DEPARTMENT OF
NETWORKED SYSTEMS
AND SERVICES

Biztonságos szoftverfejlesztés

VIHIBB01 – Kódolás és IT biztonság, 2023

Gazdag András

CrySyS Lab, BME
andras.gazdag@crysys.hu

www.crysys.hu

MŰEGYETEM 1782

SZOFTVERBIZTONSÁG MANAPSÁG

Mik a kihívások?

1. Lejt a pálya

Fejlesztői megkötések: idő, erőforrások, funkcionalitás

Támadói megkötések: motiváció és felkészültség

Mik a kihívások?

1. Lejt a pálya

Fejlesztői megkötések: idő, erőforrások, funkcionalitás

Támadói megkötések: motiváció és felkészültség

2. Biztonsági tesztelés maga

Funkcionális tesztelés: hogyan kéne működnie a rendszernek

Biztonsági tesztelés: hogyan NEM kéne működnie a rendszernek

Mik a kihívások?

1. Lejt a pálya

Fejlesztői megkötések: idő, erőforrások, funkcionalitás

Támadói megkötések: motiváció és felkészültség

2. Biztonsági tesztelés maga

Funkcionális tesztelés: hogyan kéne működnie a rendszernek

Biztonsági tesztelés: hogyan NEM kéne működnie a rendszernek

3. Gyenge üzleti motiváció

Nehéz mérni a biztonságot → a vevők nem kényszerítik ki a versenyt

Mik a kihívások?

1. Lejt a pálya

Fejlesztői megkötések: idő, erőforrások, funkcionalitás

Támadói megkötések: motiváció és felkészültség

2. Biztonsági tesztelés maga

Funkcionális tesztelés: hogyan kéne működnie a rendszernek

Biztonsági tesztelés: hogyan NEM kéne működnie a rendszernek

3. Gyenge üzleti motiváció

Nehéz mérni a biztonságot → a vevők nem kényszerítik ki a versenyt

4. Végfelhasználót éri kár

A fejlesztők nem elég motiváltak

Akkor miért is vagyunk itt?

- Nyerhetetlen harcnak tűnik,
- Ami hatalmas befektetést igényel,
- Miközben nem tudjuk tisztán mérni, hogy mennyire végeztünk jó munkát

Akkor miért is vagyunk itt?

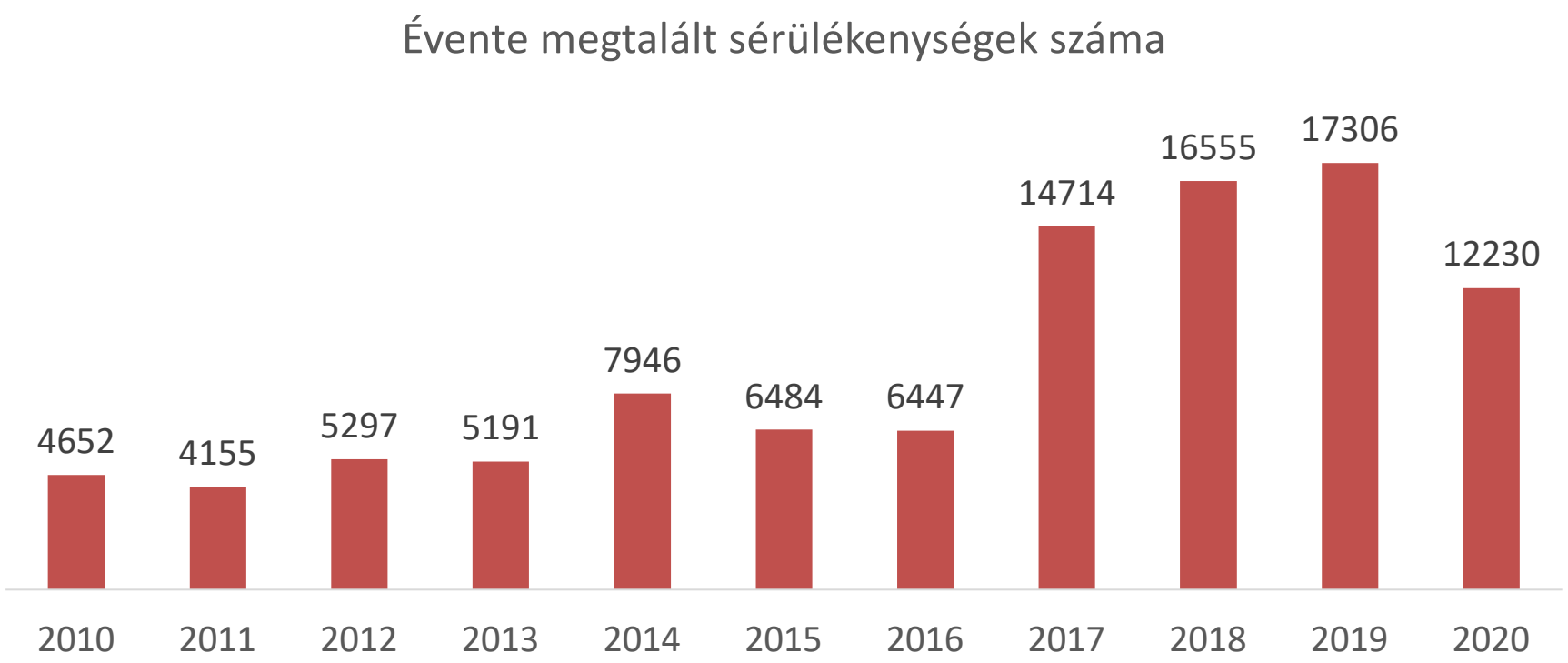
- Nyerhetetlen harcnak tűnik,
- Ami hatalmas befektetést igényel,
- Miközben nem tudjuk tisztán mérni, hogy mennyire végeztünk jó munkát

- Az incidensek 90% ugyan azokra az okokra vezethetők vissza
- Ezeket az okokat ismerjük és értjük!
- → Hatásos, olcsó és célzott védekezési mechanizmusok

- Elsőre jól csinálni nem kerül semmibe ☺

Common Vulnerabilities and Exposures

- CVE: nyilvános adatbázis a megtalált sérülékenységekről



- ~13-22 sérülékenységet fedeznek fel minden nap (!) átlagosan

Common Vulnerabilities and Exposures

- CVE: nyilvános adatbázis a megtalált sérülékenységekről

CVE-ID	
CVE-2017-16530	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
The uas driver in the Linux kernel before 4.13.6 allows local users to cause a denial of service (out-of-bounds read and system crash) or possibly have unspecified other impact via a crafted USB device, related to drivers/usb/storage/uas-detect.h and drivers/usb/storage/uas.c.	
References	
Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.	
• MISC:https://github.com/torvalds/linux/commit/786de92b3cb26012d3d0f00ee37adf14527f35c4 • MISC:https://groups.google.com/d/msg/syzkaller/pCswO77gRlM/VHuPOftgAwAJ	
Assigning CNA	
MITRE Corporation	
Date Entry Created	
20171103	Disclaimer: The entry creation date may reflect when the CVE ID was allocated or reserved, and does not necessarily indicate when this vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE.
Phase (Legacy)	
Assigned (20171103)	

BIZTONSÁGOS FEJLESZTÉSI ÉLETCIKLUS

Szoftverfejlesztési életciklusok

- A szoftverfejlesztés egy folyamat
- Sokféle tevékenységből áll

- Hogyan menedzseljük ezeket a tevékenységeket?
- → Életciklus modellek
  - Szekvenciális: vízesés, V-modell
  - Inkrementális, iteratív, spirális
  - Agilis metodológiák: scrum, extreme programming, lean, stb.

