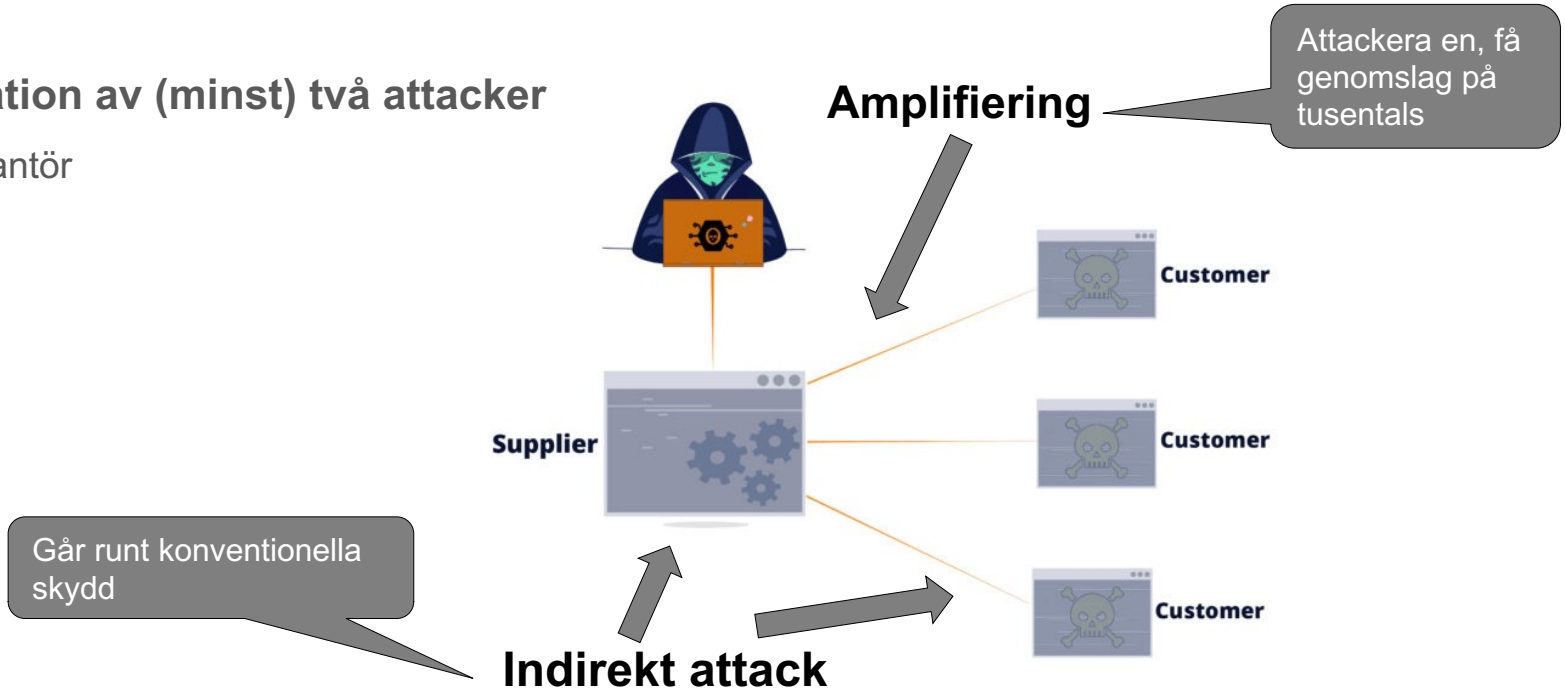
debricked

Hantera och förstå supply chain-attacker

Egenskaper för en supply chain-attack

Kombination av (minst) två attacker

1. Leverantör
2. Kund



Stort genomslag

Solarwinds, Dec 2020 — Proprietär kod

- **Leverantör**: Access till nätverk, infiltrerade byggprocessen för att inkludera skadlig kod
- **Kund**: Laddade ner (korrekt signerad) uppdatering av mjukvara. Stöld av data.
- 18000 kunder installerade uppdateringen. 425 av Fortune 500 påverkades

