

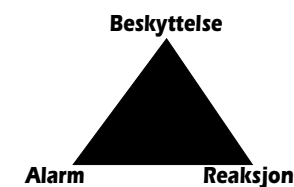
Innbruddsdeteksjon

IDS – Intrusion Detection System

I første artikkel introduserte vi IDS-systemer og deres rolle i en total sikkerhetsløsning. Videre presenterte vi ulike teknologivarianter som er i bruk, og gjennomgikk deres primære karakteristika. Turen er kommet til å se nærmere på effekten av systemene, hvordan de utvikler seg og hvilke utfordringer vi står overfor i en hverdag der truslene forandrer seg ukentlig – eller oftere.

IDS er IT-verdens innbruddsalarmer: Armert med erkjennelsen av at beskyttelse aldri kan bli vanntett og at de fleste 'forgripelser' har sin opprinnelse internt, er behovet for alarmer som supplement til regulære beskyttelsestiltak innlysende.

Avvikelser og unnvikelser



God sikkerhet fundamenteres på denne trekantede symbiosen (figuren er hentet fra seriens første artikkel).

På samme måte som for beskyttelse, er også deteksjons- og alarmsystemer utsatt for virkelighetens dynamikk: 'Fienden' blir stadig smartere, og vi kjemper en evig kamp mot tiden for ikke å bli hengende langt etter i utviklingen. Tiltakene er per definisjon reaktive: Inntrengene leter kontinuerlig etter nye veier, metoder, svakheter og triks, mens IDS-systemene skal tette hull, belyse skyggene og blokkere åpne veier.

Forkledninger og gjemsel

Denne kappestriden handler like mye om å gjøre seg usynlig – finne troverdige forkledninger – som å oppdage smutthull og bakveier. Stådig mer sofistikerte brannmurer sørger for en løpende heving av terskelen for forkledningene: De arter seg typisk som nettverkstrafikk med tilsynelatende *bona fide* innhold – kontrollmeldinger (ICMP), navnetjenertrafikk (DNS), Web-adresser (URLs) og så videre.

Slike forkledninger og gjemselsteknikker tar mål av seg til å komme gjennom signaturanalysene som ligger til grunn for de fleste avanserte brannmurer, viruskontroller og ikke minst: IDS-systemer. Signaturene er digitale mønstre som samles, lagres og kontinuerlig oppdateres i beskyttelsesutstyret.

Her er mengden, dynamikken og oppdateringene utfordringer i seg selv – en diskusjon vi skal komme tilbake til i en annen sammenheng. Imidlertid er det også slik at selve forkledningene i mange tilfeller har sine egne signaturer, som kan være mer kompliserte enn 'vanlig', men som like fullt gir muligheter for deteksjon – enten i beskyttelsesutstyret (brannmurene) eller i IDS-systemene. I IDS-sammenheng kalles dette for *Anomaly Detection*, som er mer komplisert enn alminnelig signatur-analyse, og inkludert i en rekke av de mest sofistikerte IDS-systemene på markedet.¹

¹ Mer om disse kategoriene finnes i første artikkel (forrige utgave).

Fiendens artilleri

De fleste er kjent med at det finnes hundrevis, kanskje tusenvis av programmer og verktøy til hjelp for å begå innbrudd, sabotasje og ulike andre former for angrep mot nettverk og systemer. Virus-generatorene, prosedyrer for generering av trojanske hester, og ikke minst såkalte *root-kits*, som sørger for å skaffe system-privilegier når innbruddet er et faktum, er kjente eksempler.

Langt mindre kjent er det at tilsvarende verktøy finnes for å unndra seg IDS-systemenes oppmerksomhet. Vi står kort og godt overfor den samme situasjonen som virusbeskyttelse og brannmur-beskyttelse: Et evig kappløp mellom de som leter etter nye hull og de som tetter dem.