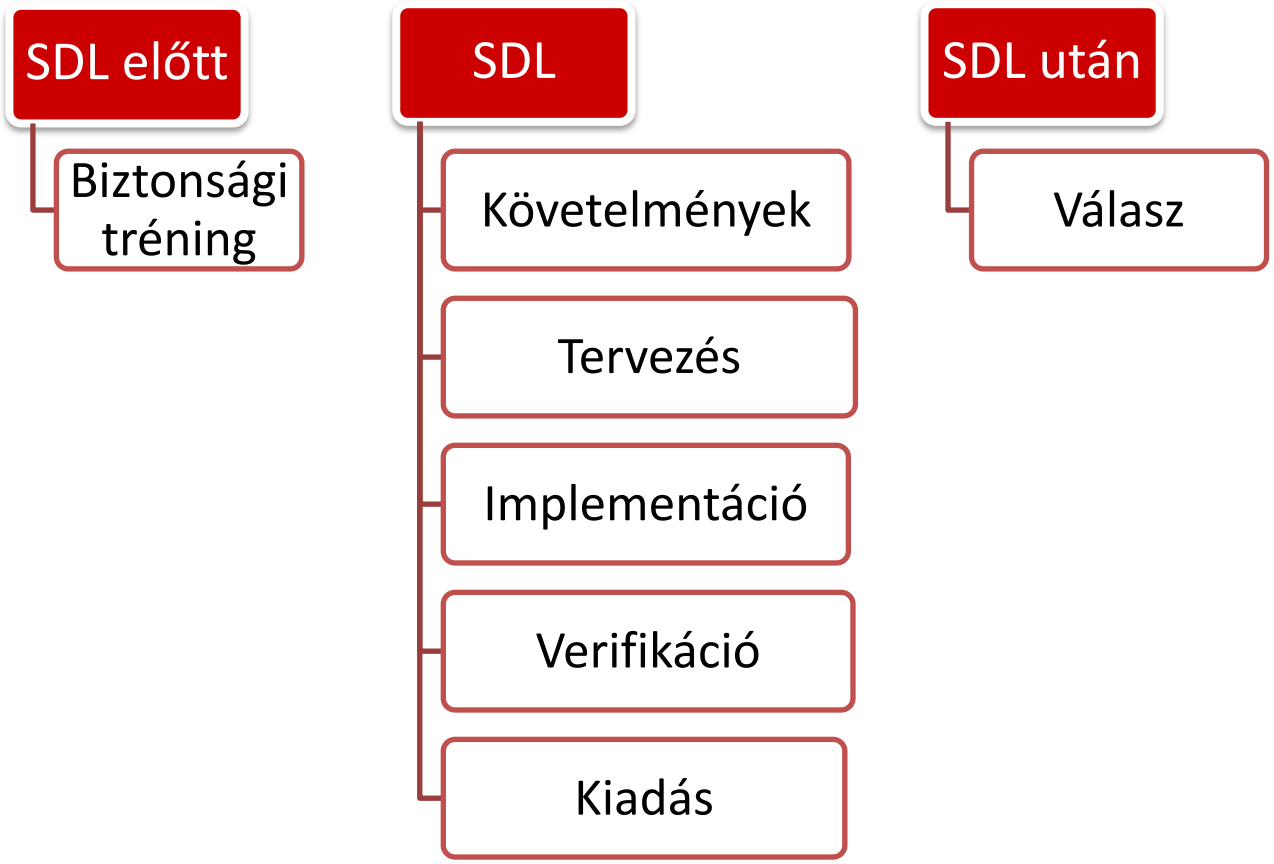
Szoftverfejlesztési életciklusok

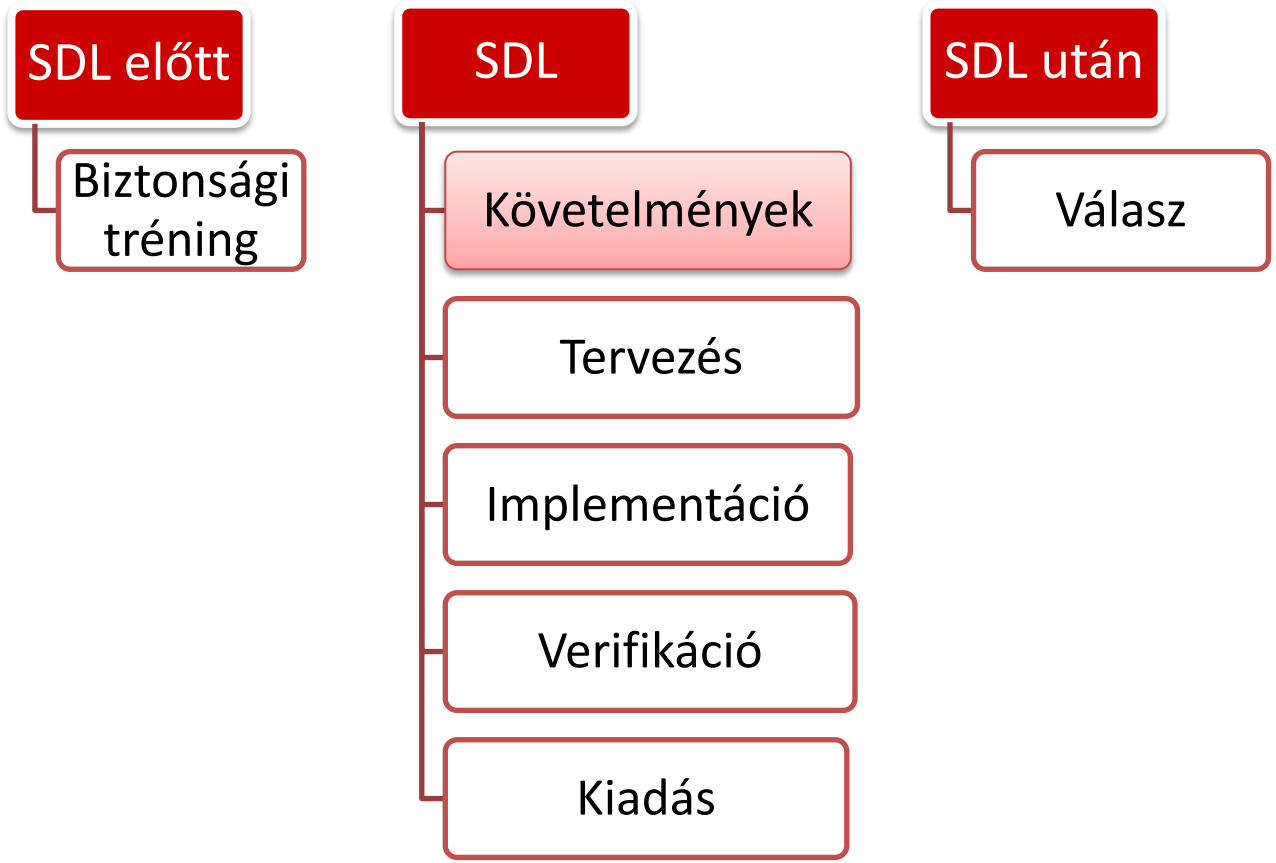
- Biztonsággal kapcsolatos tevékenységek nem jelennek meg!

- → Biztonságos szoftverfejlesztési életciklusok
  - Teljesen átalakított folyamat
    - » OWASP CLASP
    - » **Microsoft SDL**
    - » Software Security Touchpoints
  - Integrálandó tevékenységek listája
    - » OpenSAMM
    - » BSIMM
    - » SAFECode

Microsoft SDL: Áttekintés

- Csökkentsük a sérülékenységek számát fejlesztés közben
- Csökkentsük a meg nem talált biztonsági bugok súlyosságát

