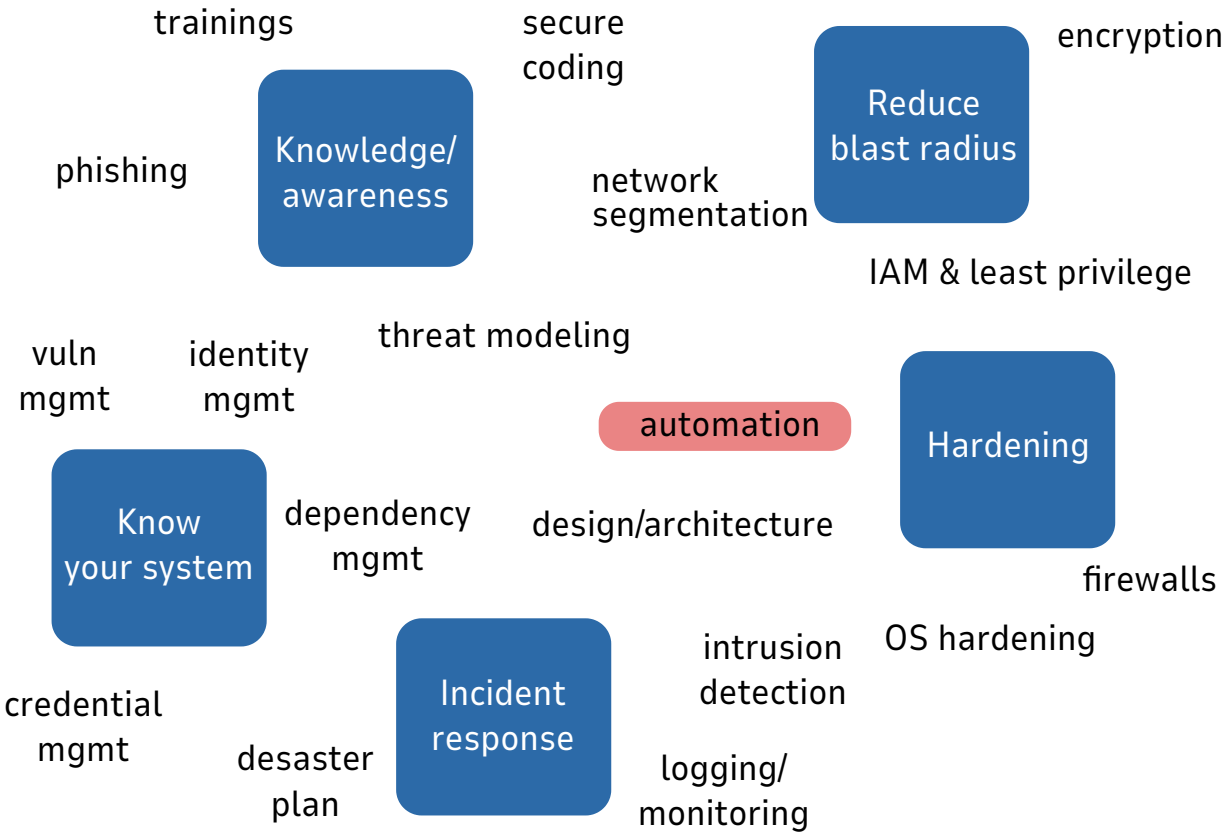
- Erteilen
  - **Least Privilege**
    - Patterns: "Just Enough Privilege", "Just in Time Privilege", ...
    - Anti-Patterns: "Confused Deputy", "Backflow of Trust", "Implicit Trust", ...
  - **Separation of Privilege**
    - Könnte man heute auch "Defense in Depth" nennen
- Durchsetzen
  - **Complete Mediation**
    - Könnte man heute auch "Zero Trust" nennen

