DCIT

Bezpečnosť webových aplikácií

Autor: Martin Zajíček
(zajicek@dcit-consulting.sk)

DCIT Consulting, http://www.dcit-consulting.sk

CONSULTING
DCIT

Obsah prezentácie

DCIT

I. Architektúra webovej aplikácie

- architektúra
- interakcia s okolitými systémami

II. Problematika návrhu

- typické webové slabiny, okruhy problémov

III. Testovanie zraniteľností

- testovanie odolnosti
- metódy, nástroje, štandardy

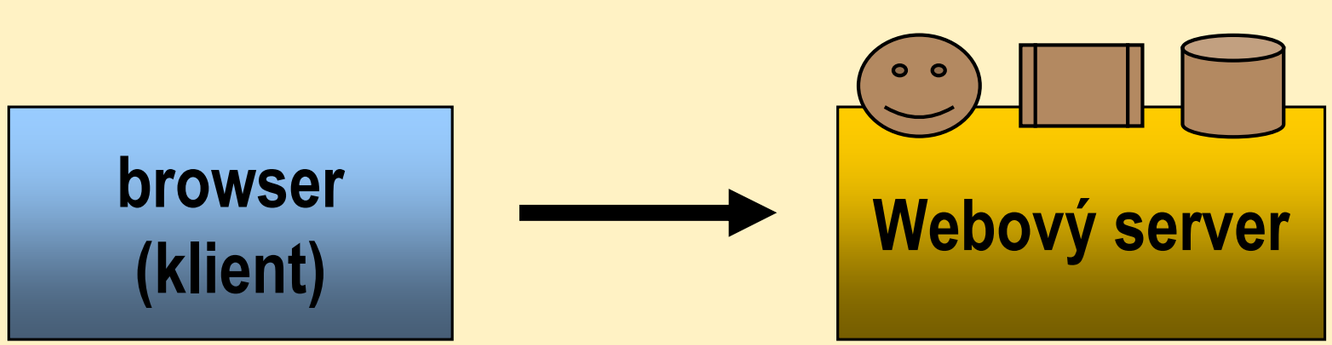
I. Architektúra webových aplikácií

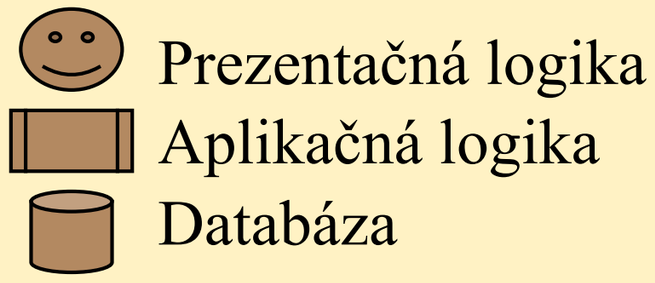
Architektúra webovej aplikácie 1

DCIT

Základná stavba „webovej aplikácie“

- (webový) klient - server
  - Všetka aplikačná a prezentačná logika na jedinom serveri – hostiteľovi





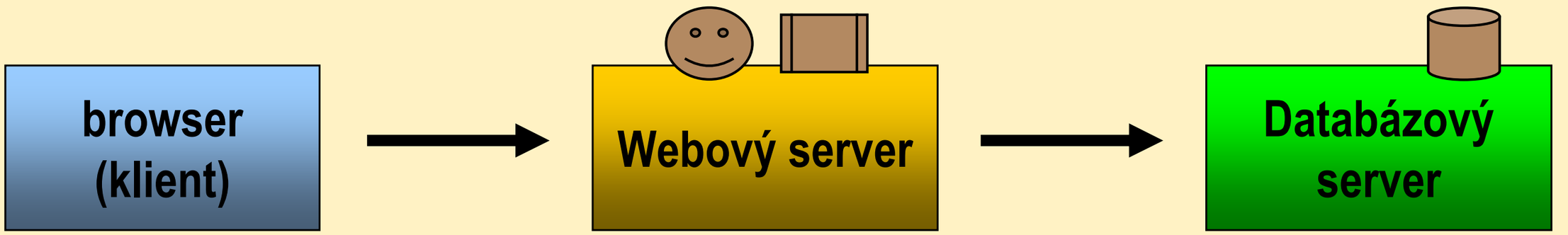
CONSULTING DCIT

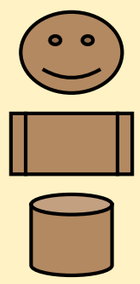
Architektúra webovej aplikácie 2

DCIT

Štruktúra komplexnejšej aplikácie

- Tradičná trojvrstvová architektúra
  - klient
  - aplikačný /prezentačný server
  - databáza





Prezentačná logika
Aplikačná logika
Databáza

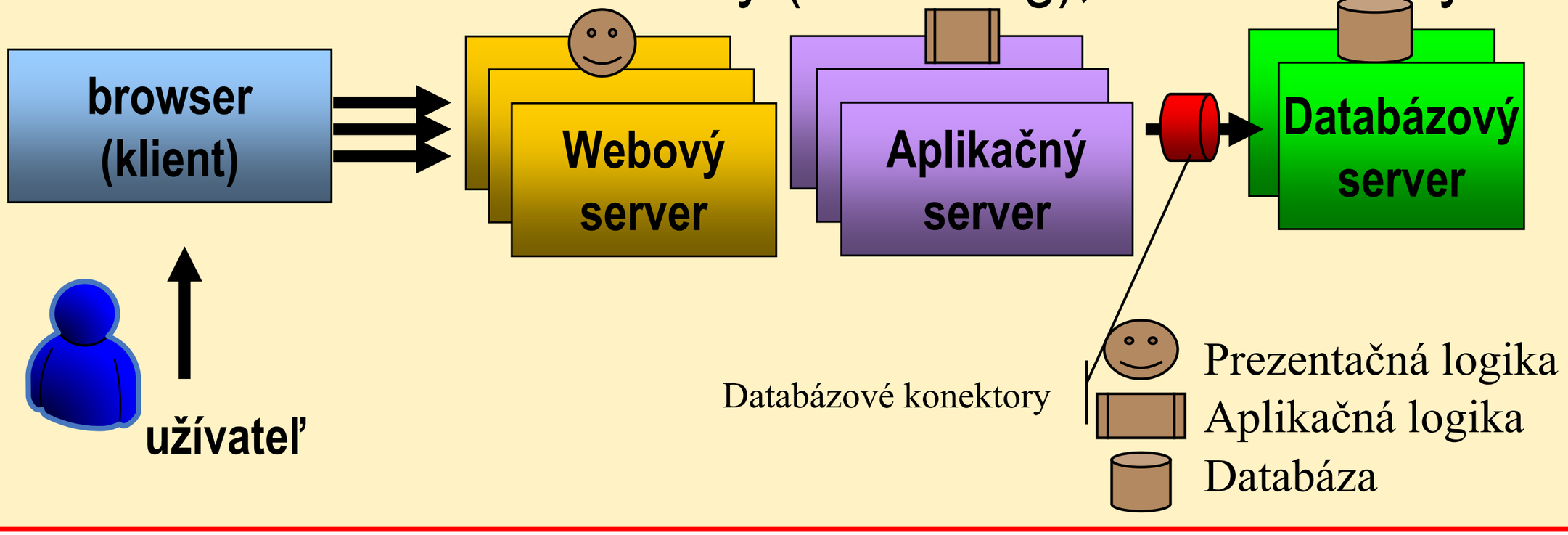
CONSULTING
DCIT

Architektúra webovej aplikácie 3

DCIT

Štruktúra n-vrstvovej architektúry

- Tradičná 3-vrstvová architektúra sa rozpadá
  - Prezentačné servery (farmy, loadbalancing)
  - Aplikačné servery (loadbalancing)
  - Databázové servery (clustering), DB konektory …



CONSULTING DCIT

Architektúra webovej aplikácie 4

DCIT

Architektúra klient - server

- klient - bežný prehliadač (MSIE, Firefox…), platformovo nezávislý (mnohé konverzie, preklady atď.),
- klient - zdieľaný mnohými (navzájom si nedôverujúcimi) aplikáciami, nutnosť „separácie“ kontextov zdieľaných aplikácií

Užívateľské rozhranie aplikácie

- realizované pomocou klienta prostriedkami WWW (HTML, CSS, JavaScript,Applety,ActiveX…), nepotrebuje ďalší klientsky SW
- prevádzka klienta úplne mimo kontroly aplikácie