Vi trenger ikke å gå lenger enn til logg-filene fra vår Web-tjener for å finne eksempler. De flommer over av innslag der ord, kommandoer og adresser er maltraktert til det ugjenkjennelige – men fortsatt vil bli forstått av tjenesten de er adressert til – dersom den finnes:

```
"GET /msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0"
```

Mens det er umulig for et IDS-system å ha signaturer som gjenkjenner alle tenkelige varianter, skal et godt system reagere på underlige kodinger: De er unormale og hører hjemme under kategorien avvik.

To spesialister fra selskapet Secure Networks Inc. laget i 1998 en studie av alle kjente 'unnaluringsteknikker',² og utviklet deretter en test som ble kjørt mot en rekke produkter på markedet – kommersielle og frie. Ingen av produktene besto prøven, hvilket demonstrerte at 'fienden' lå en langside foran forsvaret. Forholdet har uten tvil bedret seg siden, men kappløpet har på ingen måte avtatt, og 'fiendens' verktøy er blitt flere og bedre. De mest kjente er:

- ✓ **Whisker** – som først og fremst arbeider med Web-adresser (som i eksemplet ovenfor).
- ✓ **Fragrouter** – som fragmenterer IP-trafikk til det ugjenkjennelige.
- ✓ **Mutate** – et alternativ til Whisker.
- ✓ **Snort** – genererer tilfeldig nettverkstrafikk, er avledet av beskyttelsesverktøyet Snort, og fokuserer spesielt på å lure nettopp Snort.³
- ✓ **Nmap** – kartlegging av nettverk, et verktøy vi har presentert ved tidligere anledninger, og som er minst like nyttig i forsvar som i angrep.

IDS-arkitektur

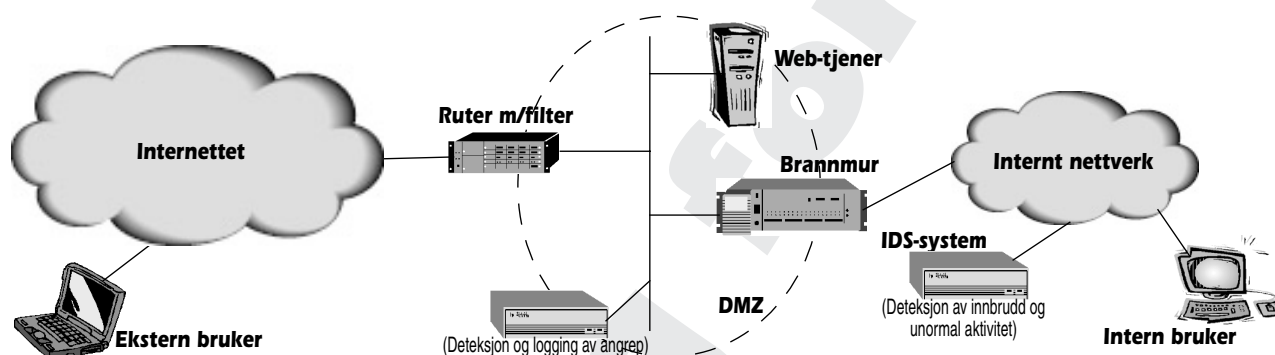
Intensjonen med forkledd trafikk kan være å kartlegge, å kommunisere med trojanske hester som allerede er på plass, å utnytte mer eller

2 Ptacek & Newsham: INSERTION, EVASION AND DENIAL OF SERVICE: ELUDING NETWORK INTRUSION DETECTION (1998) [<http://www.robertgraham.com/mirror/Ptacek-Newsham-Evasion-98.html>] – et høyst interessant dokument for interesserte.

3 Snort er et av de mest populære OPEN SOURCE IDS-produktene på markedet – som vi kommer tilbake til i neste utgave.

mindre kjente svakheter i systemer og programvare, eller rett og slett å skape problemer (*Denial of Service*, sabotasje). Det betyr at våre alarm-systemer bør være like mye på vakt mot angrep som mot innbrudd. Riktignok er det brannmurer og dybdeforsvar som har hovedansvaret for å beskytte mot slike angrep, men det er ikke dermed unyttig eller ulogisk at også alarmsystemet yter sitt: De ulike elementene i sikkerhetsarkitekturen har forskjellige måter å analysere sine omgivelser på, og dermed varierende muligheter til å oppdage ulike former for angrep.

