Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Webové stránky a jejich přístupnost, testování, bezpečnostní rizika

Michal Čada

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačních technologií

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Čada Michal

Informatika

Název práce

Webové stránky a jejich přístupnost, testování, bezpečnostní rizika

Anglický název

Web sites and their accessibility, testing, security risks

Cíle práce

Diplomová práce je tematicky zaměřena na problematiku přístupnosti www stránek, jejich testování a bezpečnostní rizika. Hlavním účelem práce je analyzovat současný stav v jednotlivých zmiňovaných kategoriích.

Dílčí cíle diplomové práce jsou:
- vytvořit přehled řešené problematiky
- analyzovat vybrané aspekty jednotlivých kategorií
- navrhnout řešení popisovaných problémů

Metodika

Metodika řešení problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Praktická část je zaměřena na vypracování případové studie analyzující vybrané aspekty přístupnosti, testování a bezpečnostních rizik www stránek a navržení jejich řešení. Na základě syntézy teoretických poznatků a výsledků praktické části práce budou formulovány závěry diplomové práce.

Harmonogram zpracování

1) Příprava a studium odborných informačních zdrojů, upřesnění dílčích cílů práce a volba postupu řešení: 6/2011
2) Zpracování přehledu řešené problematiky dle informačních zdrojů: 7/2011 – 9/2011
3) Vypracování analytické části práce, diskuze a zhodnocení výsledků. 10/2011 – 11/2012
4) Tvorba finálního dokumentu diplomové práce: 12/2011 – 2/2012
5) Odevzdání diplomové práce a teze: 3/2012

Rozsah textové části
60 - 80 stran

Klíčová slova
www stránky, programování, přístupnost, testování, bezpečnostní rizika

Doporučené zdroje informací

1. CEDERHOLM, Dan. Webdesign s webovými standardy. 1. vydání. Brno: Zoner press, 2004. 254 s. ISBN: 80-86815-15-32
2. HUSEBY, Sverre H. Zranitelný kód. Computer Press, 2006. 208 s. ISBN: 80-251-1180-6
3. VRÁNA, Jakub. PHP triky. [online]. 2005. Dostupný z www: <http://php.vrana.cz/>
4. Interval.cz. [online]. Dostupný z www: <http://interval.cz/>
5. KOSEK, Jiří. Vše o WWW. [online]. 1999. Dostupný z www: <http://www.kosek.cz/>

Vedoucí práce
Brechlerová Dagmar, RNDr., Ph.D.

Termín odevzdání
březen 2012

doc. Ing. Zdeněk Havlíček, CSc.
Vedoucí katedry

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE
Provozně ekonomická fakulta

prof. Ing. Jan Hron, DrSc., dr.h.c.
Děkan fakulty

V Praze dne 21.11.2011

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Webové stránky a jejich přístupnost, testování, bezpečnostní rizika" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 4.4.2012 Michal Čada

Poděkování

Rád bych touto cestou poděkoval Rndr. Dagmar Brechlerové, Ph.D., za konzultace a vhodná doporučení v průběhu psaní této diplomové práce.

Webové stránky a jejich přístupnost, testování, bezpečnostní rizika

---

Web sites and their accessibility, testing, security risks

Souhrn

Webové aplikace jsou v dnešní době součástí prezentace téměř každé firmy, subjektů státní správy, a v neposlední řadě i osob a spolků z veřejného či soukromého života. Mnoho těchto www stránek však opomíjí pravidla přístupnosti webových aplikací a činí tak obsah nepoužitelný pro mnoho hendikepovaných uživatelů. Kromě přístupnosti je často zanedbáno i zabezpečení kódu proti útokům na webové aplikace. Firmy tak mohou vystavit citlivá data nebezpečí, aniž by si toho byly vědomi. Pro dodržení pravidel přístupnosti a dostatečnému zabezpečení zdrojového kódu je potřeba aplikaci testovat v průběhu celého jejího vývoje. Tato diplomová práce se zabývá tvorbou přístupných a bezpečných www aplikací a způsoby jejich testování.

