

---

# BEZPEČNOSŤ SOFTVÉRU A OCHRANA PROTI ŠKODLIVÉMU KÓDU



Financované  
Európskou úniou  
NextGenerationEU

PLÁN [OBNOVY]



KOMPETENČNÉ  
CENTRUM  
KYBERNETICKEJ  
BEZPEČNOSTI

STU

SLOVENSKÁ TECHNICKÁ  
UNIVERZITA V BRATISLAVE

---

---

# OBSAH

- 1. Analýza moderných hrozieb a zraniteľností**
- 2. Viacvrstvové obranné stratégie a technológie**
- 3. Proaktívny prístup k bezpečnosti softvéru: Princíp "Shift-Left"**
- 4. Riadenie, reakcia a budúce trendy**

---

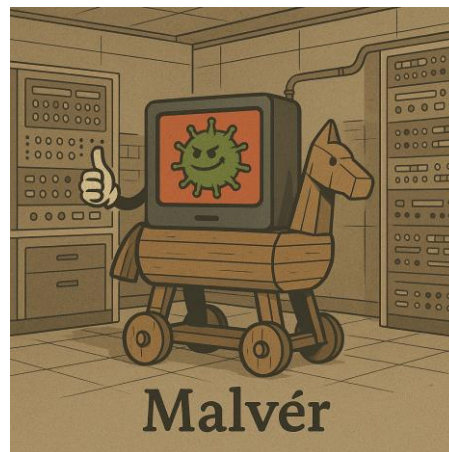
# ÚVOD: KONVERGENCIA BEZPEČNOSTI

## Dve strany tej istej mince

- V súčasnom digitálnom ekosystéme sa hranica medzi zraniteľnosťou softvéru a úspešným kybernetickým útokom stáva čoraz tenšou.
- Tradičné modely založené na obrane perimetra strácajú účinnosť v prostredí, kde perimeter prakticky neexistuje.
- Tento dokument pristupuje k bezpečnosti softvéru a ochrane proti malvéru nie ako k dvom oddeleným disciplínam, ale ako k neoddeliteľne prepojeným súčasťam.
- **Cieľ:** Poskytnúť komplexný prehľad stratégií na zabezpečenie softvéru počas celého jeho životného cyklu a na implementáciu robustnej, viacvrstvovej obrany.

# ČASŤ 1: ANALÝZA MODERNÝCH HROZIEB A ZRANITEĽNOSTÍ : ČO JE MALWARE?

**Malvér** (malware) je škodlivý softvér, ktorý sa dostane do počítača alebo mobilu a robí tam veci, ktoré nechceme — napríklad kradne údaje, poškodzuje systém, špehuje nás alebo sa šíri ďalej bez nášho vedomia.. Pochopenie jeho rôznych foriem, cieľov a metód šírenia je nevyhnutné pre návrh efektívnej obrany.



---

# TAXONÓMIA MALVÉRU

- Slovo **malware** vzniklo ako novotvar zložením z dvoch anglických slov **malicious** a **software**, čo v preklade znamená „zlomyselný a zákerný softvér alebo program“
- Ide o súhrnné pomenovanie pre všetky druhy **škodlivého softvéru**, ktoré zahŕňa rôzne typy programov.



---

# TAXONÓMIA MALVÉRU

- **Vírusy:** Pripájajú sa k legitímnym súborom a na šírenie vyžadujú ľudskú interakciu.
- **Červy (Worms):** Samostatné programy, ktoré sa autonómne replikujú a šíria po sieti zneužívaním zraniteľností.
- **Trójske kone (Trojans):** Maskujú sa za legitímny softvér; po spustení vytvárajú zadné vrátka (backdoors) alebo kradnú dáta.
- **Ransomware:** Šifruje súbory obete a požaduje výkupné. Moderné verzie kombinujú šifrovanie s krádežou dát (dvojité vydieranie).
- **Spyware/Adware:** Tajne monitoruje aktivitu používateľa (Spyware) alebo zobrazuje nechcené reklamy (Adware).

---

# POKROČILÉ HROZBY: BEZSÚBOROVÝ MALVÉR (FILELESS MALWARE)

Útok, ktorý nezanecháva stopy na disku

- **Princíp:** Neukladá škodlivé spustiteľné súbory na disk, čím sa vyhýba detekcii tradičnými antivírusmi .
- Dnešné fileless malvéry už nie sú samostatné „červy“ ako Slammer alebo Code Red, ale skôr techniky používané pokročilými útokmi.

Najtypickejší predstavitelia dnes:

- Kovter, Poweliks, Posh, Emotet (fileless fáza), TrickBot, Cobalt Strike Beacon, Carbanak/FIN7, SamSam a RobinHood ransomware

---

# POKROČILÉ HROZBY: BEZSÚBOROVÝ MALVÉR (FILELESS MALWARE)

Bežne používané techniky v moderných fileless útokoch:

*(Fileless sa často týka skôr spôsobu spustenia než samotného malware)*

- PowerShell (Invoke-Expression, EncodedCommand)
- WMI eventy
- Registry-based persistence
- Reflective DLL injection
- .NET assembly in-memory loading
- Living-off-the-Land Binaries (LOLbins)

---

# POKROČILÉ HROZBY:

## PowerShell-based fileless útoky (POSH, PowerWare, PoshC2)

- Bežia v pamäti cez PowerShell.
- Nevyžadujú zapísanie spustiteľného súboru na disk.
- Mimoriadne populárne v ransomvéroch aj APT útokoch.

---

# POKROČILÉ HROZBY:

**Living-off-the-Land Binaries (LOLbins)** sú „legitímne systémové nástroje“, ktoré už existujú priamo v operačnom systéme – a útočníci ich zneužívajú na vykonanie škodlivých činností bez potreby sťahovať alebo spúšťať nové súbory.

Ide o kľúčový prvok moderných fileless útokov.

---

# POKROČILÉ HROZBY: LOLBINS

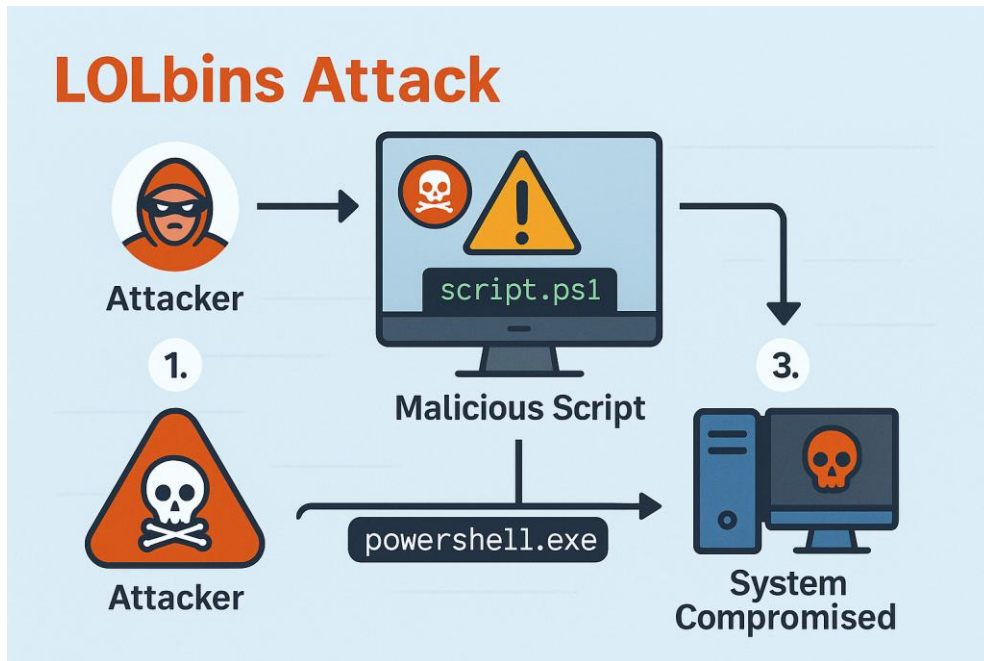
Útočník neprináša na počítač nič nové – “živí sa tým, čo už tam rastie”.

