Øvelse 38 – Tilpasset detektering med egne loglinjer og decoders

Formål

I denne øvelse skal du lære at oprette dine egne dekodere og regler i Wazuh, så du kan detektere loglinjer, du selv har defineret. Du arbejder med både en "forældre"- og en "barn"-dekoder og bygger en regel, der reagerer på logindholdet.

Formålet er at give dig praktisk erfaring med custom detection engineering, så du kan udvide Wazuh til at dække egne behov – fx ved at overvåge interne scripts, specialiserede systemer eller sikkerhedsrelevante hændelser, der ikke dækkes af standardregler.

Øvelsen bygger videre på forståelsen fra øvelse 36a og 36b – men her skaber du dine egne logformater og detektionslogik fra bunden.

Baggrund

Wazuh (ligesom de fleste SIEM-systemer) detekterer hændelser baseret på logfiler. Når agenten ser en ny loglinje, sendes den til Wazuh managerens loganalyse, som forsøger at matche linjen med et kendt mønster. Hvis linjen genkendes, og der findes en tilhørende regel, kan der genereres en hændelse.

Herunder ses, hvordan en loglinje bevæger sig gennem Wazuhs detection pipeline:

```
[Bruger genererer loglinje (fx i otest.log)]
            ↓
[Wazuh-agent læser logfilen]
            ↓
[Agent sender loglinjen til Wazuh manager]
            ↓
[Manager forsøger at matche dekoder → strukturering af log]
            ↓
[Manager forsøger at matche regel → evt. hændelse genereres]
            ↓
[Hændelse vises i Wazuh Dashboard → Security Events]
```

I denne øvelse detekteres der på følgende loglinje:

fag: Systemsikkerhed fejl: BurdeHeddeOperativSystemSikkerhed rettelse: NyNavngivning

Linjen ligner ikke en typisk log file, dette er for at understrege at Wazuh kan reagere på mønstre i en hvilken som helst tekst.

Hvad er regex?

I denne øvelse skal du oprette en såkaldt dekoder, der kan forstå og strukturere dine loglinjer. For at kunne udtrække værdier fra loggen – som fx fag: Systemsikkerhed – bruger Wazuh en teknik kaldet regex (regular expressions).

En regex er et mønster, der beskriver, hvordan tekst ser ud. Den gør det muligt at finde og navngive bestemte dele af loglinjen, så de kan bruges videre i regler og analyser.

Eksempel fra denne øvelse:

`fag: (\w+) fejl: (\w+) rettelse: (\w+)`

Denne regex matcher linjer, der starter med fag:, efterfulgt af ét ord, så fejl:, et ord mere, og til sidst rettelse: og et sidste ord.
De tre (\w+) er såkaldte capture groups – de gemmer hver sin del af teksten i felterne fag, fejl og rettelse.

Instruktioner

Nu skal der oprettes forældre dekoderer samt barne dekoder, og en regel.

Udgangspunktet er detektering af denne linje:

fag: Systemsikkerhed fejl: BurdeHeddeOperativSystemSikkerhed rettelse: NyNavngivning

---

Trin 1 – Opret forældre-dekoder (Wazuh manager)

En forældre-dekoder bruges til at identificere, om en loglinje overhovedet er relevant for videre analyse. Den fungerer som et filter, der ser på starten af loglinjen og bestemmer, om en mere detaljeret dekodning skal aktiveres.

1. Åbn filen med brugerdefinerede dekodere: `sudo nano /var/ossec/etc/decoders/local_decoder.xml`
2. Tilføj følgende blok:

```
<decoder name="otest">
  <prematch>^fag:</prematch>
```

```
</decoder>
```

Denne dekoder tjekker udelukkende, om loglinjen starter med fag:. Den udtrækker ikke felter, men fungerer som et filter, der sender relevante loglinjer videre til child-dekoderen.

1. Genstart Wazuh manager-
2. Wazuh manageren her en applikation der hedder wazuh-logtest. Eksekver den binær applikation, som har placeringen `/var/ossec/bin/wazuh-logtest`, og Indtast tekststrengen `fag: Systemsikkerhed fejl: BurdeHeddeOperativSystemSikkerhed rettelse: NyNavngivning` og verificer, at otest dekoderen matcher og ramme Phase 2, hvor _name: er otest.

Det er altid navnet på parent-dekoderen, der vises i wazuh-logtest, selv hvis det er child-dekoderen, der laver selve udtrækket.

Trin 2 – Opret barn-dekoder (Wazuh manager)

Når Wazuh har identificeret loglinjen som relevant, skal en barn-dekoder udtrække specifikke felter som fag, fejl og rettelse.