Summary

Web applications are part of the presentation of almost every company, government organizations, and also the persons and associations of public or private life. Many of these websites, however, ignore the rules of accessibility of web applications and make the content unusable for many of handicapped users. Beyond accessibility is often neglected also code security against attacks on web applications. Companies can expose sensitive data risk, without being aware of it. To comply with the rules of accessibility and adequate security of source code is needed to test the application throughout its development. This diploma thesis deals with the creation of accessible and secure web applications and methods of testing.

Klíčová slova: www stránky, programování, přístupnost, testování, bezpečnostní rizika

Keywords: www sites, programming, accessibility, testing, security risks

Obsah

1 ÚVOD

Tématem diplomové práce jsou webové aplikace a jejich přístupnost, testování a bezpečnostní rizika. Všechny tři oblasti jsou spojeny se samotným vývojem webové aplikace. Přístupnost aplikace, tedy použitelnost pro hendikepované uživatele, spočívá v dodržení pravidel dle určené metodiky při psaní zdrojového kódu. Obdobným způsobem, tedy dodržením pravidel, respektive ošetřením zdrojového kódu proti známým útokům, probíhá i snížení bezpečnostních rizik spojených s webovými aplikacemi. V průběhu celého vývoje pak probíhá testování aplikace z mnoha hledisek, týkajících se přímo zdrojového kódu (validita, zabezpečení, rychlost, přístupnost) nebo použitelnosti aplikace.

V této práci je nejprve proveden teoretický přehled řešené problematiky. U přístupnosti jsou popsány hendikepy a z nich plynoucí omezení týkající se prohlížení webových aplikací, a metodiky obsahující doporučená pravidla k tvorbě přístupného webu. U testování stránek jsou popsány nejběžnější metody testování a testované faktory. Teoretický přehled bezpečnostních rizik zahrnuje seznámení s hrozbami spojenými s útoky na webové aplikace a popis nejpoužívanějších typů útoků.

Stěžejní částí práce je analýza vybraných aspektů jednotlivých kategorií a jejich praktické využití. V oblasti přístupnosti je provedena analýza grafiky, ovládacích prvků, textu a struktury kódu tabulek a formulářů. V oblasti bezpečnosti webových aplikací zanalyzuji rizika spojená s vybranými nejčastějšími typy útoků a způsoby ochrany, tedy ošetření kódu. V části zaměřené na testování www aplikací zanalyzuji vybrané faktory a způsob jejich testování.

Výsledky analýzi jsou aplikovány na webové stránky tréninkové skupiny Milana Kováře atletického oddílu TJ Dukla Praha. Jsou prezentovány části kódu popisující řešení problémů přístupnosti. Dále jsou provedeny útoky na nezabezpečenou aplikaci, její následné zabezpečení a ověření funkčnosti. Nakonec jsou stránky otestovány z hlediska přístupnosti, datové velikosti a rychlosti nahrávání.

Závěrem práce je souhrn doporučení při tvorbě webové aplikace.

2 CÍL PRÁCE A METODIKA

Diplomová práce je tematicky zaměřen na problematiku přístupnosti www stránek, jejich testování a bezpečnostní rizika. Hlavním účelem práce je analyzovat současný stav v jednotlivých zmiňovaných kategoriích.

Dílčí cíle diplomové práce jsou:

- Vytvořit přehled řešené problematiky
- Analyzovat vybrané aspekty jednotlivých kategorií
- Navrhnout řešení popisovaných problémů

Metodika řešení problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Praktická část je zaměřena na vypracování případové studie analyzující vybrané aspekty přístupnosti, testování a bezpečnostních rizik www stránek a navržení jejich řešení.
Na základě syntézy teoretických poznatků a výsledků praktické části práce jsou formulovány závěry diplomové práce.

3 PŘEHLED ŘEŠENÉ PROBLEMATIKY

Webové aplikace jsou v dnešní době součástí prezentace téměř každé firmy, subjektů státní správy, a v neposlední řadě i osob a spolků z veřejného či soukromého života. Všeobecně fenomén internetu již dávno pronikl do všech sfér společnosti a našel si cestu do většiny domácností a tím k milionům uživatelů. Spolu s touto uživatelskou základnou roste i procento uživatelů s určitým hendikepem, znemožňujícím plné a pohodlné využití všech služeb poskytovaných webovou aplikací. Přesto mnoho programátorů www stránek na tuto nezanedbatelnou část uživatelů zapomíná, ať již nevědomě, či úmyslně, a vytváří aplikace vizuálně působivé, plné informací a po všech stránkách líbivé, ale pro hendikepované (a nejen pro ně) z mnoha důvodů nepřístupné.

