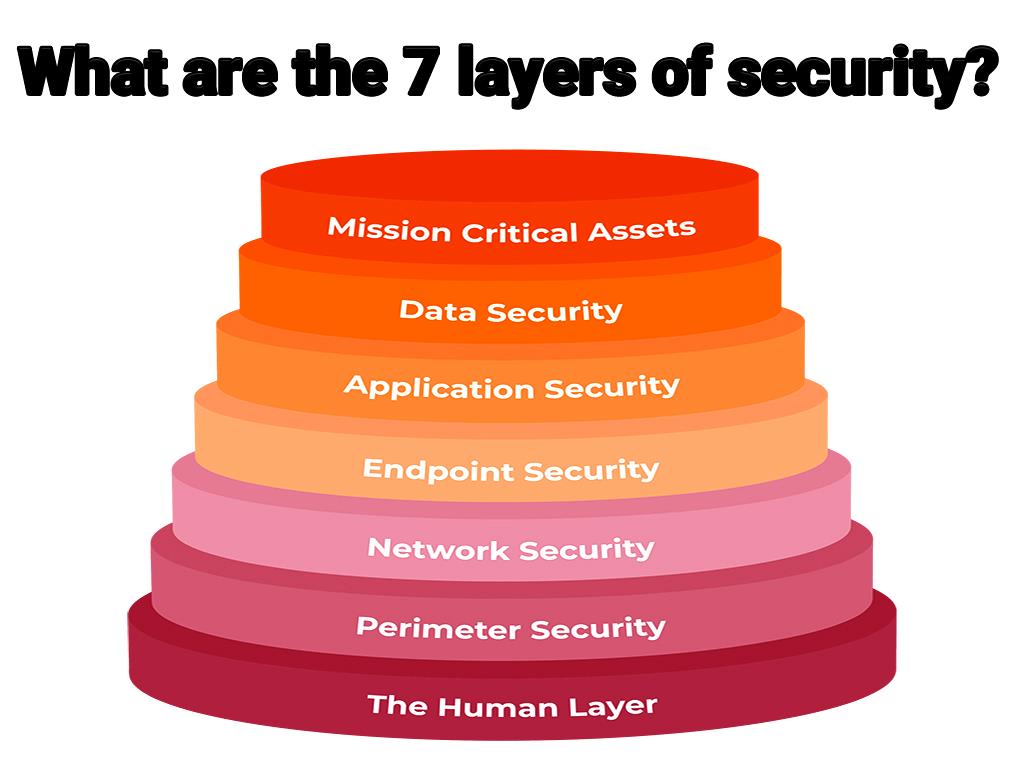
# Incidenthantering och loggar

Incidenthantering och loggar är två ämnen som blir väldigt intimt förknippade med varandra, i ett modernt cybersecurity-landskap. Med incidenthantering menas här att hantera en säkerhetsincident – alltså en godtycklig läcka av information, eller påvisat ”försök till hack” eller ”försök till brytande” av de olika säkerhetslager som finns för att skydda ett företags tillgång.



Varje försök att bryta något lager i säkerhetsmodellen – alltså det mänskliga lagret (genom t.ex. social engineering/phishing etc.), permiterlagret (genom att försöka komma in i lokalerna t.ex.) nätverkslagret (genom olika nätverksattacker, som spoofing, sniffning,... etc.), endpointlagret (som t.ex. virusattacker, ransomware etc.), applikationslagret (xss, injections), eller datalagret (för att läsa och/eller ändra filer, t.ex.) för att komma åt våra mest värdefulla tillgångar – ska ses som en incident.

En incident betyder inte att ”attacken” eller ”hacket” har lyckats – bara att vi kunnat identifiera ett försök till att bryta något av dessa lager av säkerhet.

