Øvelse 54: Design af detekteringspipeline – 7 abstraktionslag

Formål med øvelsen

Øvelsen her er individuel, men opsamling laves i jeres grupper, se tidsplanen

Formålet med denne øvelse er at træne jeres evne til at analysere og designe en detekteringspipeline ved hjælp af modellen *detekteringens 7 abstraktionslag*. Øvelsen styrker jeres forståelse for, hvordan angreb manifesterer sig i systemet, og hvordan man strukturerer effektiv overvågning, klassificering og alarmering.

Øvelsen er en grundlæggende del af **detection engineering** – altså den faglige disciplin, der handler om at udvikle, justere og dokumentere systematisk detektering af angreb og trusler. I skal lære og anvende en struktureret metode til at analysere og designe detektion. Dette er en kvalificeret metode der kan anvendes til at redegøre for en tilgang til detektering.

Det, I lærer her er en **metode til at opstille en systematisk tilgang til detektering**.
Modellen kan bruges både som analyseværktøj, dokumentationsstruktur og designmetode i jeres fremtidige arbejde med detekterings regler, logning og hændelseshåndtering.

Tidsramme: ca. 60 minutter
Inkluderer læsning, analyse, udformning af pipeline og refleksion.

Hvad er en detekteringspipeline?

En **detekteringspipeline** er den kæde af trin, hvor system- og netværksdata bliver omsat til en meningsfuld alarm:

> Logs → Events → Alarmer → Reaktion

Det handler om at opdage angreb hurtigt og præcist – og om at filtrere støj fra, for at undgå falske positiver.

For at designe en effektiv pipeline kræver det, at man forstår både:

- Hvor angreb kan observeres
- Hvordan de bedst detekteres

- Hvornår og hvordan man bør alarmere

 En pipeline kan baseres på mange forskellige typer data – fx logfiler, procesaktivitet, netværkstrafik, eller **ændringer i kritiske systemfiler** (filintegritet). Man bør overveje hvilken datatyper, der bedst viser indikatorer på potentielle angrebet.

Hvad er *detekteringens 7 abstraktionslag*?

Modellen med detekteringens 7 abstraktionslag stammer fra bogen
Engineering Trustworthy Systems – Get Cybersecurity Design Right the First Time
af O. Sami Saydjari (2021, side 228). Den er bredt anerkendt som en struktureret metode til at analysere, designe og dokumentere detekteringsstrategier i komplekse systemer.

De 7 lag er en **struktureret tankemodel** der kan hjælpe med at:

- Modeller detection pipeline
- Opdele potentielle angreb i håndterbare dele
- Dokumentere jeres detekteringsstrategi
- Identificere svagheder i eksisterende alarmer
- En vejledning til implementering af den enkelte detektering

De 7 lag forklaret

Modellen *detekteringens 7 abstraktionslag* er en struktureret metode til at analysere og designe detektion. Hvert lag repræsenterer et vigtigt spørgsmål i en detekteringspipeline – fra det første tekniske tegn på angreb til den endelige alarmering.

Tabellen herunder forklarer hvert lag og introducerer desuden et ekstra perspektiv: hvordan man **kan inddrage MITRE ATT&CK** som støtte i analysen. Det er ikke en forudsætning for at bruge modellen, men en nyttig måde at styrke arbejdet med klassificering, datakilder og detekterings mønstre.

MITRE-tilknytningen er altså ét muligt redskab blandt flere – og kan bruges når det giver mening i jeres analyse.

Lag	Spørgsmål	Forklaring
1. Feature Selection	Hvor og hvordan manifesterer angrebet sig?	Identificer hvilke systemaktiviteter, netværkshændelser eller adfærdsmønstre der potentielt indikerer et angreb. Dette er det *tekniske udtryk* for angrebet.

Lag	Spørgsmål	Forklaring
2. Feature Extraction	Hvor findes informationen?	Find ud af hvilke datakilder (fx logs, sensorer, auditdata) der indeholder de nødvendige oplysninger for at observere den valgte feature.
3. Event Selection	Hvilken del af datastrømmen er relevant?	Udpeg de specifikke datapunkter eller loglinjer, der kan være interessante – og filtrér støjen fra. Dette skaber fokus for detektionen.
4. Event Detection	Hvad definerer en konkret hændelse?	Formuler regler eller mønstre, der kan genkende en enkelt hændelse som muligvis ondsindet – fx en mistænkelig kommando eller ændring.
5. Attack Detection	Hvornår er det et angreb?	Kombinér flere hændelser eller kontekstuelle signaler og vurder, om der er tale om en *sammenhængende trusselsaktivitet*.
6. Attack Classification	Hvilken type angreb er det?	Klassificér angrebet ud fra en kendt trusseltype eller rammeværk – fx MITRE ATT&CK. Her angives både *teknikken* (hvordan angrebet udføres, fx T1059 – Command and Scripting Interpreter) og den *taktik* det tilhører (fx Execution). Dette muliggør standardiseret analyse og bedre prioritering.
7. Attack Alarming	Hvem skal alarmen nå og hvordan?	Beslut hvem der skal informeres, hvordan det skal ske (mail, dashboard, automatisk reaktion), og med hvilken alvorlighed/alarmtype. Dette muliggør effektiv reaktion og prioritering.

