Øvelse 30 – Checksum af en fil

Information

Formålet med denne øvelse er at forstå, hvordan **checksums** kan anvendes til at verificere **dataintegritet**.
Ved hjælp af hashfunktioner kan selv de mindste ændringer i en fil opdages, hvilket gør checksums særligt anvendelige i forbindelse med **sikkerhed, dataintegritet og verifikation**.

Systemer til file integrity monitoring (FIM), såsom Wazuh, anvender ofte checksums til at opdage ændringer i filer ved at sammenligne hashværdier over tid.

Auditd anvender derimod en anden tilgang, hvor systemkald overvåges for at registrere, hvornår og hvordan filer bliver ændret.
Fordelen ved denne tilgang er, at der ikke skal beregnes hashværdier for filer, hvilket kan være ressourcekrævende – især ved mange eller store filer.

To forskellige måder at opdage filændringer:

1. Checksum-baseret (FIM)
    - Sammenligner filer over tid
    - Finder ændringer *efter de er sket*
2. Auditd (event-baseret)
    - Registrerer handlingen i det øjeblik den sker
    - Viser hvem og hvordan ændringen skete

Kort sagt:

- FIM = “Er filen ændret?”
- Auditd = “Hvem ændrede filen – og hvordan?”

I praksis kombineres disse metoder ofte for at opnå både detektion af ændringer og indsigt i, hvordan de er sket.

I denne øvelse skal I arbejde med den **checksum-baserede tilgang**, som anvendes af mange HIDS- og SIEM-løsninger til file integrity monitoring.

Baggrund

For at sikre integriteten af data benytter man ofte en **hashværdi** som checksum. Checksums bruges blandt andet til:

- Verificering af filers integritet efter overførsel eller lagring
- Opdagelse af uautoriserede ændringer i system- og logfiler
- Kontrol af, om en fil er identisk med en referenceversion

En **hashfunktion** genererer en fast længde outputværdi baseret på inputdata.
Hvis blot ét enkelt tegn ændres i inputtet, vil hashværdien ændre sig markant. Dette gør checksums til en pålidelig metode til at opdage ændringer.

Instruktioner

Husk at notere dine observationer i dit cheat sheet.

1 Opret og verificér checksum

1. Opret en testfil med følgende indhold: `echo "Hej med dig" > testfil.txt`
2. Generér en checksum af filen med SHA-256: `sha256sum testfil.txt`

   > Notér checksummen.
3. Generér checksummen igen uden at ændre filen: `sha256sum testfil.txt`

   > Checksummen bør være identisk med den forrige.

---

2 Ændr filen og observer forskellen

1. Tilføj et ekstra tegn til filen: `echo "f" >> testfil.txt`
2. Generér en ny checksum: `sha256sum testfil.txt`

   > Bemærk, at checksummen nu er helt anderledes, selvom ændringen i filen er minimal.
   > Overvej: Hvorfor ændrer hele checksummen sig, selv ved en meget lille ændring i filen?

Yderligere anvendelser

Checksums anvendes bredt i IT-verdenen, blandt andet til:

- Digitale signaturer.
- Dataintegritet i backup- og gendannelsessystemer

- Bekræftelse af softwaredownloads (fx ISO-filer)

For at verificere en fil mod en kendt checksum kan du anvende: `sha256sum -c checksumfil.txt`

Checksum og logfil-integritet

I en efterforskningsproces er det afgørende at sikre, at **logfiler ikke er blevet manipuleret**. Checksums kan bruges til at:

- Verificere, at en logfil er uændret mellem indsamling og analyse
- Dokumentere filintegritet i forbindelse med compliance eller retssager
- Understøtte sporbarhed og tillid til logdata

Eksempel på oprettelse af checksum for en logfil: `sha256sum /var/log/audit/audit.log > log_checksum.sha256`

Senere verifikation: `sha256sum -c log_checksum.sha256`

Hvis checksummen matcher, er filen uændret. Hvis ikke, kan der være sket manipulation.

Refleksion

- Hvorfor er checksums særligt vigtige i forbindelse med logfiler?
- Hvilke svagheder har checksums, hvis en angriber har fuld adgang til systemet?
- Hvordan kan checksums kombineres med andre sikkerhedsforanstaltninger (fx fjernlogning eller SIEM)?

Links

- hash alot man page
- sha256sum man page

Last update: 2026-03-24 07:57:47