Paradoxně to odporuje samotné myšlence internetových prezentací. Zakladatel technologie WWW a ředitel konsorcia W3C, Tim-Barnes Lee, sám pronesl větu:

„Síla webu je v jeho univerzalitě. Přístup pro každého nezávisle na schopnostech je jeho základní prvek.“[1]

Kdyby se všichni tvůrci webových stránek drželi této základní a jednoduché myšlenky, nebyl by internet plný stránek kladoucích mnoha uživatelům spoustu nepřekonatelných bariér. Teoreticky by každý programátor mohl těmto omezením předejít při důsledném testování výsledné aplikace, či lépe již v průběhu její tvorby. Důležité je ovšem testování v celém jeho dostupném spektru. Nikoliv jen validitity a optimalizace pro nejrozšířenější prohlížeče, případně marketingovými odděleními tolik omýlané optimalizace pro vyhledávače (SEO), ale i testování přístupnosti, použitelnosti, funkčnosti atd.

Nepřímo s předchozím souvisí i bezpečnost www aplikací. Tvůrce webu by měl znát hlavní bezpečnostní rizika prezentování na internetu, spojená převážně s odcizením dat či jinými typy útoků. Krom toho může jít o útoky z hlediska bezpečnosti dat neškodné, ale mohou zamezit uživatelům pohodlný přístup k informacím. Bezpečnost webových aplikací je poměrně specifické odvětví, jelikož klasické bezpečnostní prvky, jako firewall, sice

[1] ŠPINAR, D., Tvoříme přístupné webové stránky, s. 11

ochrání od případných útoků servery, ale nikoliv samotné www aplikace. Proto je potřeba znát alespoň základní typy útoků a s nimi spojenou ochranu zdrojového kódu.

3.1 Přístupnost www aplikací

Každá firma či společnost, jejichž www prezentace je spojena se ziskem, ví, že každý zákazník (návštěvník) je důležitý. Přístupnost jejich prezentace by tak měla být jedním z klíčových aspektů. Bohužel tomu tak mnohdy není. Zadavatel obvykle řeší přístupnost na úrovni kompatibility s prohlížeči, zařízeními a podobně. Na základní aspekt přístupnosti, tedy zpřístupnění stránek hendikepovaným uživatelům, ale již nemyslí.

Nejprve obecně ke slovu **přístupnost** (v angličtině *accessibility).* Pod pojmem přístupnost můžeme chápat takový stav, kdy daná věc neklade svým uživatelům při používání žádné zásadní překážky. Přístupnou budovu mohou tedy například používat vozíčkáři a přístupný web zase například slabozrací. Synonymem pro přístupnost může být slovo **bezbariérovost.**[2]

Dle W3C přístupnost internetu znamená, že lidé s postižením mohou používat web. Přesněji řečeno, webová přístupnost znamená, že lidé s postižením mohou vnímat, pochopit, ovládat a pracovat s webem, a mohou na web připívat.[3]

Přístupnost webových stránek je tedy využití principu přístupnosti při technickém řešení funkčnosti www stránek. Mnoha studiemi bylo zjištěno, že hendikepovaných uživatelů internetu je až třetina. Je si třeba ujasnit, že pojmem hendikepovaný není v tomto případě myšlen jen uživatel se zdravotním postižením, ale i uživatel, který není schopen z různých důvodů správně vnímat obsah webu či po technické stránce ovládat jeho funkčnost. Patří sem tedy i uživatelé alternativních technických nebo softwarových vybavení, méně zkušení uživatelé, apod.

Uvedu zde několik základních skupin takto „hendikepovaných“ uživatelů.