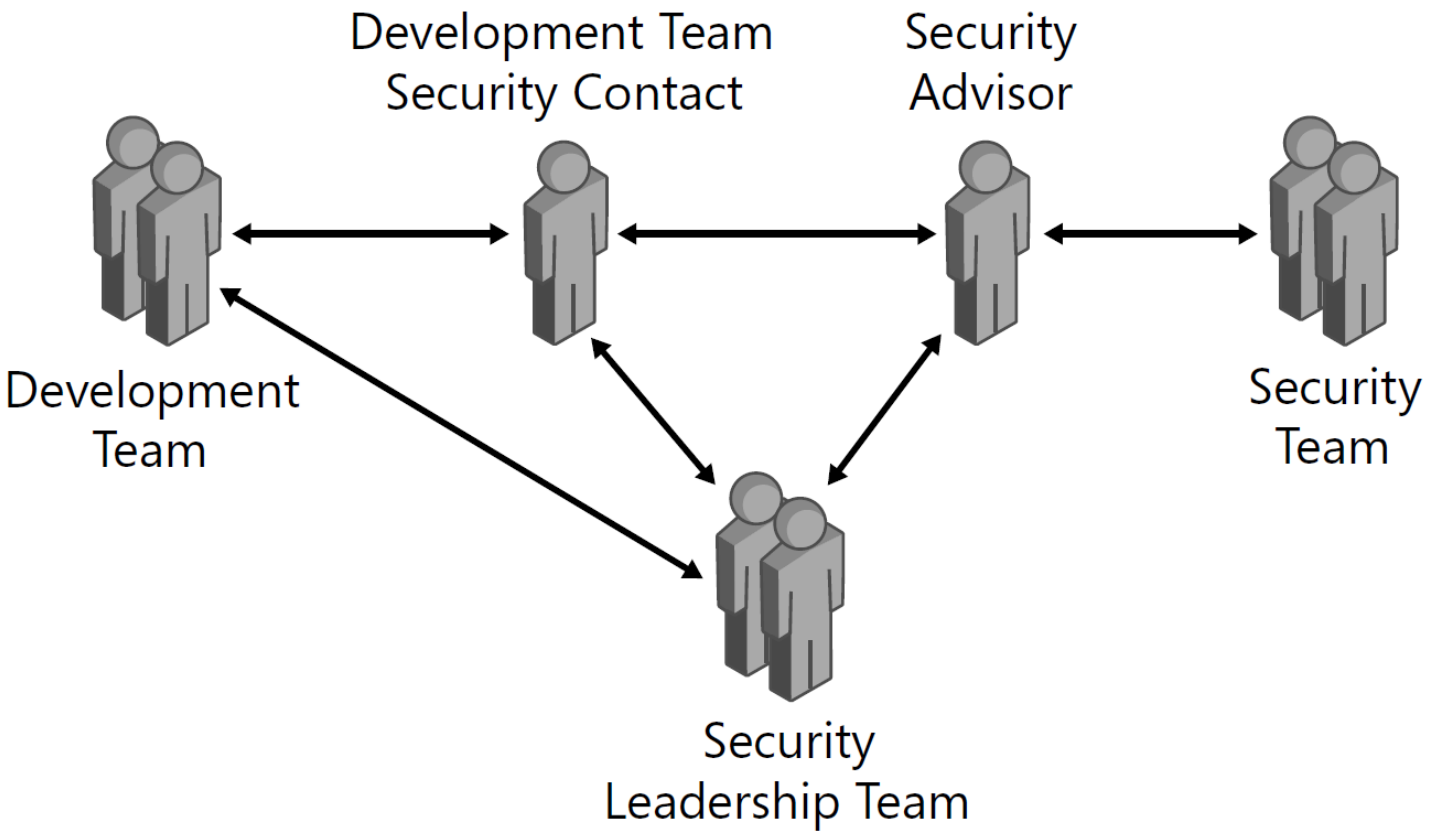




KÖVETELMÉNYEK FÁZIS

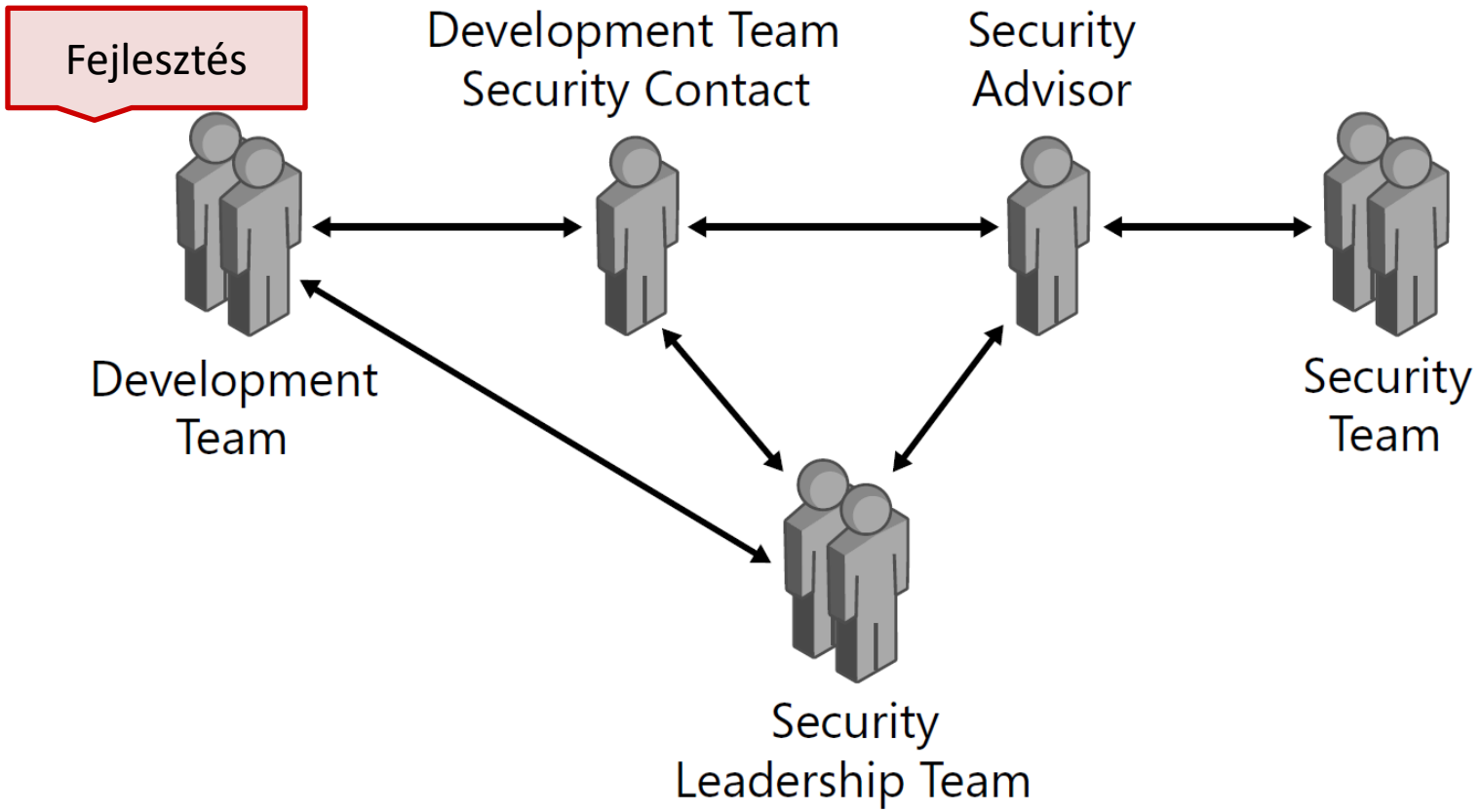
SDL: Követelmények fázis

Biztonság, mint szempont, képviselete



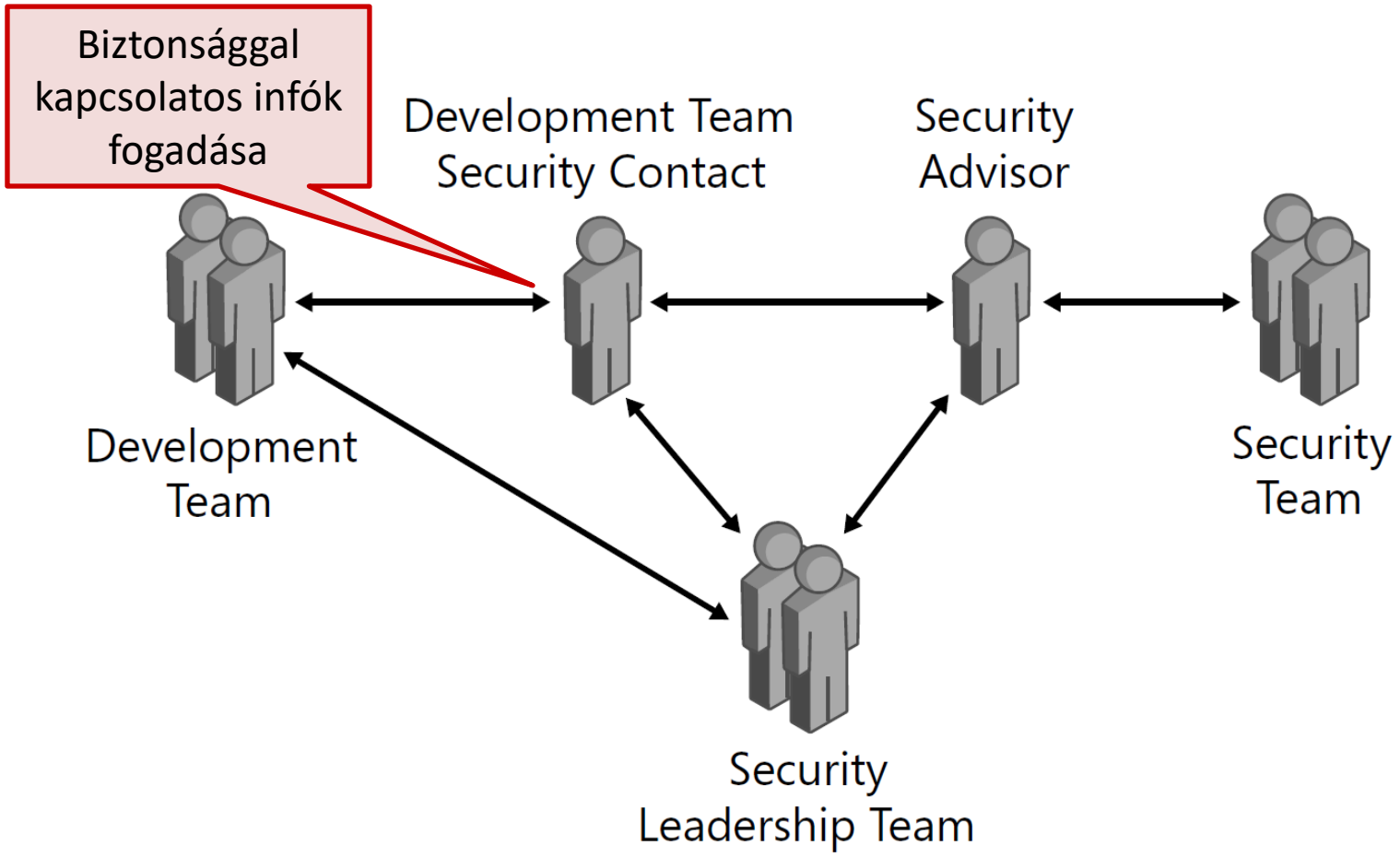
SDL: Követelmények fázis

Biztonság, mint szempont, képviselete



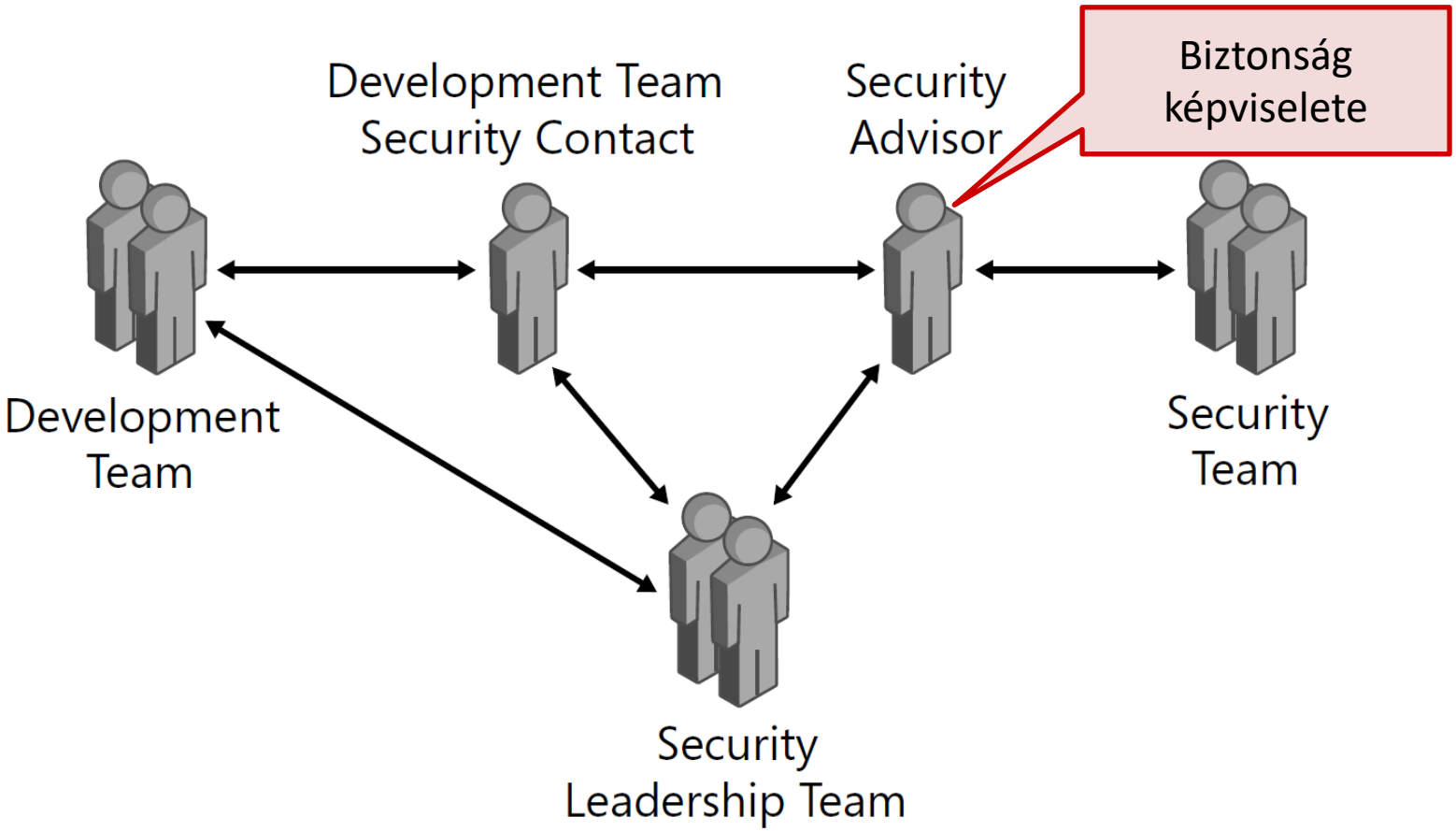
SDL: Követelmények fázis

Biztonság, mint szempont, képviselete



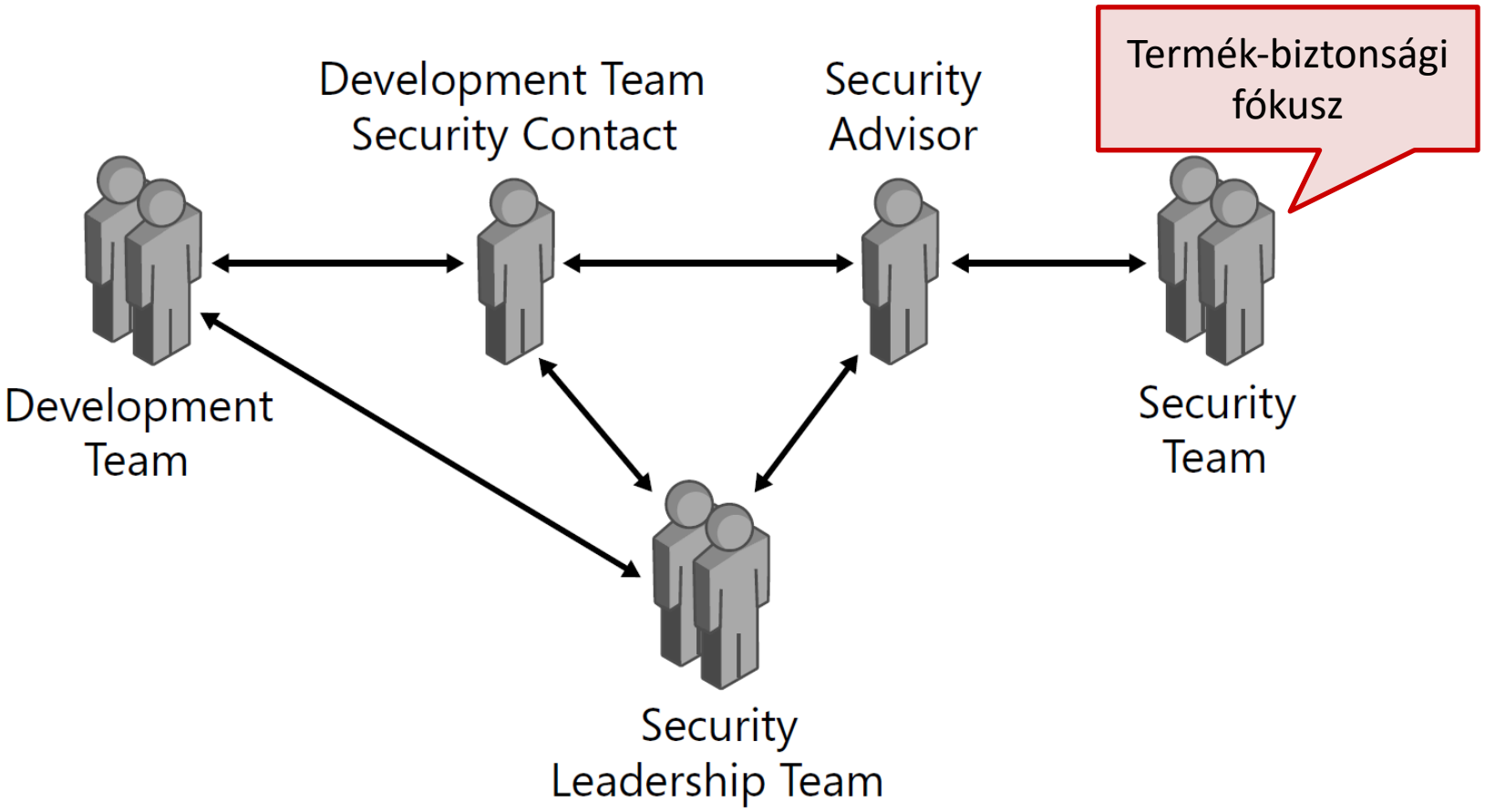
SDL: Követelmények fázis

Biztonság, mint szempont, képviselete



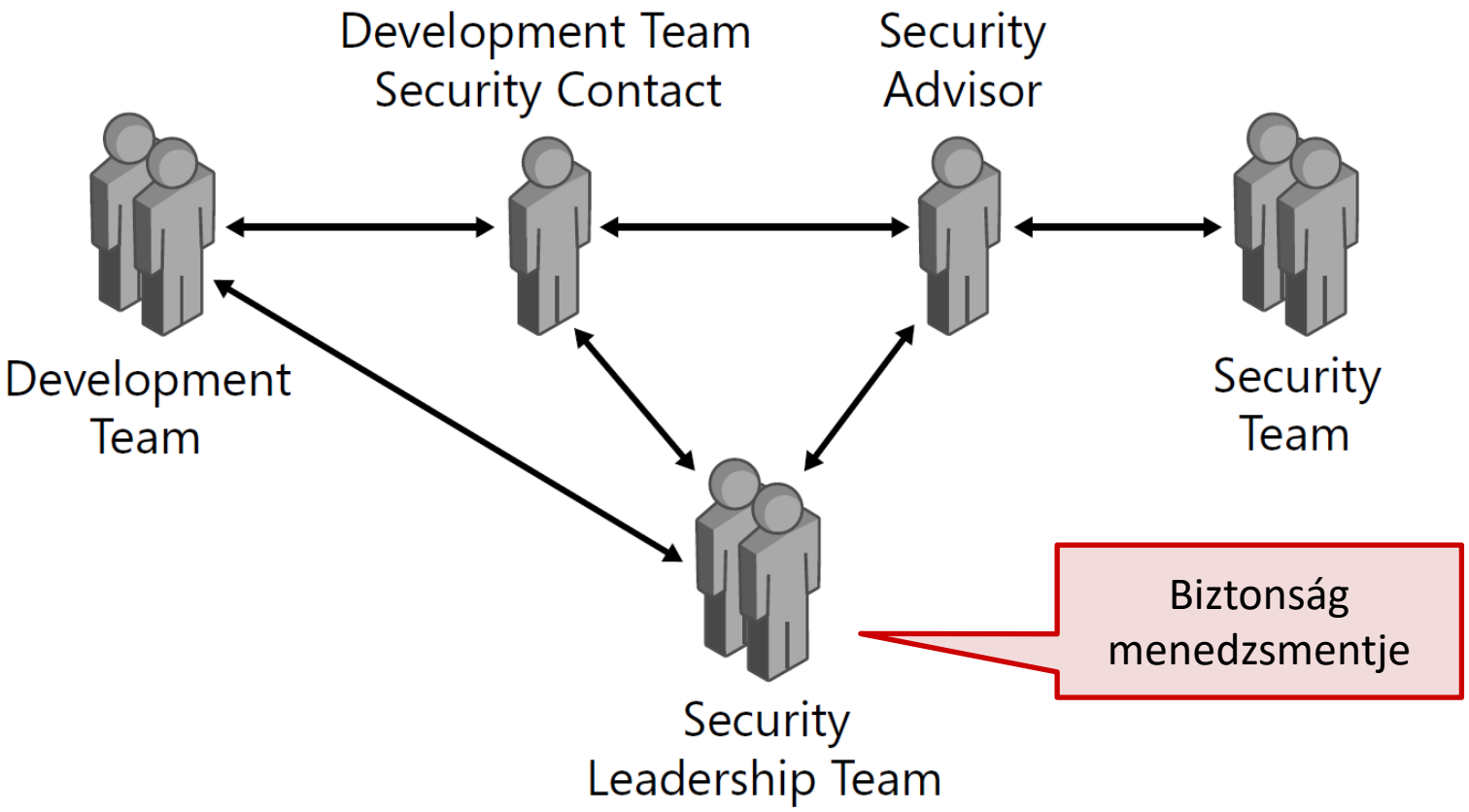
SDL: Követelmények fázis

Biztonság, mint szempont, képviselete



SDL: Követelmények fázis

Biztonság, mint szempont, képviselete



SDL: Követelmények fázis

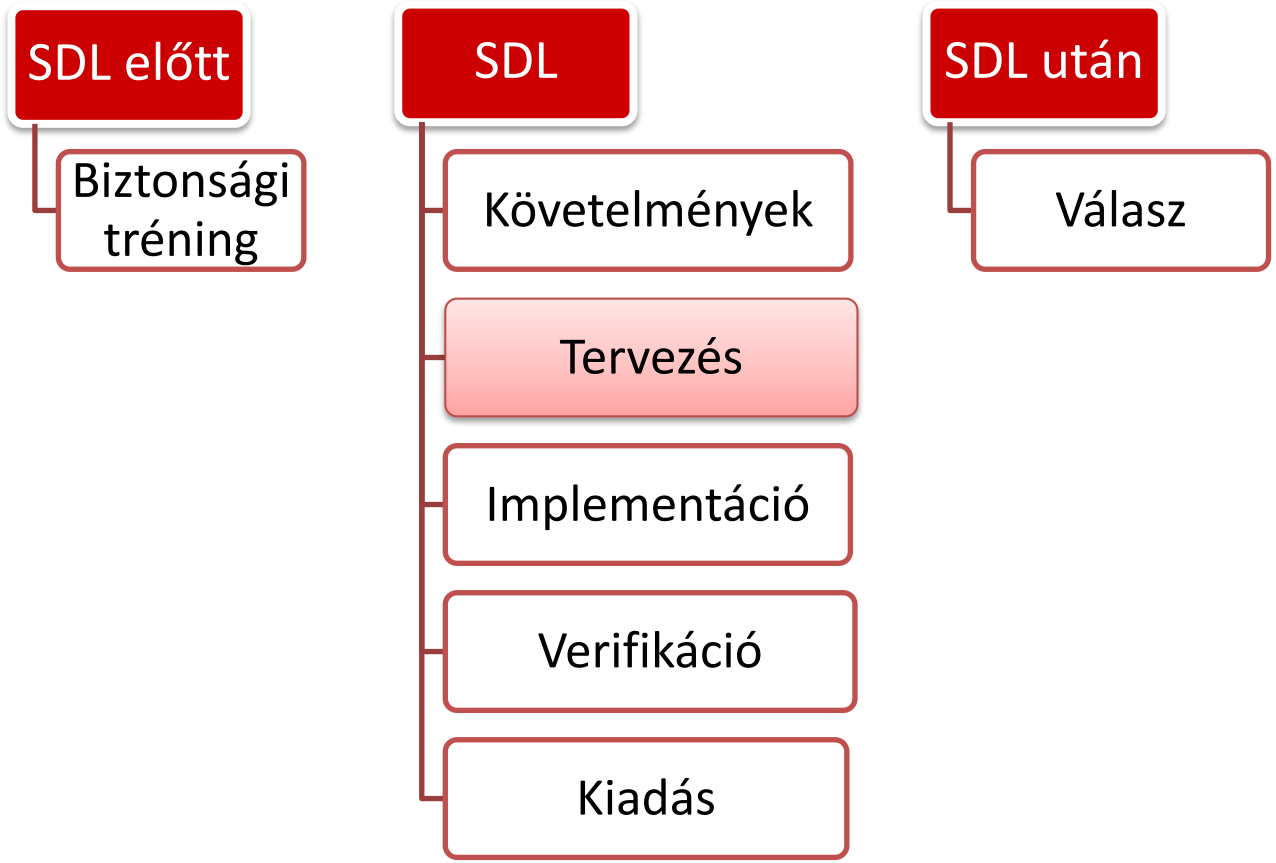
Minimálisan elérendő biztonsági és privacy kritériumok

- Threat, Risk, Vulnerability Analysis
  - Cél: dokumentáljuk a biztonsági mechanizmusok beépítésének indokait
    → Azonosított veszélyforrások és biztonsági/privacy követelmények
    → Szükséges biztonsági funkcionalitások listája
  - Általában brainstromingoló megbeszélések keretei között
  - Ha a rendszer változik, újra kell csinálni!

SDL: Követelmények fázis

Bug-követő rendszer felállítása

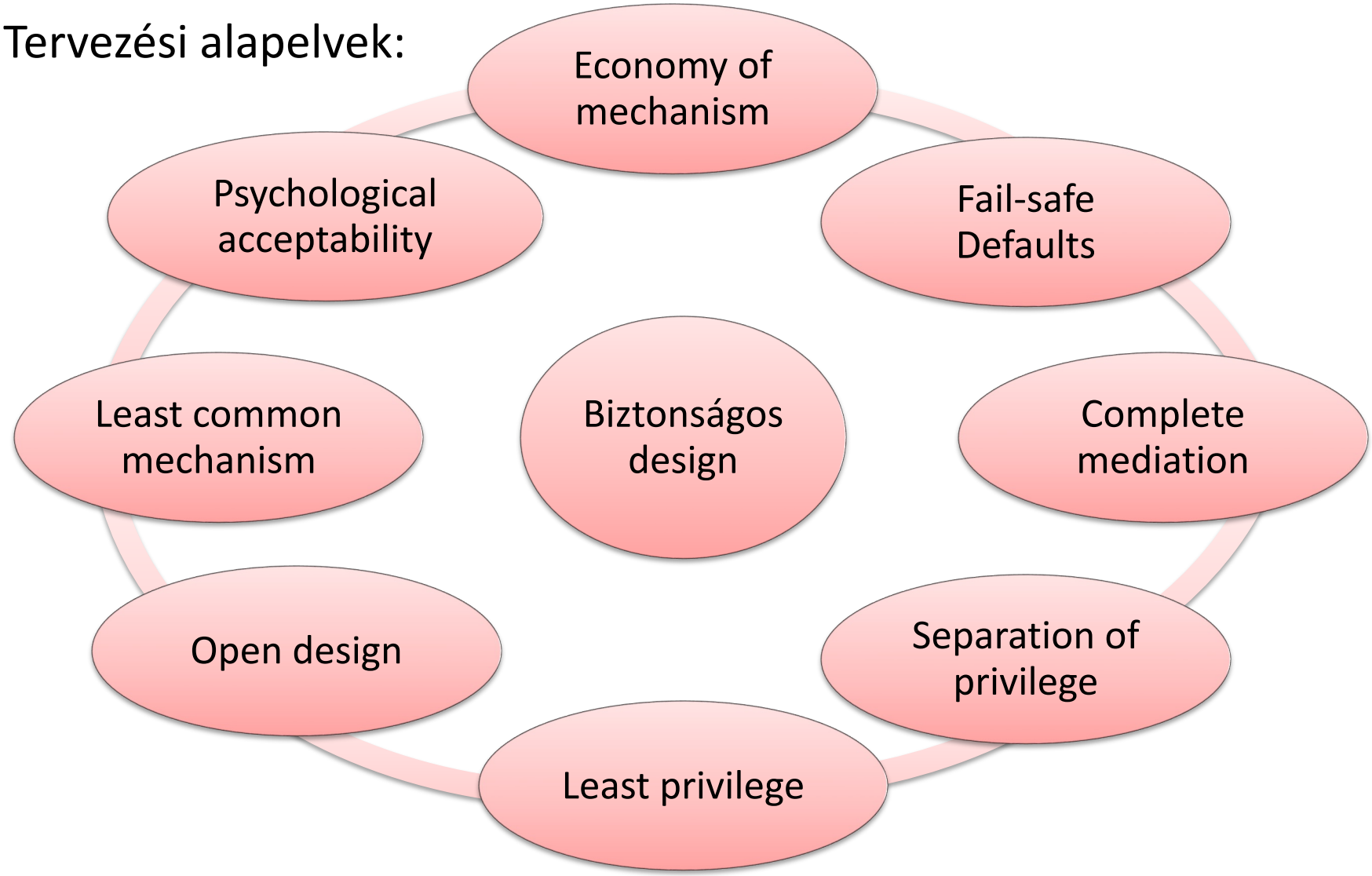
- Bugokat tartalmazó adatbázis
- Biztonsággal/privacy-vel kapcsolatos infókat is csatoljunk!
- Szükséges mezők: `Ok`, `Hatás`



TERVEZÉS FÁZIS

SDL: Tervezés fázis

Tervezési alapelvek:



SDL: Tervezés fázis – Open Design

- Ne bízz a tervek titkosságában
  - Szaknyelven: security by obscurity

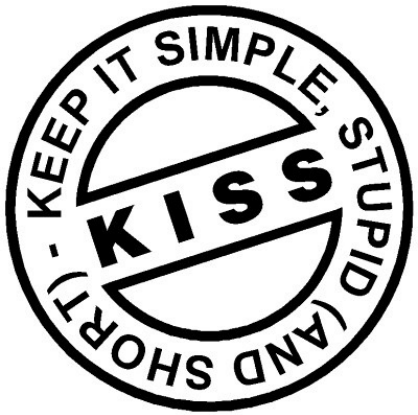