**Miesto spustenia vlastného malware použije:**

- vstavané systémové programy,
- administračné nástroje,
- skriptovacie jazyky,
- a OS funkcie, ktoré sú normálne povolené.

*Tým je útok nenápadnejší, ťažšie detekovateľný a často prejde bezpečnostnými produktmi, ktoré sledujú hlavne spustenie nových súborov.*

# POKROČILÉ HROZBY: LOLBINS



## Top 10 LOLBins



**PowerShell**  
powershell.exe



**CMD**  
cmd.exe



**Rundll32**  
rundll32.exe



**Regsvr32**  
regsvr32.exe



**Regsvr32**  
regsvr32.exe



**mshta.exe**  
mshta.exe



**Certutil**  
certutil.exe



**Msiexec**  
msiexec.exe



**Bitsadmin**  
bitsadmin.exe



**Reg**  
reg.exe

# POKROČILÉ HROZBY: LOLBINS

## AKO POWERSHELL SPŮŠŤA SKRIPT Z PAMÄTE



PowerShell vie vykonať skript uložený v premennej:

```
$code = "Write-Host 'Hello from RAM!'"Invoke-Expression $code
```

PowerShell interpretuje „premennú“ ako kód. Tak je možné spúšťať aj škodlivý payload.

PowerShell umožňuje spustiť skripty zakódované v Base64

```
powershell.exe -EncodedCommand <base64_string>
```

- \* Obsah Base64 sa rozbalí iba v pamäti
- \* PowerShell ho priamo interpretuje
- \* Na disk sa nič nezapisuje

● Bežne sa takto šíri Emotet, TrickBot či Cobalt Strike

# POKROČILÉ HROZBY: LOLBINS

## Ako PowerShell spúšťa kód z pamäte

Technika	Spôsob	Zapíše sa na disk?
 Invoke-Expression	Spustenie reťazca uloženého v pamäti	 Nie
 Encoded-Command	Spustenie Base64 príkazu	 Nie
 Download-String + IEX	Načítanie skriptu z webu → RAM	 Nie
 .NET Assembly	Spustenie DLL z bajtov v pamäti	 Nie
 Shellcode injection	Spustenie strojového kódu v RAM	 Nie

---

# POKROČILÉ HROZBY: LOLBINS RIZIKÁ

- Nepotrebný nový .exe → nič na disku
- Bežný antivírus to často nezachyti.
- Sú podpísané Microsoftom
- Vyzerajú dôveryhodne.
- Sú “povolené” administrátormi
- Väčšina IT oddelení ich bežne používa.
- Umožňujú fileless execution
- Kód môže bežať iba v pamäti.

---

# POKROČILÉ HROZBY: ADVANCED PERSISTENT THREATS (APT)

## Trpezliví a cílení lovci

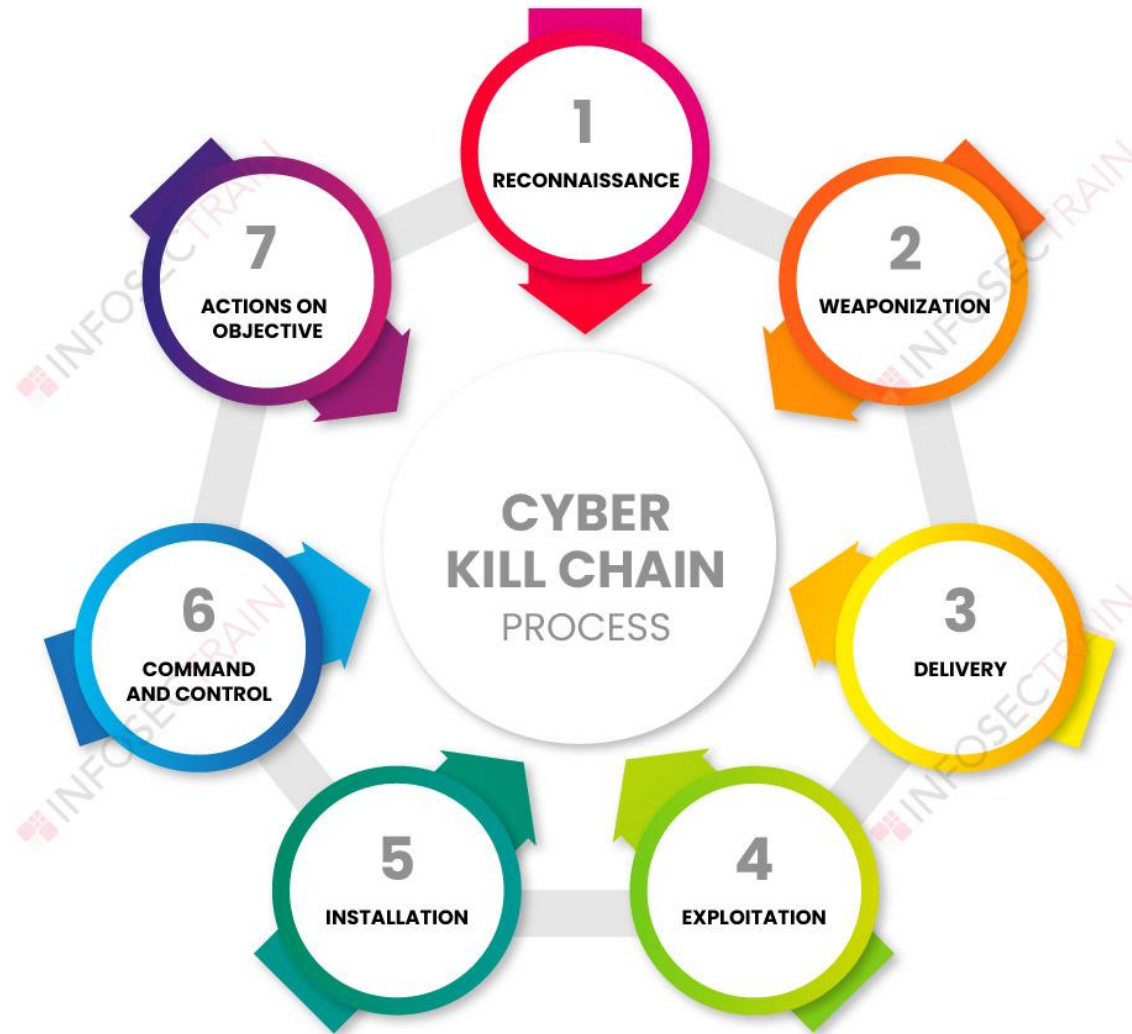
- **Definícia:** APT sú vysoko cílené, dlhodobé a skryté kybernetické útoky vykonávané sofistikovanými útočníkmi (často štátom sponzorované skupiny).
- **Charakteristika:**
  - **Advanced (Pokročilosť):** Použitie vlastných nástrojov a zero-day zraniteľností.
  - **Persistent (Pretrvávajúcnosť):** Snaha udržať si prístup mesiace až roky.
  - **Threat (Hrozba):** Jasne definovaný cieľ (špionáž, sabotáž).

---

# ŽIVOTNÝ CYKLUS ÚTOKU & MITRE ATT&CK

- **Fázy útoku:**
  1. Prieskum (Reconnaissance)
  2. Počiatočná kompromitácia (Initial Compromise)
  3. Vytvorenie oporného bodu (Establishing Foothold)
  4. Eskalácia privilégií (Privilege Escalation)
  5. Interný prieskum a laterálny pohyb (Internal Reconnaissance & Lateral Movement)
  6. Udržanie prístupu (Maintaining Access)
  7. Dosiahnutie cieľa (Mission Accomplishment)
- **MITRE ATT&CK®:** Globálna znalostná báza taktík, techník a procedúr (TTPs) útočníkov. Je to nevyhnutný nástroj na popis, analýzu a modelovanie reálnych útokov.