[2] ŠPINAR, D., Tvoříme přístupné webové stránky, s. 11
[3] W3C Web Accessibility Initiative: Introduction to Web Accesibility [online].
< http://www.w3.org/WAI/intro/accessibility.php>

1. **Zdravotně postižení**
   Do této kategorie spadají osoby s trvalou či dočasnou nemožností používat horní končetiny, nevidomí, barvoslepí, slabozrací, apod.

2. **Méně zkušení uživatelé internetu**
   Patří sem uživatelé, které nemají zažité mechanismy ovládání www stránek, typickým příkladem může být senior, ale i osoby se sníženou schopností učení jako dyslektici.

3. **Technické vybavení**
   Někteří uživatelé nechtějí nebo nemohou využívat myš, někteří mají černobílý monitor, nebo monitor s malým rozlišením. Patří sem i uživatelé mobilních a alternativních zařízení.

4. **Softwarové vybavení**
   Uživatelé jiných operačních systému či internetových prohlížečů.

5. **Vyhledávací roboti**
   V této kategorii je jediným „uživatelem“ vyhledávací robot indexující obsah pro vyhledávače. Tento robot je zároveň „nejhendikepovanějším uživatelem“.

Všechny tyto kategorie, respektive uživatelé spadající do nich, by měly být zájmovou skupinou všech webových aplikací, protože, jak již bylo řečeno, potencionálně přicházíme až o 30% návštěvníků (zákazníků). Výhodou dodržování zásad přístupnosti je i s tím spojená použitelnost. Pokud výsledná aplikace bude plně přístupná, je tím zvýšena i použitelnost pro všechny uživatele. Na webu se lépe a rychleji orientují, snáze se dostanou k potřebným informacím, a podobně.
Někdo by si mohl myslet, že tvorba přístupného webu musí být finančně náročnější, jelikož je zde nutno dodržovat mnoho pravidel při psaní kódu, případně pak výslednou aplikaci podrobit důkladnému testování. Opak je ale pravdou. Takové stránky přinesou nejenom více spokojených uživatelů a s tím spojený zisk, ale zároveň ušetří náklady

spojené s dodatečným vyřizováním různých záležitostí se zákazníkem, který nemohl získat potřebné informace na internetu a musí tak využít jiných informačních kanálů.

Další výhodou je lepší viditelnost webu ve vyhledávačích. Převážná většina uživatelů začíná svoji pouť po síti na stránkách internetových vyhledávačů. Pokud se stránky zobrazují na prvních pozicích vyhledávání, jde o obrovskou konkurenční výhodu. Toho si je samozřejmě většina firem a institucí dobře vědoma a proto dodatečně vkládají své finance do SEO optimalizací a doufají, že se v dohledné době posunou o několik příček vzhůru (pokud nevyužijí přímo placené možnosti zviditelnění ve vyhledávačích).
Přitom web dodržující pravidla přístupnosti je vyhledávači zpravidla automaticky upřednostňován, jelikož vyhledávací roboti takovému webu plně porozumí.

Jak tvrdí přední český odborník na webovou přístupnost David Špinar, přístupný web je zároveň i "robot -friendly", protože [4]:

- Veškeré informace jsou přítomny v textové podobě. A to buď pouze v ní, nebo v podobě textových alternativ grafických prvků.
- Žádný obsah nezůstal před roboty ukrytý například ve Flashi nebo JavaScriptu
- Všechny odkazy jsou plně funkční a robot se pomocí nich dostává na další stránky
- Je dodrženo sémantické označování textů. Tvůrce webu jasně vyznačil název stránky (značka <***title***>), jednotlivé nadpisy (značky <***hl***> až <***h6***>) nebo různá zdůraznění (značky <***em***> a <***strong***>). Z těchto sémantických značek robot pozná, který obsah má na stránce prioritu
- Nejdůležitější obsah je umístěn na začátku stránky, kde existuje větší pravděpodobnost, že si ho robot vyhledávače všimne

Tvorba přístupných webových stránek je tedy výhodná v mnoha směrech a každý webkodér by měl mít snahu, aby produkt jeho práce odpovídal alespoň nejdůležitějším pravidlům přístupnosti webových stránek.
Pro lepší pochopení problémů, s kterými musí bojovat hendikepovaní při prohlížení internetových stránek, blíže popíši vybrané hendikepy a s tím spojené bariéry.