- A terveket elérhetővé kell tenni, hogy minél szélesebb körben elemezhessék
- Több szem többet lát

SDL: Tervezés fázis – KISS

- Minél komplexebb a szoftver, annál valószínűbbek a bugok
- Kisebb kódbázist könnyebb karbantartani

- Amerikai haditengerészettől ered
- Kelly Johnson repülőgép-mérnökhöz kötik

- A „kicsit” sose az egyszerűség rovására érjük el!



SDL: Tervezés fázis – Least Common Mech.

- Minimalizáljuk azon mechanizmusok számát, amik
  1. Több felhasználó számára is elérhetőek, és
  2. Minden felhasználó számára szükségesek

- A megosztás egy információ átviteli csatorna
- Különböző mechanizmusok (akár különböző instance-ek!) nagyobb rugalmasságot nyújtanak

- Példa: folyamatok memóriájának szeparációja

SDL: Tervezés fázis – Fail-Safe Defaults

- Mi van, ha valami nem a várakozásoknak megfelelően történik?

- Feketelista
  - Alapértelmezés: a hozzáférés **megengedett**
  - **Kivéve**, ha van olyan szabály, ami tiltja
- → Fals pozitív döntések
  - A hozzáférést megadjuk, pedig nem kellett volna

SDL: Tervezés fázis – Fail-Safe Defaults

- Mi van, ha valami nem a várakozásoknak megfelelően történik?

- Feketelista
- → Fals pozitív döntések
  - A hozzáférést megadjuk, pedig nem kellett volna
  - **A felhasználó nem fogja jelezni!**

- Fehérlista
  - Alapértelmezés: hozzáférés **megtagadva**
  - Kivéve, ha valamilyen szabály engedi
- → Fals negatív döntések
  - Megtagadjuk a hozzáférést, pedig engedni kellett volna
  - **Ezt jelezni fogja a felhasználó ☺**

SDL: Tervezés fázis – Complete Mediation

- Ellenőrizzünk minden elérést minden objektumhoz
- Szkepticizmus: meg kéne engednem?