# KILL CHAIN & MITRE ATT&CK



---

# ČASŤ 2: VIACVRSTVOVÉ OBRANNÉ STRATÉGIE (DEFENSE IN DEPTH)

## Jedna línia obrany nikdy nestačí

- **Princíp:** Efektívna obrana si vyžaduje implementáciu viacvrstvovej stratégie.
- Žiadna jednotlivá bezpečnostná kontrola nie je dokonalá.
- Ochrana nie je založená na jednom bezpečnostnom prvku, ale na kombinácii viacerých (napr. firewall, antivírus, segmentácia siete, monitoring).
- Každá vrstva pokrýva inú časť obrany (sieť, zariadenia, dáta, aplikácie, používateľov).

---

# ČASŤ 2: (DEFENSE IN DEPTH) KLÍČOVÉ VRSTVY

## Viacvrstvová obranná stratégia

Mnohovrstevný bezpečnostný prístup, ktorý chráni systém kombináciou viacerých nezávislých vrstiev ochrany.

- Fyzická bezpečnosť
- Perimeter / sieťová ochrana
- Ochrana koncových bodov
- Aplikačná bezpečnosť
- Identita a prístupy
- Dátová vrstva
- Monitoring, detekcia a reakcia
- E-mailová a komunikačná bezpečnosť
- Politíky a procesy
- Vzdelávanie používateľov

---

# ČASŤ 2: VIACVRSTVOVÉ OBRANNÉ STRATÉGIE (DEFENSE IN DEPTH)

Cieľom je vytvoriť sériu prekrývajúcich sa obranných mechanizmov a minimalizovať dopad útoku, aby útočník, ktorý prekoná jednu vrstvu, narazil na ďalšiu.

---

# ČASŤ 2: (DEFENSE IN DEPTH) KLÚČOVÉ VRSTVY

## Fyzická bezpečnosť

- Zamknuté serverovne, prístupové karty, kamery
- Ochrana pred fyzickým odcudzením či manipuláciou so zariadeniami
-

---

# ČASŤ 2: (DEFENSE IN DEPTH) KLÚČOVÉ VRSTVY

## Perimeter / Sieťová ochrana

- Firewally
- IDS/IPS systémy
- Network segmentation (oddelenie sietí)
- VPN
-

---

# ZABEZPEČENIE SIETE: NGFW & IDS/IPS

- **Next-Generation Firewall (NGFW):**
  - Poskytuje pokročilé schopnosti nad rámec tradičných firewallov.
  - **Kľúčové funkcie:**
    - **Application awareness:** Identifikácia a kontrola špecifických aplikácií.
    - **Deep Packet Inspection (DPI):** Analýza obsahu sieťových paketov.
    - **SSL/TLS decryption:** Kontrola šifrovanej prevádzky.
- **IDS vs. IPS:**
  - **IDS (Systém detekcie narušenia):** Pasívny systém, ktorý len generuje upozornenia.
  - **IPS (Systém prevencie narušenia):** Aktívny systém, ktorý dokáže podozrivú prevádzku v reálnom čase zablokovať.

---

# ČASŤ 2: (DEFENSE IN DEPTH) KLÚČOVÉ VRSTVY

## Ochrana koncových bodov (Endpoint Security)

- Antivírus / Antimalware
- EDR (Endpoint Detection & Response), XDR
- Hardening OS (zakázanie nepotrebných služieb, pravidlá firewallu)

Choose the best endpoint protection strategy for modern threats.



Antivirus

Basic protection against known threats



EDR

Advanced threat detection and response

# ČASŤ 2: KLÚČOVÉ VRSTVY OCHRANA KONCOVÝCH BODOV: EVOLÚCIA AV -> EDR

Technológia	Fokus	Metóda Detekcie	Pokrytie Hrozieb
<b>Tradičný Antivírus (AV)</b>	Prevenencia	Signatúry známeho malvéru.	Známy, súborový malvér.
<b>Next-Gen Antivirus (NGAV)</b>	Prevenencia	Strojové učenie, behaviorálna analýza.	Známe, neznáme (zero-day), bezsúborové hrozby.
<b>Endpoint Detection &amp; Response (EDR)</b>	Detekcia a Reakcia	Nepretržitý zber a analýza telemetrie.	Pokročilé hrozby, APTs, aktivity po kompromitácii.

---

# ČASŤ 2: (DEFENSE IN DEPTH) KLÍČOVÉ VRSTVY

## Identita a prístupy (IAM / PAM)

- Silná autentifikácia (MFA)
- Privilegovaný prístup pod kontrolou (PAM)
- Least Privilege (najmenšie privilégia)
- Oddelenie rolí

---

# ZERO TRUST ARCHITECTURE (ZTA)

## Nikdy nedôveruj, vždy overuj

- **Filozofia:** ZTA nie je produkt, ale strategický prístup k bezpečnosti.
- **Základný princíp:** Každá požiadavka na prístup musí byť overená, bez ohľadu na to, odkiaľ pochádza (z internej alebo externej siete). Žiadna implicitná dôvera.
- **Praktická implementácia:** Kombinácia technológií ako:
  - Silná správa identít (IAM) s MFA.
  - Mikro-segmentácia siete.
  - Nepretržité monitorovanie (EDR, SIEM).
  - Hardening všetkých systémov.

---

# ČASŤ 2: (DEFENSE IN DEPTH) KLÚČOVÉ VRSTVY

## Dátová vrstva

- Šifrovanie dát (at-rest / in-transit)
- Data Loss Prevention (DLP)
- Zálohovanie + test obnovy (backup & recovery)

---

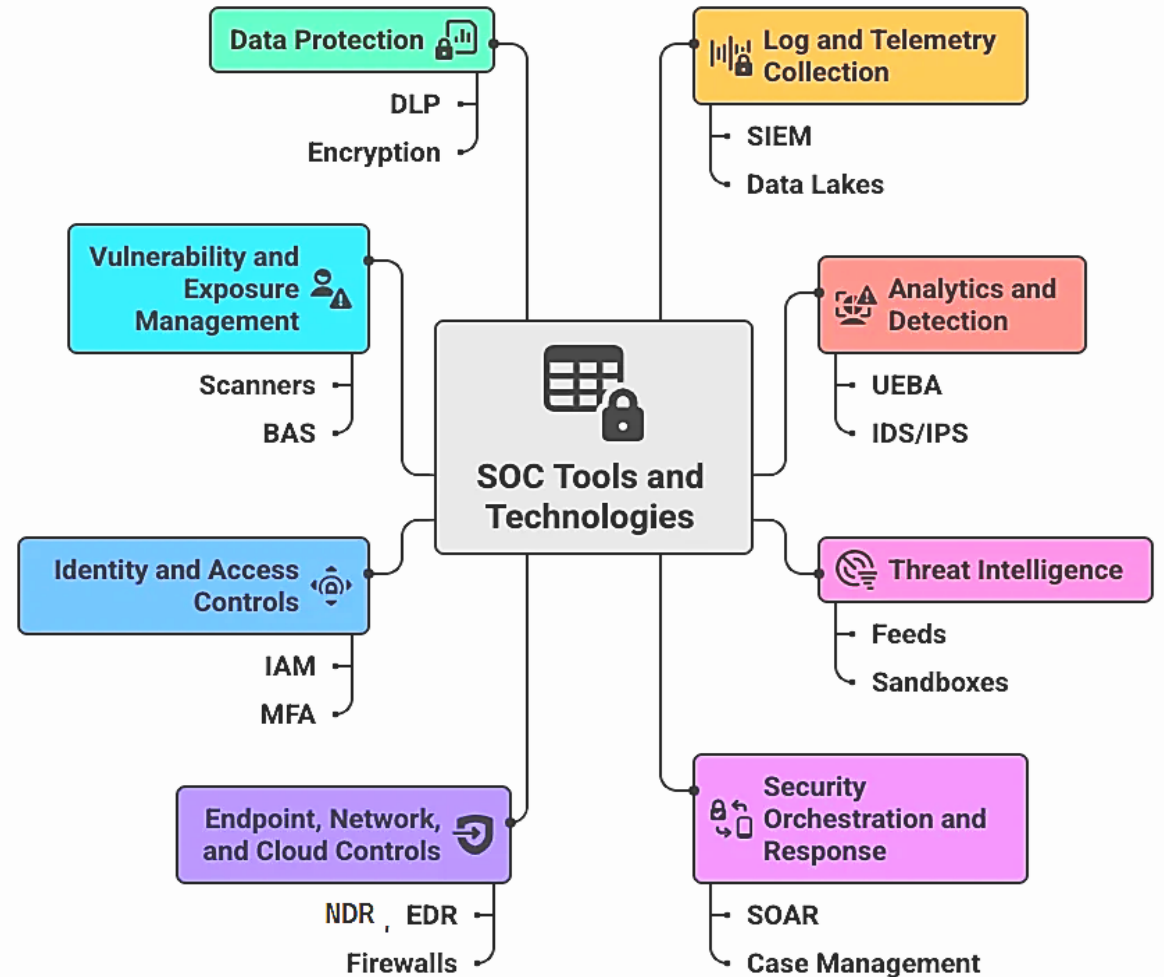
# ČASŤ 2: (DEFENSE IN DEPTH) KLÚČOVÉ VRSTVY

