Felhőbiztonság

Ki felel miért — és miért a legtöbb felhős incidens nem az AWS, hanem az ügyfél hibája.

ISMÉTLÉS & MAI TERV

Honnan indulunk — és hova jutunk el

1–2. blokk — Megosztott felelősség modell, majd IAM (identitás- és hozzáférés-kezelés)

3–4. blokk — Hálózati biztonság (SG, NACL) és titkosítás (KMS, ACM)

5–6. blokk — Monitorozás/naplózás és DDoS-/alkalmazásvédelem

7–8. blokk — Esettanulmány egy valós támadási láncon, majd tipikus hibák és megfelelőség

A 2. órán megígértük: *„előbb értenünk kell, mit épít fel helyettünk az AWS, mielőtt eldöntjük, mi a mi felelősségünk."* Most ezt váltjuk be.

1. BLOKK

Megosztott felelősség modell

Visszatérünk a 2. órán látott táblázathoz — és megnézzük, mit jelent a gyakorlatban az, hogy egy sort az „ügyfél” tölt ki.

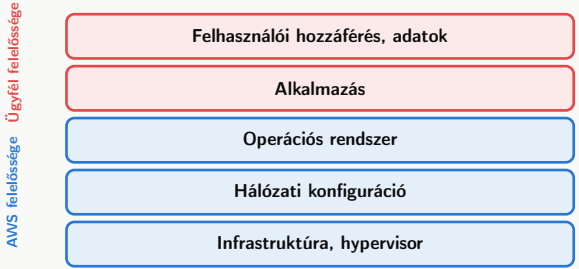
A tábla, amit a 2. órán megígértünk

Réteg	SaaS	PaaS	IaaS
Felhasználói hozzáférés, adatok	mindig az ügyfél felelőssége		
Alkalmazás	AWS	ügyfél	ügyfél
Operációs rendszer	AWS	AWS	ügyfél
Hálózati konfiguráció	AWS	AWS	ügyfél
Hypervisor, fiz. infra.	AWS	AWS	AWS

⚠ Mi EC2-t használunk — tehát **IaaS**. A táblázat szerint az operációs rendszer, a hálózati konfiguráció **és** minden fölötte lévő réteg a mi felelősségünk. A mai óra pontosan ezekről a sorokról szól.

Ugyanaz, rétegként lerajzolva

Minél **alacsonyabb** szintű szolgáltatást használunk (IaaS → PaaS → SaaS), annál **több réteg** tartozik a mi felelősségi körünkbe.



A legtöbb incidens nem az alsó (kék), hanem a **felső, piros** rétegek hibás konfigurációjából ered.

2. BLOKK

Identitás- és hozzáférés-kezelés

IAM — ki, mihez, hogyan férhet hozzá. Az „ügyfél” oszlop első, és talán legfontosabb sora.

IAM alapok és a legkisebb jogosultság elve

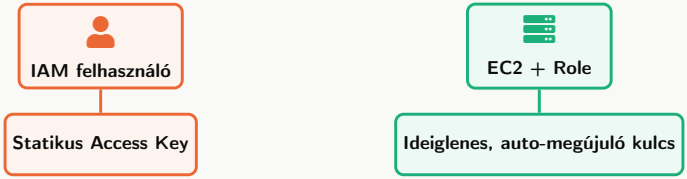
Az AWS **IAM** (Identity and Access Management) szabályozza, ki, milyen szolgáltatáshoz, milyen módon férhet hozzá.

Legkisebb jogosultság elve (least privilege): mindenki csak azt a jogosultságot kapja meg, ami a feladatához feltétlenül szükséges — soha ne adjunk alapból `AdministratorAccess`-t egy alkalmazásnak vagy felhasználónak.

Ez pontosan ugyanaz a gondolat, mint amit a Security Groupoknál a gyakorlaton már alkalmaztunk: *csak azt engedjük be/azt a jogot adjuk ki, amire valóban szükség van* — nem többet, „hátha kell majd".