En arkitektur som utnytter dette egenskapsspekteret maksimalt, har derfor IDS-systemer både på innsiden og utsiden av det interne nettverket (figur 1): Mens det eksterne systemet (typisk i en DMZ-soner) fokuserer på å oppdage angrep, er det interne orientert mot å finne inntrengere og ureglementerte aktiviteter.



Figur 1 Et IDS-system kan plasseres utenfor det interne nettverket for å avdekke (og loggføre) angrepsforsøk mot systemer i DMZ (Web-tjener) og det interne nettverket (brannmur). Et slikt oppsett fungerer også dersom DMZ-nettverket står på et eget ben ut fra brannmuren.

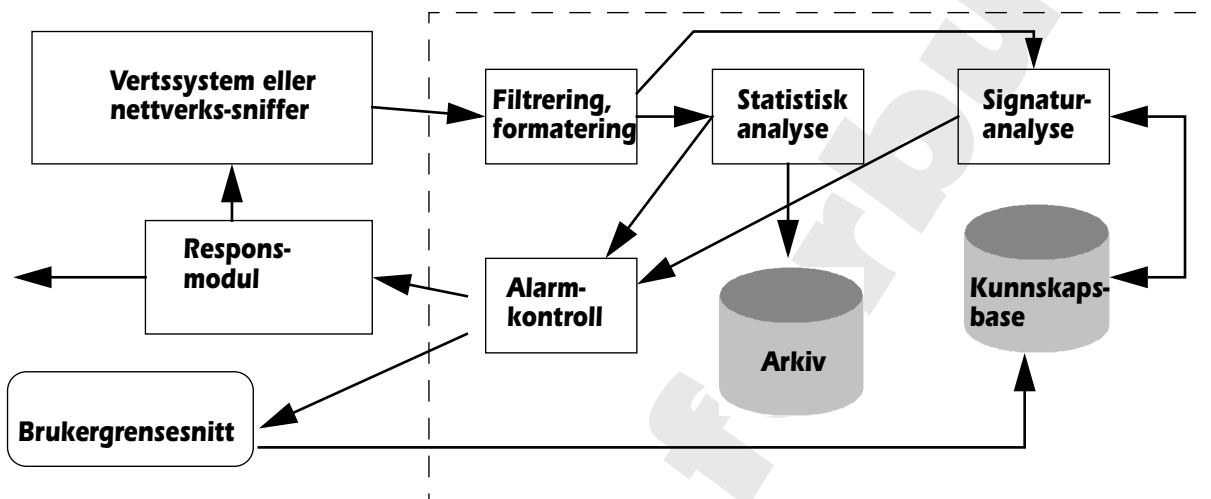
Et slikt både-og scenario er ideelt – og sjeldent: Vi vil i de fleste tilfeller være nødt til å prioritere det ene eller det andre – av økonomiske og ressursmessige årsaker. Da vil det interne IDS-systemet alltid komme i første rekke. Dersom det imidlertid hersker tvil om behovene, kan det være nødvendig å begynne med angrepsdeteksjon, som vil avdekke en virkelighet der fravær av alarmsystem fremstår som ren galskap.

Med 20-års erfaring

Siden tankegangen og ideene rundt IDS-systemer ble formalisert på 80-tallet,⁴ har en rekke modeller og arkitekturer sett dagens lys. De fleste har hatt sin opprinnelse i akademiske miljøer, og har fungert som eksperimenter – tidvis med sponning fra offentlige institusjoner i USA. Øvelsene har gitt mye erfaring og lagt grunnlaget for en rekke av produktene på markedet – kommersielle såvel som frie (*Open Source*). Målsettinger om å etablere et rammeverk og en samling standarder, slik at produkter med ulik opprinnelse lett kan fungere effektivt sammen, har imidlertid ikke lyktes.