[4] ŠPINAR, D., Tvoříme přístupné webové stránky, s. 17

3.1.1 Hendikepy a bariéry

Opomeneme hendikepy spojené s technickými a softwarovými záležitostmi, jelikož tyto problémy jsou zpravidla akceptovány a většina webů je pro tyto případy optimalizována.

Zbylé hendikepy si tedy můžeme rozdělit do následujících kategorií [5]:

1. Zrakově postižení
2. Sluchově postižení
3. Pohybově postižení
4. Uživatelé s poruchami učení a soustředění

Zrakově postižení

Nejčastěji zmiňovaná skupina hendikepovaných uživatelů internetu, nikoliv však nejpočetnější. Vzhledem k tomu, že internet, respektive www stránky, jsou hlavně médiem vizuálním, je tato skupina často vnímána jako hlavní.

Podle typu zrakového postižení a s tím spojenými bariérami můžeme dále dělit zrakové postižené do těchto podkategorií:

Zcela nevidomí a jinak těžce zrakově postižení

Uživatelů se zcela či většinově nepoužitelným zrakem sice ve společnosti není tak velké množství, ale jsou tu. Tito uživatelé používají jako výstupní zařízení ***hlasový výstup*** nebo ***braillský řádek***. Tyto pomůcky dokáží interpretovat pouze text a z tohoto faktu vyplývá i jejich základní potřeba - obsah musí být prezentován formou dobře strukturovaného textu, grafické a multimediální prvky musí mít textové alternativy, navigace musí být použitelná apod. Jako vstupní zařízení pak namísto myši používají ***klávesnici***, buďto klasickou, nebo speciální upravenou.[6]

[5] ŠPINAR, D., Tvoříme přístupné webové stránky, s. 29
[6] Přístupnost: Hendikepovaní uživatelé internetu [online]. <http://pristupnost.nawebu.cz/texty/hendikepovani-uzivatele.php>

Uživatelé s vadou zraku

Slabozrací uživatelé mohou svůj zrak používat, byť jen omezeně. Jejich základní potřebou je možnost s obsahem stránky manipulovat. Většinou ho zvětšovat.
Tito uživatelé tedy především potřebují možnosti si zvětšit text, a to tak, aby zůstal vůči svému okolí i nadále čitelný. Proto je třeba velikost písma i jiných prvků na stránce definovat pomocí *relativních* jednotek a použitelnost stránky zkontrolovat i při radikálně zvětšeném písmu.[7]

Mezi tyto vady patří:

Katarakta – neboli šedý zákal, jedná se o zakalení čočky. Výsledkem je rozmlžení a rozostření obrazu. Snížená vnímavost na nižší barevné kontrasty

Diabetická retinopatie – poškození oční cévy jakožto příznak cukrovky. Projevuje se jako výpadky v zorném poli

Glaukom – neboli zelený zákal, jedná se o poškození očního nervu poklesem nitroočního tlaku. Výsledkem je trubicové vidění a ztráta periferního vidění. Postižený tedy vidí jen malé okolí prostoru, na který se dívá

Degenerace sítnice – existuje více druhů a více příčen. Příkladem může být opak trubicového vidění, tedy tzv. makulární degenerace typická pro osoby staršího věku

Barvoslepí

Forem sníženého barevného citu je celá řada. V podstatě jde ale vždy o totéž. Někteří uživatelé mají problém rozeznávat od sebe některé škály barev. Nastane-li tento problém u barvy pozadí a popředí na webové stránce, stává se taková stránka nepřístupnou.
Hlavní potřebou takových uživatelů je tedy to, aby barvy na stránce měly dostatečný kontrast a v rámci důležitého obsahu se na konkrétní barvu nespoléhalo vůbec. [8]

[7] Přístupnost: Hendikepovaní uživatelé internetu [online].
< http://pristupnost.nawebu.cz/texty/hendikepovani-uzivatele.php>
[8] Tamtéž.

