9. Přednáška: OWASP Top 10

Bezpečnost ICT 2

Willi Lazarov

Vysoké učení technické v Brně
lazarov@vut.cz

2022

Informační bezpečnost

Základní informace

- **OWASP (Open Web Application Security Project)** je projekt a otevřená komunita, která se primárně zaměřuje na bezpečnost webových a mobilních aplikací.
- Komunitu a veškeré projekty sdružuje mezinárodní nezisková organizace **OWASP Foundation**.
- OWASP otevřeně nabízí:
  - bezpečnostní nástroje a standardy,
  - návody pro bezpečnostní testování,
  - doporučení pro bezpečný vývoj,
  - mezinárodní konference,
  - vzdělávací projekty.
- Všechny nástroje, standardy a další dokumenty jsou dostupné **zdarma** všem, kdo se zajímají o zlepšení bezpečnosti aplikací.

Známé dokumenty

- Jednou z hlavních činností je standardizace bezpečnostního testování webových a mobilních aplikací:
  - OWASP **Testing Guide**,
  - OWASP **ASVS** (Application Security Verification Standard)[1],
  - OWASP **MASVS** (Mobile ASVS),
  - OWASP **WSTG** (Web Security Testing Guide),
  - OWASP **MSTG** (Mobile Security Testing Guide).
- Nejznámějším dokumentem je žebříček **OWASP Top 10**:
  - obsahuje 10 nejkritičtějších webových zranitelností,
  - popisuje zranitelnosti a způsoby jejich zamezení,
  - roky vydání: 2004, 2007, 2010, 2013, 2017 a 2021,
  - laboratorní úloha OWASP Top 10 2021 (BPC-IC2).

[1] Podrobněji probíráno v předmětu MPC-CT3.

OWASP Top 10 2010 a 2013

Tabulka: Srovnání kategorií OWASP Top 10 2010 a 2013

OWASP Top 10 – 2010		**OWASP Top 10 – 2013**
A01 – Injection		A01 – Injection
A02 – Cross-Site Scripting (XSS)	↑	A02 – Broken Authentication and Session Management
A03 – Broken Authentication and Session Management	↓	A03 – Cross-Site Scripting (XSS)
A04 – Insecure Direct Object References		A04 – Insecure Direct Object References
A05 – Cross-site Request Forgery (CSRF)	↑	A05 – Security Misconfiguration
A06 – Security Misconfiguration	+	A06 – Sensitive Data Exposure
A07 – Insecure Cryptographic Storage	+	A07 – Missing Function Level Access Control
A08 – Failure to Restrict URL Access	↓	A08 – Cross-site Request Forgery (CSRF)
A09 – Insufficient Transport Layer Protection	+	A09 – Using Components With Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards		A10 – Unvalidated Redirects and Forwards

Pozn.: ↑ zvýšení kategorie, ↓ snížení kategorie, + nová kategorie.

OWASP Top 10 2013 a 2017

Tabulka: Srovnání kategorií OWASP Top 10 2013 a 2017

OWASP Top 10 – 2013		**OWASP Top 10 – 2017**
A01 – Injection		A01 – Injection
A02 – Broken Authentication and Session Management		A02 – Broken Authentication
A03 – Cross-Site Scripting (XSS)	↑	A03 – Sensitive Data Exposure
A04 – Insecure Direct Object References	+	A04 – XML External Entities (XXE)
A05 – Security Misconfiguration	+	A05 – Broken Access Control
A06 – Sensitive Data Exposure	↓	A06 – Security Misconfiguration
A07 – Missing Function Level Access Control	↓	A07 – Cross-Site Scripting (XSS)
A08 – Cross-site Request Forgery (CSRF)	+	A08 – Insecure Deserialization
A09 – Using Components With Known Vulnerabilities		A09 – Using Components With Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	+	A10 – Insufficient Logging & Monitoring

Pozn.: ↑ zvýšení kategorie, ↓ snížení kategorie, + nová kategorie.