- Sok rendszer cache-elni a „hozzáférés engedélyezve” eredményt, majd a cache-t használja adott ideig
- Mi van, ha közben visszavonták az engedélyt??

SDL: Tervezés fázis – Least Privilege

- Csak annyi jogosultságot használjunk, ami feltétlenül szükséges a feladat elvégzéséhez
- → Kisebb kárt okoznak a balesetek és hibák
  - Kódot szúrt be a támadó → a fertőzött program jogosultságaival fog futni
- Magasabb jogosultságot csak időlegesen szerezzünk!
- Példák: hadászati ”need-to-know”, sandboxok

SDL: Tervezés fázis – Separation of Priv.

- Többféle feltételnek is tegyenek eleget, mielőtt megadjuk az engedélyt
- → Robusztusabb és rugalmasabb rendszer
  - Egyetlen ellenőrzés: kijátszható, kudarcot vallhat

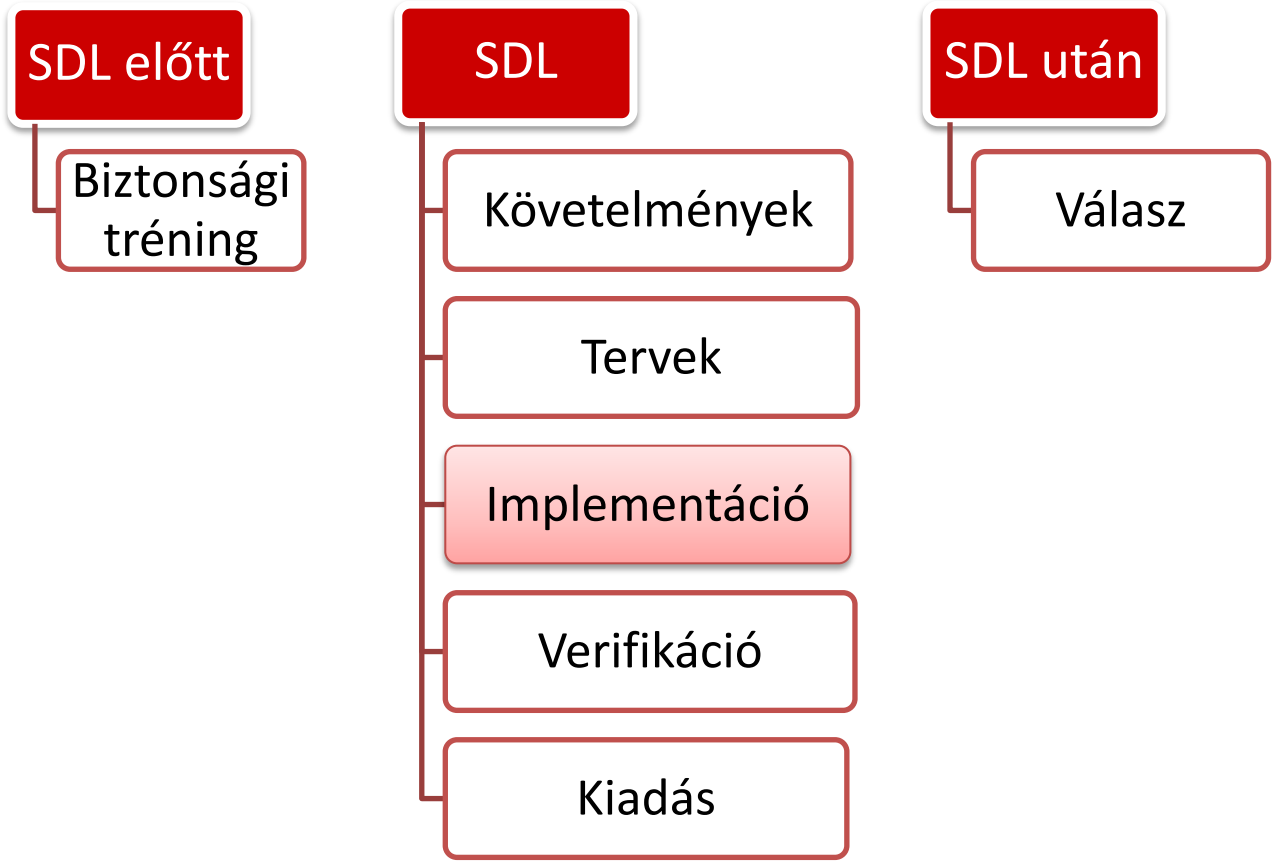
- Előfeltétel: részekre osztás
  - Bontsuk fel a rendszert kisebb egységekre
  - → Minden egység más-más feltételt ellenőrizhet
- Példák: kétfaktoros autentikáció, UNIX `sudo su`

SDL: Tervezés fázis – Psychological Accept.

- Ha a felhasználók nem fogadják el, meg fogják próbálni kikerülni
- Figyeljünk az emberi tényezőre!

- A rendszer használata legyen egyszerű és intuitív
- Az erőforrás-hozzáférés maradjon egyszerű

- Példa: ssh login privát-publikus kulcspárral



IMPLEMENTÁCIÓ FÁZIS

SDL: Implementáció fázis

- Kezdjünk kódolni!
- Ugyan mi sikerülhet félre?

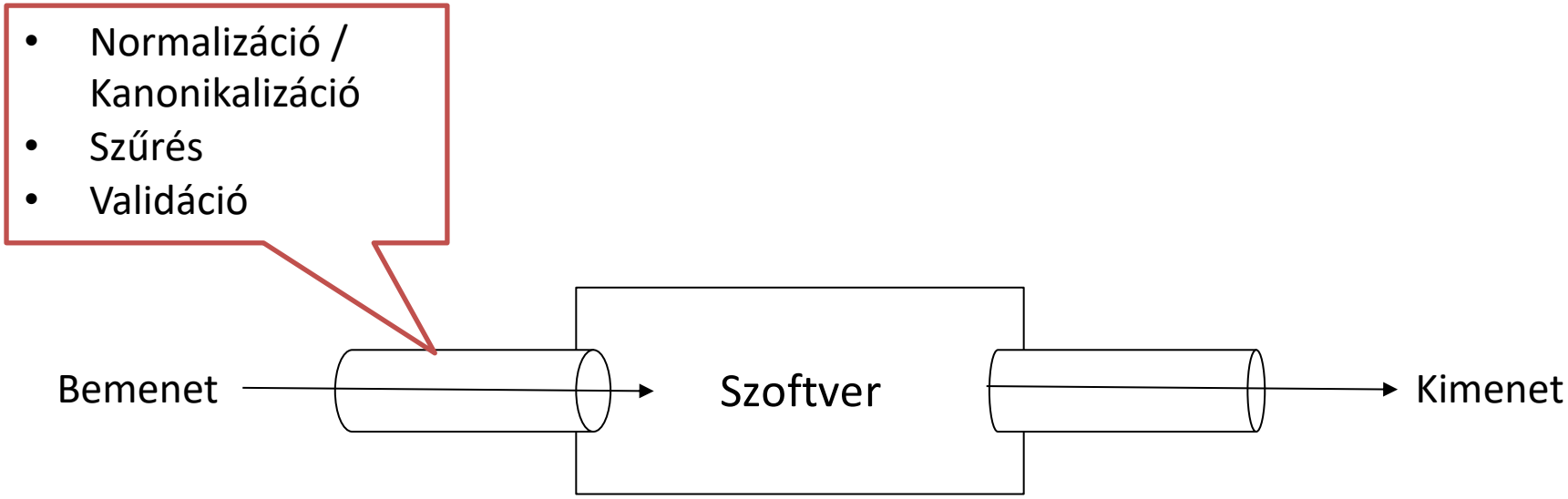
SDL: Implementáció fázis

- Kezdjünk kódolni!
- Ugyan mi sikerülhet félre?
  - Input validáció
  - Hibakezelés
  - Naplózás
  - Versenyhelyzet
  - Stb.

- Vegyük figyelembe a szoftver **támadási** felületét!
  - Minden végrehajtási út, min keresztül adat/parancs érkezik/távozik
  - Kód, ami ezeket a végrehajtási utakat védi
  - Minden értékes adat, amit a szoftver használ
  - Kód, ami ezeket védi

SDL: Implementáció fázis – Adatfeldolgozás

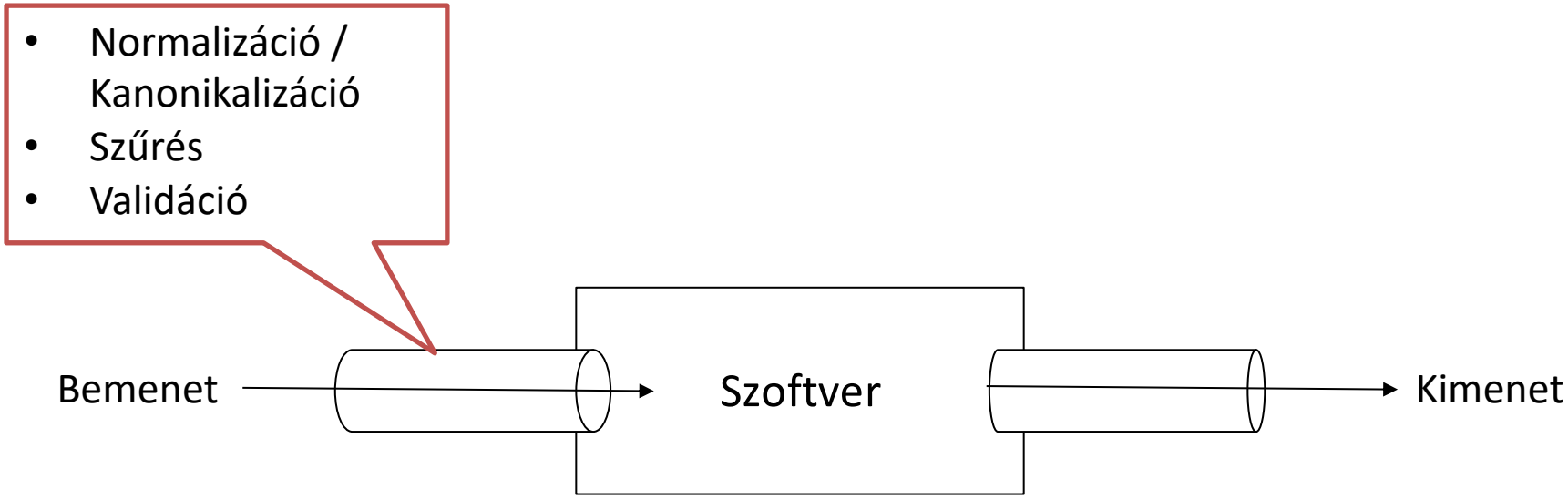
- Bemenetet a felhasználó kontrollálja → **megbízhatatlan**



- Normalizáció / kanonikalizáció: átalakítás a legegyszerűbb (és elvárt) formára
  - Pl. minden sztring Unicode-dá alakítása, üres XML tagek

SDL: Implementáció fázis – Adatfeldolgozás

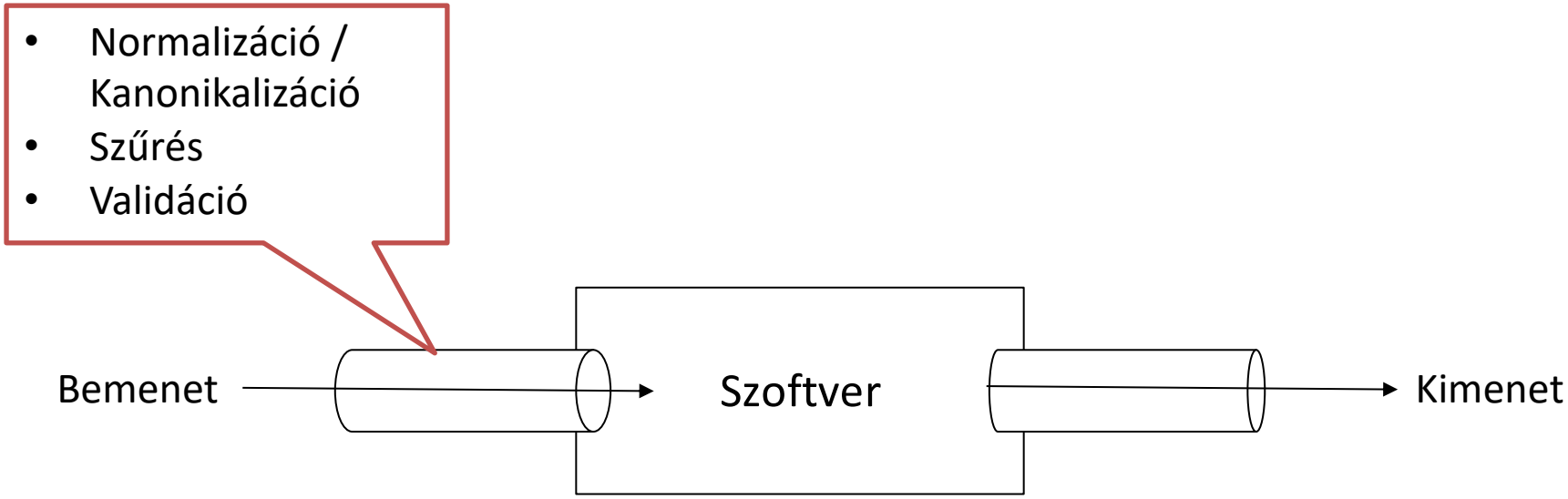
- Bemenetet a felhasználó kontrollálja → **megbízhatatlan**