Mezi tyto vady patří ***dichromazie***, kdy má člověk sníženou schopnost vnímání jedné barvy, a to:

- Protanopie – nevidí červenou barvu
- Protanomálie – vidí hůře červenou barvu
- Deuteranoie – nevidí zelenou barvu
- Deuteranomálie – vidí hůře zelenou barvu
- Tritanopie – nevidí modrou barvu, nejméně častá

Monochromazie je pak stav, kdy člověk vidí pouze ve stupních šedi.

Uživatelé s dočasně zhoršenou možností vidět

Do této skupiny spadají uživatelé, kteří netrpí žádnou formou vady zraku, ale v důsledku vnějších vlivů mají zhoršenou viditelnost. Může se jednat o přesvětlenou místnost, vybledlé barvy monitoru apod., což zapříčiní zhoršení kontrastu barev na stránce.

Sluchově postižení

Sluchově postižených uživatelů je na internetu mnoho a na velké většině www stránek nemají s prohlížením sebemenší problémy. Výjimkou mohou být www prezentace, které se spoléhají převážně na audiovizuální stránku a s tím spojenou prezentaci obsahu. Všeobecně internet zažívá rozmach multimédií, což se odráží i v rozšiřující specifikaci HTML5. Je vhodné proto k těmto multimediálním datům nabídnout alternativní text.
Se sluchovým postižením souvisí ještě jedna indispozice, týkající se od narození sluchově postižených uživatelů. Ti mají, kvůli svému hendikepu, často menší slovní zásobu a dělá jim problém porozumět obtížně strukturovanému a složitému textu. A na tuto stránku, tedy vhodně strukturovaný a srozumitelný text, bychom měli při tvorbě webu myslet. Koneckonců, je to jedna z podmínek SEO optimalizace.

Pohybově postižení

Z hlediska ovladatelnosti webových stránek můžeme do této kategorie řadit převážně uživatele s pohybovým omezením horních končetin. Tento hendikep jim zamezuje

používat klasické vstupní zařízení, tedy myš, v některých případech dokonce i klávesnici. K ovládání tak používají například upravené klávesnice (pro ovládání jednou rukou), trackbally, v extrémních případech i trubičky reagující na dech uživatele.

Do této kategorie dále patří i uživatelé s dočasným omezením pohyblivosti, například při zlomenině apod. V neposlední řadě sem patří také mnoho seniorů, jejichž motorika již není dostatečně jemná pro ovládání myší.

Všechna výše zmíněná vstupní zařízení pracují převážně na bázi klávesnice. Minimem při tvorbě webu je tak použitelnost pouze za použití klávesnice, tedy dostupnost všech ovládacích prvků, formulářových polí atd.

Uživatelé s poruchami učení a soustředění

Do této kategorie, v souvislosti s prohlížením webových stránek, patří hlavně uživatelé trpící dysleksií. Neschopnost správně vnímat a zpracovávat jazykové informace je nejčastější poruchou učení. Některou její formou trpí 15 - 20% populace. Dysleksie je mezi nimi nejvíce rozšířená.[9]

Krom dysleksie sem spadají i uživatelé s poruchou soustředění. Příkladem je hyperaktivita často se vyskytující u mladistvých. Všichni tito uživatelé mají problémy s pochopením příliš dlouhých textů, chaotické navigace a dalších pro ně obtížně pochopitelných prvků. Tato oblast není zatím v kontextu webové prezentace příliš prozkoumaná, neví se ani přesně, jestli problém spadá do přístupnosti, nebo použitelnosti.

Uživatelé s poruchami učení a soustředění tedy potřebují přehledné, strukturované a jednoduše pochopitelné webové stránky se strukturovaným obsahem, přehlednou navigací. Písmo spíše větší, krátké odstavce, hodně nadpisů, více vizuálních "zarážek" pro oči, více obrázkových symbolů místo slov apod.[10]

[9] ŠPINAR, D., Tvoříme přístupné webové stránky, s. 43
[10] Přístupnost: Hendikepovaní uživatelé internetu [online].
< http://pristupnost.nawebu.cz/texty/hendikepovani-uzivatele.php>