Felhasználó (statikus kulcs) vs. szerepkör (role)

Egy önálló IAM felhasználóhoz tartozó **Access Key** statikus, hosszú élettartamú titok — egy EC2/Lambda-hoz rendelt **szerepkör (role)** ezzel szemben időben korlátozott, automatikusan lejáró és megújuló hitelesítő adatokat ad.



Ahol lehet, **roles**-t használjunk felhasználó + statikus kulcs helyett — nincs mit ellopni, mert nincs hosszú élettartamú titok.

MFA és az Access Key-ek veszélyei

Többfaktoros azonosítás (MFA)

Különösen a **root** és adminisztrátori fiókoknál alapkövetelmény — egy ellopott jelszó önmagában nem elég a bejutáshoz.

Access Key-ek kezelése

Soha nem szabad forráskódba/verziókezelőbe (pl. GitHub) beleégetni — ez a felhős adatszivárgások **egyik leggyakoribb oka**.

⚠ Emlékezzünk: a gyakorlaton az `aws configure` parancs pont egy ilyen Access Key-t/Secret Key-t kért be — ugyanez a pár az, amit **soha** nem szabad megosztani vagy commitolni.

Egy IAM policy anatómiája

Egy jogosultság a gyakorlatban nem egy egyszerű be/ki kapcsoló, hanem egy **policy dokumentum** — ennek három fő eleme van.

Effect

`Allow` vagy `Deny` — engedélyezzük vagy kifejezetten tiltjuk a lenti műveletet.

Action

Mely API-műveletek — pl. `s3:GetObject`, `ec2:StartInstances`.

Resource

Mely konkrét erőforrásra vonatkozik — pl. egyetlen S3 bucket ARN-je, nem „minden S3".

A legkisebb jogosultság elve (előző dia) **ezen a három mezőn keresztül** érvényesül a gyakorlatban: minél szűkebbre húzzuk az Action-t és a Resource-t, annál kisebb a kockázat egy kiszivárgott hitelesítő esetén.

A root fiók — csak vészhelyzetre

A regisztrációkor használt e-mail+jelszó a **root** fiók — ennek korlátlan, **policy-val nem korlátozható** hozzáférése van mindenhez, a számlázástól a fiók törléséig.

⚠ Mivel a root jogosultsága nem szűkíthető IAM policy-val, egy ellopott root hitelesítő a **teljes fiók** elvesztését jelentheti — ezért a napi munkát mindig IAM felhasználóval/szerepkörrel végezzük.

✓ Gyakorlati teendő regisztráció után: azonnal **MFA-t kapcsolni a root fiókra**, és létrehozni egy adminisztrátori jogú IAM felhasználót a napi munkához — a root bejelentkezést innentől tartsuk fenn kivételes esetekre.

3. BLOKK

Hálózati biztonság

A gyakorlaton már megépítettünk egy Security Groupot — most megnézzük, mitől „stateful”, és mi a különbség a NACL-hez képest.

Security Group — gyakorlati recap, *stateful*

A **Security Group** EC2 (vagy más erőforrás) szintjén ható tűzfal — ezt a gyakorlaton már beállítottuk inbound/outbound szabályokkal.

Stateful: ha egy bejövő kapcsolatot engedélyezünk, a rá érkező **válaszforgalmat automatikusan is engedi** — nem kell külön outbound szabályt írnunk a válaszhoz.

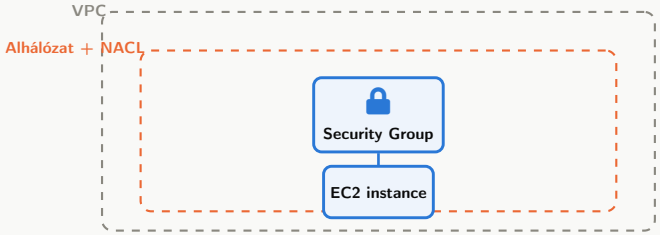
Network ACL (NACL) — *stateless*

A **NACL** alhálózat szintjén ható szűrő — a bejövő és kimenő szabályokat **külön-külön, explicit módon** kell megadni, mert nem emlékszik a kapcsolat állapotára.