## Monitoring, detekcia a reakcia

- SIEM (Security Information & Event Management)
- SOC monitoring
- Threat hunting
- Incident Response (IR)
- Forezná analýza

# ČASŤ 2: VIACVRSTVOVÉ OBRANNÉ STRATÉGIE (DEFENSE IN DEPTH)

## SOC Tools and Technologies



---

# ŽIVOTNÝ CYKLUS REAKCIE NA INCIDENTY (NIST SP 800-61)

## Pripravenosť je kľúčom k efektívnej reakcii

- Napriek všetkým opatreniam je nevyhnutné predpokladať, že k incidentom dôjde.
- Efektívna reakcia nie je chaotická aktivita, ale systematický proces.
- Schopnosť organizácie **reagovať** je priamo úmerná tomu, ako dobre má zvládnutú **prípravu**.

---

# FÁZY REAKCIE NA INCIDENTY

## 1. Príprava (Preparation):

- Vytvorenie plánu a tímu (CSIRT), zabezpečenie nástrojov (SIEM, EDR), pravidelné tréningy a simulácie.

## 2. Detekcia a Analýza (Detection & Analysis):

- Identifikácia incidentu z alertov alebo hlásení a analýza jeho rozsahu a dopadu.

## 3. Zadržanie, Odstránenie a Obnova (Containment, Eradication & Recovery):

- Izolácia postihnutých systémov, odstránenie malvéru a bezpečné obnovenie prevádzky.

## 4. Aktivity po incidente (Post-Incident Activity):

- Analýza "**Lessons Learned**" na identifikáciu slabých miest a neustále zlepšovanie.

---

# ČASŤ 2: (DEFENSE IN DEPTH) KLÚČOVÉ VRSTVY

## Emailová a komunikačná bezpečnosť

- Ochrana proti phishingu
- Spam filtering
- Ochrana príloh (sandboxing)

---

# POKROČILÁ OBRANA: SANDBOXING A TECHNIKY OBCHÁDZANIA

- **Sandboxing:** Spustenie podozrivého kódu v bezpečnom, izolovanom prostredí na pozorovanie jeho správania (dynamická analýza). Kľúčové pre detekciu zero-day malvéru.
- **Techniky Obchádzania (Evasion Techniques):**
  - **Detekcia sandboxu:** Malvér hľadá stopy virtualizačného prostredia (špecifické súbory, ovládače, MAC adresy).
  - **Časové bomby:** Škodlivá aktivita sa odloží o hodiny alebo dni, aby sa predišlo krátkej analýze v sandboxe.
  - **Vyžadovanie interakcie používateľa:** Malvér čaká na pohyb myšou alebo kliknutie, ktoré v automatizovanom prostredí nemusí nastať.

---

# ČASŤ 2: (DEFENSE IN DEPTH) KLÍČOVÉ VRSTVY

## Politiky a procesy

- Bezpečnostné smernice
- Pravidlá používania IT
- Riadenie zraniteľností (Vulnerability Management)
- Penetračné testy

---

# ČASŤ 2: (DEFENSE IN DEPTH) KLÚČOVÉ VRSTVY

## Vzdelávanie používateľov

- Školenia kyberbezpečnosti
- Simulácie phishingu
- Bezpečné návyky pri práci

---

# RÁMCE KYBERNETICKEJ BEZPEČNOSTI

- **Synergia:** NIST CSF definuje "čo a prečo", ISO 27001 poskytuje systém riadenia a CIS Controls dávajú konkrétny technický "návod", ako to implementovať.

Aspekt	NIST Cybersecurity Framework (CSF)	ISO/IEC 27001	CIS Critical Security Controls
Účel	Strategické riadenie rizika.	Procesné riadenie (ISMS).	Taktická implementácia technických kontrol.
Detail	Čo dosiahnuť, nie ako.	Zameraný na procesy a dokumentáciu.	Vysoko preskriptívny, "ako" to urobiť.
Certifikácia	Nie.	Áno, medzinárodne uznávaná.	Nie.

---

# RÁMCE KYBERNETICKEJ BEZPEČNOSTI NIST CSF



---

# ČASŤ 2: (DEFENSE IN DEPTH) KLÍČOVÉ VRSTVY

## Aplikačná bezpečnosť

- Patch management (pravidelné záplaty)
- **Bezpečný vývoj (SSDLC)**
- Web Application Firewall (WAF)
- Ochrana API

---

# KYBERNETICKÁ HYGIENA: HARDENING & PATCH MANAGEMENT

Základy, ktoré sa nedajú preskočiť

- **System Hardening:**
  - Proces konfigurácie systému tak, aby bol čo najodolnejší.
  - Cieľom je minimalizovať útočnú plochu (attack surface) odstránením nepotrebných služieb, aplikácií a zmenou predvolených nastavení.
  - **CIS Benchmarks®:** Sú de facto štandardom a poskytujú detailné návody na hardening.
- **Patch Management:**
  - Systematický proces identifikácie, testovania a nasadzovania bezpečnostných záplat.
  - **Životný cyklus:** Inventarizácia -> Identifikácia -> Testovanie -> Nasadenie -> Overenie.