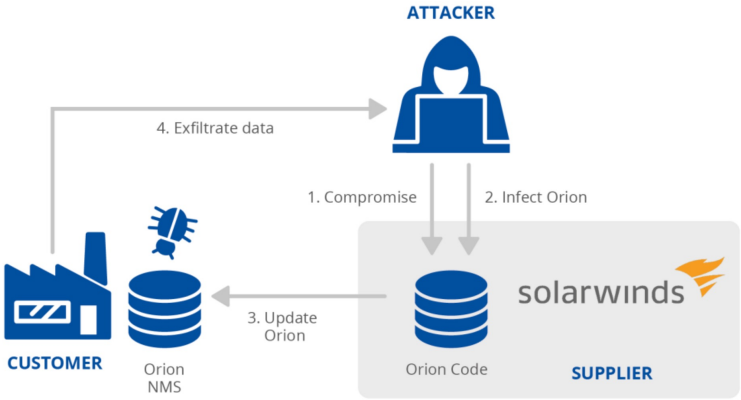


Bild från: *Enisa threat landscape for supply chain attacks, 2021*

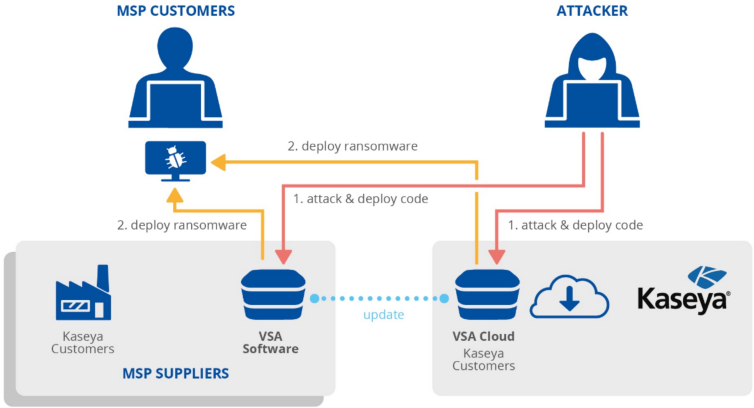
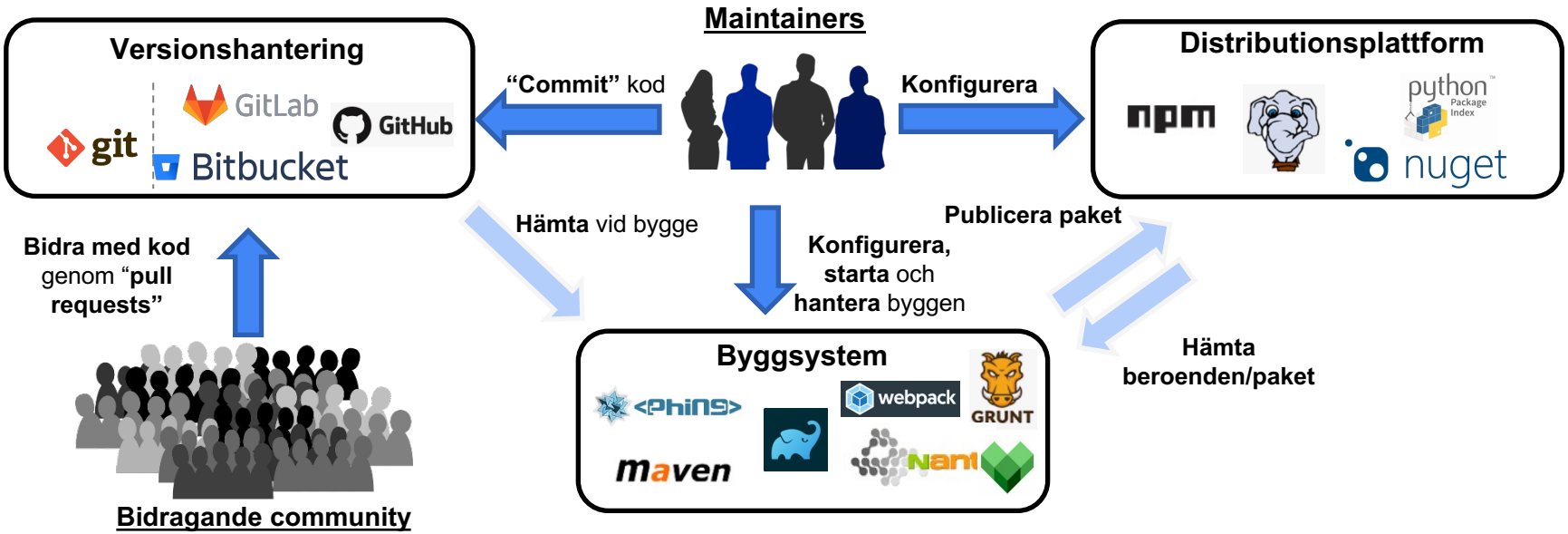