OWASP Top 10 2017 a 2021

Tabulka: Srovnání kategorií OWASP Top 10 2017 a 2021

OWASP Top 10 – 2017		**OWASP Top 10 – 2021**
A01 – Injection	↑	A01 – Broken Access Control
A02 – Broken Authentication	↑	A02 – Cryptographic Failures
A03 – Sensitive Data Exposure	↓	A03 – Injection
A04 – XML External Entities (XXE)	+	A04 – Insecure Design
A05 – Broken Access Control	↑	A05 – Security Misconfiguration
A06 – Security Misconfiguration	↑	A06 – Vulnerable and Outdated Components
A07 – Cross-Site Scripting (XSS)	↓	A07 – Identification and Authentication Failures
A08 – Insecure Deserialization	+	A08 – Software and Data Integrity Failures
A09 – Using Components With Known Vulnerabilities	↑	A09 – Security Logging and Monitoring Failures
A10 – Insufficient Logging & Monitoring	+	A10 – Server-Side Request Forgery (SSRF)

Pozn.: ↑ zvýšení kategorie, ↓ snížení kategorie, + nová kategorie.

A01 – Broken Access Control

- **Řízení přístupu** (access control) prosazuje zásady tak, aby uživatelé nemohli jednat v rozporu s přiděleným oprávněním.
- Selhání obvykle vede k neoprávněnému přístupu, modifikaci nebo zničení dat nebo vykonání vnitřní funkce systému **bez uživatelského oprávnění**.
- Mezi běžné zranitelnosti řízení přístupu patří:
  - porušení zásady minimalizace oprávnění bez autorizace,
  - obejití (bypass) procesu autorizace úpravou URL,
  - přístup k API bez předchozí autorizace (chybějící token),
  - povýšení vlastního oprávnění na úroveň administrátora,
  - přístup, úprava nebo odstranění cizí entity pomocí ID,
  - manipulace s metadaty (access control token, cookie aj.).

A01 – Broken Access Control

- **Opatření pro zamezení narušení řízení přístupu**:
  - Ve výchozím stavu zakázat přístup ke všem zdrojům kromě těch, které jsou předem určené jako veřejně dostupné.
  - Jednotně implementovat řízení přístupu napříč celou aplikací.
  - Zakázat výpis adresářů a zajistit, aby se v kořenovém adresáři nenacházela metadata souborů (např. adresář .git) a zálohy.
  - Omezit maximální počet žádostí o přístup pro daný časový interval (rate limiting a API throttling).
  - Po odhlášení uživatele zneplatňovat relaci na straně serveru.
  - Zaznamenávat selhání řízení přístupu a v konkrétních situací upozornit správce systému (např. při opakovaném selhání).

A01 – Broken Access Control

- **Scénář č. 1**:
  - Aplikace vrací záznam podle přijatého query parametru `id`:
    - `pstmt.setString(1, request.getParameter("id"));`
    - `ResultSet results = pstmt.executeQuery();`
  - Útočník dokáže jednoduchou úpravou parametru `id` v URL získat přístup k libovolnému uživatelskému účtu: https://example.com/app/account-info?id=1
  - Prvním uživatelským účtem (`id=1`) v databázi bývá často administrátor nebo jiný vysoce privilegovaný uživatel.

A01 – Broken Access Control

- **Scénář č. 2**:
  - Pro přístup ke stránce „admin-getapp-info“ by mělo být vyžadováno oprávnění administrátora: https://example.com/app/admin-getapp-info
  - Pokud může přistoupit k webové stránce „admin-getapp-info“ autentizovaný uživatel, který však není administrátorem, jedná se o zranitelnost v řízení přístupu, a ta může dovést útočníka na další nedostatečně chráněné administrátorské stránky.