	Security Group	NACL
Szint	EC2 / erőforrás	Alhálózat
Állapot	Stateful	Stateless
Szabályok	Csak „engedélyezés”	Engedélyezés **és** tiltás
Válaszforgalom	Automatikus	Külön szabály kell

Mélységi védelem (defense in depth)

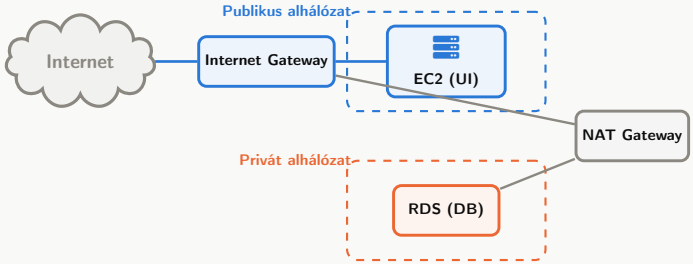
A gyakorlatban ezeket a rétegeket **egymásra rétegezve** érdemes alkalmazni, hogy egyetlen hibásan beállított szabály se jelentsen azonnal teljes kompromittálódást.



Minél **belső** rétegig jut el egy támadó, annál kevésbé valószínű, hogy a következő gyűrűt is áttöri — ez a „ne egyetlen ponton múljon minden" elv.

Publikus/privát alhálózat — gyakorlati recap

A gyakorlaton látott VPC-nek nem minden alhálózata egyforma — ez is a mélységi védelem egyik eszköze.



A DB-nek **nincs közvetlen publikus IP-je/útvonala** — kimenő (pl. frissítés) forgalomhoz a **NAT Gateway**-en keresztül éri csak el az internetet, bejövő kapcsolatot viszont onnan kívülről senki nem tud kezdeményezni felé.

4. BLOKK

Titkosítás

Két teljesen más kérdés: mi történik az adattal, amíg *mozog* a hálózaton — és mi történik vele, amíg *nyugalomban* van egy tárolón.

Nyugalmi vs. adatátviteli titkosítás



Adatátviteli (in transit)

A hálózaton mozgó adat védelme, jellemzően **TLS/HTTPS**-sel — egy publikus alkalmazásnál ma alapkövetelmény, nem opció.

Nyugalmi (at rest)

A tárolt adat titkosítása, jellemzően **AWS KMS** kulcsokkal — EBS, S3, RDS mind támogatja, sokszor egy checkbox-szal.

AWS KMS — kulcstípusok

Az előző dián látott „AWS KMS kulcs” mögött valójában **kétféle** kulcs lehet.

AWS managed key

Ingyenes, az AWS hozza létre és kezeli (rotáció is automatikus) — a legtöbb szolgáltatás alapértelmezett választása, ha csak bepipáljuk a titkosítást.

Customer managed key (CMK)

Mi hozzuk létre, mi szabjuk meg rajta a jogosultságokat (IAM policy!) és a rotációt — ha pl. egy csapattól vissza kell vonni a hozzáférést egy adathoz, ezt csak CMK-nál tehetjük meg finomhangoltan.

A kulcs **maga** is egy IAM-mal védett erőforrás — ugyanaz a legkisebb jogosultság elv vonatkozik rá, mint bármelyik más AWS szolgáltatásra (2. blokk).

ACM — a TLS tanúsítvány automatizálva

A **Certificate Manager (ACM)** ingyenesen ad ki és **automatikusan megújít** TLS tanúsítványokat AWS-es szolgáltatásokhoz (CloudFront, Load Balancer).

⚠ Hagyományos (nem felhős) üzemeltetésben a **lejárt tanúsítvány** az egyik leggyakoribb, teljesen elkerülhető szolgáltatáskiesési ok — valaki elfelejti időben megújítani.

✓ Az ACM ezt a kézi lépést teszi feleslegessé — ezzel biztosítja gyakorlatban az előző dián látott „adatátviteli (in transit)" titkosítást egy publikus webalkalmazásnál.

5. BLOKK

Monitorozás és naplózás

Ha egy védelmi vonal mégis átszakad, honnan tudjuk meg — és mikor?