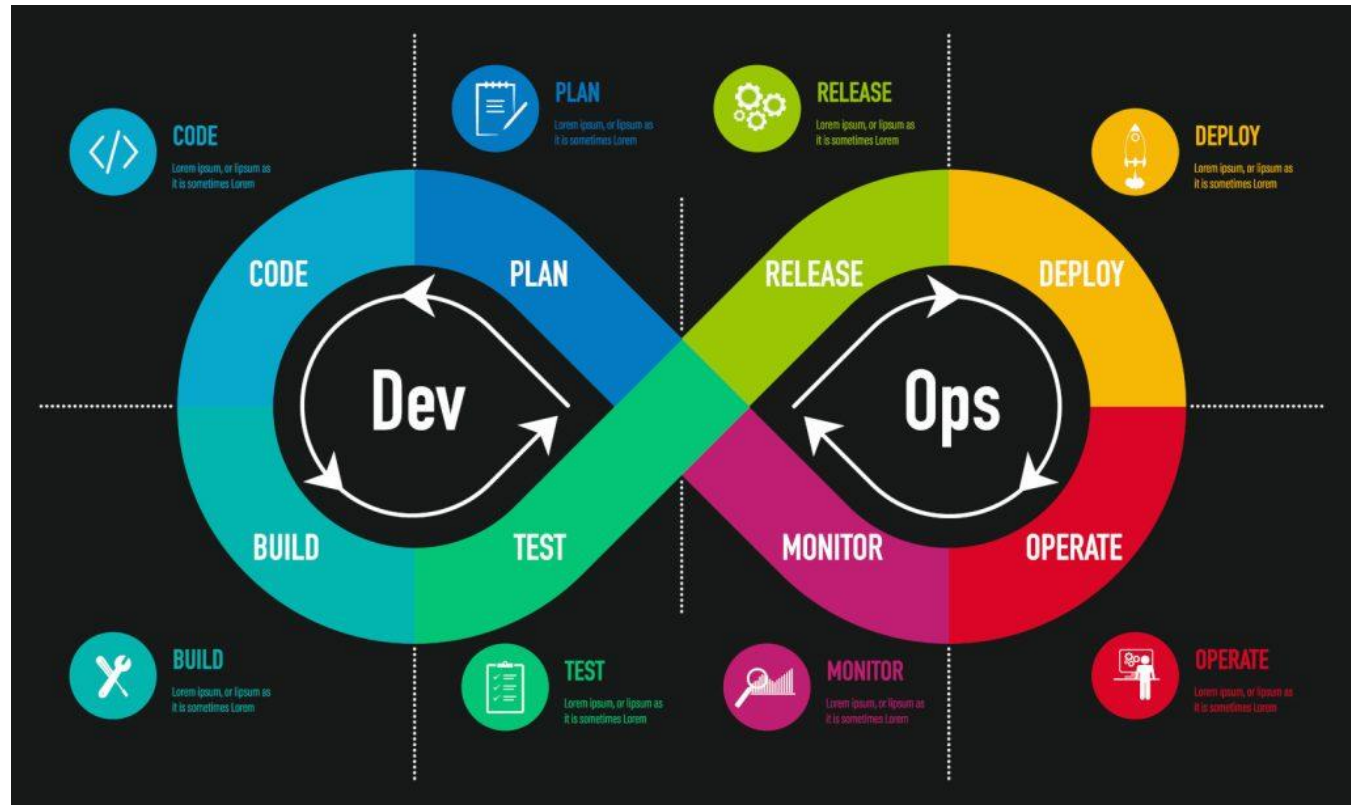
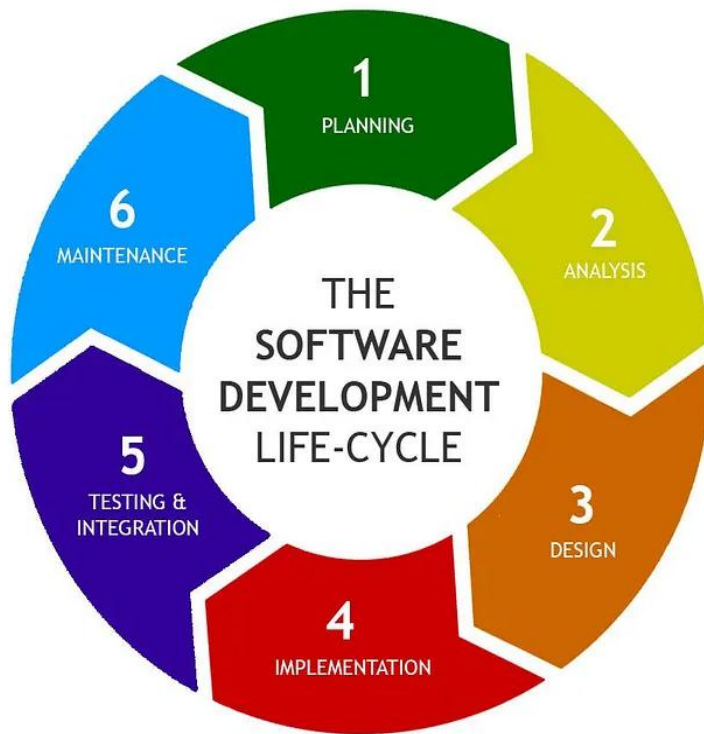
---

# KYBERNETICKÁ HROZBA: EXPLOIT

- *Exploit je v informatike špeciálny program, dáta alebo sekvencia príkazov, ktoré využívajú programátorskú chybu, ktorá spôsobí pôvodne nezámernú činnosť softvéru a umožňuje tak získať nejaký prospech.*
- Exploit umožní napríklad spustiť nežiadúcu inštaláciu softvéru (napr. nejaký druh malvéru), získať privilégia alebo obísť bezpečnostné kontroly.

***SSDLC pomáha znižovať počet zraniteľností, aby ich útočníci nemohli zneužiť pomocou exploitov.***

# TRADIČNÝ ŽIVOTNÝ CYKLUS VÝVOJA SOFTVÉRU SDLC (DEVOPS)



---

# ČASŤ 3:

## PROAKTÍVNY PRÍSTUP - PRINCÍP "SHIFT-LEFT"

Prečo hasiť požiar, keď mu môžeme predchádzať?

- **"Shift-Left"** je fundamentálny princíp integrácie bezpečnostných opatrení do najskorších fáz životného cyklu softvéru.
- Predstavuje paradigmatický posun od tradičného **reaktívneho prístupu**, kde sa bezpečnosť riešila až na konci (napr. penetračným testovaním).
- Oprava zraniteľností po nasadení je extrémne **nákladná a riziková**.
- Integráciou bezpečnosti do každej fázy budujeme inherentne **odolnejšie a bezpečnejšie aplikácie**.

---

# BEZPEČNÝ ŽIVOTNÝ CYKLUS VÝVOJA SOFTVÉRU (SSDLC)

## Bezpečnosť ako súčasť DNA, nie ako náplast'

- **Definícia:** SSDLC je rámec, ktorý systematicky integruje bezpečnostné aktivity a kontroly do každej fázy tradičného SDLC.
- Bezpečnosť nie je vnímaná ako izolovaná fáza, ale ako **nepretržitý proces** od plánovania až po údržbu.
- Vyžaduje si kultúrnu transformáciu smerom k modelu **DevSecOps**, kde vývojové, operačné a bezpečnostné tímy zdieľajú zodpovednosť za bezpečnosť.
- Výsledok: Bezpečnosť sa stáva neoddeliteľnou vlastnosťou produktu ("**security by design**").

---

# KLÚČOVÉ BENEFITY SSDLC DEVSECOPS – NOVÁ FILOZOFIA PRÍSTUPU FUNKCIONALITA & BEZPEČNOSŤ

- **Redukcia nákladov:** Bezpečnosť sa považuje za tradičnú prekážku pre inovácie a často sa vníma negatívne. S posunom zabezpečenia do životného cyklu ide o pomoc vývojovému tímu tak, aby mohlo dochádzať k inováciám súčasne so zabezpečením.
- **DevSecOps** je zdravý model, ktorý predpokladá, že je za bezpečnosť zodpovedný každý. Nie je mysliteľné, aby jeden tím vytvoril aplikáciu a odovzdal ju inému tímu, ktorý by sa mal postarať o jej bezpečnosť. Oboje sa musí určovať a vykonávať.

---

# KLÍČOVÉ BENEFITY SSDLC

- **Redukcia nákladov:** Oprava chýb v skorých fázach je výrazne lacnejšia.
- **Zníženie rizika:** Proaktívnym riešením zraniteľností sa minimalizuje pravdepodobnosť úspešných útokov a únikov dát.
- **Zvýšená dôvera:** Zákazníci a partneri viac dôverujú softvéru, ktorý je vyvíjaný s dôrazom na bezpečnosť.
- **Proaktívna mitigácia hrozieb:** Umožňuje identifikovať a zmierňovať hrozby ešte predtým, ako sa prejavia.
- **Zlepšenie súladu s predpismi:** Uľahčuje dodržiavanie požiadaviek ako GDPR, HIPAA alebo PCI DSS.