Bild från: *Enisa threat landscape for supply chain attacks, 2021*

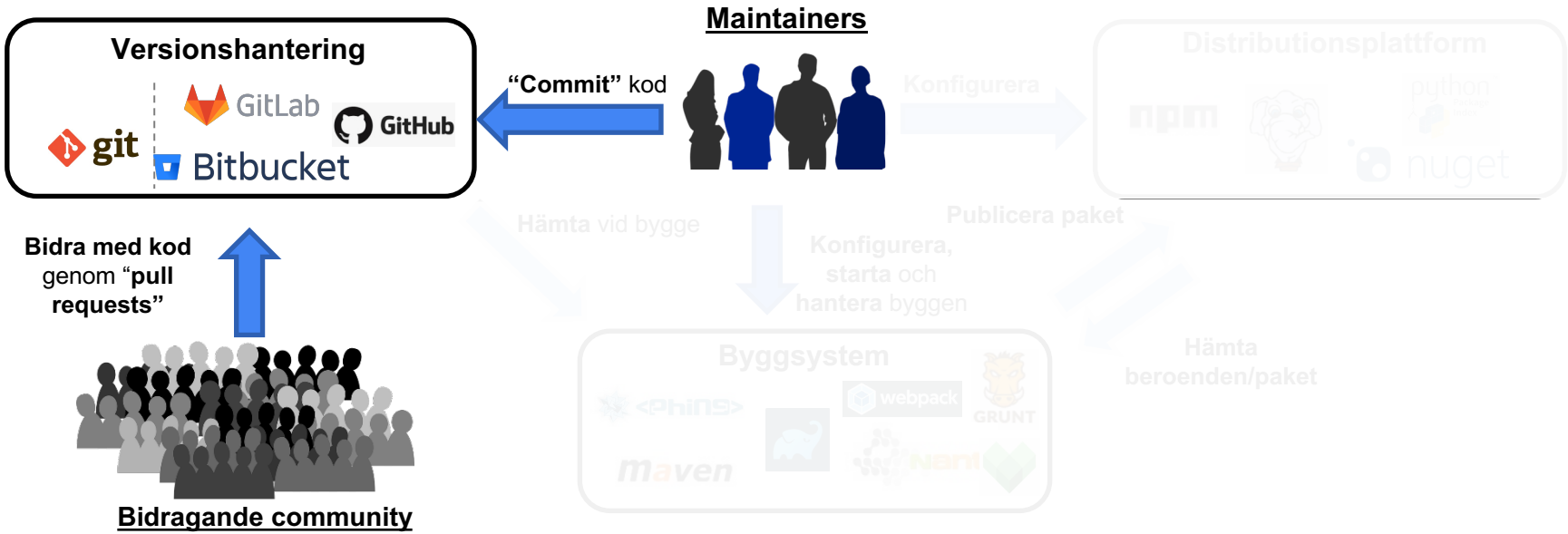
Kaseya, Juli 2021 — Proprietär kod

- **Leverantör**: Sårbarhet i Keseyas system som tillåta attackerare att uppdatera mjukvara som användes av kunder
- **Kund**: Mjukvara uppdaterades, vilken innehöll ransomware
- I Sverige känd som “coop-attacken”