A02 – Cryptographic Failures

- Kategorie Cryptographic Failures představuje slabiny, které jsou způsobené **kryptografickými nedostatky**.
- Citlivé údaje, jako jsou např. uživatelská hesla, čísla kreditních karet, zdravotní záznamy, osobní údaje a obchodní tajemství, vyžadují zvýšenou ochranu.
- Mezi běžné kryptografické nedostatky patří:
  - přenos dat v otevřené nezašifrované podobě (např. HTTP),
  - používání slabých kryptografických algoritmů a protokolů,
  - aktivní komunikace s nedůvěryhodnými certifikáty,
  - použití zastaralých hashovacích funkcí (MD5, SHA1, ...),
  - zneužitelné kryptografické chybové zprávy nebo informace získatelné z postranních kanálů (side-channel attack),
  - nízká nebo žádná míra náhodnosti kryptografického systému,
  - žádné nebo nedostatečné generování inicializačního vektoru.

A02 – Cryptographic Failures

- **Opatření pro zamezení kryptografických nedostatků**:
  - Klasifikovat citlivá data a neukládat nepotřebné citlivé údaje.
  - Pokud data nejsou nadále potřebná, měla by být zničena.
  - Zajištění šifrované komunikace a všech uložených citlivých dat.
  - Použití prokazatelně silných a doporučených kryptografických algoritmů, protokolů a hashovacích funkcí (viz NIST).
  - Hashování uživatelských hesel s použitím kryptografické soli.
  - Zajištění vysoké kryptografické náhodnosti (vysoká entropie).
  - Zakázat cachování citlivých dat získaných z odpovědi serveru.
  - Vhodně zvolit inicializační vektory pro daný provozní mód.
  - Používat autentizované šifrování namísto pouhého šifrování.
  - Nezávisle ověřit účinnost konfigurace a nastavení systému.

A02 – Cryptographic Failures

- **Scénář č. 1**:
  - Aplikace šifruje čísla kreditních karet uložených v databázi pomocí automatického šifrování. Tato citlivá data jsou však při načítání aplikace automaticky dešifrována, kvůli čemuž lze získat čísla kreditních karet v otevřeném textu.
- **Scénář č. 2**:
  - Databáze hesel používá k ukládání hesel všech uživatelů slabou hashovací funkci nebo nepoužívá kryptografickou sůl.
  - Pokud se útočníkovi podaří předchozí kompromitací prolomit zabezpečení a získat uložená hesla, dokáže zlomit neosolené hashe pomocí duhové tabulky přes předem vypočtené hashe.
  - Hash, který byl vygenerován slabou hashovací funkcí, může být s dostatečným výkonem GPU prolomen, i přestože byla použita kryptografická sůl (např. MD5, SHA-1 aj.).

A03 – Injection

- **Injekční útok** představuje napadení aplikační, systémové nebo databázové vrstvy přes neošetřený vstup aplikace.
- Nejčastěji injekční útoky cílí na:
  - SQL, NoSQL a ORM (objektově relační mapování),
  - JS (JavaScript) a EL (Expression Language),
  - OGNL (Object Graph Navigation Library),
  - příkazy operačního systému.
- Mezi běžné příčiny injekčních útoků patří:
  - Data zadaná uživatelem nejsou aplikací validována, ani nijak před jejich zpracováním sanitizována.
  - Dotazy nebo neparametrizovaná volání bez kontextového escapování se používají přímo v interpretu prog. jazyka.
  - Neošetřená data se používají v parametrech vyhledávání, dynamické části dotazu, příkazu nebo uložené procedury.

A03 – Injection

- **Opatření pro zamezení injekčních útoků**:
  - Všechny externí vstupy musí být validovány na straně serveru.
  - Pro všechny dynamické dotazy vypustit speciální znaky pomocí specifické escape syntaxe pro daný programovací jazyk.
  - Preferovanou možností je použití bezpečného rozhraní API, které se zcela vyhne interpretaci, poskytne parametrizované rozhraní nebo použije nástroje pro objektově relační mapování.
  - V SQL dotazech používat LIMIT a další kontrolní prvky, aby bylo zabráněno hromadnému zpřístupnění záznamů v případě útoku SQL injection (opatření pro zmírnění dopadu útoku).
  - Důrazně se doporučuje testování všech parametrů, hlaviček, URL, cookies a datových vstupů JSON, SOAP a XML.