---

# KLÍČOVÉ BENEFITY SSDLC DEVSECOPS – EFEKTÍVNOST

Ryan O’Leary, šéf výskumu zabezpečenia v spoločnosti White Hat:

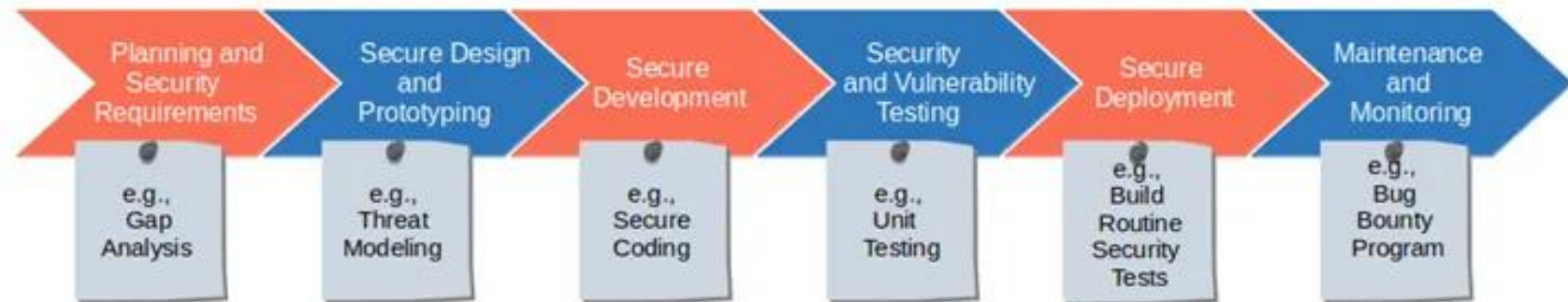
- Priemerná spoločnosť potrebuje 113 dní, na odstránenie zraniteľnosti zistené vo vývoji pomocou statickej analýzy. Zatiaľ čo spoločnostiam využívajúcim DevSecOps stačí len 51 dní.
- Na odstránenie zraniteľnosti nájdennej pri použití dynamickej analýzy v produkčnom prostredí je potreba 174 dní. Spoločnosti ktoré implementovali DevSecOps, to však zvládnu iba za 92 dní.

# SDLC VS. SSDLC

## Software Development Life Cycle (SDLC) Process



## Secure Software Development Life Cycle (SSDLC) Process



---

# FÁZY SSDLC (1) - PLÁNOVANIE A NÁVRH ZÁKLADY BEZPEČNEJ ARCHITEKTÚRY

## Fáza 1: Požiadavky a plánovanie

V plánovacej fáze je kľúčové vytvoriť „bezpečnostne orientovaný“ prístup. Zahŕňa to modelovanie hrozieb, pri ktorom sa identifikujú potenciálne riziká a určujú stratégie ich zmiernenia. V tejto fáze sa stanovujú aj hlavné ciele, ako napríklad požiadavky na ochranu údajov, regulačná zhoda a ochrana infraštruktúry.

**Bezpečnostné aktivity:** Definovanie bezpečnostných požiadaviek (napr. pre autentifikáciu, autorizáciu) , počiatočné posúdenie rizík a klasifikácia dát.

---

# FÁZY SSDLC (2) - PLÁNOVANIE A NÁVRH

## ZÁKLADY BEZPEČNEJ ARCHITEKTÚRY

### Fáza 2: Analýza a návrh

Vo fáze návrhu je bezpečnosť priamo zapracovaná do architektúry. Princípy „security-by-design“ pomáhajú tímom predvídať hrozby, ako sú SQL injection alebo cross-site scripting, a zabudovať kontrolné mechanizmy priamo do aplikácie.

**Bezpečnostné aktivity: Modelovanie hrozieb (Threat Modeling)** na identifikáciu hrozieb v architektúre , aplikácia princípov bezpečného návrhu (Defense in Depth , Least Privilege , Fail Securely, Šifrovanie, Autentifikácia a Autorizácia).

---

# FÁZY SSDLC (3) - VÝVOJ A TESTOVANIE

## OD BEZPEČNÉHO KÓDU K ODHALENIU ZRANITEĽNOSTÍ

### Fáza 3: Vývoj a implementácia

Bezpečné programovanie je pre vývojárov neustály proces, pričom existuje niekoľko všeobecných stratégií, ktoré by mali dodržiavať.

Zavedenie týchto postupov zabezpečuje, že zraniteľnosti sú riešené už v raných fázach, čím sa znižujú potenciálne útočné vektory.

**Bezpečnostné aktivity:** Dodržiavanie štandardov bezpečného kódovania (OWASP) , používanie bezpečných API , revízia kódu (Code Review) a bezpečná správa tajomstiev (Secrets Management), nástroje pre SAST

---

# FÁZY SSDLC (4) - VÝVOJ A TESTOVANIE

## OD BEZPEČNÉHO KÓDU K ODHALENIU ZRANITEĽNOSTÍ

### Fáza 4: Testovanie

SSDLC integruje bezpečnostné testovanie ako nepretržitú súčasť vývoja. Tento krok overuje bezpečnosť kódu ešte pred jeho nasadením. Integrácia automatizovaných testov do CI/CD pipeline poskytuje priebežnú istotu, že bezpečnostné štandardy sú dodržané, a zabezpečuje včasné odhalenie a vyriešenie problémov.

### Bezpečnostné aktivity:

**SAST:** Analýza zdrojového kódu.

**DAST:** Testovanie bežiacej aplikácie.

**IAST:** Kombinácia SAST a DAST.

**SCA:** Skenovanie knižníc tretích strán.

**Penetračné testovanie:** Simulácia reálneho útoku.

Metodika	Princíp	Fáza v SDLC	Kľúčové Výhody	Kľúčové Nevýhody
<b>SAST</b>	White-box (analýza kódu)	Vývoj, CI/CD	Včasná detekcia, nízke náklady na opravu.	Vyšší počet falošných poplachov.
<b>DAST</b>	Black-box (útok na bežiacu app)	Testovanie, Staging	Simuluje reálne útoky, nízky počet falošných poplachov.	Neskorá detekcia v SDLC.
<b>IAST</b>	Grey-box (agent vnútri app)	Testovanie, QA	Kombinuje výhody SAST a DAST, presná lokalizácia chyby.	Vyžaduje inštrumentáciu aplikácie.

## TESTOVANIE APLIKAČNEJ BEZPEČNOSTI (AST)

---

# FÁZY SSDLC (5) - NASADENIE A ÚDRŽBA BEZPEČNOSTĚ V PRODUKCII A PO NEJ

## Fáza 5: Nasadenie

Počas nasadzovania sa konfigurácia a riadenie prístupu stávajú kľúčovými. Tímy by mali:

- Uplatňovať princíp najmenších privilégii v prostrediach na nasadzovanie.
- Používať konfigurácie špecifické pre jednotlivé prostredia, aby sa znížili riziká spojené so zdieľanými prihlasovacími údajmi.
- Monitorovať nasadzovanie pomocou bezpečnostných logov pre potreby auditov.
- Zabezpečením procesu nasadzovania môžu tímy udržiavať bezpečnosť aj vtedy, keď sa zmeny presúvajú do produkčných prostredí.

**Bezpečnostné aktivity:** Bezpečná konfigurácia (**Hardening**) systémov podľa štandardov ako CIS Benchmarks a zabezpečenie CI/CD Pipeline.

---

# FÁZY SSDLC (6) - NASADENIE A ÚDRŽBA BEZPEČNOSTĚ V PRODUKCII A PO NEJ

## Fáza 6: Údržba

Bezpečnost' sa nasadením nekončí. Neustále monitorovanie, správa záplat a reakcia na incidenty zabezpečujú, že zraniteľnosti sú rýchlo riešené.

**Bezpečnostné aktivity:** Nepretržité monitorovanie logov, pravidelná správa záplat (**Patch Management**) a pripravený Plán reakcie na incidenty (Incident Response Plan).

---

# ANALÝZA SOFTVÉROVEJ KOMPOZÍCIE (SCA)

## Čo sa skrýva vo vašom dodávateľskom reťazci?

- **Problém:** Moderné aplikácie sú z veľkej časti zložené z open-source komponentov, ktoré môžu obsahovať zdedené zraniteľnosti.
- **SCA:** Automatizovaný proces, ktorý skenuje závislosti a knižnice tretích strán na identifikáciu známych zraniteľností (CVEs) a licenčných problémov.
- **Výstup: Software Bill of Materials (SBOM)** – detailný zoznam všetkých komponentov, z ktorých je softvér zložený.
- **Význam:** Základný kameň pre zabezpečenie softvérového dodávateľského reťazca.

---

# ANALÝZA SOFTVÉROVEJ KOMPOZÍCIE POŽIADAVKY NA SBOM :

- **Zoznam všetkých komponentov**  
(knižnice, moduly, závislosti – vrátane open-source)
- **Verzie komponentov**  
umožňujú sledovať zraniteľné alebo zastarané verzie.
- **Informácie o dodávateľovi**  
kto vytvoril jednotlivé časti softvéru.
- **Licenčné informácie**  
typ licencie pre každý komponent.
- **Väzby a závislosti**  
ako sú komponenty prepojené a od čoho závisia.
- **Formát čitateľný strojom**  
napr. SPDX, CycloneDX, SWID.
- **Aktualizácia a dostupnosť**  
SBOM musí byť aktuálny a dostupný počas celého životného cyklu softvéru.