Varje sådant försök skall hanteras (och på ett – för företaget bra sätt) – utifrån vad som är mest effektivt vad det gäller resurser som tid, pengar, tekniskt kunnande etc. Ofta brukar detta omsättas i praktiken med att ”säkerhet är allas ansvar”, och där ALLA användare av resurser på ett företag – utifrån sina egna erfarenheter – förväntas föreslå åtgärder för att kunna upprätthålla säkerheten, att undvika incidenter, samt att hantera dessa.

Nästan alla böcker (ja – det finns långa månghundrasidiga böcker), artiklar, filmer och websidor brukar beskriva just incidenthanteringsarbetet i någon form av ”incident response plan” (IRP) – en enkelt begriplig plan för att hantera (alla sorters) incidenter – med handfasta tips om vad man ska göra ”om något händer”.

## IRP

Enligt alla standarder borde en IRP innehålla åtminstone sektioner om

* Förberedelser – innan en incident inträffar
* Identifiering av säkerhetsincidenter
* ”Cointainment” – hur man hanterar data/nätelement/servrar/källkod som är påverkat – för att undvika vidare spridning
* ”Eradication” – hur man tar bort skadan
* Återställning
* Lessons learned (vad vi kunnat lära oss av incidenten)

Vi ska ta en titt på de här olika punkterna – var och en för sig och se vad varje punkt kan innehålla för åtgärder vi kan vidta – för att antingen undvika incidenter, eller hantera dessa. Vårat s.k. ”incident response”.

### Förberedelser

Det här är det första steget i vårt planerade säkerhetsarbete. Vi talar om ”system av system” för att t.ex. detektera olika sorters attacker. Detta system kan vara ett system av t.ex. SIEM, brandväggar (och waf), antivirussystem, endpoint security, filintegritetskontroller och andra system eller delar av system som kan ge oss tidiga indikationer på att något är ”fara å färde”.

Vårt ”system av system” är vår första linje av försvarsmekanismer – tänkt som en ”underrättelseåtgärd”, där vi alltså – genom övervakning av miljöerna på olika sätt kan detektera en incident.

### Identifiering av säkerhetsincidenter

Har vi nu förberett oss ordentligt (och har ett system av system för att säkra upp oss med) handlar identifieringen av våra säkerhetsincidenter om olika sorters larmfunktioner eller ”fällor” i våra system, där vi snabbt – och automatiskt – kan identifiera olika sorters säkerhetsincidenter. Incidenter i ”det mänskliga lagret” är emellertid svåra att upptäcka automatiskt, utan kräver normalt sett ett antal manuella operationer – för att både upptäcka, men också registrera – säkerhetsincidenter.

I vårt ”system av system” har vi emellertid nu ett IDS – ett ”intrusion detection system”. Detta är normalt sätt byggt som en kombination av flera system – men ofta är dessa byggda kring samma principer som vi talade om på förra lektionen – kring kryptologiska skydd för filintegritet/nätverkssäkerhet/attacksignaturer etc.

Vi kan också identifiera säkerhetsincidenter efter det att de inträffat – med hjälp av vårt SIEM-system. Genom att jämföra olika loggmeddelanden mot kontrollnivåer – vår ”baseline” kan vi i efterhand detektera att vi redan HAR varit utsatta för en säkerhetsincident. Vårt SIEM kan vi också använda ”forensiskt” – för att finna spår på vad som hänt, och för att då vi sedan använder det i vår plan, röna ut ”root cause” och vad vi kan lära oss av incidenten, för att undvika att den händer igen.

Ett tredje sätt att upptäcka säkerhetsincidenter är att ta till vara information från t.ex. Endpoint security clients, som ju alla har en egen databas för identifiering av viss(a sorters) attack(er) och attackvektorer.

Det sista sättet att identifiera säkerhetsincidenter är att upptäcka dessa manuellt. Det kräver emellertid en hel del erfarenhet av säkerhetsarbete sedan tidigare – men det är inte ovanligt att just säkerhetsincidenter i det mänskliga lagret upptäcks just av människor, utan tekniska hjälpmedel.