A03 – Injection

- **Scénář č. 1**:
  - Aplikace používá nedůvěryhodná data během konstrukce následujícího zranitelného SQL dotazu: `String query = "SELECT FROM users WHERE id='"+ request.getParameter("id") + "'";`
  - Útočníkovi stačí pro injekční útok upravit `id` parametr v URL: https://example.com/app/account-info?id=' or '1'='1
- **Scénář č. 2**:
  - Webová aplikace vypisuje jméno uživatele následujícím řádkem kódu: `<div>Uživatelské jméno: ${auth.username}</div>`
  - Útočník si uživatelské jméno nastaví na hodnotu: `Zdenda<script src="https://hacker.com/foo.js"></script>`
  - Prohlížeč vykreslí „`Uživatelské jméno: Zdenda`“ a na pozadí spustí útočníkův skript.

A04 – Insecure Design

- **Nezabezpečený návrh** je široká kategorie představující různé nedostatky označované jako „chybějící nebo neúčinný návrh“.
- Nedostatky v návrhu a implementaci rozlišujeme z určitého důvodu, mají různé příčiny a způsoby nápravy.
- Nezabezpečený návrh nelze opravit dokonalou implementací, protože potřebná bezpečnostní kontrola proti útokům a dalším hrozbám **nebyla v rámci návrhu vytvořena**.
- Jedním z faktorů, které přispívají k nezabezpečenému návrhu, je **nedostatečné profilování rizik obchodní logiky**, které je vlastní vyvíjenému software nebo systému, a tedy neschopnost určit, jaká úroveň návrhu zabezpečení je potřebná.

A04 – Insecure Design

- **Opatření proti nezabezpečenému návrhu**:
  - Zavedení a používání bezpečného životního cyklu vývoje, který pomáhá vyhodnocovat a navrhovat bezpečnostní opatření.
  - Vytvoření a používání knihoven bezpečných návrhových vzorů.
  - Použití modelování hrozeb (threat modeling) pro ověřování řízení přístupu, obchodní logiky a dalších klíčových prvků.
  - Nasadit unit a integrační testy pro automatické ověřování odolnosti vůči použitému modelu hrozeb.
  - Kontrolovat důvěrnost a bezpečnost na každé úrovni aplikace.
  - Segregovat vrstvy na systémové a síťové vrstvě v závislosti na potřebách jejich zabezpečení a celkové zvýšení ochrany.
  - Omezení použitých prostředků podle uživatele nebo služby.

A04 – Insecure Design

- **Scénář č. 1**:
  - Aplikace používá bezpečnostní otázky a odpovědi pro ověření identity uživatelů (např. pro obnovu uživatelského hesla).
  - Otázkám a odpovědím nelze věřit jako důkazu identity, protože odpovědi může znát nebo dohledat jiná osoba.
- **Scénář č. 2**:
  - Řetězec kin nabízí slevy pro skupinové rezervace a vyžaduje uhradit zálohu maximálně pro patnáct účastníků.
  - Útočníci by mohli tento tok hrozeb modelovat a vyzkoušet, zda mohou najednou v několika požadavcích rezervovat 600 míst pro všechna kina, čímž by způsobili obrovskou ztrátu příjmů.

A05 – Security Misconfiguration

- Tato kategorie představuje žádnou, nedostatečnou a chybnou **bezpečnostní konfiguraci** aplikací, frameworků, aplikačních serverů, databáze a dalších důležitých komponent.
- Bezpečnostní konfigurace by měla být definována, prováděna a udržována, protože výchozí hodnoty zvyšují riziko zneužití.
- Příčiny nesprávné bezpečnostní konfigurace:
  - Chybějící konfigurace zabezpečení v jakékoliv části.
  - V systému jsou nainstalovány a používány nepotřebné funkce.
  - Při chybě se uživatelům zobrazují informativní chybové zprávy.
  - Konfigurace nepoužívá dostatečně bezpečné hodnoty.
  - Výchozí jména a hesla jsou v systému přítomná a nezměněná.
  - Server neodesílá bezpečnostní hlavičky ani jiné direktivy.
  - Použitý software je zastaralý nebo obsahuje zranitelnosti.