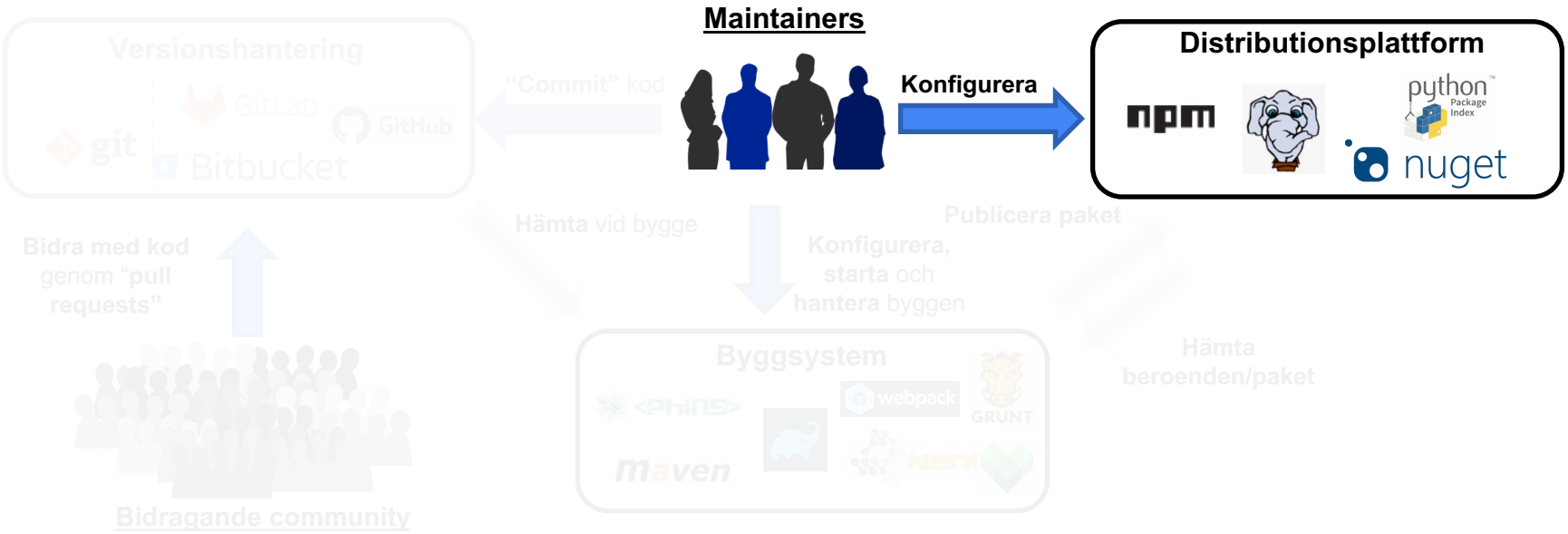
Ekosystem för öppen källkod



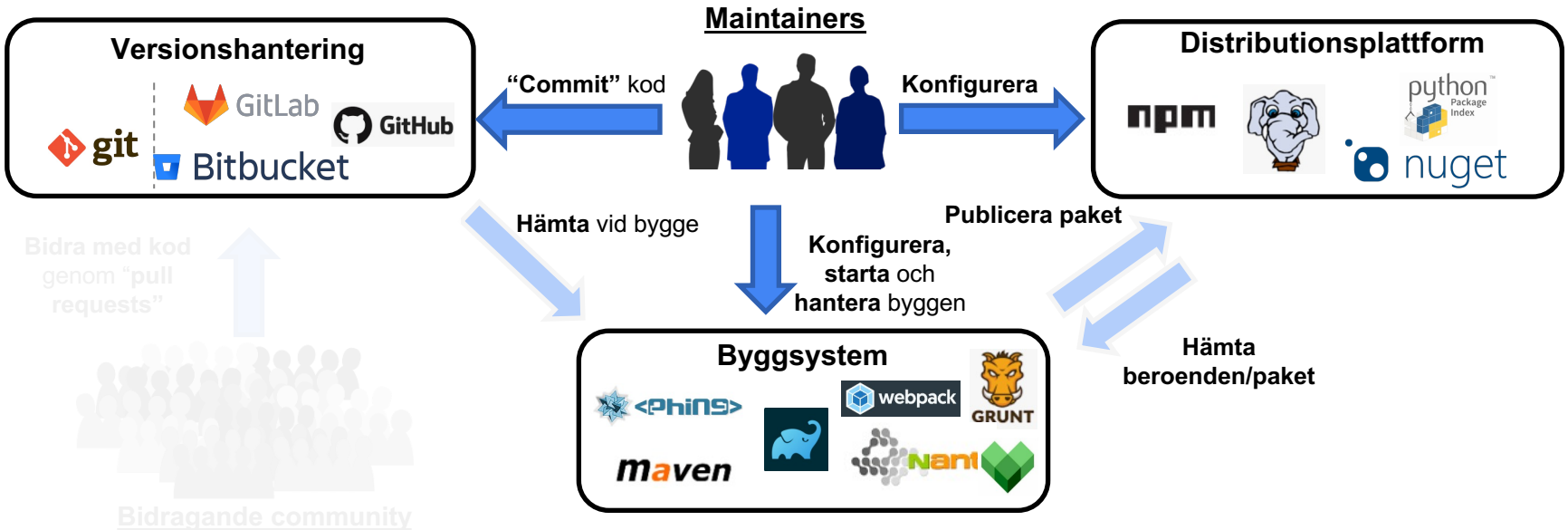
Ekosystem för öppen källkod



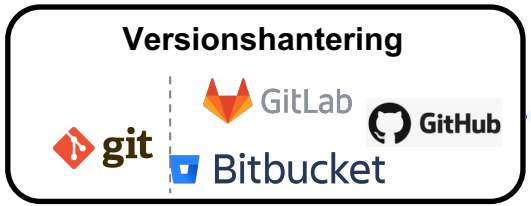
Ekosystem för öppen källkod



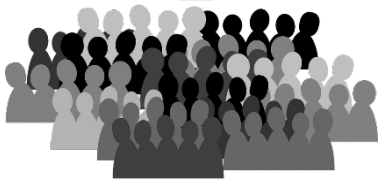
Ekosystem för öppen källkod



Attacker mot ekosystemet för öppen källkod



Bidra med kod genom “**pull requests**”