Software Architektur

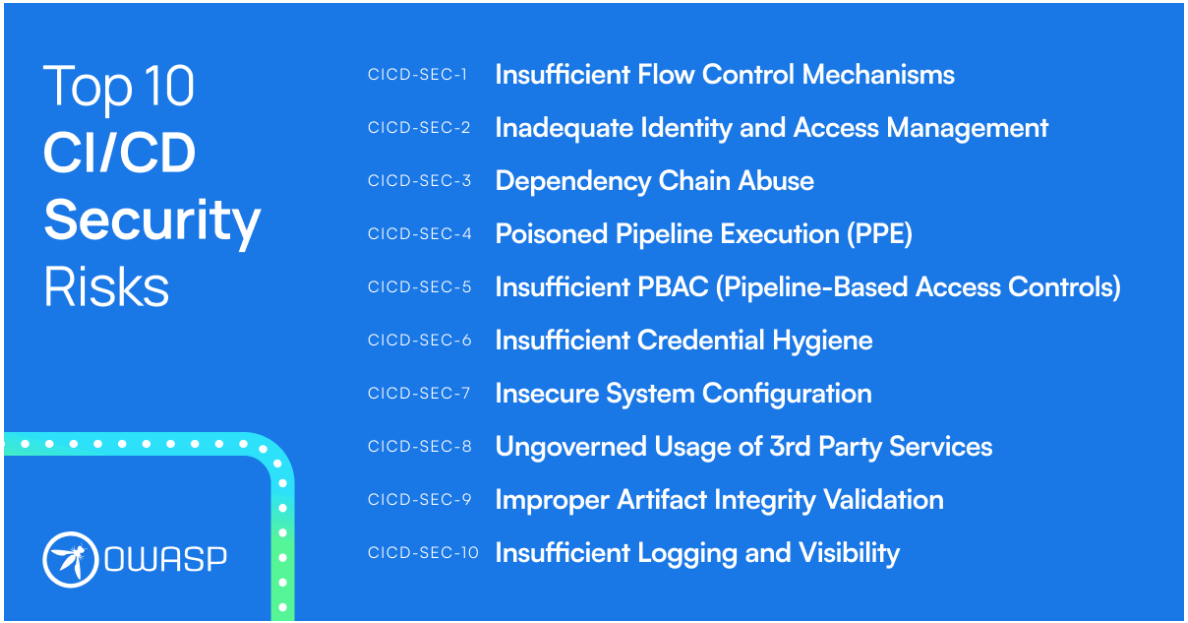
- **Economy of Mechanism**
  - Könnte man heute auch "KISS" nennen
- **Least Common Mechanism**
  - Keine zu starke Verallgemeinerung
- **Fail-safe Defaults**
  - Könnte man heute auch "Deny by Default" nennen
- **Open Design**
  - Könnte man heute auch "No Security by Obscurity" nennen