Komunikácia

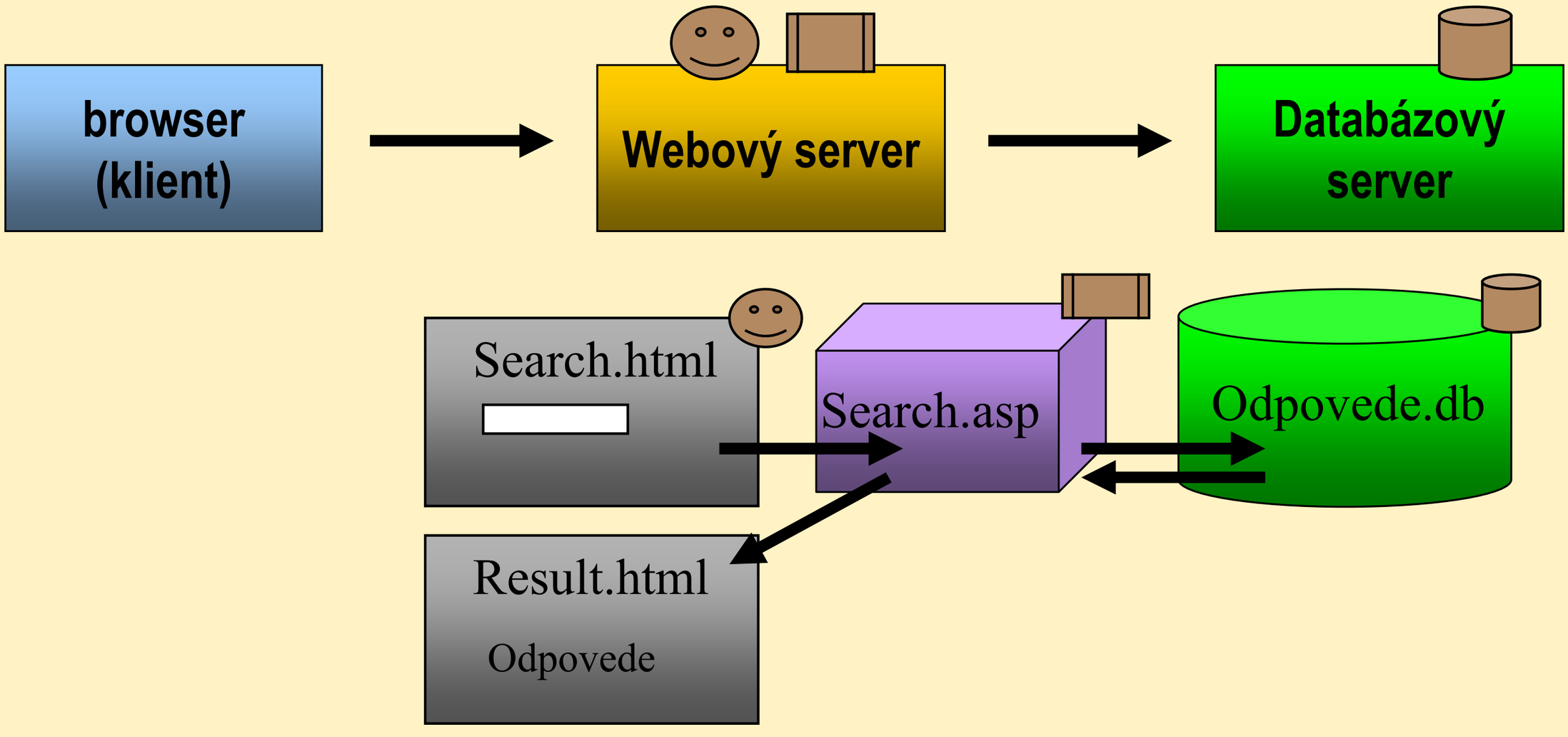
- nedôveryhodné prostredie (rozľahlé paketové siete - Internet)
  - **Klient – WiFi – ISP – Internet – ISP –** WebServer
- preklady formátov a kódovania, problémy so správnou (jednoznačnou) interpretáciou
  - **Dátový záznam** - Databáza (SQL) – aplikačný server (XML) – prezentačný server (HTTP/S) – klient (DOM) - **užívateľ**

CONSULTING DCIT

Architektúra webovej aplikácie 5

DCIT

Princíp činnosti webovej aplikácie



CONSULTING DCIT

Bezpečnostné špecifiká 1

DCIT

Dostupnosť

- široký okruh „užívateľov“, napr. anonymný prístup do celej aplikácie či verejne prístupný vstupný bod (LoginForm)

Prepojenie internej a externej siete

- **web. aplikácie sprístupňujú (čiastočne) interné systémy**
- interné systémy nie sú pripravené na rizikové prostredie (s nulovou dôverou k protistranám)

Unikátnosť

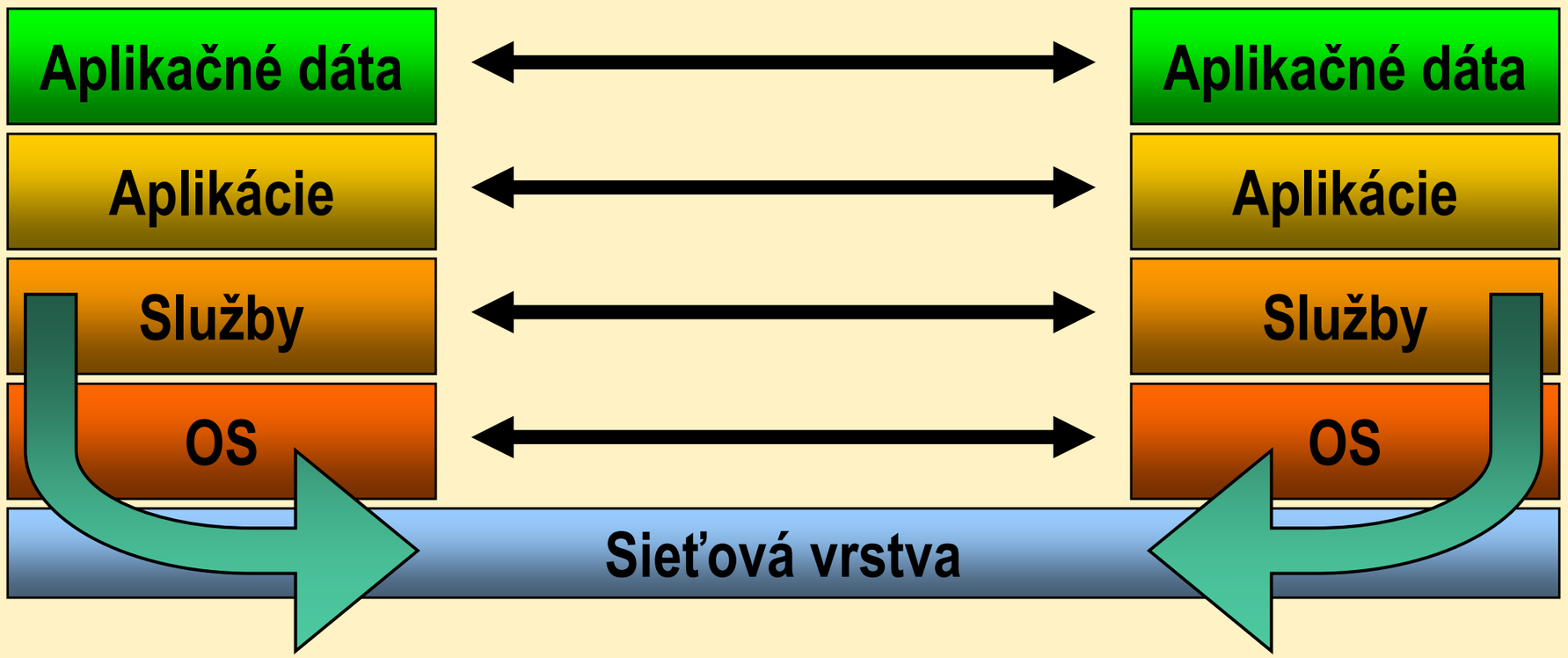
- rôznorodosť - exotické platformy, heterogénne prostredie
- vývoj neštandardnej aplikácie nevyhnutne produkuje „neštandardné“ chyby - zlyhávajú metódy automatického testovania zraniteľností
- **problém z hľadiska štandardizácie testovania zraniteľností**
- i štandardné aplikácie a platformy použité pre konštrukciu aplikácií mnohokrát obsahujú chyby !!!