Eksempler

Det er vigtigt at være opmærksom på, at anvendelsen af modellen *detekteringens 7 abstraktionslag* resulterer i en **modellering af jeres detekteringspipeline** – ikke nødvendigvis en komplet eller præcis teknisk implementering.

Modellering er ikke en ensrettet praksis med ét korrekt svar. I nogle tilfælde beskrives F.eks. logkilder meget konkret (fx en bestemt fil), mens der i andre nævnes mere overordnede systemkomponenter eller værktøjer, som kan levere informationen.

Eksempelvis:

- I **Eksempel 1** under *Feature Extraction* angives en specifik fil: `/var/log/auth.log`
- I **Eksempel 2** angives generelle services som `auditd`, `netstat` og `journald`

Begge tilgange er gyldige. Modellering handler ikke nødvendigvis om præcision, men om at etablere en konceptuel model, der kan understøtte en praktisk implementering og videre analyse.

Eksempel 1: SSH Brute Force

Lag	Indhold
Feature Selection	Mange mislykkede loginforsøg
Feature Extraction	`/var/log/auth.log`
Event Selection	Linjer med `Failed password`
Event Detection	> 5 fejl fra samme IP på 1 minut
Attack Detection	Gentagelsesmønster detekteret
Attack Classification	Brute force
Attack Alarming	Mail, dashboard, Bloker ip adressen med Wazuh agenten eller andet værktøj F.eks. fail2ban

---

Eksempel 2: Uautoriseret sudo

Lag	Indhold
Feature Selection	`sudo` fra ukendt bruger
Feature Extraction	`auth.log`, `audit.log`
Event Selection	Bruger ikke whitelisted

Lag	Indhold
Event Detection	`sudo`-kommando fra uautoriseret bruger
Attack Detection	Afvigelse fra brugerrolle/tidspunkt
Attack Classification	Privilege Escalation
Attack Alarming	Mail til admin, logflag i dashboard, Lås bruger konti

---

Instruktioner

Denne øvelse skal udføres individuelt.
Men i næste opgave skal I præsentere og diskutere øvelsen med jeres gruppe i den næste opgave.

Trin 1 – Vælg et scenarie

- Brute force login via SSH
- Reverse shell fra kompromitteret maskine
- Ændring i systemfil (fx `/etc/shadow`)
- Uautoriseret `sudo`
- Egne scenarier fra semesterprojekt

Trin 2 – Brug denne skabelon til at designe din pipeline:

```
### Angreb:
...

### 1. Feature Selection
Hvordan manifesterer angrebet sig?

### 2. Feature Extraction
Hvor kan informationen hentes? (logkilder, processer, sockets)

### 3. Event Selection
Hvordan udvælger I det relevante?

### 4. Event Detection
Hvordan identificeres en begivenhed teknisk?
```

```
### 5. Attack Detection
Hvordan bestemmes, at det er et angreb?

### 6. Attack Classification
Hvordan navngives og kategoriseres hændelsen?

### 7. Attack Alarming
Hvordan og til hvem alarmeres der?
```

Refleksionsspørgsmål

```
- Hvilket lag var mest uklart at definere?
- Hvad kunne give falske positiver i jeres pipeline?
- Hvordan ville en angriber forsøge at undgå jeres detektion?
- Hvordan kan I bruge denne tilgang i eksamensprojektet?
```

Tip: Brug de 7 lag som en metode til at klarlægge og strukturere jeres detekteringsstrategi – både i arbejdet med overvågning og når I dokumenterer hændelseshåndtering, SIEM i semesterprojektet.

Perspektiv: Detection in Depth og Detection in Breadth

Modellen med de 7 lag hjælper jer med at analysere og strukturere en enkelt detekteringspipeline. Men i virkelige systemer kræver effektiv overvågning mere end ét godt design – det kræver **dybde og bredde** i detektion.

Detection in Depth betyder, at I observerer *samme angreb* på flere niveauer og med forskellige datakilder.

> Eksempel: En reverse shell kan opdages via både netværksaktivitet, proceslogs og auditd – og alle lag kan bidrage med deres egne vinkler.

Detection in Breadth betyder, at I har detektering for *flere forskellige angrebstyper* og teknikker – så I dækker et bredt trusselsbillede.

> Eksempel: At kunne opdage både brute force-login, privilege escalation og ændringer i systemfiler.

Brug de 7 lag som analyseværktøj til *hver* type angreb – og tænk derefter over, hvordan jeres system samlet set dækker både i dybden og i bredden.

 I kan bruge dette som metode i jeres eksamensprojekt: Lav pipelines for flere angrebstyper og vis, hvordan de samlet dækker systemet.

Links

- Bog: Engineering trustworthy systems

2025-04-08 05:23:58