CloudTrail és GuardDuty

CloudTrail

Minden API-hívást naplóz (ki, mikor, milyen szolgáltatáson, milyen műveletet) — utólagos vizsgálathoz (audit), illetéktelen hozzáférés kiderítéséhez nélkülözhetetlen.

GuardDuty

Gépi tanuláson és fenyegetettségi adatbázisokon alapuló, folyamatos anomália-felismerés — pl. szokatlan API-hívási minták, ismert rosszindulatú IP-ről érkező forgalom.

CloudWatch — amit a 2. órán már láttunk

A 2. órán megismert **CloudWatch** biztonsági szempontból is releváns: riasztásokat (alarm) állíthatunk be szokatlan erőforrás-felhasználásra, ami akár kompromittálódás jele is lehet.



Ez egy tipikus **detektálási lánc**: a napló (CloudTrail) adja az adatot, az anomália-felismerés (GuardDuty) értelmezi, a riasztás (CloudWatch Alarm) pedig cselekvésre — emailre, SMS-re, akár automatikus válaszra — készteti az üzemeltetőt.

AWS Config — folyamatos megfelelőség-ellenőrzés

Míg a CloudTrail **eseményeket** (ki mit csinált) naplóz, az **AWS Config** az erőforrások **aktuális konfigurációját** figyeli folyamatosan, és szabályokkal ellenőrzi.

Példa szabályok: „minden EBS kötet legyen titkosítva”, „egyetlen Security Group se engedje a 22-es portot `0.0.0.0/0`-ról” — eltérés esetén azonnal naplóz, riaszthat.

Ezzel a 7–8. blokkban látott **tipikus hibák egy részét automatikusan, folyamatosan** ellenőrizhetjük — nem csak akkor derül ki, ha már baj lett belőle.

6. BLOKK

DDoS és alkalmazásszintű védelem

Mi történik, ha nem egy belső hiba, hanem egy külső támadás próbálja megbénítani a rendszert?

AWS Shield, WAF, CloudFront

AWS Shield

Hálózati/szállítási rétegbeli (L3/L4) DDoS védelem — alapszinten minden ügyfélnek automatikusan, **ingyenesen** jár.

AWS WAF

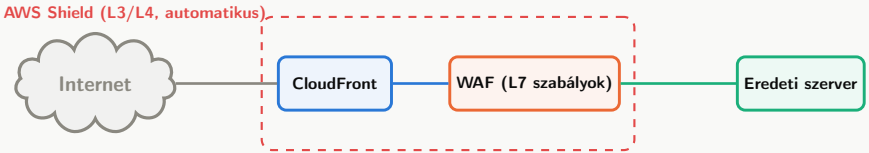
Alkalmazásrétegbeli (L7) szűrés: SQL injection, XSS és egyéb tipikus webes mintázatok ellen, és rate limiting.

CloudFront (CDN)

A felhasználóhoz földrajzilag közelebb elhelyezett gyorsítótár — elrejti az eredeti szervert, csökkenti a támadási felületet.

A három szolgáltatás együtt védi a pert

A Shield és a WAF **nem külön állomás** a hálózaton — mindkettő ugyanahhoz az éli (edge) ponthoz, a CloudFronthoz (vagy egy Load Balancerhez) kapcsolódik.



A **Shield** a CloudFront/ALB infrastruktúráját védi volumetrikus támadások ellen (ezért rajzoltuk köré), a **WAF** ugyanitt, de alkalmazás-szinten szűri a kéréseket — mire a forgalom eléri az eredeti szervert, nagy része már kiszűrve/eltérítve lett, és az eredeti szerver IP-je a gyakorlatban nem is nyilvános.

AWS Shield — Standard vs. Advanced

	Standard	Advanced
Ára	Ingyenes, mindenkinek	Fizetős előfizetés
Védelem	Gyakori L3/L4 DDoS minták	Kifinomultabb, nagy volumenű támadások
Láthatóság	Nincs külön riportálás	Részletes támadás-elemzés, riportok
Support	—	24/7 DDoS Response Team
Költségvédelem	—	Skálázásból adódó extra költség térítése

A Standard **automatikusan, mindenkinél bekapcsolva** fut — az Advanced azoknak éri meg, akiknél egy sikeres DDoS valódi üzleti/pénzügyi kockázatot jelent.