Bezpečnostné špecifiká 2

DCIT

Útoky na webové aplikácie

- obvykle vedené na aplikačnej vrstve (HTTP/S)
  - mimo dosah sieťových obranných mechanizmov(IDS/IPS)
- viac-menej **potenciálne zraniteľné sú všetky vrstvy** webovej aplikácie:



CONSULTING DCIT

Bezpečnostné špecifiká 3

DCIT

Potenciálne slabé miesta

- **Klient**
  - slabiny prehliadača, útoky s podsúvaním kódu
- **Komunikácia**
  - odposluch na nechránených kanáloch (HTTP)
  - podpora slabého šifrovania
  - útoky MITM (presmerovanie šifrovanej komunikácie)
- **Webový server**
  - slabiny v software a konfigurácii servera
- **Aplikácia a aplikačné dáta**
  - útoky na autentizáciu, riadenie prístupu, validácia vstupov, chyby aplikačnej logiky
  - manipulácia s uskutočňovaním databázových dotazov

II. Problematika návrhu

Problematika návrhu 1

- Problematické miesta môžu byť v celej architektúre, vo všetkých vrstvách, platformách, technológiách, protokoloch…
- Môžu byť spôsobené chybou kódu (štandardné platformy či aplikácie), použitím nevhodných prostriedkov, nesprávnou konfiguráciou…

Dôsledok:

- tradičný prístup testovania štandardných zraniteľností nefunguje…

Zraniteľnosti nie sú „štandardné“

Problematika návrhu 2

DCIT

- abstrakcia návrhu/testovania do úrovne mimo platformy, systémových a ďalších „technických“ problémov
- zavedenie pojmov a definícií popisujúcich logické (v podstate procesné) chovanie aplikácie
  - Z chýb typu „vo formulári XY opraviť chybové hlásenie pri návratovej chybe MS SQL servera“ je potrebné abstrahovať na úroveň „nevhodné spracovanie chýb“...

Nutnosť abstrakcie do úrovne umožňujúcej štandardizáciu

CONSULTING DCIT

Problematika návrhu 3

DCIT

Abstrakcia do procesného popisu slabín

1) Unvalidated Input
2) Broken Access Control
3) Broken Autentication and Session Management
4) Cross-site scripting
5) Buffer Overflows
6) Injection Flaws
7) Improper Error Handling
8) Insecure Storage
9) Denial of Service
10) Insecure Configuration Management

Problematika návrhu 4

DCIT

Unvalidated Input 1

Nedostatočná kontrola vstupných parametrov:

- prílišná dôvera k obsahu vstupných parametrov
  - v HTTP hlavičkách požiadaviek (napr. URL adrese, parametrov POST dotazov, hodnotách cookie) atď.

Typické príklady:

- Cross Site Scripting, Buffer Overflows, SQLinjection

Vhodná ochrana:

- testovanie všetkých parametrov **na povolené** hodnoty
- iné metódy (zakázané hodnoty, úprava hodnôt) ne sú bezpečné (mnoho výnimiek)

Problematika návrhu 5

DCIT

Unvalidated Input - praktická ukážka:

Validácia dát na strane klienta

- aplikácia kontroluje dáta zadávané do prehliadača pomocou JavaScriptu v atribúte **onsubmit** tagu **<form>**
- útočník vyplní formulár a zadá do prehliadača adresu **javascript:void(document.forms[0].submit())**
- prehliadač dáta formulára odošle, bez toho, aby spustil **onsubmit** a bola uskutočnená akákoľvek kontrola dát (na klientskej strane)
- existuje mnoho ďalších ciest ako dosiahnuť rovnaký výsledok (proxy, debugger atď.)
- validácia dát na strane klienta je úplne neúčinná!

Problematika návrhu 6

DCIT

Broken Access Control

Chybná kontrola prístupu (autorizácie)

- nesprávne prideľovanie prístupových práv k jednotlivým prostriedkom (funkciám) webovej aplikácie **už autentizovaným** užívateľom
- zneužitím chyby v prístupe možno získať vyššie privilégia v rámci aplikácie (v extrémnom prípade i administrátorské, root privilégia)

Typické príklady:

- nevhodná ochrana prístupu k administračnému rozhraniu, autorizácia na klientovi, Obídenie kontrol pri login-e, Client Side Caching, Súborové oprávnenia

Problematika návrhu 7

DCIT

Broken Access Control - praktická ukážka:

Path traversal

- aplikácia pracuje s URL typu **https://banka.cz/display.asp?doc=dokument**
- z parametru **doc** je vytvorené meno súboru, ktorý bude zobrazený, napr. **c:\docs\dokument.html**
- útočník pošle **doc=../../../../inetpub/wwwroot/display.asp%00**
- aplikácia zobrazí **c:\docs\..\..\..\..\inetpub\wwwroot\display.asp\0.html** (\0 je znak s kódom 0), čiže **c:\inetpub\wwwroot\display.asp**, teda svoj zdrojový kód stránky
- analogicky možno čítať aj iné súbory

CONSULTING DCIT

Problematika návrhu 8

DCIT

Broken Autentication and Session Managment

Chybná autentizácia a riadenie relácie

- všeobecne akýkoľvek problém v autentizácii užívateľov a v správe relácií (sessions) - správa relácií je náročný problém v kombinácií s bezstavovým HTTP/HTTPS
  - nesprávny mechanizmus autentizácie (overenie identity) užívateľa a riadenie aktívnych užívateľských relácií
  - nevhodné spôsoby zmeny hesla, obsluha „zabudnutých“ hesiel, ukladanie hesiel v prehliadačoch, exspirácia účtov a pod.

Typické príklady:

- problém kvality, použitia a bezpečného uloženia hesla, implicitná dôvera medzi aplikáciami – zákaz implicitnej dôvery

Problematika návrhu 9

DCIT

Broken Autentication and Session Managment - praktická ukážka:

Timing side-channel

- proces prihlasovania nerozlišuje chybné užívateľské meno vs. správne užívateľské meno a chybné heslo
  - rôzna odozva odpovedí (v prvom prípade 200 ms vs. 500 ms) - stačí zmerať rýchlosť odozvy aplikácie

Password bruteforcing

- užívatelia – pridelené heslo (PIN) dĺžky 5 cifier (10^5), blokácia po 3 neúspešných pokusoch, počet pokusov prihlásenia na rôzne účty z PC (IP) neobmedzované
- predpoklad: PIN-y generované náhodne (uniformné rozdelenie)
  - pre ľubovoľný PIN je pravdepodobnosť > 62% nájdenia užívateľa
  - pre 2 pokusy (2 rôzne hodnoty PIN-u) pravdepodobnosť > 86 %

Problematika návrhu 10

DCIT

Cross-site Scripting (XSS)

- **podsunutie vlastného kódu** (napr. JavaScript) do dynamicky generovanej HTML stránky (pomocou parametrov stránky, vstupov do formulárov a pod.)
  - z hľadiska servera ide o vloženie nedôveryhodných dát do generovaných výstupov

Typické príklady:

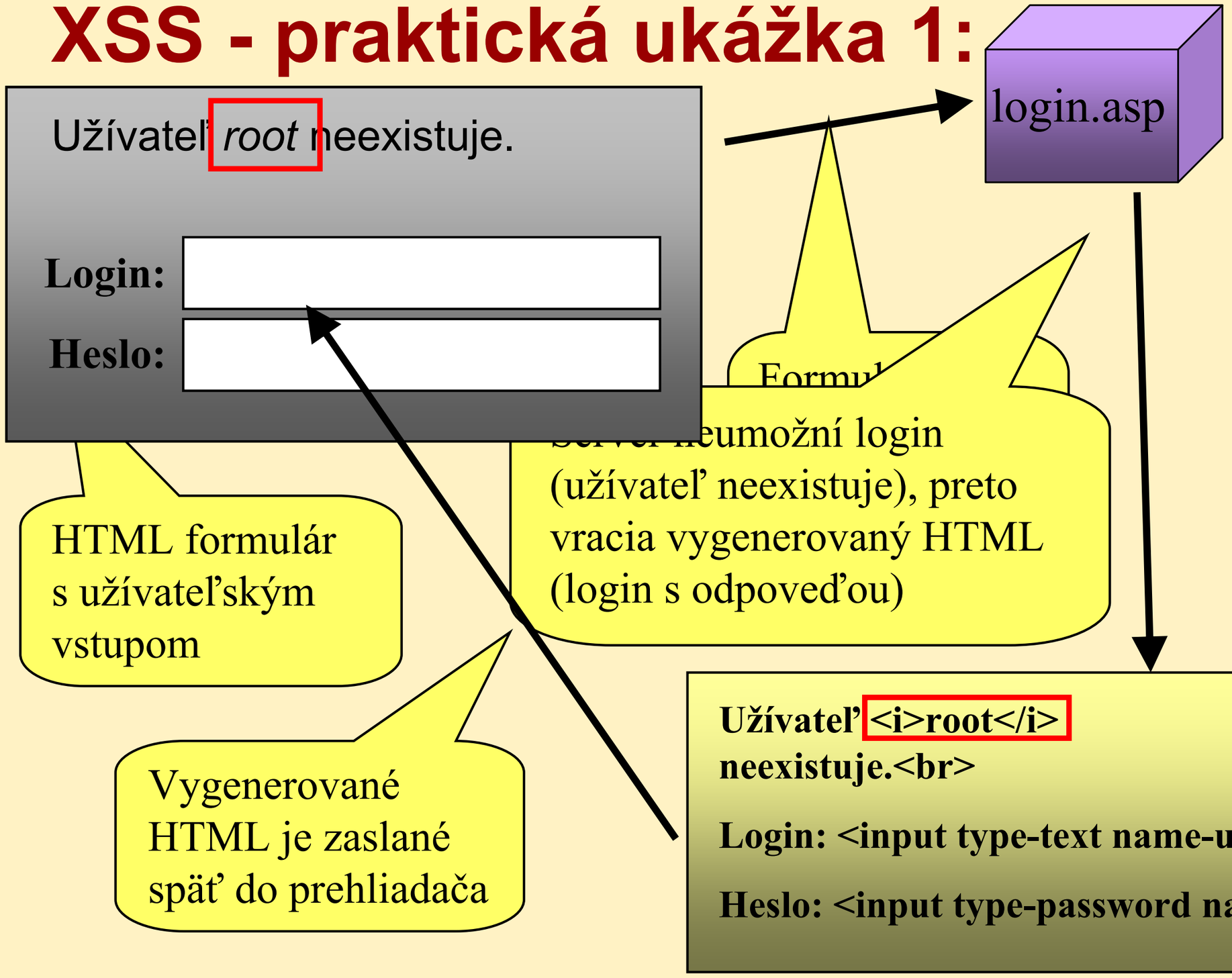
- zmena obsahu, odposluch hesla, práca pod cudzou identitou
- **XSS - útok proti klientskej časti (užívateľovi)**