- Szűrés: biztonyos kritériumok alapján egyes elemek eltávolítása a bementből (nem feltétlen biztonsági okból)
  - Pl. < és > karakterek eltávolítása webes formból származó input esetén

SDL: Implementáció fázis – Adatfeldolgozás

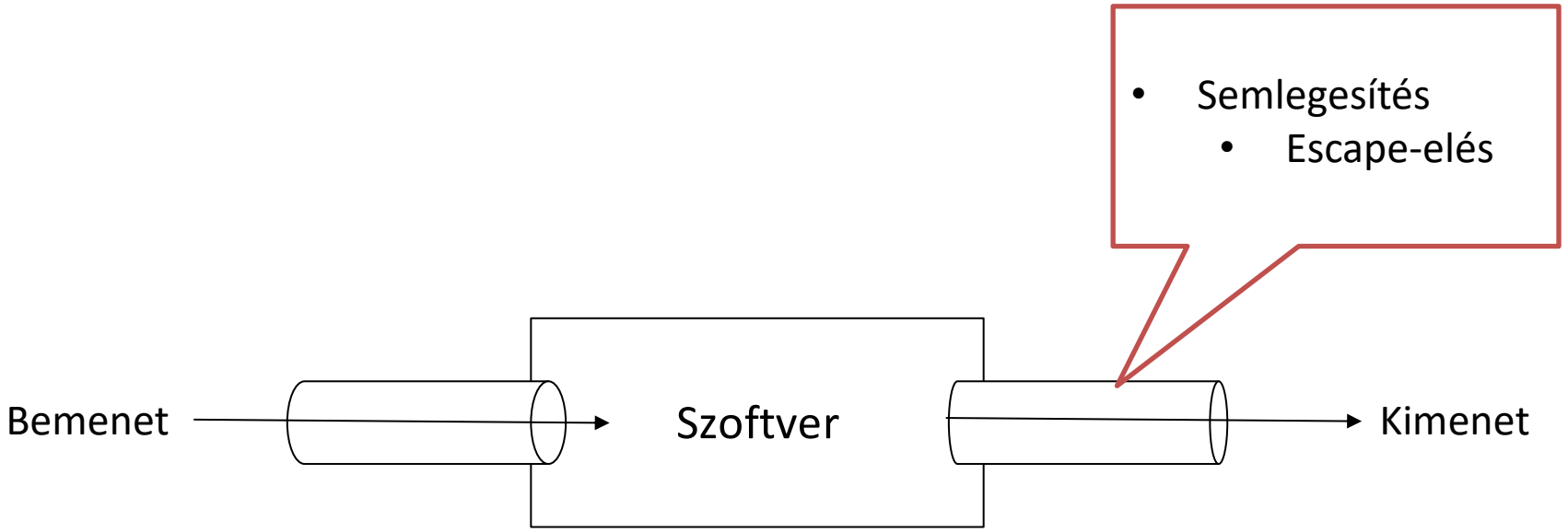
- Bemenetet a felhasználó kontrollálja → **megbízhatatlan**



- Validáció: az bemenet ésszerűségének ellenőrzése (ide értve létezést, típust, korlátokat, konzisztenciát, stb.)
  - Pl. `YYY-MM-DD` formátumban érkező dátum → struktúra + szemantika

SDL: Implementáció fázis – Data Processing

- Kimenetet más komponens felhasználhat → **lépcső**



- Semlegesítés: valamilyen biztonsági kritériumnak megfelelő kimenet előállítása
  - Escape-elés: olyan kimenet előállítása, amiben nincsenek káros elemek

SDL: Implementáció fázis – Naplózás

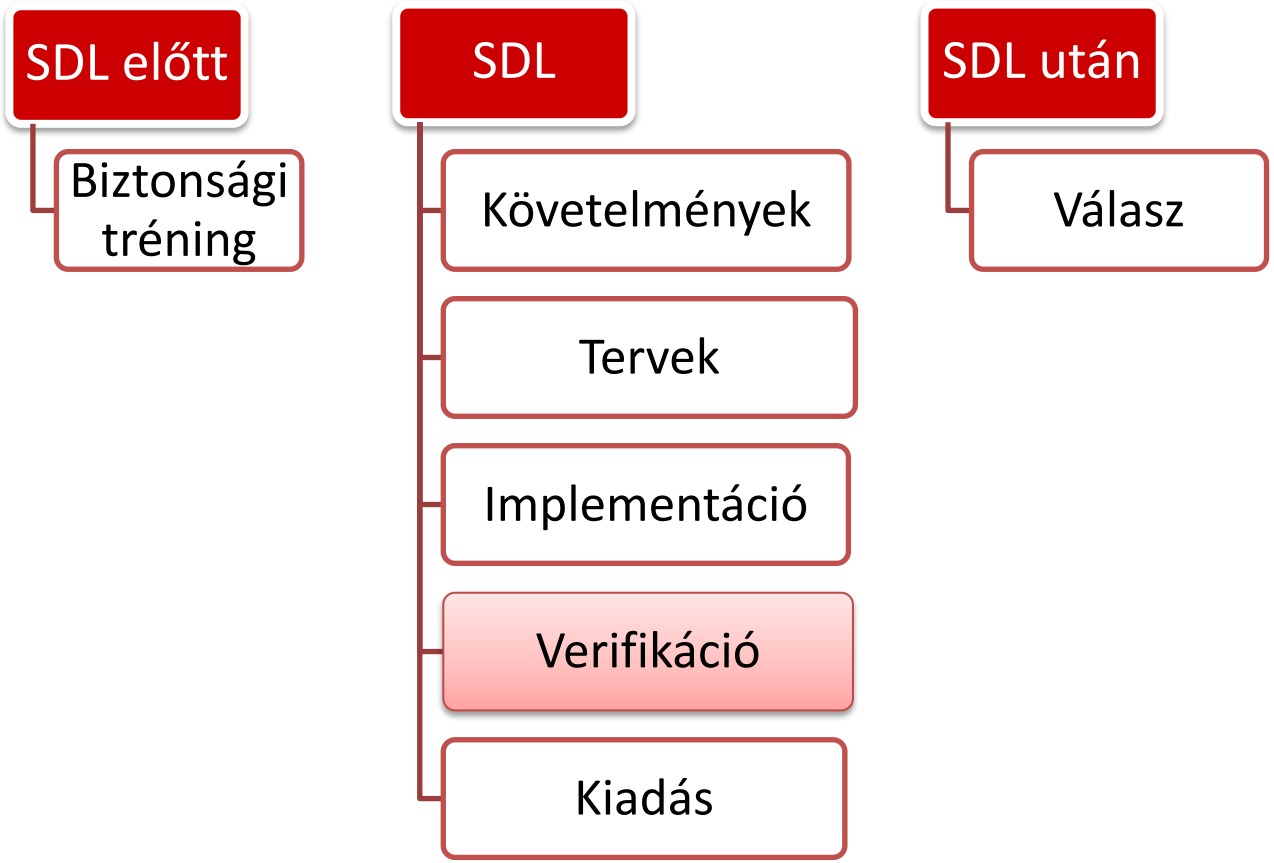
- A naplófájlok fontos adatforrások:
  - Biztonsági incidensek azonosítása
  - Szabályszegések monitorozása
  - Letagadhatatlansági mechanizmusok támogatása
  - Incidensek kivizsgálása

- Hogyan készítsünk naplóbejegyzéseket?
  - Mikor: naplóbejegyzés dátuma és ideje, esemény dátuma és ideje
  - Hol: komponens azonosítók, címek, kódban hely, geolokáció, szolgáltatás
  - Ki: IP cím, felhasználói azonosító (felhasználónév, licensz száma, …)
  - Mit: esemény típusa, súlyossága, leírása, biztonsággal kapcsolatos-e?

SDL: Implementáció fázis – Naplózás

- Milyen eseményeket naplózzunk mindenképp?
  - Adatfeldolgozási hibák
  - Autentikáció eredménye (sikeres vagy sem)
  - Autorizációs hiba
  - Session menedzsment
  - Hibák és rendszeresemények
  - Naplózás kezdete

- Mik NE szerepeljenek a bejegyzésekben?
  - Kulcsok,
  - Jelszavak,
  - Forráskód,
  - Tokenek,
  - Más érzékeny információk
- Gondolkodásmód: naplófájlokat bárki olvashatja!

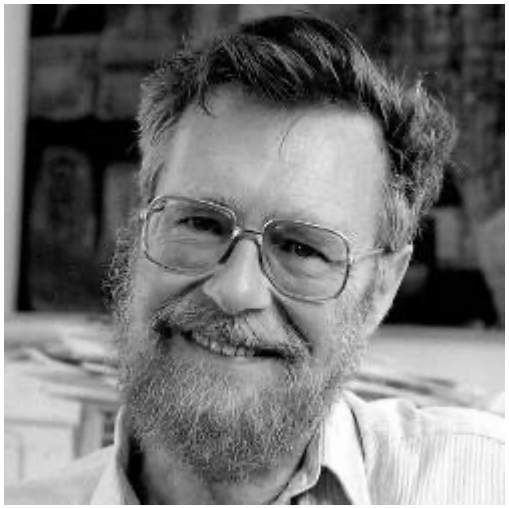


VERIFIKÁCIÓ FÁZIS

SDL: Verifikációs fázis

- Cél: ellenőrizzük a CIA hármast + 3A-t
- Minden életciklus fázisban kell verifikálni!