### ”Containment”

Containment handlar om att isolera säkerhetsincidenten – och de tjänster, datorer, system etc. som är berörda – så att incidenten inte kan sprida sig till andra system/delsystem/funktioner etc.

Men det handlar också om att vi inte kan genomföra något slags forensik eller återställande åtgärder på systemet, eller delsystemet, eller funktionen – utan att först isolera incidenten och begränsa den så att den inte växer eller vi förlorar kontrollen över den.

(Här kommer den flitige eleven säkert ihåg händelserna kring säkerhetsincidenterna på 1177 för några få år sedan – där man \_inte\_ skötte containment och därför fick hålla på i flera dagar och ”städa”, både tekniskt och medialt. Vi kommer alla ihåg uttalanden från företaget som var ansvariga som ”internetsladd” (blev tom en viral sång, som bl.a. fortfarande finns på youtube), ”avancerade kommandorörelser” etc.)

Men all meningsfull containment innebär en komplett avstängning av all nätverkstrafik – både mot Internet och inåt mot ”intranät”-strukturer – för att köpa tid att åtgärda problemen.

### Eradication

Eradication handlar om att ta bort problemen som incidenten kan ha ställt till med – men också att förhindra att de kan uppstå igen. Ofta börjar ”eradication-processen” med en forensiskt analys – där man går igenom samtliga spår som finns av attacken/incidenten, i form av loggar (SIEM igen), eventuella distribuerade loggar, etc.

Man går också igenom eventuella föreslagna patchningsförfaranden (har någon annan vi känner till någon gång råkat ut för något liknande? Hur löste de problemet?) etc.

Normalt sätt genomgår man steg som att ”blåsa om” maskiner (nyinstallerar dessa med ett uppdaterat/patchat OS eller liknande), letar efter bakdörrar eller konton som skapats av någon annan (filintegritetskontroll, tripwire etc.) och fastställer ”root cause” till incidenten.

Denna brukar fastställas och beskrivas i en RCA – ”Root Cause Analysis” – ett dokument som går igenom inte bara vad som hänt, utan varför det faktiskt hände. Normalt sett innehåller ett sådant dokument saker som en tidslinje av vad som hänt, vilka åtgärder som tagits av vem och när – och i många fall också varför. En RCA skall också innehålla vilka åtgärder som vidtagits för att undvika att felet/incidenten återuppstår – både tekniska åtgärder (såsom ”patcha databasservern”) som mer administrativa procedurer (”ingen nyanställd får tillgång till administrationskontot i utvecklingsmiljön under de första 3 månadernas provanställning...”

### Återställning

Vid återställning efter en incident gäller det att få upp de system som är berörda till ett läge snarlikt, eller bättre, än innan incidenten så fort som möjligt. Detta för att minska eventuell ekonomisk förlust kopplad till ”nedtid” i systemet.

Med detta i bakhuvudet är det samtidigt viktigt att inte försöka sig på någon form av återställning av systemet innan man gjort en fullständig containment på det – annars ökar risken för att incidenten kommer tillbaka.

### Lessons learned

Efter att man gått igenom eradication och återställningen efter en incident är det viktigt att man går igenom vilka lärdomar man själv kan dra av säkerhetsincidenten. Finns det t.ex. procedurer och/eller processer som måste ändras? Finns det tekniska slutsatser att dra? Hur tar vi till vara på de lärdomarna vi kan dra av en genomförd incident?

Ofta ska detta rapporteras på olika sätt – t.ex. i något kvalitetssäkringssystem (HACCP, ISO 9000, PCI/DSS etc.) och dokumenteras inför framtiden för att man ska kunna gå tillbaka och se vad man lärt sig av olika incidenter och vad det har ändrat processer, tillvägagångssätt och dagliga rutiner i arbetet.

Vi ska ta och titta lite på just ”Incident response plan” och en liten film för att befästa kunskaperna.