Mitigationen

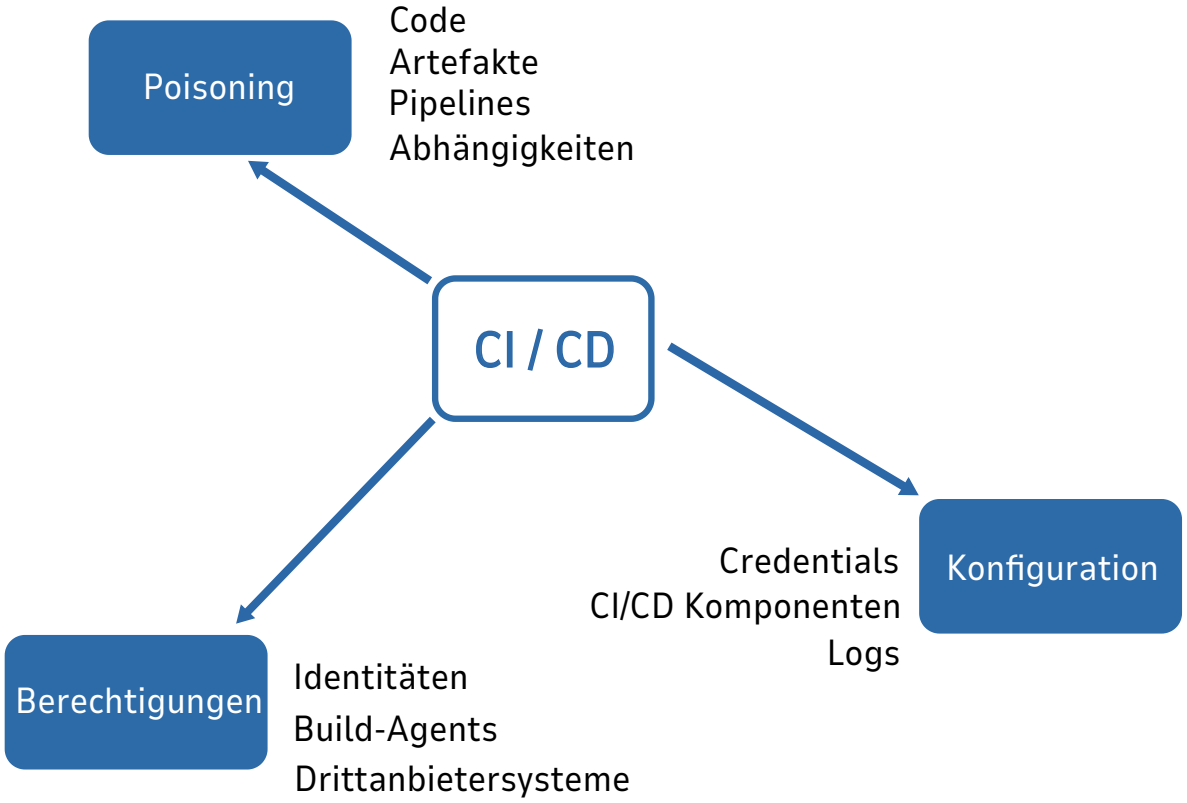
- Was?
  - **Work Factor**
    - Könnte man heute auch "Reasonable Security" nennen
- Wie?
  - **Psychological Acceptability**
    - Könnte man heute auch "User-centric Security" nennen
  - **Compromise Recording**
    - Könnte man heute auch "Transparenz" nennen


trainings
secure coding
encryption
Knowledge/ awareness
Reduce blast radius
phishing
network segmentation
IAM & least privilege
threat modeling
vuln mgmt
identity mgmt
automation
Hardening
Know your system
dependency mgmt
design/architecture
firewalls
OS hardening
intrusion detection
Incident response
credential mgmt
desaster plan
logging/ monitoring

Automatisierung a.k.a CI/CD



Quelle: https://github.com/cider-security-research/top-10-cicd-security-risks/blob/main/Images/risks.png


Poisoning
Code
Artefakte
Pipelines
Abhängigkeiten
CI / CD
Credentials
CI/CD Komponenten
Logs
Konfiguration
Berechtigungen
Identitäten
Build-Agents
Drittanbietersysteme