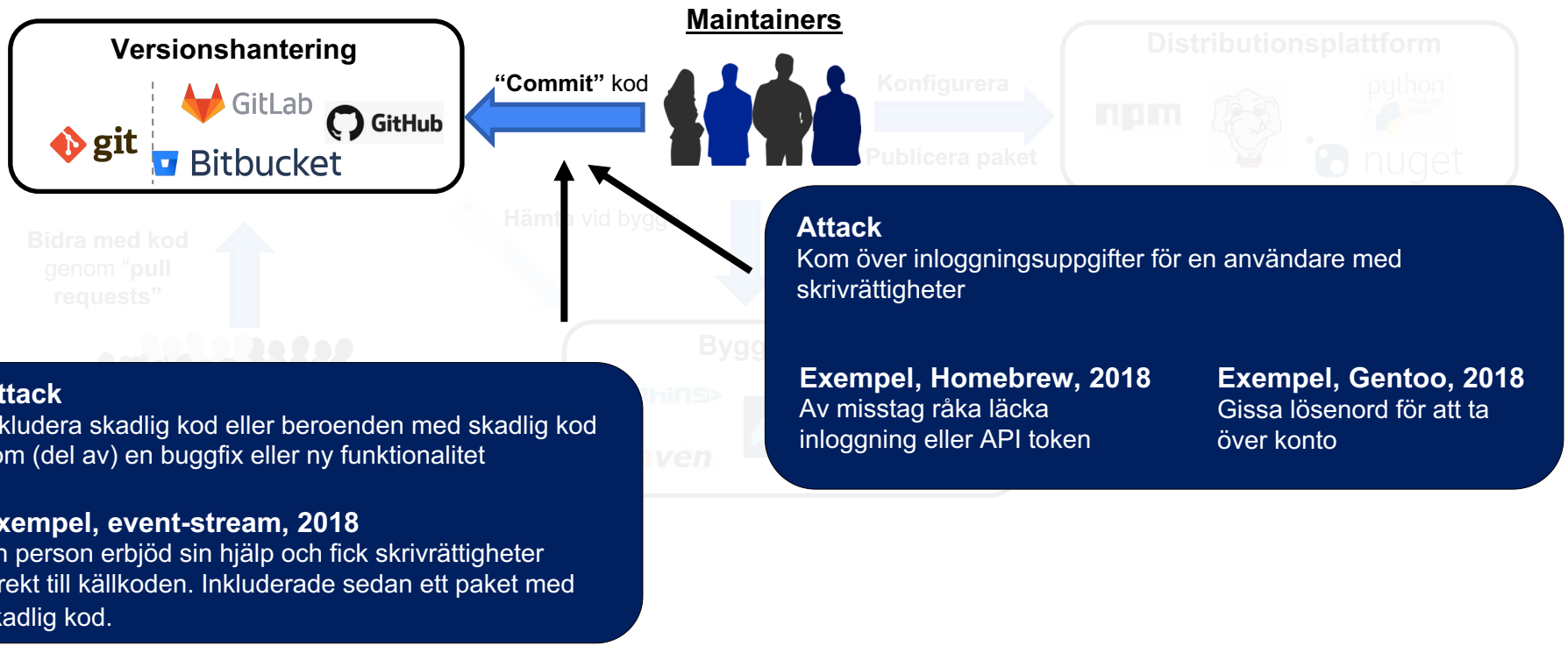


Bidragande community

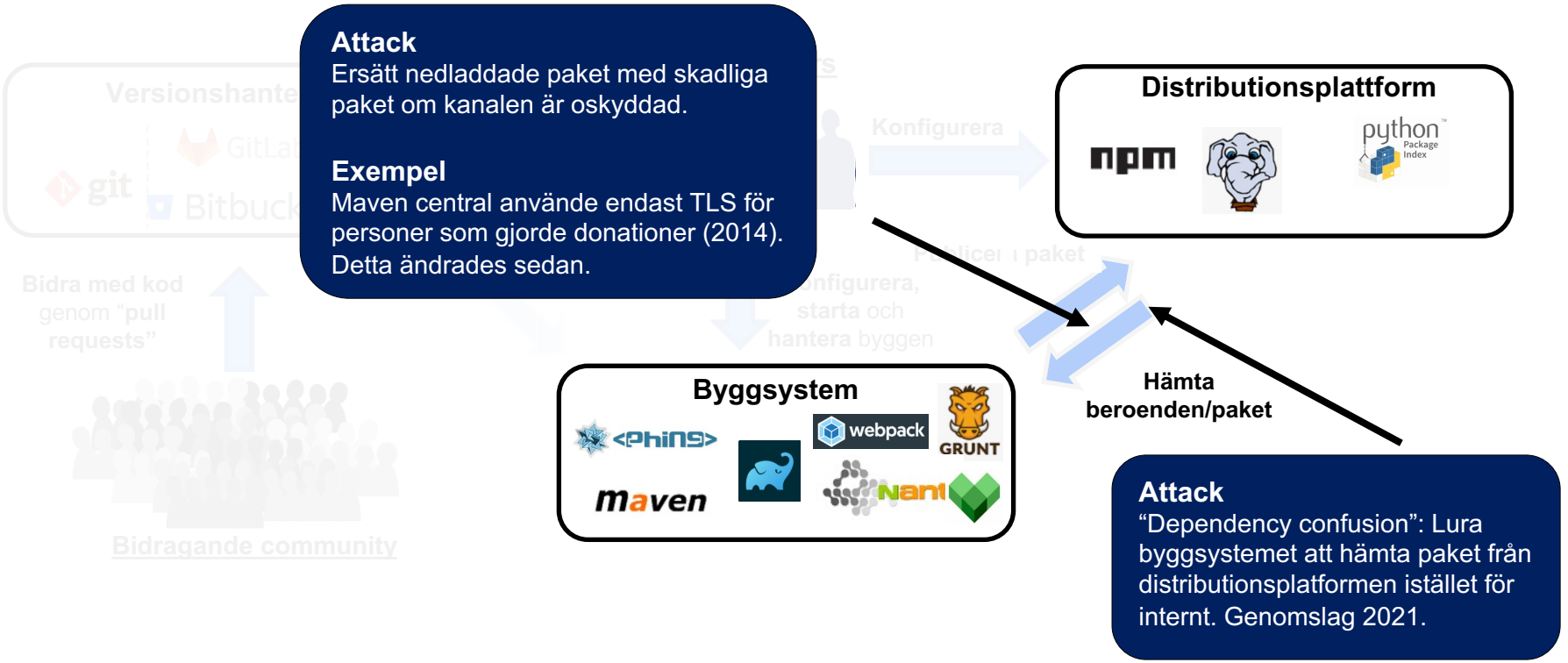
Attack
Inkludera skadlig kod eller beroenden med skadlig kod som (del av) en buggfix eller ny funktionalitet