Vhodná ochrana:

- dôsledné testovanie obsahu (hlavičiek, cookies, polí, skrytých polí) na povolené hodnoty - „čo nie je povolené, je zakázané“

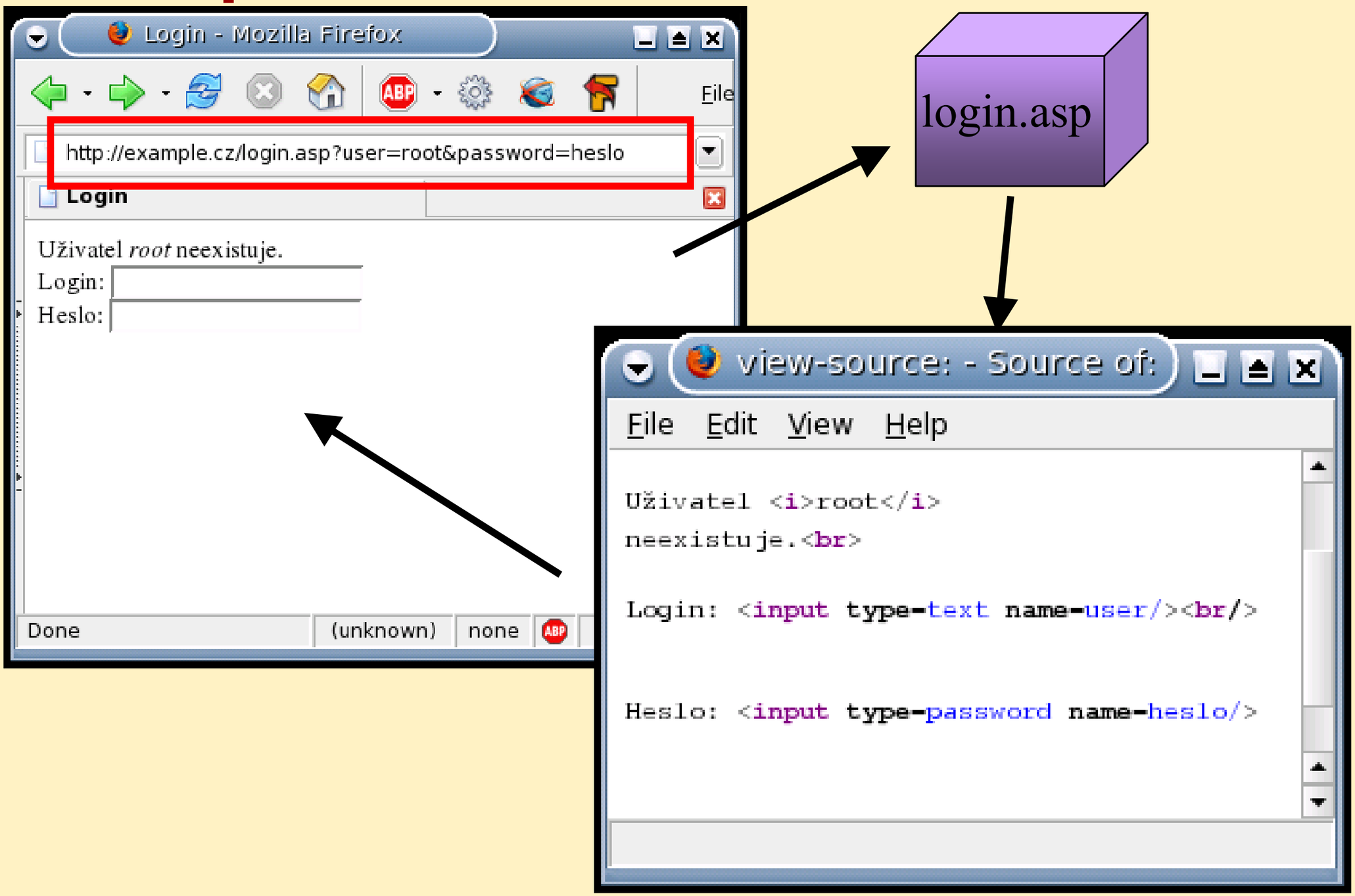
Problematika návrhu 11

XSS - praktická ukážka 1:



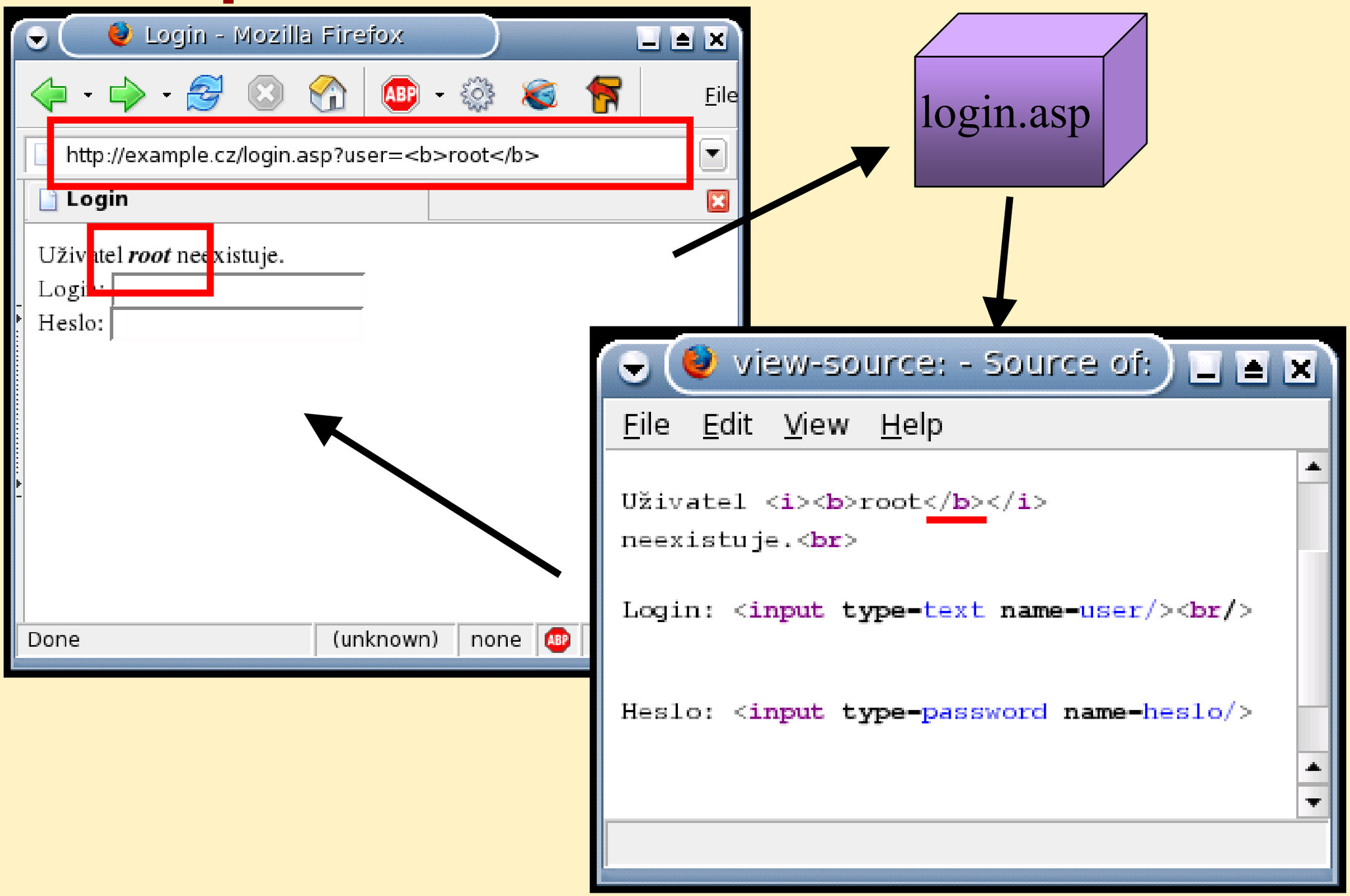
Problematika návrhu 12

XSS - praktická ukážka 2:



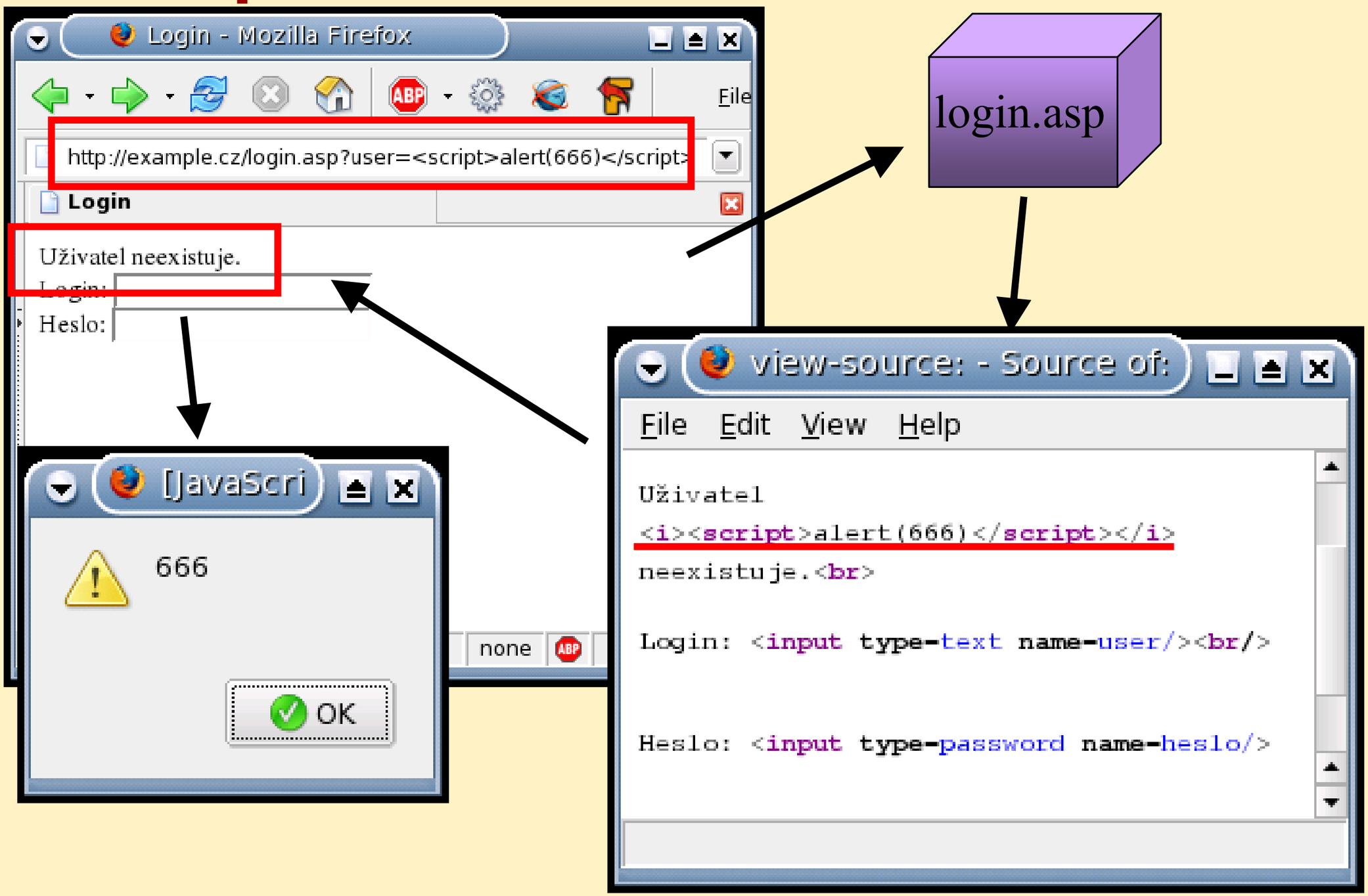
Problematika návrhu 13

XSS - praktická ukážka 3:



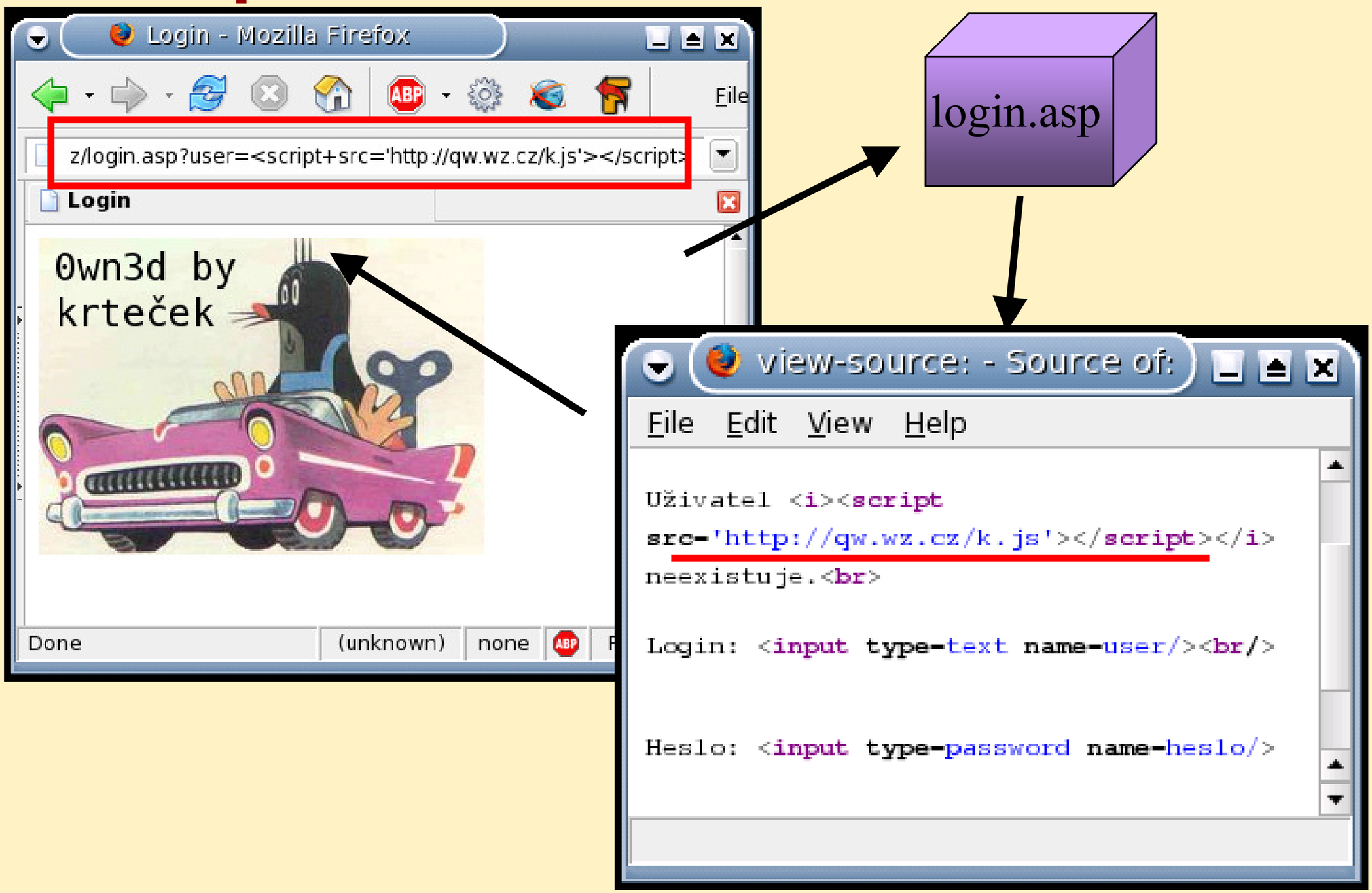
Problematika návrhu 14

XSS - praktická ukážka 4:



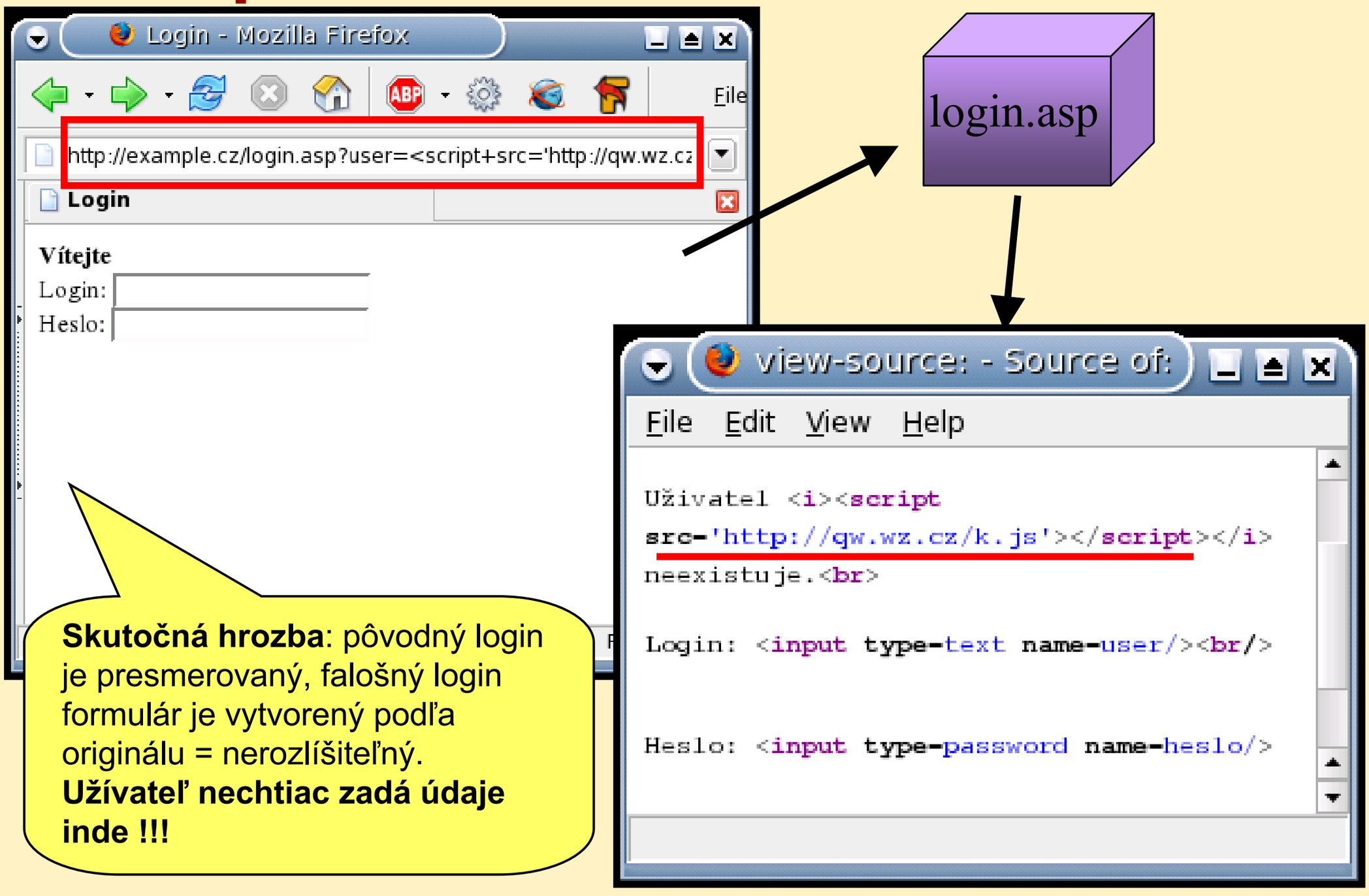
Problematika návrhu 15

XSS - praktická ukážka 5:



Problematika návrhu 16

XSS - praktická ukážka 6:



Problematika návrhu 17

DCIT

XSS - praktická ukážka 7:

„obfuscated“ XSS

- aplikácia doslova opisuje svoje vstupy do dokumentu, ale bráni sa výskytu reťazca script a rada podobných textov bežne využívaných pre XSS (pokiaľ niečo z toho vo vstupe nájde, hlási chybu)
- útočník nájde miesto, kde aplikácia vkladá vstup do atribútu a ako hodnotu, ktorá má byť vložená, predloží svojej obeti text **"+style="width:\65xpression(\x64ocument.\x77rite(...));**
- filter nenájde na texte nič podozrivého, avšak prehliadač obete (ak to je to MSIE) tam vidí príkaz **document.write(...)**, ktorý tiež vykoná

Problematika návrhu 18

DCIT

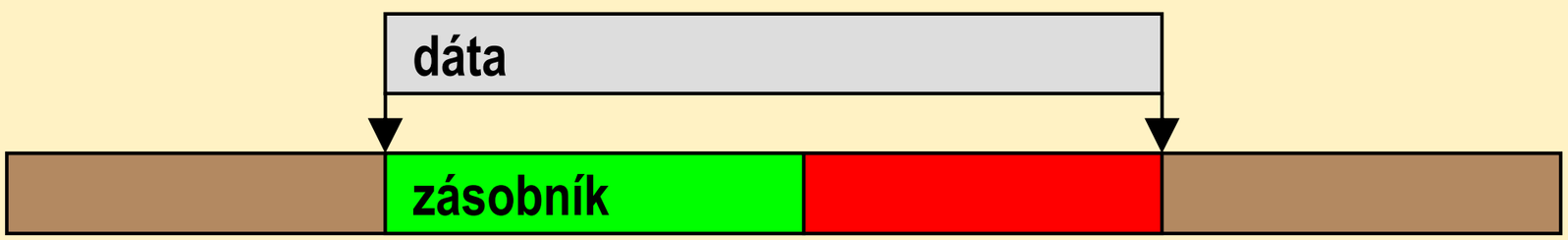
Buffer Overflows

Pretečenie zásobníka

- nevhodná kontrola veľkosti vstupných dát + pokus o ich uloženie do nedostatočne veľkého zásobníka ➔ BO (historicky najrozšírenejší SW problém)
  - klasický problém nižších programovacích jazykov (mienený najmä C), v samotných webových aplikáciách zriedka, ALE často v podporných SW komponentoch: HTTP servery, OS, špecializované knižnice (a tiež prehliadačoch!)

Vhodná ochrana:

- pravidelná aktualizácia webových a aplikačných serverov a ďalších prvkov



CONSULTING DCIT

Problematika návrhu 19

DCIT

Buffer Overflows - praktická ukážka:

Stack buffer overflow

- v aplikácii je kód tohto tvaru **auto char buf[100];**
  strcpy(buf, vstup);
- útočník predloží text (primerane) dlhší ako 100 znakov, aplikácia sa pokúsi text vložiť do poľa **buf** a začne pri tom modifikovať pamäť za jeho koncom
- **buf** je ako lokálna premenná v zásobníku a nad ním sa nachádza i návratová adresa, ktorá bude prepísaná údajom zo vstupu
- pri návrate z aktuálne uskutočňovanej funkcie sa použije upravená návrat.adresa a začne sa vykonávať kód útočníkom určený alebo dokonca preložený (ak ukazuje adresa na nejaké miesto, kde sú uložené ďalšie dáta od útočníka)

CONSULTING DCIT

Problematika návrhu 20

DCIT

Injection Flaws

Injektovanie kódu

- vloženie vykonávaného kódu do aplikácie/komponentu pomocou vstupných parametrov.
  - Najznámejší je SQL injection - manipulácia s SQL požiadavkou: zmena/odcudzenie dát z databázy, ovládnutie celého DB systému, prípadne celého OS. Mimo SQL možno manipulovať i ASP,PHP a ďalšie. Špeciálnym prípadom injekcie kódu je i XSS.

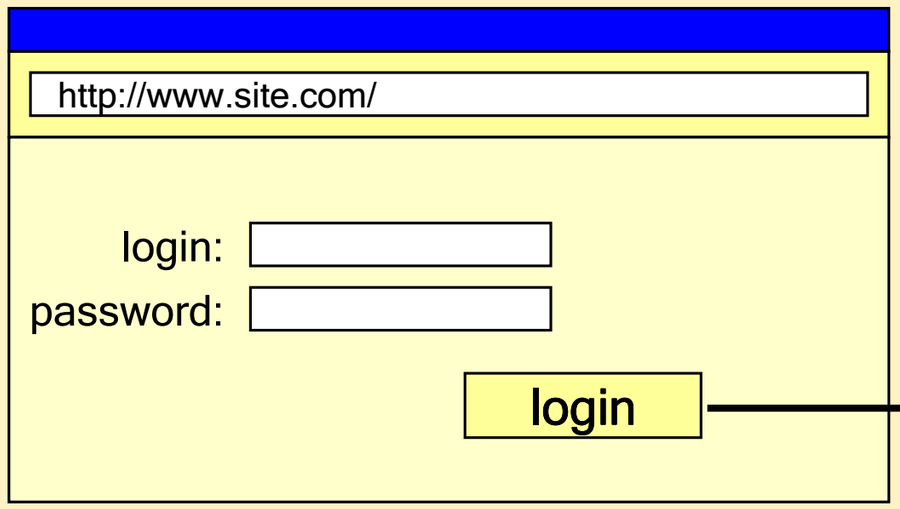
Vhodná ochrana:

- minimalizácia využitia externých interpreterov, uložené procedúry/príkazy, prevádzka aplikácie s min. privilégiami
- mechanizmus riešenia chýb/výnimiek/timeoutov
- dokumentácia testov, návratových kódov

Problematika návrhu 21

Injection Flaws - praktická ukážka:

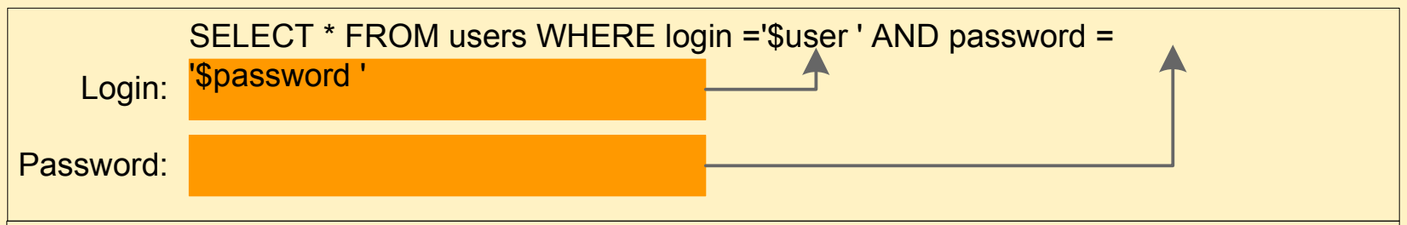
Klientska strana – browser

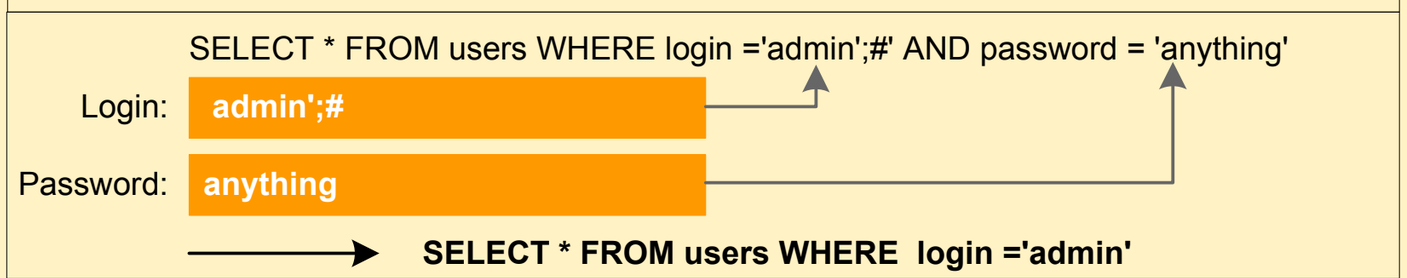


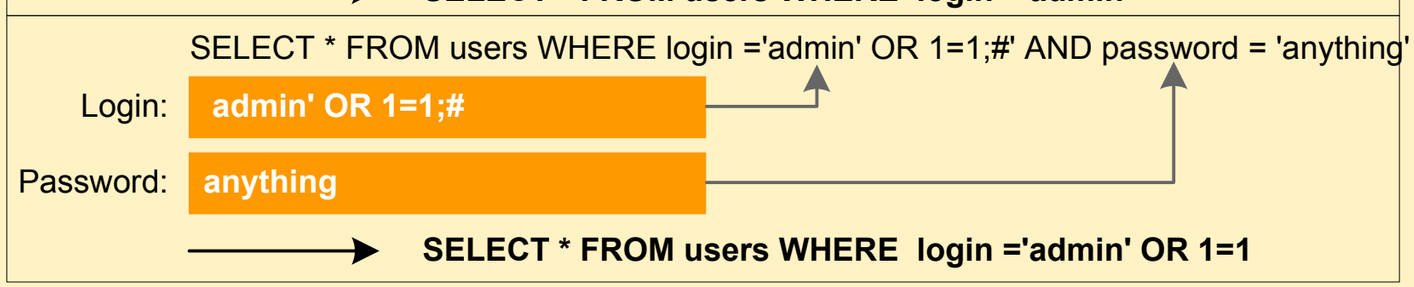
login.asp – na serveri

```
Query = „SELECT * FROM users
WHERE login ='$user' AND
password = '$password'"

If QueryResult(Query) = "" Then
  Authenticated = 0
Else
  Authenticated = 1
End If
```







Problematika návrhu 22

DCIT

Improper Error Handling

Nezvládnutie chybovej operácie

- akákoľvek forma problému v spracovaní chybových stavov

Typické príklady:

- detailné chybové výpisy (typy, verzie komponentov atď.), detailné info o aplikácii ➔ zefektívnenie útoku

Vhodná ochrana:

- reakcie na akékoľvek možné chyby **vopred definovanou informáciou** o chybe
- interná log. chyb (**detekcia** implementačných slabín, pokusov o prienik – opakovaný neúspešný pokus = poplašná udalosť)
- Pozn.: Dokumentácia politiky spracovania chýb je neoddeliteľnou súčasťou návrhu aplikácie ☺

Problematika návrhu 23

DCIT

Improper Error Handling – praktic. ukážka:

Príliš detailné chybové hlásenia

- aplikácia príjme požiadavku
  https://banka.cz/hledani.asp?dotaz=abc%27xyz
- a odpovie **Microsoft OLE DB Provider for SQL Server error '80040e14'**
  Line 1: Incorrect syntax near 'xyz'.
    - útočník hneď vidí slabinu typu SQL injection

chybové hlásenia ako postranný kanál

- útočník dokáže vložiť kód, ale nedokáže priamo čítať odpovede
- Použije prenos cez chybové hlásenia:
    - ako súčasť chybového hlásenia (viď **'xyz'** vyššie),
    - v parametroch hlásení (druh chyby, miesto výskytu chyby),
    - v samotnom fakte, či chyba nastala alebo nie

CONSULTING DCIT

Problematika návrhu 24

DCIT

Insecure Storage

Nedostatočná ochrana (citlivých) dát

- aplikácie často pracujú s heslami, kľúčmi… Ich nevhodná ochrana v prípade kompromitácie výrazne zhoršuje následky útoku

Typické príklady:

- nevhodné šifry, chyby v implementácii, nezabezpečené ukladania kľúčov/cert./hesiel, pseudonáhodné generátory, snaha o „vývoj“ proprietárnych šifrovacích algoritmov

Možná ochrana:

- Obmedzené, kontrolované využitie kryptograf. prostriedkov
  - neprenášať problém ochrany hesla („hashovanie“ namiesto ochrany kľúča šifry), „rozumné“ hashovacie funkcie
  - vždy využívať verejne dostupnú/preverenú/využívanú kryptografickú knižnicu (žiadna ľudová tvorivosť)
  - hlavný kľúč by mal byť uchovávaný (pokiaľ vôbec) po častiach

Problematika návrhu 25

DCIT

Insecure Storage – praktická ukážka:

Šifrované parametre

- aplikácia chráni parametre generovaných URL pred neoprávnenou modifikáciou tým, že ju šifruje, napr. miesto **par=123** vidí užívateľ **Encrypted=CkpLRUpMRkpFTEtGSkVMRktKTEVKRmxrc2psZnNqbGZranNs**
- útočník zistí, že nejaká funkcia aplikácie akceptuje parameter **par** v otvorenom tvare (napr. ako súčasť dát zadávaných do formulára) a odovzdáva ho do nadväzujúcej funkcie už v šifrovanom tvare
- útočník pomocou zmienenej funkcie získa zašifrovaný tvar ním požadovanej hodnoty parametru a nahradí s ním pôvodne šifrovaný parameter v URL, na ktorú chce uskutočniť útok
- aplikácia predpokladá, že šifrovanie ochránilo parameter pred modifikáciou, a akceptuje jeho hodnotu bez ďalšej kontroly

Problematika návrhu 26

DCIT

Denial of Service (DOS)

Odopretie (zneprístupnenie) služby

- Vyradenie aplikácie z prevádzky za použitia ich vlastných funkcií
  - typicky zahltenie náročnými operáciami
  - DoS – problém vďaka netriviálnemu rozpoznaniu legitímnej požiadavky od zahlcovania serveru
- Veľké množstvo (HTTP) požiadaviek ➔ vyradenie komponentu ➔ ohrozenie dostupnosti celej aplikácie
  - útoky nemusia byť založené na hrubej sile (slabé miesta v logike aplikácie - napr. blokácie účtov opakovaním neúspešných prihlásení)

Možná ochrana:

- obmedziť dostupné systémové prostriedky pre 1 užívateľa
- kvóta max. zaťaženia požiadavky (napr. 1 požiadavka na užívateľa v danom čase)
- neautentizovaný užívateľ - obmedziť prístup k prostriedkom (napr. zákazom výkonovo náročných operácií)

Problematika návrhu 27

DCIT

Denial of Service – praktická ukážka:

Zamykanie účtov

- aplikácia má ľahko predikovateľné mená účtov
    - každý účet je zablokovaný po niekoľko neúspešných pokusoch o prihlásenie
    - útočník môže prejsť veľké množstvo účtov a úmyselne ich zablokovať

Bombardovanie SMS správami

- aplikácia umožňuje autentizáciu užívateľa s použitím SMS
    - užívateľské účty sú často identifikované predikovateľným číslom (napr. telefón užívateľa), a taktiež možno metódou pokus-omyl nájsť mnoho platných mien
- útočník prinúti aplikáciu, aby bombardovala mobily svojich užívateľov, čo nielen obťažuje užívateľa, ale tiež spôsobuje priame náklady prevádzkovateľovi aplikácie

Problematika návrhu 28

DCIT

Insecure Configuration Management

Nevhodná konfigurácia servera

- konfiguračné parametre často v **default nastaveniach**
  - nadbytočné (k prevádzke nepotrebnej) služby
  - demo/testovacie súbory, scripty (ladiace funkcie, info o verziách)
  - nadbytočné chybové hlásenia
  - prevádzka pod zbytočne vysokým oprávneniami
  - slabé heslá pre admin. prístup do aplikácie („zabudnuté“ aktívne účty z doby inštalácie, vývoja a testovania)
- **Nezáplatované** bezpečnostné diery na serveri
- Nesprávne použitie SSL, certifikátov
  - verzie, dôveryhodnosť, exspirácia

Možná ochrana:

- pravidelné záplatovanie, test zraniteľností, definovaný jednotný konfiguračný štandard

Problematika návrhu 29

DCIT

Insecure Config. Man. – praktická ukážka:

Nežiaduce sprístupnenie administrátorskej funkcie

- z Internetu je voľne prístupný testovací server **test.banka.cz**
  - pod **https://test.banka.cz/soap** prístupné rozhranie vrátane administrátorských funkcií, využiteľných k inštalácií vlastného komponentu, majúci cez std.javovské triedy prístup k FS, sieti atď.

Podpora slabých šifier a SSLv2

- server akceptuje slabé (exportné) šifry a SSLv2
- klienti sú obvykle tiež konfigurovaní tak, aby ich podporovali
  - útočník sa pokúsi presvedčiť klienta i server, aby použili protokol SSLv2 (rollback attack) a slabú šifru
  - následne útočník rozlúšti 40-bitový „exportný“ kľúč a dešifruje komunikáciu, prípadne do nej aktívne vstúpi

III. Testovanie zraniteľností

Testovanie zraniteľností 1

DCIT

Rekapitulácia poznatkov

- **webová aplikácia**
  - heterogénny, distribuovaný systém
  - komponenty nemajú vzájomnú dôveru
  - niektoré časti nemožno kontrolovať vôbec
  - komunikácia cez rozľahlé, nedôveryhodné prostredie
- **problémové miesta (zraniteľnosti) aplikácií**
  - všetky vrstvy (sieť, OS, služby, aplikácie, dáta)
  - slabiny možno kategorizovať (10 oblastí)
  - oblasti sa prekrývajú, reálne útoky kombinujú zraniteľnosti

Možno teda vôbec vytvoriť bezpečnosť?

Testovanie zraniteľností 2

DCIT

1. Koncepčný návrh aplikácie

- so zreteľom na známe zraniteľnosti (oblasti)
- s predpokladom modifikácie bezpečnostných funkcionalít

2. Bezpečnostné testovanie

- invazívne testy pred nasadením nových verzií
- záťažové testy
- pravidelne opakované prevádzkové testy zraniteľností

3. Dôsledná konfigurácia

- prevádzková konfigurácia všetkých komponentov
- jednotný konfiguračný štandard
- priebežná údržba prevádzkových systémov (záplaty, aktualizácia)

Testovanie zraniteľností 3

DCIT

1. Koncepčný návrh aplikácie

- Akceptovanie aktuálnych poznatkov oboru (zoznámenie s okruhom zraniteľností, viď 10 oblastí)
  - **Prekvapivé množstvo i renomovaných vývojárskych firiem nemá o zaistení bezpečnosti aplikácií ani potuchy !!!**
- Štandardizácia kľúčových bezpečnostných prvkov
  - PKI, certifikáty, práca s heslami, šifrovacie algoritmy atď.
- Vývoj je dynamický, je potrebné pripraviť sa na možnosť zásahu do kódu
  - zmeny bezpečnostných funkcií – hashovanie, šifrovanie, doplnenie kontrol…
    - **Zásady štruktúrovaného (objektového) návrhu platí i tu**

CONSULTING DCIT

Testovanie zraniteľností 4

DCIT

2. Bezpečnostné testovanie

=Test chovania a odolnosti aplikácie

- pri vysokom zaťažení
- pri užívateľskej chybe
- pri úmyselnej neoprávnenej manipulácii

Metódy testovania

- Neinvazívne
  - **audit** zdrojových kódov, konfigurácií („whitebox“)
- Invazívne
  - **penetračné** testy („blackbox“ **hacking**)

CONSULTING DCIT

Testovanie zraniteľností 5

DCIT

3. Dôsledná konfigurácia

Konfiguračný štandard

- Definícia požiadaviek na infraštruktúru
  - prevádzkové parametre - nastavenie serverov, databáz, filtrácií

- Možno špecifikovať aj požiadavky na odolnosť aplikácií
  - voči špecifikovaným typom útokov (viď „10 oblastí“)
  - testovacie „minimum“, zmluvné zabezpečenie (reklamácie)
  - viď ukážka možnej špecifikácie vybraných útokov

CONSULTING DCIT

Testovanie zraniteľností 6

Špecifikácia útokov (konfiguračný štandard)

Typ útoku	Popis
SQL injection	nedůsledné ošetření vstupů v některých případech umožňuje útočníkovi modifikovat SQL dotazy spouštěné v databázi, což může vést k neoprávněnému získání/modifikaci dat nebo dokonce ke spuštění vlastního kódu na databázovém serveru
Cross site scripting (XSS)	WWW server trpící touto slabinou může být zneužit jako prostředník při útoku spočívajícím ve spuštění kódu (obvykle Javascript) v prohlížeči uživatele-oběti, který považuje zranitelný server za důvěryhodný
URL tampering	manipulace se strukturou URL a/nebo parametry předávanými v rámci URL může u zranitelné aplikace vést k provedení „neočekávaných“ akcí
Hidden field manipulation	útok na často citlivá data předávaná v rámci HTML formulářů v tzv. HIDDEN polích (nejsou viditelná pro běžného uživatele)
HTTP response splitting attack	specifický útok, při kterém je útočník schopen nežádoucím způsobem rozdělit HTTP hlavičku a vytvořit „falešnou“ odpověď pocházející ze zranitelného serveru
Cross site tracing (XST)	možnost zneužití metody TRACE k získání citlivých informací z http hlaviček (session cookies, autentizační údaje)

Testovanie zraniteľností 7

Štandardizácia v odbore

- Nie sú oficiálne štandardy, ale:
  - Metodika penetračného testovania (nie len webové aplikáce) **- OSSTMM Open Source Security Testing Methodology Manual, [www.osstmm.org]**
  - Projekt OWASP, zaoberajúci sa vývojom metodík tvorby bezpečných aplikácií **- The Open Web Application Security**, **[www.owasp.org]**
    - **Zaujímavosť: využíva mapovanie na štruktúru opatrení COBiT**

Zmieňovaných „10 oblastí“ použitých i v OWASP „TopTen Most Critical Vulnerabilities“

Testovanie zraniteľností 8

DCIT

- **Audit kódu**
  - priechod kódom je pomalé = drahé riešenie
  - žiaľ, nemožno preveriť štandardné platformy (a ich chyby)
- **Invazívne testy**
  - simulácia reálneho útoku
  - prieskum chovania interagujúcich komponentov ako celku (efektívnejšie riešenie)
  - možno preveriť nielen úroveň zabezpečenia ale i reakcie obranných mechanizmov

CONSULTING DCIT

Poznámky k testovaniu

- **Penetračné testy webových aplikácií sú špecifické**
  - nemožno použiť automatické testovanie
    - unikátne chyby – nie sú v databázach zraniteľností
  - závisí na skúsenostiach testovacieho tímu, nie na použitých nástrojoch
    - prakticky všetky potrebné testovacie nástroje sú dostupné ako OpenSource

Záver

Webové aplikácie (IB nevynímajúc):

- komplex platforiem, štandardného a užívateľského kódu, prepojený cez nedôveryhodné prostredie
- na konci reťazca je užívateľ (často „zlý“)

Unikátnosť riešenia vyžaduje:

- koncepčný návrh a implementáciu so zapracovaním bezpečnostných požiadaviek (dodržovanie konfiguračných štandardov)
- priebežné invazívne, manuálne testovanie

= Kľúč k tvorbe bezpečných webových aplikácií

Ďakujem za pozornosť