Automatisierung - Die Helle Seite

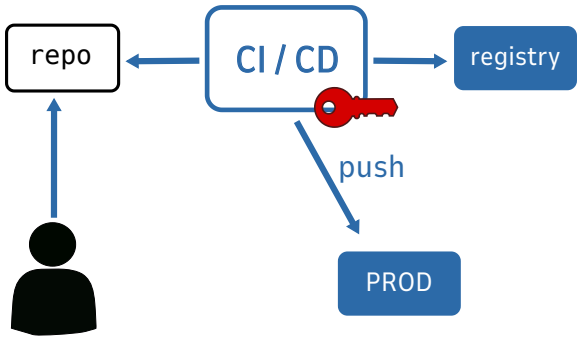
TNG TECHNOLOGY CONSULTING

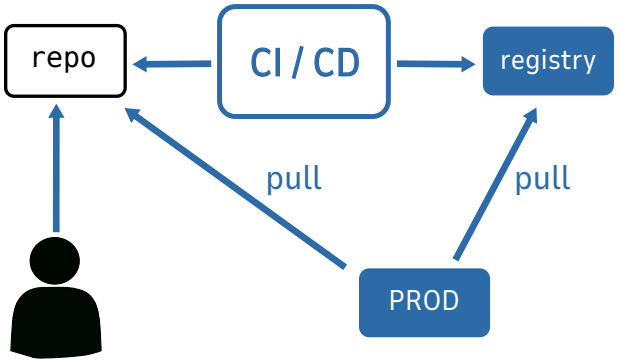
- Reduziere den *"Problemfaktor Mensch"*
- System wird dokumentiert und verständlich
- Übertrage Rechte: Es braucht keine Deploy-Berechtigungen, um deployen zu können
  - Sehr eng definierte Berechtigungen

Automatisierung - Die Dunkle Seite

- Eine Form der **Rechteausweitung**
  - Wer Code pushen kann, kann auch die Produktivumgebung deployen
- Kann ein **single point of failure** sein
  - Berechtigungen, Zugänge, Credentials, … alles am gleichen Platz

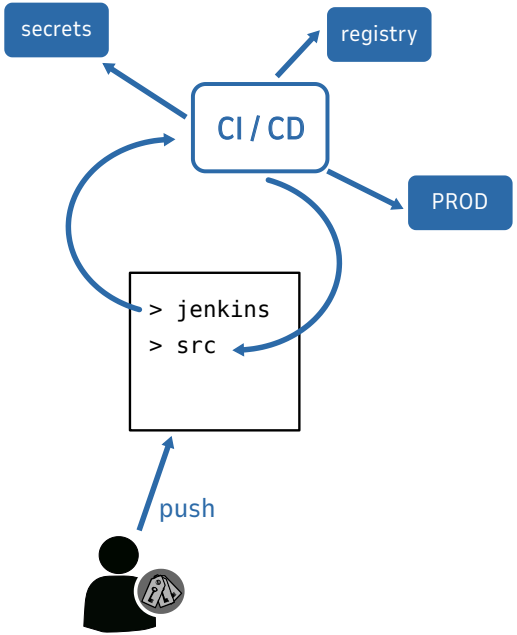
Push vs. Pull deployment





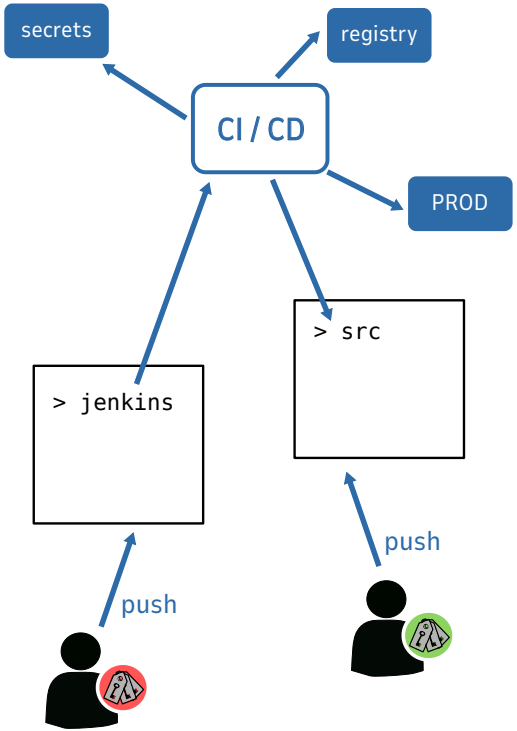
Pipeline Poisoning - Berechtigungen

- Typisches Layout eines Repositories:
  - Code und CI/CD Pipelines zusammen im gleichen Repository
  - das macht so auch Sinn...
- Wie kann man die Pipeline manipulieren?
  - Direkt den Code abändern? -> Schutz: 4-Augen-Prinzip
  - Pipelines für PRs starten? -> möglich, wenn CI fehlkonfiguriert
    - Das ist eine starke Rechteausweitung
      - "PR erstellen" bedeutet dann "Zugriff auf PROD"