Exempel, 2021
University of Minnesota bidrog med kod till Linuxkärnan som innehöll ett flertal sårbarheter.

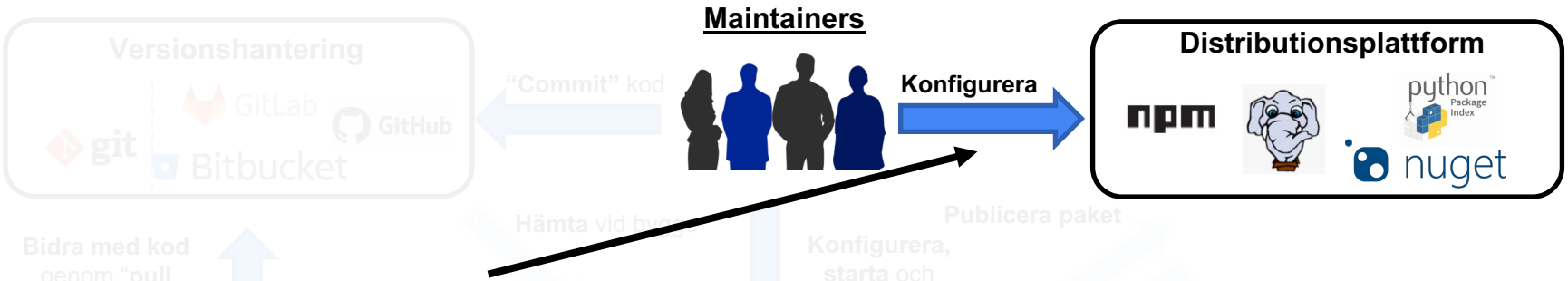
Attacker mot ekosystemet för öppen källkod



Attacker mot ekosystemet för öppen källkod



Attacker mot ekosystemet för öppen källkod



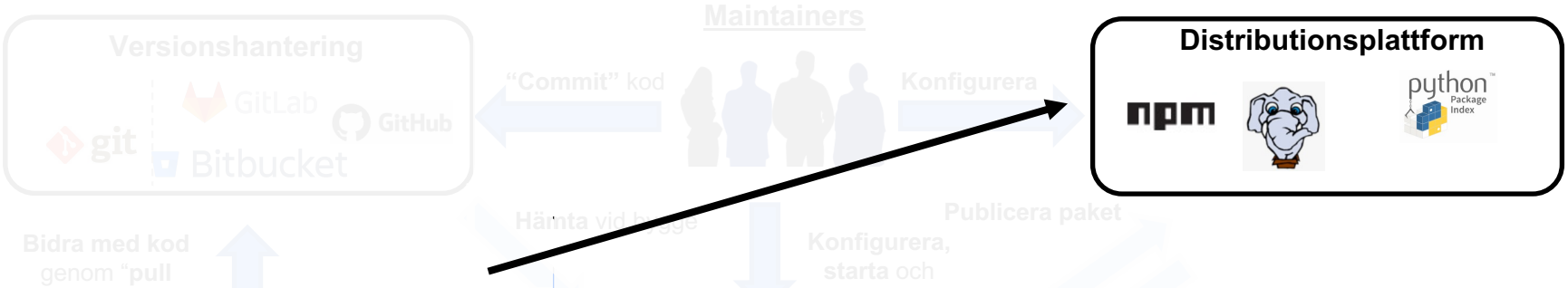
Attack
Kom över inloggningsuppgifter för en användare med skrivrättigheter

Exempel, 2017
Gissade och läckta lösenord tillät access till 14% av alla paket på npm. 54% inräknat indirekta beroenden.

Exempel, eslint, 2018
Lösenord läckt och sedan återanvänt.

Exempel, UAParser.js, 2021
Lösenord läckt och skadliga versioner laddades upp på npm.