1. Tilføj følgende blok efter den forrige dekoder i local_decoder.xml:

```
<decoder name="otest_child">
  <parent>otest</parent>
  <regex>fag: (\w+) fejl: (\w+) rettelse: (\w+)</regex>
  <order>fag, fejl, rettelse</order>
</decoder>
```

Denne child-dekoder knytter sig til otest og bliver kun brugt, hvis parenten matcher. Den anvender en regex til at udtrække tre felter fra loglinjen og gemmer dem i feltnavne, der angives i order.

1. Genstart Wazuh manager.
2. Test igen med wazuh-logtest og verificer, at felterne fag, fejl og rettelse vises. Test med tekst strengen `fag: Systemsikkerhed fejl: BurdeHeddeOperativSystemSikkerhed rettelse: NyNavngivning`
   Obs! Det er navnet på parent-dekoderen som vises – det er helt normalt.

Trin 3 – Test med Ruleset Test (Dashboard)

Wazuh webinterfacet har et værktøj til at teste, hvordan loglinjer behandles af dekodere og regler. Brug dette til at verificere, at dine dekodere virker, før du skriver regler.

1. I Wazuh webinterfacet, klik på dropdown menuen i øverste venstre hjørne, og vælg: *Server management → Ruleset Test*
2. Indsæt tekst strengen `fag: Systemsikkerhed fejl: BurdeHeddeOperativSystemSikkerhed rettelse: NyNavngivning` og klik på Test
3. Bekræft, at dekoderen anvendes og felterne vises korrekt.

Regle testeren i web interfacet er det samme værktøj som den i tidligere har anvendt i CLI.

Trin 4 – Opret regel (Wazuh manager)

Nu hvor loglinjen kan dekodes, skal du skrive en regel, der reagerer på loggen og genererer en hændelse.

1. Åbn lokal regel-fil: `/var/ossec/etc/rules/local_rules.xml`
2. Tilføj en simpel testregel:

```
<group name="otest">
  <rule id="100022" level="10">
    <decoded_as>otest</decoded_as>
    <description>Reglen virker!</description>
  </rule>
</group>
```

Denne regel matcher alle loglinjer, der er dekodet af otest. Den tildeler dem et alert-niveau og angiver en beskrivelse, som vises i Wazuh-dashboardet, når hændelsen trigges.

1. Genstart Wazuh manager.
2. Test med logtest eller Ruleset Test og bekræft, at reglen udløses.

Trin 5 – Overvåg logfil med Wazuh agent (på overvåget host)

Nu skal du forbinde det hele ved at få agenten til at overvåge en rigtig file.

1. Opret logfilen i et directory: `touch otest.log`
2. Redigér agentens konfigurations file: `/var/ossec/etc/ossec.conf`
3. Tilføj følgende block i en ossec_config blok:

```
<localfile>
  <log_format>syslog</log_format>
  <location>Filsti/otest.log</location>
</localfile>
```

1. Genstart wazuh agenten.
2. Tilføj testlinjen til logfilen: `echo "fag: Systemsikkerhed fejl: BurdeHeddeOperativSystemSikkerhed rettelse: NyNavngivning" >> /sti/til/otest.log`
3. Gå til *Wazuh-dashboard* → *Threat Hunting* → *Events* og søg efter: `rule.id:100022`

Du bør nu se en hændelse med teksten “Reglen virker!” samt dine tre felter.

Opsummering

Øvelsen viser overordnet viser at wazuh kan konfigureres til at arbejde med en hvilken som helst tekst file. Der skal blot konfigureres en analyse pipeline med dekodere og regel konfigurering.

Essens er at Wazuh kigger efter mønstre i en given file

Nyttige links

- Custom rules and decoders - Wazuh
- Wazuh decoders
- Regex 1010
- Wazuh - Regex

Refleksionsspørgsmål

- Hvad er fordelen ved at kunne lave sine egne dekodere og regler?
- Hvordan kan en fast struktur i loglinjer gøre detektering nemmere?
- Hvilke typer interne hændelser i en organisation kunne man bruge denne metode til?
- Hvordan ville du kvalitetssikre din egen dekoder og sikre, at den virker stabilt over tid?

CIS Controls – Kobling

CIS Control: 08 – Audit Log Management
Titel: Strukturér og analyser logdata
Relevans: Øvelsen viser, hvordan logdata tilpasses og indgår i analysepipeline

CIS Control: 17 – Incident Response Management
Titel: Håndtér tilpassede hændelser
Relevans: Egne regler giver mulighed for hurtigt at opdage ikke-standard hændelser

Denne øvelse viser, hvordan Wazuh kan udvides til at understøtte overvågning af organisationens egne systemer og behov – ikke kun standard-software.

Last update: 2026-04-12 17:32:46