<https://youtu.be/NL1ShMo4Gm4> (3:48)

## Loganalys

Som vi såg i vår incident response plan kommer just loggning och analys av loggning in på många olika ställen i processen när vi ska förebygga, hantera men också städa upp efter en säkerhetsincident.

Även om ”bra loggning” inte är det enda som gör just incident response möjligt är det ett oerhört viktigt verktyg i verktygslådan för systemsäkerhet (i stort) och IT-säkerhet (i synnerhet) och för utvecklare – både av mjukvara som faktiskt analyserar loggar, men också mjukvara som producerar loggar på olika sätt – är detta yttermera viktigt att veta – och lägga vikt vid.

För att vi ska vinna något alls på att logga något alls måste vi alltså veta \_varför\_ vi loggar något och \_till vem\_. Loggfilen måste kunna läsas av antingen A) en annan person – som sedan ska kunna dra rimliga slutsatser utefter vad hen faktiskt läser eller B) ett annat system – som ska kunna utföra någon form av ”AI”, eller ”automatisk analys” av loggens olika meddelanden. I en kurs om ”IT-säkerhet för utvecklare” måste vi ta hänsyn till hur den webbapplikation eller den mjukvara vi utvecklar för stunden faktiskt meddelar sig med både utvecklare och administratörer med just loggar.

Vi vill att loggmeddelanden ska vara ”lagom långa” – tillräckligt långa för att vi ska kunna förstå vad det menar och inte behöva tolka in för mycket data själva (det kommer nämligen att leda till feltolkningar av den som läser loggfilen), men också tillräckligt korta för att inte diskytor och loggservrar och olika loggningsprocesser helt ska svämmas över av ovidkommande logg.

Vi ska ta att titta lite på loggar ur perspektivet logganalys. Vi brukar enkelt skilja på logganalys utifrån två olika perspektiv – utifrån tiden när vi tittar på loggen:

- Real time analysis

Svårare än "efter att nåt hänt" - incidenthantering

- Post event analysis

Spåra vad som hänt - forensik

Som utvecklare ser man snabbt att här finns ”ett produktformat hål i marknaden” för olika sorters automatisering av logganalys – för att kunna ta fördelar av vår loggfunktion. Inte minst vad det gäller ”Real Time Analysis” – där vi alltså kan använda datat (i realtid) ur vår logg för att faktiskt detektera (som snubbeltråd) vad som händer i vår applikation eller program, när just den är målet i en säkerhetsincident.

Vid post event analysis kontrollerar man loggarna för att få kännetecken på vad som faktiskt hänt. Olika processer kommer att lämna olika ”loggavtryck” och dessa kan kombineras för att t.ex. genomföra olika sorters forensik.

Vi ska – återigen – kolla in youtube och se det här lilla klippet om just loganalys – vad det är och varför man gör det:

Loganalys - vad det är och varför - 2:30 - <https://youtu.be/fGVFCaVQnWw>

Chainsaw är ett windowsverktyg som använder windows event logs för att hitta ”hot”. Det har dels en inbyggd databas, kopplat till sin ”parser” för eventloggar – som gör det lätt att skriva filter till den för att söka efter kända windowsbuggar.

MEN – och det finns ett men här: det använder också ”sigma” (där ”sig” i ”sigma” står för ”signature”) – en öppen databas för attacker (tillbaka till när vi pratade om kryptologiska skydd i vår säkra utvecklingsmiljö) som i det här fallet är attacksignaturer baserade på olika programs eventloggar.

Chainsaw - Windows Event log threat detection - 18:47 <https://youtu.be/YN_kffuC6a8>

För den yttermera intresserade av loggning (på windows) finns här en guide för att skriva – och tolka – loggmeddelanden, som kan användas som inspirationskälla när man själv skriver eventloggar.

Förstå och skriva eventloggar för windows: <https://docs.microsoft.com/en-us/previous-versions/windows/desktop/bb226812(v=vs.85)>