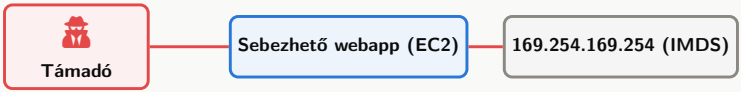
7. BLOKK

Esettanulmány: egy valós támadási lánc

Nem elmélet — nézzük meg, hogyan kapcsolódik össze a mai órán látott 6 téma egyetlen, valóban előforduló típusú incidensben.

1. lépés — sebezhető webalkalmazás + SSRF

A webalkalmazásban van egy **SSRF** (Server-Side Request Forgery) sebezhetőség — a támadó rá tudja venni a szervert, hogy *saját maga nevében* küldjön ki egy HTTP kérést egy általa megadott célra.



rosszindulatú URL-paraméter a szerver *saját magának* kéri le

⚠ A kérés **a szerver belsejéből indul** — sem a Security Group, sem a NACL (3. blokk) nem szűri, hiszen ez nem egy kívülről érkező kapcsolat.

2. lépés — IMDSv1 válaszol, kérdés nélkül

Ha a gép a régi **IMDSv1**-et használja, a metaadat-szolgáltatás **token nélkül** válaszol — a támadó megkapja az EC2-höz rendelt szerepkör (role) aktuális, ideiglenes hitelesítő adatait.



⚠ Innentől a támadó **mindent megtehet**, amire a role jogosult — pl. letöltheti egy bizalmas adatot tartalmazó S3 bucket tartalmát (lásd a 8. blokk „nyilvános S3 bucket" tipikus hibáját — itt most nem is kellett, hogy publikus legyen).

Hol állt volna meg — 1. védelmi vonal

Az **IMDSv2** megköveteli, hogy előbb egy külön `PUT` kéréssel kérjünk egy ideiglenes tokent, és **csak ezzel a tokennel** fogadja a metaadat-lekérdezést.

Egy egyszerű, csak továbbküldést (GET) megengedő SSRF sebezhetőség emiatt sok esetben **nem elég** ehhez a támadáshoz — a `PUT` kérés továbbítása jellemzően bonyolultabb kihasználni.

Éppen ezért szerepelt ma már egyszer, a 8. blokk tipikus hibái között is — ez az egyik legjobb ár/érték arányú védelmi beállítás, amit egyetlen EC2 opcióval bekapcsolhatunk.

Hol állt volna meg — 2. védelmi vonal

Ha az IMDSv2 ellenére mégis sikerülne a hitelesítő-lopás, a **legkisebb jogosultság elve** (2. blokk) korlátozza a kárt.

Ha a role csak egyetlen, konkrét bucket **olvasására** jogosult (gondoljunk vissza az IAM policy Resource mezőjére), a támadó nem tud más erőforráshoz hozzáférni, és nem tud írni/törölni sem.

A **GuardDuty** (5. blokk) kifejezetten figyeli az IMDS-hez kapcsolódó szokatlan mintákat és a lopott hitelesítővel végzett, szokatlan helyről érkező API-hívásokat — riasztást kapunk akkor is, ha minden korábbi réteg elbukott.

Mit tanulhatunk belőle

- Egyetlen hiba (itt: IMDSv1) **ritkán önmagában** katasztrófa
- A védelem mindig **több réteg együttes** hatása — amit az egyik réteg elenged, a következő megállítja vagy legalább észreveszi
- Ez pontosan a 3. blokk „mélységi védelem" gondolata — csak most egy konkrét, végigkövetett példán
- A 8. blokkban következő „tipikus hibák" lista nem elvont — mindegyik mögött ilyen, ehhez hasonló valós láncolat áll

8. BLOKK

Gyakorlati tanulságok

Elmélet helyett: mi az, ami a valóságban tényleg elromlik — és hogyan igazolja ezt egy szolgáltató a jog felé?

Tipikus felhős biztonsági hibák

Nyilvános S3 bucket

Bizalmas adatot tartalmazó bucket publikusan elérhetővé állítva — a leggyakoribb valós incidens-forrás.