---

# MODELOVANIE HROZIEB (THREAT MODELING)

## Identifikácia hrozieb pred napísaním kódu

- **Účel:** Štruktúrovaný proces identifikácie, enumerácie a prioritizácie potenciálnych hrozieb a zraniteľností.
- **Kedy:** Primárne vo fáze návrhu softvéru, ešte pred napísaním kódu.
- **Cieľ:** Odpovedať na kľúčové otázky:
  1. Na čom pracujeme?
  2. Čo sa môže pokaziť?
  3. Čo s tým urobíme?
  4. Urobili sme to dobre?

---

# METODIKY MODELOVANIA HROZIEB

- **STRIDE:** Metodika od Microsoftu, ktorá klasifikuje hrozby do šiestich kategórií.
- **PASTA (Process for Attack Simulation and Threat Analysis):** Risk-centrická, 7-fázová metodika, ktorá spája technické hrozby s biznisovými dopadmi.
- **DREAD:** Model na kvantitatívnu prioritizáciu hrozieb.

---

# METODIKY MODELOVANIA HROZIEB : STRIDE

- **STRIDE** je osvedčená metodika modelovania hrozieb vytvorená spoločnosťou Microsoft, ktorá sa časom vyvinula a stala sa jednou z najefektívnejších dostupných metodík. Táto technika efektívne identifikuje hranice systému, udalosti a entity ich aplikáciou na diagramy toku dát (DFD).

# METODIKY MODELOVANIA HROZIEB : STRIDE

Hrozba	Ohrozená vlastnosť	Definícia hrozby
Spoofing	Autentifikácia	Útočník sa vydáva za niekoho iného.
Tampering	Integrita	Zmena kódu alebo dôležitých dát.
Repudiation	Neodvolateľnosť	Nemožnosť preukázať aktivitu používateľa.
Information Disclosure	Dôvernosť	Únik citlivých alebo súkromných údajov.
Denial of Service	Dostupnosť	Zablokovanie prístupu oprávneným osobám.
Elevation of Privilege	Autorizácia	Získanie práv bez oprávnenia.

---

# METODIKY MODELOVANIA HROZIEB : PASTA

- Proces simulácie útokov a analýzy hrozieb (Process for Attack Simulation and Threat Analysis - PASTA) je 7-kroková metodika modelovania hrozieb zameraná na riziko. Keďže sa PASTA viac zameriava na hrozby s najvyšším rizikom, pomáha venovať viac času a zdrojov dôležitým zraniteľnostiam a menej pozornosti venuje hrozbám s malým dopadom. V skutočnosti PASTA prikladá väčší význam obchodnému kontextu ako iné metodiky modelovania hrozieb, ako napríklad STRIDE.

# METODIKY MODELOVANIA HROZIEB : PASTA

Krok	Popis
1. Identifikácia aktív a architektúry	Zistenie, čo aplikácia obsahuje a ako je navrhnutá.
2. Definovanie prostredia hrozieb	Určenie, akým hrozbám aplikácia čelí.
3. Funkčná dekompozícia	Rozdelenie funkcií a spôsobov, ako môžu útočníci využiť slabiny.
4. Identifikácia útokových scenárov	Určenie najdôležitejších možných útokov.
5. Analýza scenárov (STRIDE)	Použitie rámca STRIDE na detailné vyhodnotenie.
6. Identifikácia hroziacich aktérov	Kto môže útoky vykonať (interní/externí útočníci).
7. Prioritizácia a mitigácia	Určenie priorít a návrh protopatrení.

---

# METODIKY MODELOVANIA HROZIEB : DREAD

- **DREAD** je metodika modelovania hrozieb vyvinutá spoločnosťou Microsoft, ktorá je skratkou pre Damage Potential (potenciál poškodenia), Reproducibility (reprodukovateľnosť), Exploitability (zneužitelnosť), Affected users (ovplyvnení používateľia) a Discoverability (zistiteľnosť).
- Táto metodika slúži ako rámec, ktorý pomáha používateľom identifikovať hrozby a posúdiť úroveň rizika spojenú s každou z nich.
- DREAD sa často používa v kombinácii s modelom STRIDE, kde STRIDE identifikuje hrozby a DREAD sa potom používa na hodnotenie závažnosti hrozieb. Tento model vám poskytuje číselné hodnotenie, ktoré potom môžete použiť na stanovenie priorít hrozieb a príslušných stratégií zmiernenia.

# METODIKY MODELOVANIA HROZIEB : DREAD

Metrix	Popis (SK)
Damage potential	Maximálna možná škoda, ktorú hrozba môže spôsobiť (10 = extrémna škoda, napr. úplné obídenie bezpečnostných kontrol).
Reproducibility	Ako ľahko je útok reprodukovateľný; 10 = funguje vždy, nižšie = len pri špecifických podmienkach.
Exploitability	Potrebné zručnosti a zdroje na vykonanie útoku; 10 = môže spraviť aj script kiddie, nižšie = len štátne aktéru).
Affected users	Podiel postihnutých používateľov (1 = 0–10%, 2 = 11–20%, ..., 10 = 91–100%).
Discoverability	Pravdepodobnosť, že útočník objaví zraniteľnosť; 10 = takmer isté, 1 = malá pravdepodobnosť.

---

# PRINCÍPY BEZPEČNÉHO KÓDOVANIA

## Stavebné kamene odolného softvéru

- **Definícia:** Disciplína písania kódu, ktorý je odolný voči útokom.
- Ide o praktickú implementáciu bezpečnostných princípov definovaných vo fáze návrhu.
- Je to najúčinnější spôsob, ako eliminovať celé triedy zraniteľností priamo pri ich zdroji – v kóde.
- **Kľúčové referenčné rámce:**
  - [OWASP Secure Coding Practices Quick Reference Guide](#)
  - [OWASP Development Guide](#)
  - [MITRE CWE/SANS Top 25 Most Dangerous Software Weaknesses](#)

---

# BEZPEČNÉ KÓDOVANIE: KLÚČOVÉ TECHNIKY (1/2)

- **Validácia vstupov (Input Validation):**
  - Všetky dáta z nedôveryhodných zdrojov musia byť validované na strane servera.
  - Preferovaný prístup je **Whitelisting** (povolenie len známych dobrých formátov).
  - Mitigované hrozby: Injection útoky (SQLi, Command Injection).
- **Kódovanie výstupov (Output Encoding):**
  - Neutralizácia špeciálnych znakov pred zobrazením dát používateľovi.
  - Použitie **kontextového kódovania** (napr. pre HTML telo, JavaScript).
  - Mitigované hrozby: Cross-Site Scripting (XSS).
- **Autentifikácia a správa hesiel:**
  - Implementácia **Multi-Faktorovej autentifikácie (MFA)**.
  - Používanie silných, solených a adaptívnych hashovacích funkcií (**Argon2, bcrypt**).

---

# BEZPEČNÉ KÓDOVANIE: KLÚČOVÉ TECHNIKY (2/2)

- **Riadenie prístupu (Access Control):**
  - Autorizačné kontroly musia byť vynucované výlučne na **strane servera**.
  - Aplikácia **princípu najnižších privilégií** (Least Privilege).
- **Bezpečnosť databáz:**
  - Používanie **parametrizovaných dopytov (Prepared Statements)** je najúčinnnejšou obranou proti SQL Injection.
- **Správa pamäte (pre jazyky ako C/C++):**
  - Dôsledná **kontrola hraníc (Bounds Checking)** na zabránenie Buffer Overflow chybám.
  - Mitigované hrozby: Out-of-bounds Write (CWE-787), Use After Free (CWE-416).