A05 – Security Misconfiguration

- **Opatření proti nesprávné bezpečnostní konfiguraci**:
  - Opakovaně aplikovat hardening, který umožňuje odhalit a zabezpečit chybějící nebo nesprávnou konfiguraci.
  - Odebrat zbytečné funkce, komponenty a další součásti aplikace.
  - Kontrolovat a aktualizovat konfigurace, aby odpovídaly všem bezpečnostním standardům, aktualizacím a vydaným záplatám pro minimalizaci zranitelností a technických nedostatků.
  - Segmentovat aplikaci pro efektivní a bezpečné oddělení komponent (virtualizace, kontejnerizace apod.).
  - Odesílat klientům pokyny pro zabezpečení jejich požadavků, například přes bezpečnostní hlavičky (Security Headers).
  - Nasadit automatizovaný proces ověřování konfigurace.

A05 – Security Misconfiguration

- **Scénář č. 1**:
  - Aplikační server je dodáván s ukázkovými (demo) aplikacemi, které nebyly odstraněny z produkčního serveru.
  - Předpokládejme, že jednou z těchto aplikací je konzole správce a výchozí přihlašovací údaje k jejímu připojení nebyly změněny.
  - V takovém případě se útočník dokáže přihlásit pomocí výchozího hesla a převzít kontrolu nad systémem.
- **Scénář č. 2**:
  - Na serveru není zakázán výpis adresářů, čehož zneužije útočník.
  - Útočník najde a stáhne zkompilované třídy jazyka Java, které dekompiluje a reverzním inženýrstvím získá zdrojový kód.
  - Získané informace útočník využije pro odhalení zranitelnosti ve zdrojovém kódu, kterou zneužije pro svůj další postup.

A06 – Vulnerable and Outdated Components

- **Zranitelné a neaktualizované komponenty** jsou známým bezpečnostním nedostatkem, který zahrnuje rizika spojená s verzí používaného software a jeho součástí.
- Příčiny přítomných zranitelností komponent:
  - Software je zranitelný, nepodporovaný nebo zastaralý.
  - Verze používaných komponent je neznámá.
  - Nedochází k pravidelné kontrole přítomnosti zranitelností a nových verzí používaných komponent.
  - Reakce na nalezené zranitelnosti a aktualizace je opožděná.
  - Vývojáři software netestují kompatibilitu aktualizovaných, upgradovaných nebo opravených komponent.

A06 – Vulnerable and Outdated Components

- **Náprava zranitelných a neaktualizovaných komponent**:
  - Odstranění nebo deaktivace všech nepoužívaných závislostí, funkcí, komponent, služeb, souborů a dokumentací.
  - Průběžná kontrola verzí komponent na straně klienta i serveru.
  - Omezit oprávnění komponent pouze na nezbytně nutný rozsah.
  - Sledování zdrojů, jako jsou databáze známých zranitelností CVE (Common Vulnerability and Exposures).
  - Všechny komponenty stahovat a instalovat pouze z oficiálních a ověřených zdrojů (např. přímo z repozitáře autora).
  - Nahrazení komponent, které nejsou nadále udržovány nebo pro jejich zranitelné verze nejsou vydány bezpečnostní záplaty.
  - Pokud záplatování není možné, nasazení virtuální záplaty pro monitorování, detekci nebo ochranu před přítomným rizikem.

A06 – Vulnerable and Outdated Components

- **Scénář č. 1**:
  - Komponenty disponují stejným oprávněním jako aplikace.
  - Nedostatky mohou být zcela náhodné (např. chyba v kódování) nebo úmyslné (např. backdoor pro vzdálené připojení).
  - Pokud útočník úspěšně kompromituje zranitelnost v takové komponentě, může získat kontrolu nad celou aplikací.
- **Scénář č. 2**:
  - Webová aplikace používá framework Apache Struts v2.3.14.
  - Pro přítomnou verzi 2.3.14 je evidována známá zranitelnost CVE-2017-5638, která spočívá ve vzdáleném spuštění kódu.
  - Útočník může tuto zranitelnost zneužít pro spuštění vlastního kódu na straně serveru a ovládnout část napadené aplikace.