Túl megengedő Security Group

Pl. `0.0.0.0/0` minden portra nyitva — pont az ellentéte annak, amit a 3. blokkban gyakoroltunk.

Hardcode-olt Access Key

Forráskódba/verziókezelőbe kerülő hozzáférési kulcsok — lásd a 2. blokk MFA/Access Key diáját.

Elavult IMDSv1

Az EC2 metaadat-szolgáltatás régi (v1) verziója SSRF-sebezhetőség esetén jogosultsági adatokat szivárogtathat — lásd a 7. blokk esettanulmányát.

Megfelelőség (compliance)

A felhőszolgáltatók iparági/jogi szabványoknak (**ISO 27001**, **SOC 2**, **PCI-DSS**) való megfelelést tanúsítanak — de ez csak a megosztott felelősség modell **szolgáltatói** felét fedi le.

⚠ Hogy egy konkrét alkalmazás/adatkezelés megfelel-e például a **GDPR**-nak, továbbra is az üzemeltető felelőssége — a megfelelő **régióválasztástól** a hozzáférés-szabályozásig.

ℹ Ugyanaz a gondolat, mint az 1. órán: a nagy szolgáltatók régió szintjén garantálják, hogy az adat a kiválasztott adatközpont-régión belül marad, de ezt **tudatosan be kell állítani** — nem elég „belefutni" az alapértelmezett beállításokba.

Biztonsági eszköztár — gyors referencia

Szolgáltatás	Mire való	Blokk
IAM	Ki, mihez férhet hozzá (felhasználó, role, policy)	2
KMS	Nyugalmi állapotú (at rest) titkosítási kulcsok	4
ACM	TLS tanúsítvány kiadása, automatikus megújítás	4
Security Group	Stateful tűzfal, erőforrás (pl. EC2) szinten	3
NACL	Stateless szűrő, alhálózat szinten	3
CloudTrail	Minden API-hívás naplózása (audit)	5
GuardDuty	Gépi tanulásos anomália-/fenyegetésfelismerés	5
AWS Config	Erőforrás-konfiguráció folyamatos ellenőrzése	5
Shield / WAF / CloudFront	DDoS- és alkalmazásrétegbeli (L7) védelem	6

Egyik szolgáltatás sem helyettesíti a másikat — az esettanulmányban (7. blokk) pont az látszott, hogy egyszerre **több sorból** is kellett védelem ahhoz, hogy a támadás ne okozzon érdemi kárt.

ÖSSZEFOGLALÁS

Mit vittünk haza?

- A **megosztott felelősség modell** IaaS esetén (ez a mi AWS-használatunk) a legtöbb réteget a mi felelősségünkbe utalja
- **IAM**: legkisebb jogosultság elve (policy = Effect/Action/Resource), szerepkör statikus kulcs helyett, root fiók csak vészhelyzetre
- **Hálózat**: Security Group (stateful) + NACL (stateless) + publikus/privát alhálózat, rétegezve (defense in depth)
- **Titkosítás**: at rest (KMS, AWS vs. customer managed) és in transit (TLS, automatizálva ACM-mel)
- **Monitorozás**: CloudTrail (napló) → GuardDuty (anomália) → CloudWatch (riasztás), mellette AWS Config (folyamatos megfelelőség)
- **DDoS/alkalmazásvédelem**: Shield (Standard/Advanced), WAF, CloudFront — az eredeti szerver elrejtésével
- Az esettanulmány megmutatta: egyetlen hiba (IMDSv1) ritkán végzetes — a **rétegek együtt** állítják meg/veszik észre a támadást
- A valós incidensek döntő többsége **konfigurációs hiba**, nem a szolgáltató hibája — és a megfelelőség (GDPR stb.) ettől külön kérdés

KÖVETKEZŐ ÓRA: DOCKER

Kérdések?

A megosztott felelősség modelltől az IAM-en, a hálózati védelmen és a titkosításon át a monitorozásig, a DDoS-védelemig és egy végigkövetett támadási láncig — és mi szokott a valóságban elromlani.