- „Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence.” – Edsger W. Dijkstra



SDL: Verifikációs fázis – Statikus elemzés

- Az utasításokat csak értelmezzük, nem hajtjuk végre
- Az elemzés egy absztrakt domén fölött történik automatizáltan vagy manuálisan

- Előnyök
  - Skálázhatóság: nagy kódbázist is képes kezelni
  - A fejlesztés során korán el lehet kezdeni alkalmazni

- Hátrány
  - Nincs futásidejű információ→ fals pozitív eredmény: olyan utasításokat is sérülékenynek jelezhet, amit a valóságban sose lehet lefuttatni

- Példa: code review, bináris analízis

SDL: Verifikációs fázis – Dinamikus elemzés

- A szoftver futtatás közben elemezzük

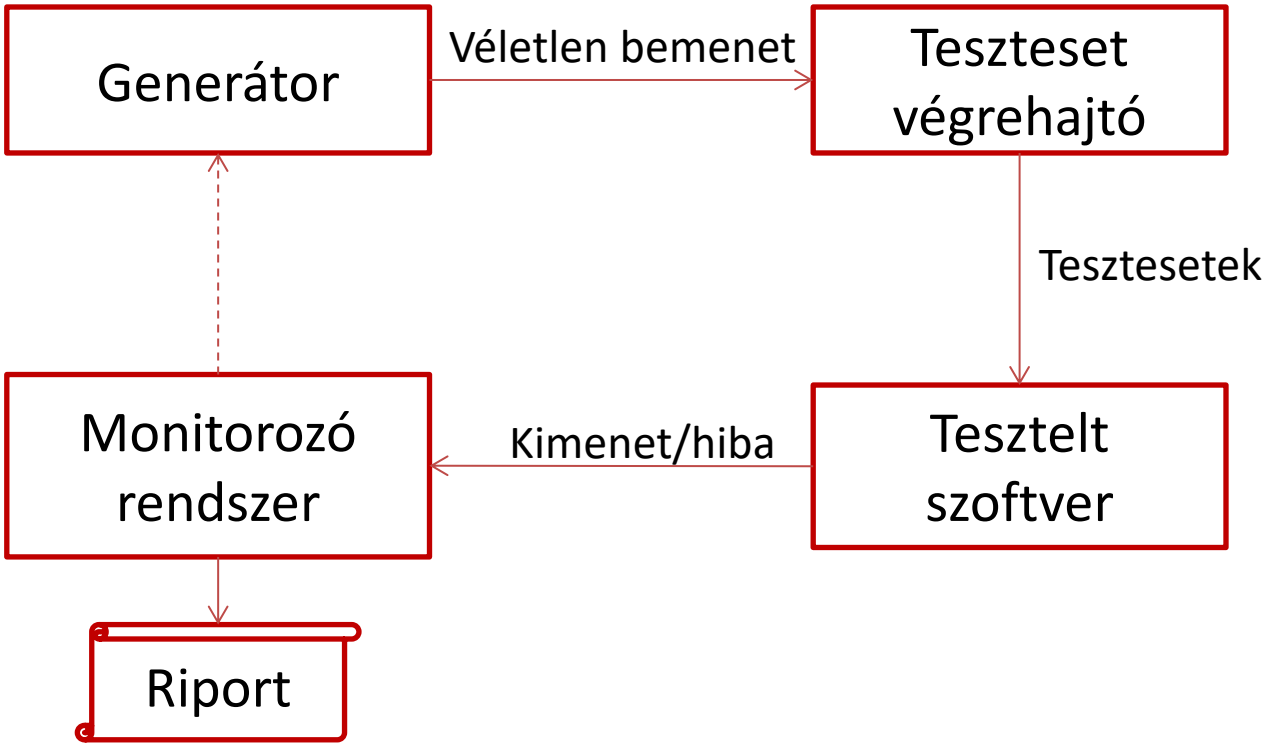
- Előnyök
  - Elérhetők futási idejű információk→ pontosabb eredmények
  - Jelzett sérülékenységek biztosan benne vannak a szoftverben

- Hátrányok
  - Instrumentációra lehet szükség (fordított kódra nem triviális!)
  - Alacsony kód-lefedettség
  - Nem tud semmit mondani olyan viselkedésről, amit nem figyelt meg

- Példa: fuzzing

SDL: Verifikációs fázis – Fuzzing

- Automatikus generált véletlenszerű bemenetek
- Szoftvert futás közben monitorozzuk hibákat keresve



- Tool: american fuzzy lop (afl)

SDL: Verifikációs fázis – Sérülékenység szkennelés

- Automatizált folyamat sérülékenységek kereséséhez
- Csak jól ismert sérülékenységeket talál meg

Completed: Feb 17, 2012 9:10 — Remove Vulnerability | Audit Trail

Filters: No Filters — Add Filter — Clear Filters

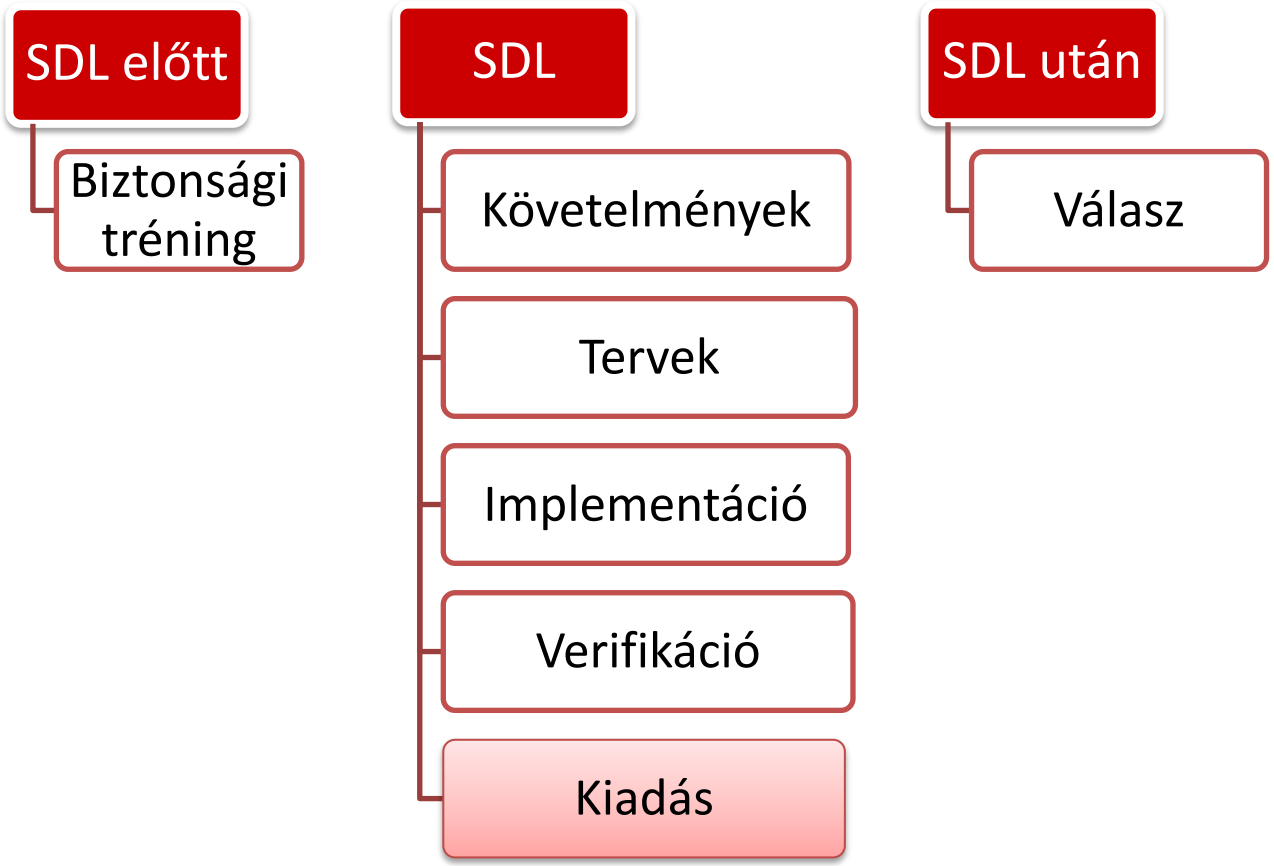
Plugin ID	Count	Severity	Name	Family
11139	1	High	CGI Generic SQL Injection	CGI abuses
42479	1	High	CGI Generic SQL Injection (2nd pass)	CGI abuses
18405	1	Medium	Microsoft Windows Remote Desktop Protocol Server Man-in-the-Middle Weakness	Windows
39466	1	Medium	CGI Generic Cross-Site Scripting (quick test)	CGI abuses : XSS
42056	1	Medium	CGI Generic Local File Inclusion	CGI abuses
44136	1	Medium	CGI Generic Cookie Injection Scripting	CGI abuses
44670	1	Medium	Web Application SQL Backend Identification	CGI abuses
49067	1	Medium	CGI Generic HTML Injections (quick test)	CGI abuses : XSS
26194	1	Low	Web Server Uses Plain Text Authentication Forms	Web Servers
30218	1	Low	Terminal Services Encryption Level is not FIPS-140 Compliant	Misc.
47830	1	Low	CGI Generic Injectable Parameter	CGI abuses
11219	2	Info	Nessus SYN scanner	Port scanners
10107	1	Info	HTTP Server Type and Version	Web Servers
10287	1	Info	Traceroute Information	General
10302	1	Info	Web Server robots.txt Information Disclosure	Web Servers
10662	1	Info	Web mirroring	Web Servers
10940	1	Info	Windows Terminal Services Enabled	Windows
11032	1	Info	Web Server Directory Enumeration	Web Servers
11874	1	Info	Microsoft IIS 404 Response Service Pack Signature	Web Servers
11936	1	Info	OS Identification	General
12053	1	Info	Host Fully Qualified Domain Name (FQDN) Resolution	General

http://thehackernews.com/2012/02/tenable-release-nessus-50-vulnerability.html

SDL: Verifikációs fázis – Penetrációs tesztelés

- Futó rendszeren (teszt vagy production) végzett elemzés
- Cél: demonstrálni, hogy mit tehet egy támadó

1. Információ gyűjtés: ismerjük meg a rendszert, amennyire lehet
   - Toolok: Nmap, Nessus, Jack the Ripper, stb.
2. Ismert sérülékenységek keresése nyilvános adatbázisban
3. Támadások indítása a szerzett infók alapján
   - Tool: Metasploit framework
4. Eredmények összefoglalása emberi fogyasztásra alkalmas formában