A07 – Identification and Authentication Failures

- Ověření identity a bezchybná správa relace jsou zásadní pro ochranu před útoky souvisejícími s **identifikací a autentizací**.
- Příčiny selhání identifikace a autentizace:
  - Aplikace nebrání slovníkovým útokům, kdy útočník získá seznam platných uživatelských jmen (user enumeration).
  - Není implementován mechanismus, který by zabránil útokům hrubou silou nebo dalším automatizovaným útokům.
  - Jsou povolena výchozí, slabá nebo známá uživatelská hesla.
  - Používají se neúčinné procesy obnovy zapomenutých hesel.
  - Hesla jsou uložena v čitelné podobě nebo jsou hashovaná prokazatelně slabou hashovací funkcí.
  - Není zavedena vícefaktorová autentizace nebo je neúčinná.
  - Relace není zneplatněna nebo je funkce zneplatnění chybná.
  - Opakovaně se lze přihlásit se stejným identifikátorem relace.

A07 – Identification and Authentication Failures

- **Prevence proti útokům na autentizační mechanismy**:
  - Implementovat vícefaktorovou autentizaci, aby se zabránilo slovníkovým útokům, opakovanému použití přihlašovacích údajů a útokům hrubou silou.
  - Nenasazovat systém s výchozími přihlašovacími údaji.
  - Kontrola pro zamezení použití slabých nebo známých hesel.
  - Aplikovat zásady pro délku, složitost a další pravidla hesel podle aktuálního standardu NIST 800-63.
  - Zajistit, aby byly cesty pro registraci, obnovení hesla a API zabezpečeny proti automatické enumeraci uživatelských účtů.
  - Zavést reaktivní opatření pro případ opakovaného neúspěšného pokusu o přihlášení (např. po třetím neúspěšném pokusu).
  - Nasadit na straně serveru zabezpečený správce relace.

A07 – Identification and Authentication Failures

- **Scénář č. 1**:
  - Aplikace nebrání slovníkovým útokům na autentizační systém.
  - V takovém případě lze provést útok postupným zkoušením hesel pro daný uživatelský účet, dokud není nalezena shoda slovníkového hesla pro daného uživatele.
- **Scénář č. 2**:
  - K většině autentizačních útoků dochází kvůli neustálému používání hesel jako jediného ověřovacího faktoru uživatele.
  - Požadavky na střídání hesel a jejich složitost (např. použití velkého znaku), které byly kdysi považovány za bezpečné, podněcují uživatele k opakovanému používání slabých hesel.
  - Provozovatelům se doporučuje, aby tyto postupy podle normy NIST 800-63 ukončili a používali vícefaktorovou autentizaci.

A08 – Software and Data Integrity Failures

- **Porušení integrity dat a použitého software** souvisí se zdrojovým kódem a nechráněnou infrastrukturou.
- Možné příčiny porušení integrity:
  - Aplikace je závislá na knihovnách, balíčcích a dalších modulech z nedůvěryhodných zdrojů, úložišť a sítí pro doručování obsahu CDN (content delivery network).
  - Nezabezpečená CI/CD pipeline může představovat riziko pro neoprávněný přístup, nahrání škodlivého kódu nebo přímou kompromitaci systému.
  - Objekty a data jsou zakódovány nebo serializovány do části aplikace, kterou může útočník modifikovat.
  - Aktualizace se automaticky provádějí bez ověření integrity.