3.1.2 Metodiky přístupnosti a právní normy

Protože je k docílení přístupnosti potřeba dodržení mnoha postupů, byla vytvořena metodika Web Content Accessibility Guidelines (WCAG) a na jejím základě vytvořená česká metodika „Blind Friendly Web“.
Jak je však pro lidi typické, pokud to není přímo nutné, aplikací těchto metod a s tím spojenou „prací navíc“ (i když by mělo jít o standardní způsoby tvorby webu) se weboví programátoři příliš nezabývají. V mnoha zemích proto fungují zákony, které obecně upravují diskriminaci menšin, mezi které se řadí i zdravotně postižení. V návaznosti na tyto zákony jsou pak vytvořeny zákonné normy pro tvorbu webových prezentací, jako v USA Section 508 Standards, nebo u nás novela Zákona č. 365/2000 Sb., o informačních systémech veřejné správy (ISVS).

WCAG 2.0

Doporučení Web Content Accessibility Guidelines 2.0, vydané konsorciem W3C koncem roku 2008, je v současnosti nejpropracovanějším materiálem o přístupnosti, které je veřejně k dispozici. Tato metodika byla vytvořena na základě předchozí verze WCAG 1.0, které již přestalo vyhovovat požadavkům dnešní doby. Starší metodika upravovala technickou stránku přístupnosti. Ale po technické stránce přístupný web nemusí být automaticky přístupný i reálně, tedy po stránce použitelnosti. To napravuje druhá verze, která procházela téměř osmi letým vývojem. Jak uvádí konsorcium W3C na svých stránkách, WCAG 2.0 zahrnuje celou řadu doporučení pro vytváření přístupnějšího webového obsahu. Na základě těchto pokynů bude obsah přístupný širšímu okruhu osob se zdravotním postižením, včetně slepoty a hluchoty, zhoršeným zrakem, ztrátou sluchu, poruchou učení, kognitivním omezením, omezením pohybu, poruchami řeči, fotosenzitivitou a jejich kombinacemi. V návaznosti na tyto pokyny bude také takovýto web často více použitelný pro uživatele obecně.[11]

Tato nová pravidla jsou flexibilnější a nadčasová. To zajišťuje oddělení principu přístupnosti od konkrétního technického řešení. Navíc je tato metodika nezávislá na technologiích a nemusí se tak uplatňovat jen při tvorbě www prezentace.

[11] W3C: Web Content Accessibility Guidelines (WCAG) 2.0 [online]. <http://www.w3.org/TR/2008/REC-WCAG20-20081211/>

WCAG 2.0 definuje 4 principy přístupnosti [12]:

- **Vnímatelnost:** informace a součásti uživatelských rozhraní musí být prezentovány tak, aby je uživatelé byli schopni vnímat
- **Ovladatelnost:** všechny součásti uživatelského rozhraní a všechny navigační prvky musí být ovladatelné
- **Srozumitelnost:** informace a ovládání uživatelského rozhraní musí být srozumitelné
- **Robustnost:** obsah musí být dostatečně robustní, aby mohl být spolehlivě interpretován širokou škálou přístupových zařízení včetně asistivních technologií

Každý tento princip definuje několik pravidel (celkem jich je 12). S nimi úzce souvisí tzv. „kontrolní kritéria“, která slouží k otestování webu a ověření jeho souladu s danými kritérii. Podle míry vyhovění danému kritériu spadá obsah do jedné ze tří kategorií A (nejnižší), AA a AAA (nejvyšší).

Blind Friendly Web

Tato česká metodika tvorby přístupného webu je zaměřena hlavně na zrakově postižené. Projekt vznikl v roce 2000 ve *Sjednocené organizaci nevidomých a slabozrakých ČR*, jako reakce na nepoužitelnost mnoha webových stránek pro zrakově postižené. Cíli tohoto projektu bylo mapování přístupných stránek a jejich zařazení na seznam Blind Friendly, tvorba metodik a návodů pro tvůrce webu a osvětová činnost.

Blind Friendly Web je **první projekt v ČR**, který se začal systematicky věnovat přístupnosti webových stránek.[13] V roce 2004 tak vznikl první metodický materiál *Best Practice – pravidla pro tvorbu přístupného webu*. Od roku 2006 pak spolu s obecně prospěšnou společností SONS pracovali na projektu přístupnosti webových stránek orgánů státní správy. Tento krok byl dokončen rokem 2008 a i nadále tento projekt pokračuje v dalším vývoji.