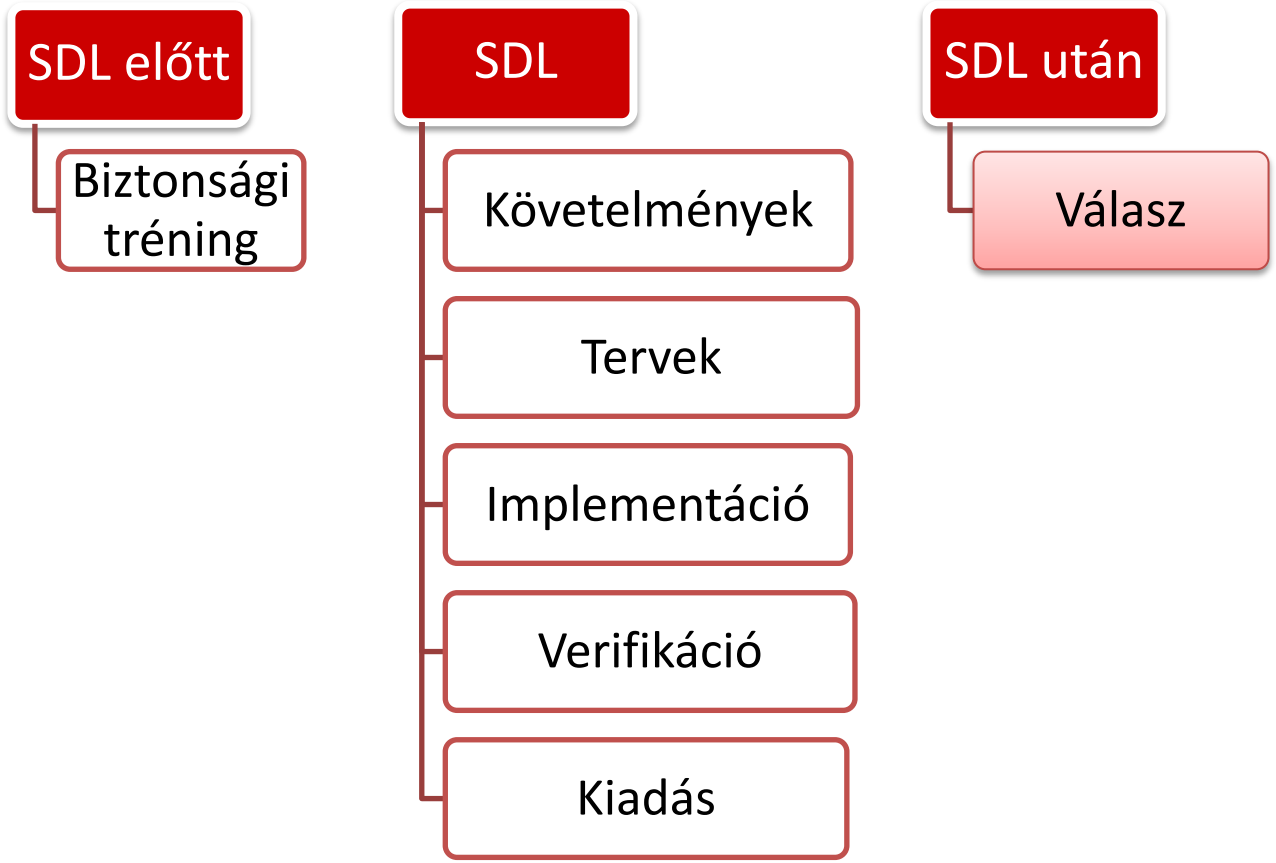


KIADÁS FÁZIS

SDL: Kiadás fázis

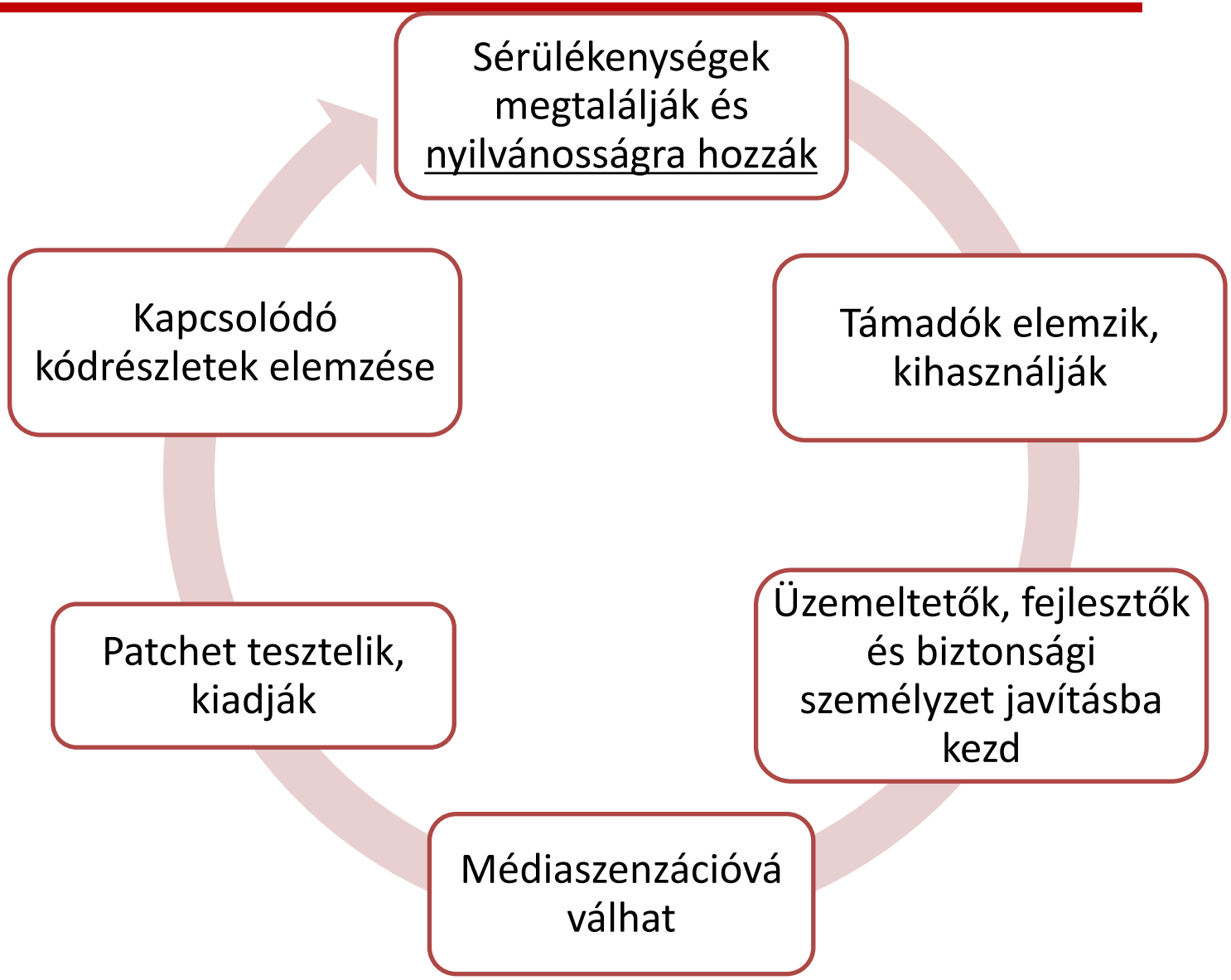
- **Security response terv** → Security Response Center
  - Új sérülékenységek fognak felbukkanni
  - Mit tegyünk, ha érintettek vagyunk?
  - Hogyan léphetnek velünk kapcsolatba, ha sérülékenységet találnak?

- **Incidens kezelési terv** → Incident Response Team
  - Mi legyen, ha a sérülékenységet nem felelősen hozták nyilvánosságra?
  - Mi van, ha megtámadják a rendszereinket?



FEJLESZTÉS UTÁN

Sérülékenységek életciklusa



SDL után: Security Response

- **Security response terv** végrehajtása: security response folyamat
- Security Response Center:
  1. Sérülkenység riportok gyűjtése, válaszolás
  2. Riport elemzése → Fejlesztők el tudnak kezdeni dolgozni
  3. Kapcsolattartás megtalálókkal, felelős nyilvánosságra hozás bátorítása
  4. Security bulletin létrehozása
  5. Ügyfelek problémáinak és a sajtó figyelése

SDL után: Security Response

- Security bulletin:

Media framework

The most severe vulnerability in this section could enable a remote attacker using a specially crafted file to execute arbitrary code within the context of a privileged process.

CVE	References	Type	Severity	Updated AOSP versions
CVE-2017-13248	A-70349612	RCE	Critical	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13249	A-70399408	RCE	Critical	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13250	A-71375536	RCE	Critical	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13251	A-69269702	EoP	Critical	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13252	A-70526702	EoP	High	8.0, 8.1
CVE-2017-13253	A-71389378	EoP	High	8.0, 8.1

SDL után: Security Response

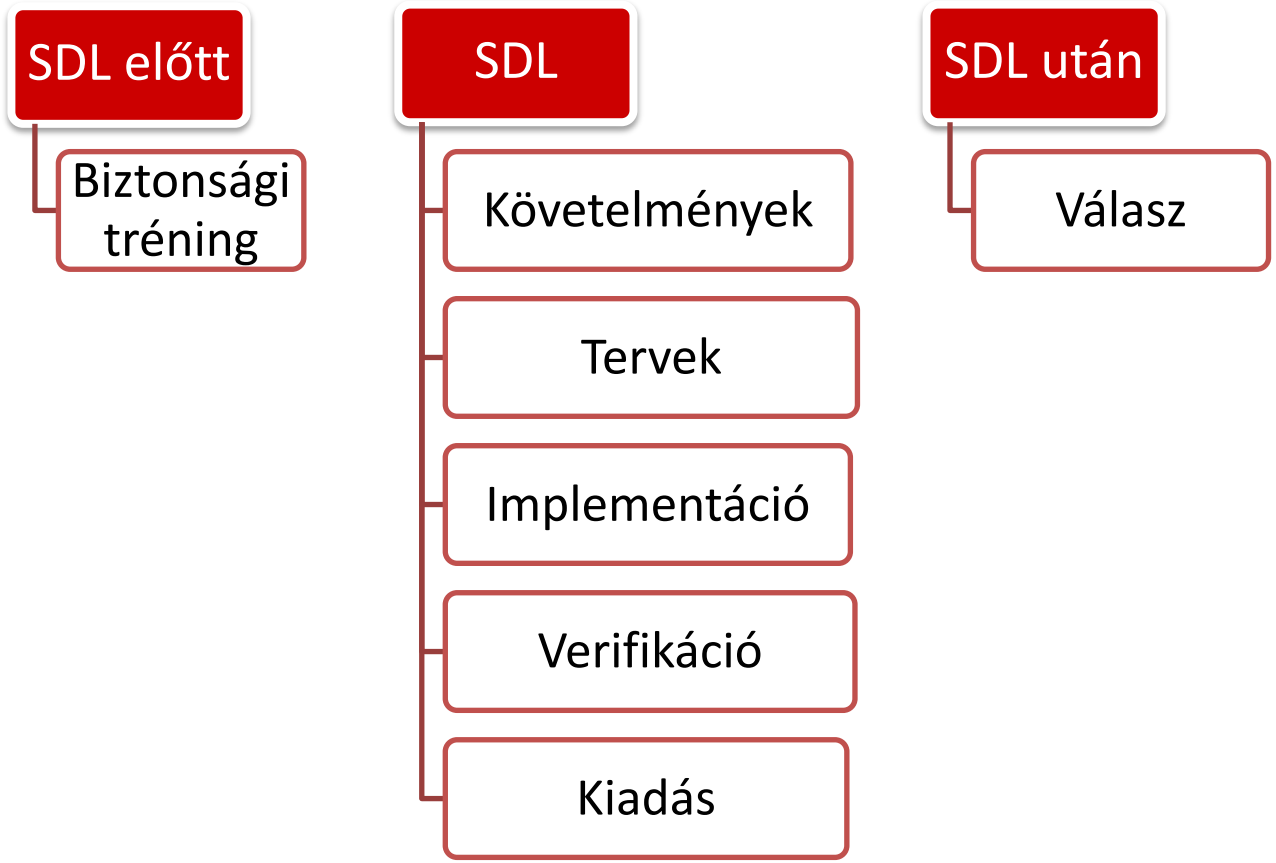
- **Security response terv** végrehajtása: security response folyamat
- Fejlesztőcsapat:
  1. Javítás implementálása: eredeti és változatok is!

- Tesztelő csapat:
  1. Eredeti problémák javításának a verifikálása (Security Team)
  2. Kompatibilitás ellenőrzése

- Tanulságok dokumentálása

SDL után: Incidens kezelés

- Felelőtlen nyilványosságra hozás, malware, stb. → **incidens kezelési plan**
- Incident response team:
  1. Környezet, ügyfelek, sajtó, stb. nyomon követése
  2. Security response team mobilizálása → security response plan
  3. Szituáció értékelése, tanácsadás, kerülő megoldások kommunikálása
  4. Információ és eszközök nyújtása a normál ügymenet helyreállításához



ÉPÍTSÉTEK BE A BIZTONSÁGOT!

ELLENŐRZŐ KÉRDÉSEK

Ellenőrző kérdések

- Mi a CVE?
- Miért nehéz biztonságos szoftvert fejleszteni?
- Milyen fázisai vannak a Microsoft SDL-nek?
- Milyen feladatokat és szerepeket kell a személyzetnek ellátnia?
- Mit nevezünk bug követő rendszernek tracking system? Hogyan kell létrehozni ahhoz, hogy sikeres legyen a bug követés?
- Mit mond a "economy of mechanism" alapelv?
- Mit mond a "fail-safe defaults" alapelv?
- Mit mond a "complete mediation" alapelv?
- Mit mond a "separation of privilege" alapelv?
- Mit mond a "least privilege" alapelv?
- Mit mond a "open design" alapelv?

Ellenőrző kérdések

- Mit mond a "least common mechanism" alapelv?
- Mit mond a "psychological acceptability" alapelv?
- Mit értünk egy szoftver támadási felülete alatt?
- A szoftvere hibás bemenetet észlel. Hogyan reagáljon?
- Milyen információkra van szükség naplózáskor?
- Miért fontosak a naplóbejegyzések?
- Milyen típusú adatot nem szabad sose naplózni?
- Milyen előnyei és hátrányai vannak a statikus elemzésnek?
- Milyen előnyei és hátrányai vannak a dinamikus elemzésnek?
- Mi a fuzzing alapötlete? Mi az egyes komponensek feladata?
- Mi a célja a penetrációs tesztelésnek? Hogyan végzünk pentesztelést?

Ellenőrző kérdések

- Mi a különbség a security response és az incidens kezelés között?
- Mit jelent a sérülékenységek felelős nyilvánosságra hozatala?
- Hogyan néz ki egy sérülékenység életciklusa?

HÁTTÉRANYAG

Háttéranyag

[1] Common Vulnerabilities and Exposures
https://cve.mitre.org/index.html

[2] Microsoft: Security Development Lifecycle
https://www.microsoft.com/en-us/SDL/process/requirements.aspx

[3]Steven M. Bellovin: Thinking Security: Stopping Next Year's Hackers. Addision-Wesley Professional, 2015

[4] US-CERT: Design Principles
https://www.us-cert.gov/bsi/articles/knowledge/principles/design-principles

Háttéranyag

[5] Stephen Northcutt, Jerry Shenk, Dave Shackleford, Tim Rosenberg, Raul Siles, Stev Mancini: Penetration Testing: Assessing Your Overall Security Before Attackers Do
https://www.sans.org/reading-room/whitepapers/analyst/penetration-testing-assessing-security-attackers-34635

[6] Ken Houghton: Vulnerabilities & Vulnerability Scanning
https://www.sans.org/reading-room/whitepapers/threats/vulnerabilities-vulnerability-scanning-1195