A08 – Software and Data Integrity Failures

- **Opatření pro zamezení porušení integrity dat a software**:
  - Použití digitálního podpisu, kontrolních součtů a podobných mechanismů k ověření, že příchozí data pocházejí z původního zdroje a nebyla nijak pozměněna.
  - Využívání pouze důvěryhodných repozitářů a dalších zdrojů.
  - Ověření, zda pro použité komponenty nejsou evidovány žádné známé zranitelnosti (CVE).
  - Pravidelná analýza zdrojového kódu, aby se minimalizovala možnost, že se do softwarového dostane škodlivý kód.
  - Implementovat CI/CD tak, aby byla zajištěna integrita během sestavení a nasazení aplikace (DevSecOps).
  - Nepodepsaná a nešifrovaná serializovaná data neodesílat nedůvěryhodné straně bez kontroly integrity nebo použití a ověření digitálního podpisu.

A08 – Software and Data Integrity Failures

- **Scénář č. 1**:
  - Mnoho routerů, switchů a dalších zařízení neověřuje aktualizace prostřednictvím podepsaného firmware.
  - Nepodepsaný firmware je stále častějším cílem útočníků a očekává se, že se tento problém bude zhoršovat.
- **Scénář č. 2**:
  - Webová aplikace využívá sadu mikroslužeb Spring Boot.
  - Vyvojáři se snažili zajistit, aby byl jejich kód neměnný.
  - Řešení, se kterým přišli, spočívá v serializaci stavu uživatele a jeho předávání v každém požadavku.
  - Útočník odhalí signaturu objektu `r00` v Base64 a pomocí Burp Suite (rozšíření Java Serial Killer) provede vzdálené spuštění kódu na serveru, na kterém se aplikace nachází.

A09 – Security Logging and Monitoring Failures

- **Selhání bezpečnostního logování a monitoringu** omezují možnost reagovat na aktivní hrozby.
- Časté nedostatky monitorování a logování:
  - Události, jako jsou neúspěšná přihlášení a další bezpečnostní aktivity, se nezaznamenávají a nikam neukládají.
  - Upozornění a chyby generují nedostatečný popis události.
  - Logy aplikace a API nejsou monitorovány z hlediska zaznamenaných podezřelých aktivit.
  - Zaznamenané události se ukládají pouze na lokální úložiště.
  - Nejsou zavedena vhodná pravidla upozornění a procesy automatické reakce, nebo jsou zavedené metody neúčinné.
  - Systém nedokáže detekovat, reagovat nebo upozornit na probíhající útoky v (téměř) reálném čase.

A09 – Security Logging and Monitoring Failures

- **Doporučení pro bezpečnostní logování a monitoring**:
  - Zaznamenávat všechny neúspěšné pokusy o přihlášení, řízení přístupu a ověřování vstupů na straně serveru.
  - Logování událostí provádět s dostatečným uživatelským kontextem pro možnost identifikovat podezřelé aktivity.
  - Logy ukládat dostatečně dlouho pro možnost pozdějšího provedení digitální forenzní analýzy.
  - Zajistit správné kódování, dekódování a ověření dat, aby bylo zabráněno injekčním útokům nebo útokům proti logovacím a monitorovacím systémům.
  - Důležité operace s daty by měly mít auditní stopu s kontrolou integrity pro zabránění neoprávněné manipulaci a odstranění.
  - Zavést plán aktivní reakce na incidenty a obnovu dat.

A09 – Security Logging and Monitoring Failures

- **Scénář č. 1**:
  - U jedné z velkých leteckých společností došlo k úniku dat, který zahrnoval osobní údaje 1 mil. cestujících za více než 10 let, včetně údajů o cestovních pasech a kreditních karet.
  - K incidentu došlo u poskytovatele cloudového hostingu třetí strany, který společnost o narušení později informoval.
- **Scénář č. 2**:
  - Poskytovatel zdravotního pojištění pro děti nedokázal včas odhalit a zastavit únik záznamů více než 3,5 milionu dětí, které útočník získal a upravil.
  - Vzhledem k tomu, že systém nebyl monitorován ani logován, mohlo narušení bezpečnosti údajů probíhat již od roku 2013, tedy po dobu více než 8 let.