Variant: Typosquatting



Attack
Skapa nya skadliga paket med liknande namn som vanliga paket

Exempel, 2017
Slovakisk säkerhetsorganisation identifierade 10 skadliga Pythonpaket.

urllib3 → urllib
urllib3 → urlib3

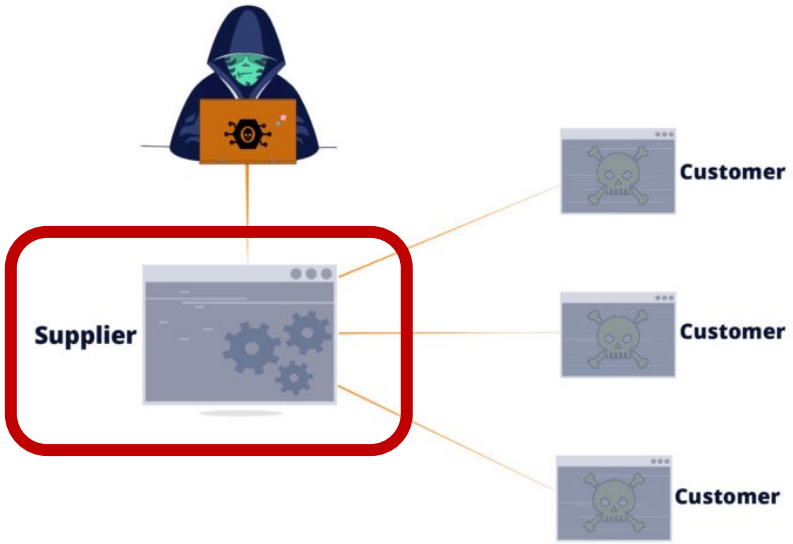
Exempel, 2017
38 paket från npm raderades.

sqlite → sqlite.js
sqlite → sqliter

Positiva aspekter

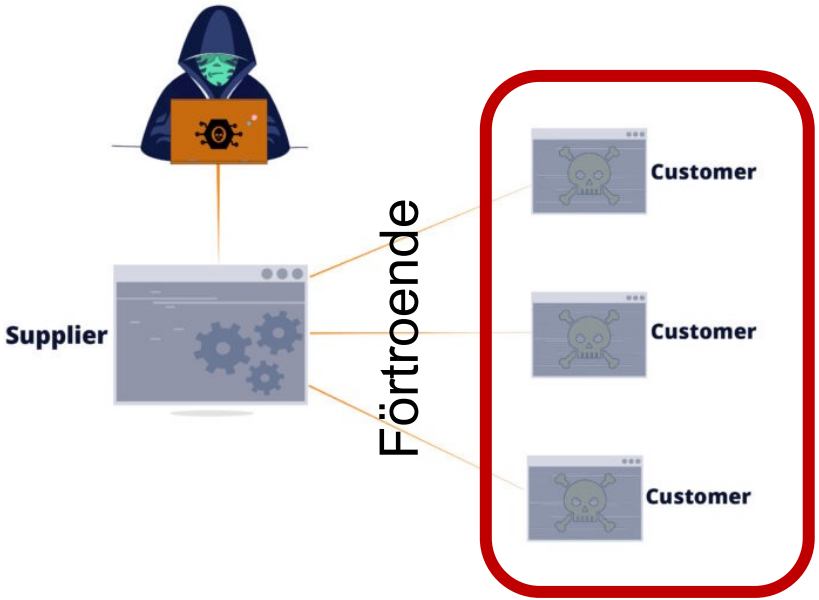
Bättre skydd för “supplier” till följd av dessa attacker

- Förbättrade policys för lösenord
- Tvåfaktorsautentisering
- Säkra kanaler



Hur skyddar man sig?

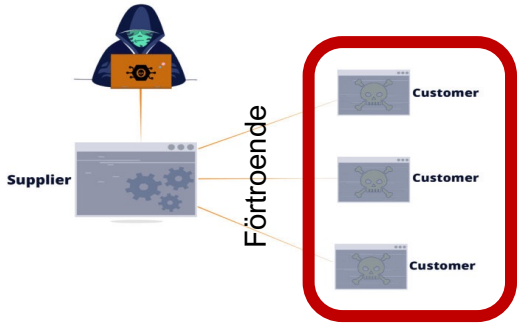
- Det finns ingen “silver bullet”
  - Någon annan attackeras



Hur skyddar man sig?