⁴ Banebrytende litteratur på området er "COMPUTER SECURITY MONITORING AND SURVEILLANCE" av James Andreson (1980), og Dorothy Denning: "AN INTRUSION DETECTION MODEL" – gjengitt i IEEE Transactions on Software Engineering, februar 1987, se <www.cs.georgetown.edu/~denning/infosec/ids-model.rtf>.

I forrige artikkel konstaterte vi at to hovedgrupper av IDS-systemer finnes: Vertsbaserte (HIDS) og nettverksbaserte (NIDS). Disse hører i sin tur til én av to kategorier: Aktive eller passive, hvilket i sin tur gir oss fire ulike typer med distinkte særtrekk og tallrike likheter: De kan alle prinsipielt beskrives av det generiske blokkdiagrammet i figur 2.



Figur 2 Prinsippskisse for et INTRUSION DETECTION SYSTEM. De fleste systemer på markedet er satt sammen etter denne modellen.

De fleste produktene på markedet er hybrider, og dekker to eller flere av disse typene. Måten de ulike modulene er implementert på og hva prioriteringene har vært med hensyn til egenskaper, varierer imidlertid sterkt, et forhold vi kommer inn på når vi diskuterer konkrete produkter i neste artikkel.

Viktige egenskaper

Utover grunnleggende funksjonalitet, må vi stille følgende hovedkrav til et IDS:⁵

- ✓ **Selvstendig:** Systemet skal normalt klare seg på egen hånd, uten tilsyn eller inngrep. Det skal ikke forstyrre vertsmaskinen eller nettverket som overvåkes, og det skal være mulig for en administrator å kontrollere hva som foregår innvendig i systemet (*Black Box* prinsippet er ikke akseptabelt for et IDS).
- ✓ **Feiltoleranse:** Systemet skal kunne gjennomleve at verten krasjer uten å tape data i arkiv eller kunnskapsbase.
- ✓ **Selvkontroll:** Systemet skal overvåke seg selv og kunne avdekke forsøk på overstyring eller tukling fra uvedkommende.

⁵ Kravene er utarbeidet av COAST, COMPUTER OPERATIONS, AUDIT AND SECURITY TECHNOLOGY ved Purdue-universitetet i USA.

- ✓ **Effektivitet:** Systemet må ikke representere en vesentlig belastning på verten eller nettverket. Et system som gir merkbart økte responstider, vil ikke bli brukt.
- ✓ **Avviksdeteksjon:** Systemet skal lett kunne detektere avvik fra normale trafikk-, belastnings- og bruks-mønstre.
- ✓ **Tilpasningsdyktig:** Alle systemer har sine karakteristiske bruksmønstre, og systemet må enkelt kunne tilpasses til disse.
- ✓ **Fleksibilitet:** Bruksmønstre forandrer seg over tid, for eksempel i forbindelse med at applikasjoner og løsninger legges til, fjernes, oppdateres eller ombygges. Systemet må kunne tilpasse seg slike forandringer uten manuell intervensjon av betydning.
- ✓ **Robust:** Mens ingen systemer er feilfrie, må vi forlange at produktet er vanskelig å lure.

Kritisk effektivitet

Kravene er enkle nok å spesifisere, men langt fra like enkle å måle eller tilfredsstille. Spesielt er effektivitet og kapasitet en stor utfordring, spesielt i forbindelse med nettverksbaserte IDS-systemer. Oppfatningen av hva som er høy båndbredde og dermed tilstrekkelig kapasitet i et system, forandrer seg år for år. Dermed blir det ofte slik at behov og produkter kommer i utakt. De fleste dedikerte IDS-bokser og vertsbaserte systemer kneler (taper data) når benyttet båndbredde passerer 20 til 50 Mbps – hvilket ikke lenger er for mye å regne i lokalnettsammenheng. Dette forholdet må vi ta hensyn til i planlegging og implementasjon: Det er bortkastet tid å anta at noe vil fungere dersom enkle regnestykker viser det motsatte.

