

□ Øvelse 57 – Fra model til implementering: Design og byg jeres egen detektering

□ Formål

Formålet med denne øvelse er at bringe jeres viden i spil i praksis: I skal designe, afprøve og implementere jeres egen detekteringsmekanisme i Wazuh.

Øvelsen samler det, I har arbejdet med i tidligere øvelser om:

- detekteringens 7 abstraktionslag (analyse og modellering)
- gruppediskussion af detection design (sammenligning og kvalificering)
- teknisk opbygning af dekodere og regler (implementering og test)

□ Øvelsen følger en **systematisk metode**:

Modellér en detektering pipeline → **Undersøg** datakilder og loglinjer → **Implementér** detektering i Wazuh

Det primære læringsmål er at kunne udvikle en sammenhængende og praktisk anvendelig detektering – fra idé til konkret implementering.

Øvelsen træner jer i at tænke som **detection engineers**: I starter med et scenario, undersøger systemets reaktion og bygger derfra en meningsfuld og struktureret detektering – en tilgang, I også kan anvende i jeres eksamensprojekt.

□ Baggrund: Fra idé til praktisk detektering

Når man bygger detektering, handler det både om at forstå angrebet og systemets reaktion på det.

Det kræver overblik over, hvilke *digitale signaler* eller spor man kan aflæse – og hvor de findes i systemets logs og hændelser.

Det handler om at oversætte begivenheder i systemet til strukturerede og målbare hændelser, som kan genkendes og analyseres.

Modellen med de 7 abstraktionslag hjælper jer med at strukturere denne proces. Den fungerer som en metode til at omsætte angrebstænkning og loganalyse til **praktisk, målrettet detektering**.

□ I denne øvelse skal I bruge modellen til at udvikle jeres egen komplette detektering – og implementere den som dekoder og regel i Wazuh.

□ *I behøver ikke nødvendigvis tage udgangspunkt i jeres tidligere pipeline, men I kan bygge videre på det arbejde, I allerede har lavet.*

□ Ekstra perspektiv: Detektering som datakilde

Når I bygger en regel, kan den også fungere som en slags **sensor**, der tæller eller observerer bestemte hændelser.

Det betyder, at I – ud over at detektere angreb – også kan:

- **samle kvantitative data** om systemets tilstand
- **visualisere hændelser** over tid (fx failed logins, scanninger, portangreb)
- **underbygge projektarbejdet** med fakta og statistik

□ Hvis I vælger at bruge jeres regel som måleinstrument, så husk: - I *Feature Selection* handler det ikke kun om "hvad der indikerer et angreb", men også om *hvad I ønsker at observere og måle*

- Formålet kan være både **realtidsdetektion** og **indsamling af empiri**

□ Instruktioner

Denne øvelse skal laves i grupper og kan kobles til jeres semesterprojekt.

□ Trin 1 – Design jeres detektering

1. Vælg et scenarie, I ønsker at detektere (fx reverse shell, uautoriseret sudo, brute force, filændringer)
2. Brug skabelonen fra øvelse 54 og dokumentér jeres design med de 7 lag
3. Undersøg hvilke logkilder der indeholder de nødvendige data
4. Hvis I er i tvivl: gå ned på hosten og find eksempler i logfilerne (`/var/log/auth.log`, `journalctl`, `audit.log` osv.)

□ Trin 2 – Find en loglinje i systemet, som I vil detektere

- Gå ned på en af jeres hosts og generér eller find en loglinje, der afspejler jeres scenario
- Brug fx `journalctl`, `tail -f`, `grep` eller `less` til at identificere strukturen i loggen

▢ *Det kan være en fordel at starte med en loglinje med en fast struktur – det gør det lettere at lave regex.*

▢ Trin 3 – Opret dekodere

1. Opret en "forældre"-dekoeder med prematch (fx `^sudo:`)
2. Opret en "barn"-dekoeder med regex og felter
3. Test jeres dekodere med `wazuh-logtest` eller Ruleset Test

▢ Trin 4 – Skriv en regel

1. Opret en regel i `local_rules.xml`, som reagerer på jeres dekoeder
2. Tilføj beskrivelse, level og evt. `group`, `frequency`, `same_source_ip` osv.

▢ Trin 5 – Overvåg en rigtig logfil

1. Konfigurer agenten til at overvåge en logfil
2. Generér testdata og verificer, at hændelsen opfanges
3. Test, at reglen virker og vises i *Threat Hunting* → *Events*

▢ Output

Metoden I har anvendt i denne øvelse, er et godt udgangspunkt for en systematisk fremgangsmåde til detektering i jeres eksamensprojekt.

For at sikre dokumentation for jeres arbejde, og så I kan vise en **reflekteret metode**, bør I som minimum producere:

- Den færdige detekteringspipeline model (De 7 abstraktionslag)
- Den loglinje I arbejder med
- Konfigurationsudsnit fra dekodere og regel
- Screenshots af test og evt. visualisering

▢ *Hvis I vil, kan I også lave en visualisering i Wazuh Dashboards – det er ikke et krav, men en god måde at følge med i jeres detektion på over tid.*

▢ Refleksionsspørgsmål

- Hvad var den største udfordring i designet?
- Hvor præcis er jeres detektering – og hvad kunne give falske positive?
- Hvad lærte I om jeres systems logning og datakilder?
- Hvordan kan I bruge jeres regel til at samle data – og hvad kunne det sige noget om?
- Hvordan passer metoden med 7 lag til jeres måde at tænke sikkerhed og systemforståelse på?

□ Nyttige links

- [Custom rules and decoders \(Wazuh Docs\)](#)
- [Regex syntax i Wazuh](#)
- [Regex101 – testværktøj](#)

□ CIS Controls – Kobling

CIS Control	Titel	Relevans
08 – Audit Log Management	Brug logdata aktivt	I opbygger skræddersyet loganalyse og detektering
13 – Security Monitoring	Opdag mistænkelig aktivitet	Detektering bygger bro mellem logs og trusler
17 – Incident Response Management	Håndtér hændelser effektivt	Regler og alerts sikrer hurtig reaktion

🕒 2025-04-04 08:43:27