Pipeline Poisoning - Berechtigungen

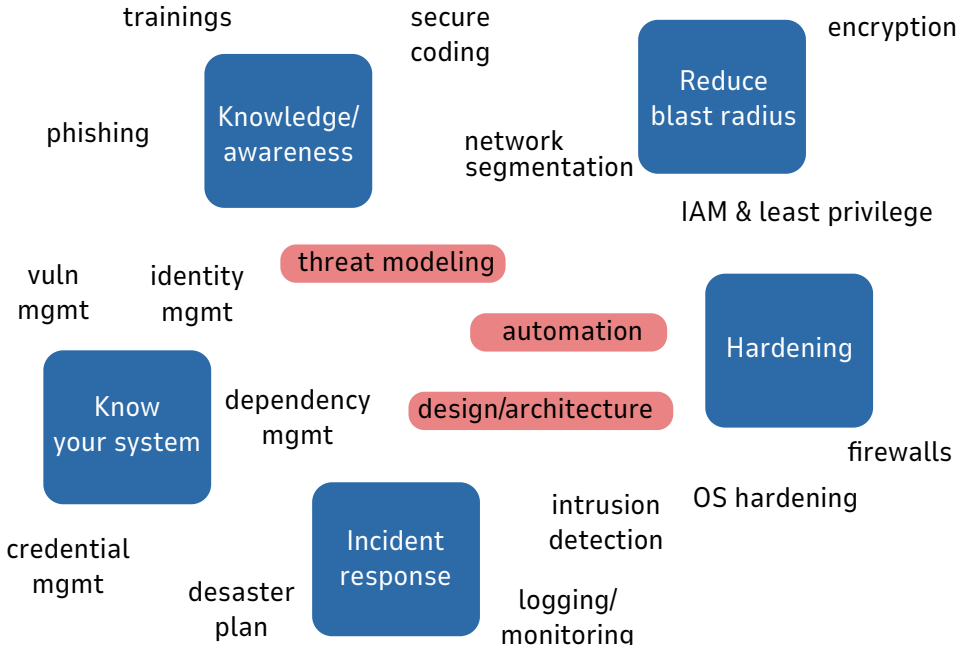
- Mögliche Lösung
- Aufteilen in verschiedene Repos, um unterschiedliche Berechtigungen durchzusetzen
  - Ich plädiere nicht dafür
  - ABER: Verschiedener Code im selben Repo kann verschiedene Sicherheitsimplikationen haben
  - Das muss beachtet werden -> Kleinster Gemeinsamer Nenner sollte die Sicherheit der kritischsten Teile sein



Wie wird meine Softwareentwicklung "Secure by Design"?

TNG TECHNOLOGY CONSULTING

- Wissen aufbauen
  - System
  - Probleme
- Fehler machen schwerer machen
  - Automatisierung
  - Komplexität reduzieren, Vereinheitlichung
  - Least Privilege
- Probleme vermeiden
  - Gefahren vorhersehen
  - Proaktiv mitigieren



Takeaways

- Design kann große Sicherheitskonsequenzen haben
- Proaktive Designentscheidungen können ganze Angriffsklassen vermeiden
- **Threat Modeling** ist eine zentrale Maßnahme
  - Probleme vermeiden
  - Das System verstehen
  - Wissen aufbauen und verteilen
- Nicht nur ein technisches Thema
  - Braucht Rückendeckung durch Management

TNG TECHNOLOGY CONSULTING

Vielen Dank!

Gibt es Fragen?

Christoph Niehoff
Senior Consultant @ TNG
christoph.niehoff@tngtech.com