Praktiske tester viser at det er vanskelig å vite når den ytelsesmessige grensen er nådd: Systemene er sjelden selv i stand til å oppdage at data går tapt. Desto viktigere er det å for det første vite hva den reelle kapasiteten er, og for det andre å sørge for korrelasjon med reell nettverkstrafikk: For eksempel kan det være hensiktsmessig å definere en alarm i nettverksstyrings-systemet som trigges når den gjennomsnittlige trafikken overstiger kapasiteten til IDS-systemet.

For å plassere denne triggeren, må vi imidlertid med en viss sikkerhet vite hvor grensen går. Leverandørenes tall er som regel for optimistiske, fordi de har fremkommet under urealistiske omgivelser. Slike tester er notorisk vanskelige å lage og å utføre, fordi kunstig generert trafikk sjelden ligner på virkeligheten, og fordi ulike algoritmevalg internt i IDS-systemene kan gjøre et gitt system supereffektivt på én type trafikk, og ineffektivt på en annen type. Løsningen er å gjennomføre tester på reell trafikk som kombineres med typiske eksempler på forventet avviks- og innbrudds-trafikk. Testene kan utføres i et 'levende' nettverk, eller ved å tappe nettverket (gjøre opptak av trafikken) over en periode, og senere 'spille av' trafikken i et testnettverk.

Alarmen går

Forsvaret er på plass, alarmene aktivert og systemene testet forlengs og baklengs. Det er god grunn til å føle en høyere grad av trygghet enn noen gang tidligere. Men hva skjer når alarmen går? Hvordan håndterer vi hendelsen, hvilke prosedyrer følger vi og hva skal vi foreta oss for å sikre bevis og annet relevant materiale, samtidig med at vi reduserer skadene til det minimale?

Det er kanskje ikke overraskende – i alle fall er det et faktum at denne delen av ligningen gjerne blir glemt eller oversett. Her ligger også en av årsakene til at så få innbrudds- (og utbrudds-) saker blir rapportert til myndighetene, og at enda færre leder til noe annet enn en rapport og en forsikringsutbetaling.

Igjen står vi overfor en vanskelig prioritering – som til slutt kan komme til å ligge i forsikringsselskapenes hender: Skal vi være fornøyde med å oppdage og stoppe overtredelser tidlig, eller skal vi stå løpet ut, og sørge for prosedyrer og verktøy for sikring og samling av bevismateriale? Inntil forsikringsselskapene forlanger sistnevnte, vil den bli nedprioritert av de fleste organisasjoner, fordi gevinsten er liten eller negativ. Sikring av bevis fører gjerne til at systemer blir ute av drift over lengre tid, og pågripelse medfører gjerne negativ publisitet. Inntil videre opprettholdes derfor dagens situasjon: Inntrengere og forbrytere blir stoppet, men sjelden identifisert og praktisk talt aldri straffet.

Uansett valget, er det nødvendig å utarbeide og teste prosedyrer og rutiner for håndtering av ulike alarmer: Hvem som har ansvaret, hva som må gjøres, informasjonsflyt, verktøy og så videre.

Også på dette området finnes det en rekke interessante *Open Source* verktøy tilgjengelige – de fleste sentrert rundt *The Coroner's Toolkit*,⁶ en fritt tilgjengelig pakke utviklet av de samme personene som i sin tid laget SATAN, som senere ble til SAINT.⁷ En tilsvarende pakke med en annen opprinnelse finnes for Windows: IRCR – *Incident Response Collection Report*.⁸

Ytterligere informasjon om tilgjengelige verktøy for sikring og samling av bevis finnes i tilleggsmaterialet til denne utgaven av Mellvik-Rapporten, se side 35 for detaljer.

Neste utgave

I neste utgave avslutter vi presentasjonen av IDS-teknologi og -systemer med en gjennomgang av prosedyrene som bør ligge til grunn for valg av system, og en oversikt over de viktigste produktene i segmentet – frie/*Open Source* såvel som kommersielle. ■

6 TCT og tilleggsverktøyene kan kjøres på alle tenkelige Unix/Linux-plattformer, se www.porcupine.org/forensics for mer informasjon.

7 Dan Farmer, Wietse Venema

8 www.incident-response.org/IRCR.htm