A10 – Server-Side Request Forgery (SSRF)

- K chybám **SSRF** dochází v případech, kdy webová aplikace komunikuje se vzdáleným zdrojem bez dalšího ověření.
- Útočník tak dokáže aplikaci přimět k odeslání vytvořeného požadavku na vybraný cíl (často pod kontrolou útočníka).
- Příčiny možnosti provedení SSRF:
  - Uživatel má určitou kontrolu nad odesílanými požadavky.
  - Kontrola řízení přístupu je implementována v jiné části, která se nachází před aplikací a při zpětném připojení ji lze obejít.
  - Rozhraní administrace naslouchá na jiném portu než aplikace, a proto nemusí být pro uživatele přímo dostupné.
  - Pro účely obnovy systému po jeho pádu umožňuje aplikace přístup ke správě z místní sítě bez nutnosti přihlášení.
  - Útočník ovládá službu 3. strany, se kterou aplikace komunikuje.

A10 – Server-Side Request Forgery (SSRF)

- **Doporučení pro ochranu před útoky SSRF**:
  - Ověřovat a sanitizovat všechna vstupní data, zejména data zadaná klientem.
  - Segmentovat funkce pro vzdálený přístup k prostředkům v oddělených sítích s cílem snížit dopad útoku SSRF.
  - Nastavit bránu firewall ve výchozím stavu jako odepřít (deny) nebo nastavit pravidla řízení přístupu k síti tak, aby byl kromě nezbytného blokován veškerý provoz.
  - Vynucení schématu URL, portu a cíle pomocí whitelist pravidel.
  - Neodesílat klientům nezpracované odpovědi ze serveru.
  - Zakázat přesměrování na HTTP a vyžadovat vždy šifrovanou komunikaci přes HTTPS.

A10 – Server-Side Request Forgery (SSRF)

- **Scénář č. 1**:
  - Útočníci mohou získat přístup k lokálním souborům nebo interním službám a získat tak citlivé informace.
  - Příklad: `file:///etc/passwd` a `http://localhost:28017/`.
- **Scénář č. 2**:
  - Útočník dokáže zneužít interní služby k dalším útokům, jako je vzdálené spuštění kódu (RCE) nebo odepření služby (DoS).
- **Scénář č. 3**:
  - Pokud síťová infrastruktura není nijak segmentovaná, mohou útočníci zmapovat vnitřní síť a na základě připojení, uplynulé doby spojení nebo odmítnutí spojení a použitím SSRF zjistit, zda jsou porty na vnitřních serverech otevřené nebo zavřené.

Shrnutí – OWAS

- Komunita **OWASP**:
  - Zaměřuje se na bezpečnost webových a mobilních aplikací.
  - Cílem je v této oblasti informovat, vzdělávat a standardizovat.
- Testování **webových aplikací**:
  - OWASP ASVS,
  - OWASP WSTG.
- Testování **mobilních aplikací**:
  - OWASP MASVS,
  - OWASP MSTG.
- Dokument **OWASP Top 10**:
  - Pojednává o nejzávažnějších webových zranitelností.
  - Vydán v letech 2004, 2007, 2010, 2013, 2017 a 2021.

Reference I

[1] OWASP Top Ten. *OWASP Foundation*. [online]. [cit. 2022-10-31]. Dostupné z: https://owasp.org/www-project-top-ten/.

[2] OWASP Application Security Verification Standard. *OWASP Foundation*. [online]. [cit. 2022-10-31]. Dostupné z: https://owasp.org/www-project-application-security-verification-standard/

[3] Projects. *OWASP Foundation*. [online]. [cit. 2022-10-31]. Dostupné z: https://owasp.org/projects/.

[4] FREDJ, O. B.; CHEIKHROUHOU, O.; KRICHEN, M.; HAMAM, H.; DERHAB, A. An OWASP top ten driven survey on web application protection methods. In: *International Conference on Risks and Security of Internet and Systems*. Springer, Cham, 2020. p. 235–252. ISBN 978-3-030-68887-5.

Děkuji za pozornost!

Dotazy ?

lazarov@vut.cz