- Det finns ingen “silver bullet”
  - Någon annan attackeras

- Bättre förståelse för vad som levereras



SBOM – Software Bill Of Materials
- Lista över alla open-sourcekomponenter
- Transparensen sätter mer press på leverantör

SPDX
Linux Foundation

CycloneDX
OWASP

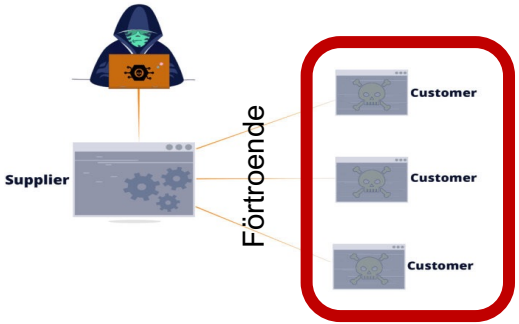
- Skapar förutsättningar för kunder att förstå mjukvaran

Analysera hälsan för att proaktivt bedöma risker med open-sourcemjukvara

Detta gör vi på Debricked

Hur skyddar man sig?

- Det finns ingen “silver bullet”
  - Någon annan attackeras
- Bättre förståelse för vad som levereras
- Automatisk patchning

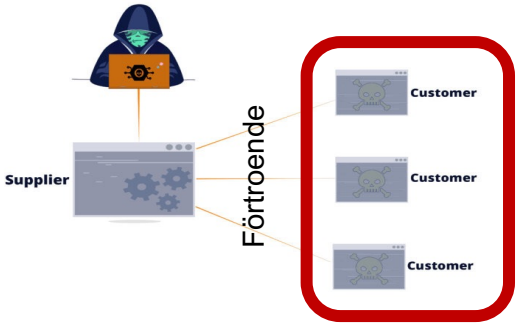


- Minimera konskvenser
- Del av “Software Composition Analysis” där sårbara/skadliga komponenter identifieras och automatiskt patchas

Detta gör vi på Debricked

Hur skyddar man sig?

- Det finns ingen “silver bullet”
  - Någon annan attackeras
- Bättre förståelse för vad som levereras
- Automatisk patchning
- Ifrågasätt förtroendet



Zero-Trust

- Paradigm/koncept som utgår från tanken att alla användare/enheter explicit måste autentiseras och vars access måste kontrolleras

Framväxande teknologier och angreppssätt

- Google SLSA (Supply chain Levels for Software Artifacts)
- **Mål:** Säkerställa riktighet i versionshantering, byggprocess och distributionsplattform

Level 1, Basic protection

↓

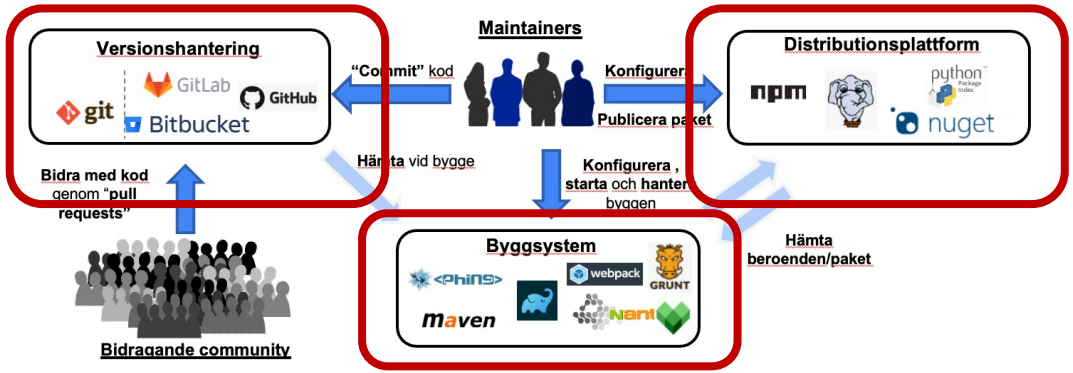
Level 2, Medium protection

↓

Level 3, Advanced protection

↓

Level 4, Maximum protection



Fokus på korrekt bygge av kod – inte riktighet för dess beroenden

Framväxande teknologier och angreppssätt

- MITRE D3FEND – Ramverk och kunskapsdatabas för försvarstekniker

- Kategorisering baserad på

Harden	Detect	Isolate	Deceive	Evict
Gör det svårare att attackera	Identifiera antagonistisk aktivitet	Skapa barriärer så att en attack blir isolerad	Locka in en antagoinist i en kontrollerad miljö	Ta bort antagonisten från miljön

Referenser och fördjupning

- Presentationen bygger på
  - https://debricked.com/blog/software-supply-chain-attacks-part-one/
  - https://debricked.com/blog/software-supply-chain-attack-part-two/
  - https://debricked.com/blog/software-supply-chain-attacks-part-three/
  - https://debricked.com/blog/software-supply-chain-attacks-part-four/
- Rekommendationer från CISA och NIST
  - https://www.cisa.gov/sites/default/files/publications/defending_against_software_supply_chain_attacks_508.pdf
- Analys av ENISA
  - https://www.enisa.europa.eu/publications/threat-landscape-for-supply-chain-attacks
- Google SLSA
  - https://slsa.dev/
- MITRE D3FEND
  - https://d3fend.mitre.org/

Thank you!

Shoot for the stars!