---

# ČASŤ 3:

## ANALÝZA MODERNÝCH HROZIEB A ZRANITEĽNOSTÍ

### Pochopenie protivníka

- Pre efektívnu obranu je nevyhnutné hĺbkové pochopenie prostredia hrozieb.
- Trojuholník porozumenia hrozbám:
  - **OWASP Top 10:** Popisuje, **čo** sa môže aplikácii stať (symptómy).
  - **MITRE CWE Top 25:** Vysvetľuje, **prečo** sa to stalo (koreňové príčiny v kóde).
  - **MITRE ATT&CK:** Detailne mapuje, **ako** útočník slabiny zneužije (taktiky a techniky).

---

# OWASP TOP 10 2021 (1/3)

- **A01:2021-Broken Access Control (Narušené riadenie prístupu):**
  - Postúpila na prvé miesto; až 94% testovaných aplikácií malo nejakú formu tejto chyby.
  - Príklad: Modifikácia URL parametra na prístup k účtu iného používateľa.
- **A02:2021-Cryptographic Failures (Kryptografické zlyhania):**
  - Zameriava sa na zlyhania pri ochrane dát "at rest" aj "in transit".
  - Príklad: Ukladanie hesiel v čitateľnej forme alebo pomocou slabých hashov (napr. MD5).
- **A03:2021-Injection (Vkladanie škodlivého kódu):**
  - Konsolidovaná kategória, ktorá teraz zahŕňa aj Cross-Site Scripting (XSS).
  - Príklad: SQL Injection na manipuláciu s databázou.

---

# OWASP TOP 10 2021 (2/3)

- **A04:2021-Insecure Design (Nezabezpečený návrh):**
  - Nová kategória zameraná na fundamentálne chyby v dizajne a architektúre.
- **A05:2021-Security Misconfiguration (Chybná bezpečnostná konfigurácia):**
  - Vyplýva z ponechania predvolených hesiel, otvorených portov alebo príliš informatívnych chybových hlášok.
- **A06:2021-Vulnerable and Outdated Components (Zraniteľné a zastarané komponenty):**
  - Riziko vyplývajúce z používania knižníc a frameworkov so známymi zraniteľnosťami (napr. Log4j).

---

# OWASP TOP 10 2021 (3/3)

- **A07:2021-Identification and Authentication Failures:**
  - Príklad: Credential stuffing, slabé procesy obnovy hesla.
- **A08:2021-Software and Data Integrity Failures:**
  - Zameraná na integritu aktualizácií, CI/CD pipeline a riziko nebezpečnej deserializácie.
- **A09:2021-Security Logging and Monitoring Failures:**
  - Nedostatočné logovanie bráni včasnej detekcii a vyšetrovaniu incidentov.
- **A10:2021-Server-Side Request Forgery (SSRF):**
  - Umožňuje útočníkovi prinútiť server, aby odoslal požiadavku do internej siete.

---

# OWASP API SECURITY TOP 10 2023

## RASTÚCI VÝZNAM ZABEZPEČENIA API

- API sú chrbtovou kosťou moderných architektúr (mikroslužby, mobilné aplikácie), a preto sa stávajú primárnym cieľom.
- **Najkritickejšie riziká podľa verzie z roku 2023:**
  - **API1:2023 - Broken Object Level Authorization (BOLA):** Najčastejšia a najkritickejšia zraniteľnosť; útočník zmenou ID v požiadavke získa prístup k cudzím dátam.
  - **API2:2023 - Broken Authentication:** Zlyhania v autentifikačných mechanizmoch API.
  - **API3:2023 - Broken Object Property Level Authorization:** Nedostatočná kontrola prístupu k jednotlivým poliam v dátovom objekte.
  - **API9:2023 - Improper Inventory Management:** Existencia "tieňových" alebo zastaraných API, ktoré nie sú zdokumentované a zabezpečené.

---

# MITRE CWE TOP 25: KOREŇOVÉ PRÍČINY

## Prečo zraniteľnosti vznikajú?

- Zatiaľ čo OWASP Top 10 popisuje symptómy, **MITRE Common Weakness Enumeration (CWE)** sa zameriava na ich koreňové príčiny – chyby v kóde, dizajne alebo architektúre.
- Zoznam CWE Top 25 je zostavovaný analýzou tisícov reálnych zraniteľností (CVE) a hodnotí slabiny na základe ich **frekvencie a závažnosti**.
- Pre expertov je to kľúčový nástroj na pochopenie, ktoré programátorské chyby vedú k najhorším následkom.

---

# CWE TOP 25: KLÚČOVÉ KATEGÓRIE

- **Problémy s bezpečnosťou pamäte (Memory Safety Issues):**
  - Dlhodobo dominujú na čele rebríčka a často vedú k RCE (Remote Code Execution).
  - **CWE-787: Out-of-bounds Write** (Zápis mimo buffera)
  - **CWE-416: Use After Free** (Použitie pamäte po uvoľnení)
  - **CWE-125: Out-of-bounds Read** (Čítanie mimo buffera)
- **Injection a Validácia Vstupov:**
  - **CWE-89: SQL Injection**
  - **CWE-79: Cross-site Scripting (XSS)**
  - **CWE-78: OS Command Injection**
- **Problémy s riadením prístupu:**
  - **CWE-862: Missing Authorization** (Chýbajúca autorizácia)
  - **CWE-863: Incorrect Authorization** (Nesprávna autorizácia)

---

# ZHRNUTIE KLÚČOVÝCH ZISTENÍ

- **Bezpečnosť musí byť integrovaná, nie pridaná:** Princíp "Shift-Left" a kultúra DevSecOps sú základom modernej a efektívnej stratégie.
- **Obrana musí byť viacvrstvová a adaptívna:** "Defense in Depth" a princípy Zero Trust sú nevyhnutné na ochranu proti sofistikovaným hrozbám.
- **Pochopenie protivníka je kľúčové:** Rámce OWASP, CWE a MITRE ATT&CK poskytujú komplexný pohľad na hrozby, ich príčiny a metódy útočníkov.
- **AI je nevyhnutnosť, ale ľudská expertíza zostáva nenahraditeľná:** Budúcnosť je v hybridnom modeli človek-stroj, kde AI posilňuje, nie nahrádza, ľudských analytikov.

---

# STRATEGICKÉ ODPORÚČANIA

## AKČNÉ KROKY PRE PRAX

1. **Zaved'te základnú kybernetickú hygienu:** Začnite s implementáciou **CIS Controls Implementation Group 1 (IG1)**.
2. **Integrujte bezpečnosť do vývoja:** Presadzujte zavedenie **SAST a SCA nástrojov** priamo do CI/CD pipeline.
3. **Zvýšte viditeľnosť na koncových bodoch:** Doplňte existujúce riešenia o technológiu **Endpoint Detection and Response (EDR)**.
4. **Implementujte modelovanie hrozieb:** Zaved'te proces modelovania hrozieb (napr. **STRIDE**) pre kľúčové aplikácie.
5. **Pripravte sa na incidenty:** Vypracujte a pravidelne testujte **Incident Response Plan** a zabezpečte centralizované logovanie (SIEM).

---

# ČASŤ 4:

## RIADENIE, REAKCIA A BUDÚCE TRENDY

### Spojenie technológie, procesov a stratégie

- Zabezpečenie softvéru a ochrana pred malvérom nie sú len technické problémy.
- Vyžadujú si robustné procesy riadenia, plánovanie reakcie na incidenty a neustále prispôsobovanie sa budúcim trendom.
- Táto časť spája technické kontroly so strategickými cieľmi organizácie.

---

# BUDÚCE TRENDY

## Čo nás čaká?

- **Zabezpečenie AI:** Prioritou sa stane ochrana samotných AI modelov pred útokmi typu Adversarial ML.
- **Kvantová hrozba:** Nástup kvantových počítačov predstavuje hrozbu pre súčasnú kryptografiu. Prechod na **post-quantovú kryptografiu (PQC)** bude obrovskou výzvou.
- **Sprísňovanie regulácií:** Očakáva sa zvýšený tlak na preukázateľné bezpečnostné programy a požiadavky na **SBOM** sa pravdepodobne stanú štandardom.

---

# ĎAKUJEM ZA POZORNOSŤ

- **Otázky a odpovede**