[12] Zroják.cz: WCAG 2.0 - začínámě [online]. < http://zdrojak.root.cz/clanky/wcag-2-0-zaciname/>
[13] Blind Friendly: O projektu [online]. < http://www. http://blindfriendly.cz/o-projektu>

Zákona č. 365/2000 Sb., o informačních systémech veřejné správy

V České republice se projekt pravidel tvorby přístupného webu datuje k roku 2004. Tato pravidla jsou zakotvena v novele zákona číslo 365/2000 Sb., o informačních systémech veřejné správy provedenou zákonem číslo 81/2006 Sb. Pro zjištěné nedostatky a zároveň v návaznosti na zveřejnění nové metodiky WCAG 2.0 byl zákon ještě několikrát novelizován, naposledy pak zákonem č. 18/2012 Sb.
Zákon o informačních systémech veřejné správy stanoví práva a povinnosti správců informačních systémů veřejné správy (ISVS) a dalších subjektů, jež souvisejí s vytvářením, užíváním, provozem a rozvojem informačních systémů veřejné správy. V návaznosti na to upravuje působnost Ministerstva vnitra jako ústředního správního úřadu pro tvorbu a rozvoj informačních systémů veřejné správy. Zákon vytváří podmínky, aby kvalitní informační systémy byly dobrým nástrojem pro výkon veřejné správy.[14]

Section 508 Standards

Jedná se o část č. 508 amerického zákona Rehabilitation Act, která byla do zákona přidána v roce 1998 a stanovuje federálním orgánům USA povinnost poskytování informací přístupným způsobem.
Konkrétněji § 508 tohoto zákona vyžaduje, pokud federální agentury vyvíjejí, opatřují, udržují nebo používají EIT, aby federální zaměstnanci se zdravotním postižením měli přístup a mohli využívat informace a data srovnatelným způsobem, jako federální zaměstnanci, kteří nejsou zdravotně postižení. [15]
Do tohoto výčtu ji zařazuji z důvodu, že v mnoha testovacích nástrojích je tato metodika k dispozici pro analýzu přístupnosti.

3.2 Testování www aplikací

Metod pro testování, či analýzu www stránek, existuje celá řada. Testování je, nebo by alespoň mělo být, součástí vývoje každé webové aplikace (a nejen webové). Často se stává,

[14] Ministerstvo vnitra České republiky: Zákon č. 365/2000 Sb., o informačních systémech veřejné správy [online]. < http://www.mvcr.cz/clanek/zakon-c-365-2000-sb-o-informacnich-systemech-verejne-spravy.aspx>
[15] Section508.gov: Standards [online]. < http://www.section508.gov/index.cfm?fuseAction=stds>

že designér webu, v rámci originality výsledného produktu, navrhne nové, neobvyklé funkce ovládání, atypické rozložení prvků stránky apod. Ve výsledku pak mohou být takovéto stránky uživatelsky nepřívětivé. K tomu je zapotřebí testování stránek ještě před jejich zveřejněním. Nejedná se jen o testování základní použitelnosti. Na stránkách můžeme testovat více méně všechno, od přístupnosti až po funkčnost odkazů. Druhotným efektem testování je pak i minimalizace nákladů na případnou rekonstrukci webu.
Před samotným testováním je zapotřebí vybrat metodu, kterou pro testování využijeme a poté rozsah testování, tedy jaké faktory konkrétně budeme testovat.

3.2.1 Metody testování

Metod testování je celá řada. Liší se jak v kvalifikaci testerů, tak v časové a finanční náročnosti a samotných výsledcích. Cílem testování je zjistit, jestli je vše správně naprogramované a co je případně potřeba zlepšit. Dále je popsáno několik základních metod testování. Většina těchto metod se využívá k testování použitelnosti. Jednoduše měřitelné faktory testování si je totiž schopen otestovat programátor sám za použití aplikací třetích stran.

Uživatelské testování

Jedná se o nejpoužívanější metodu testování použitelnosti webových stránek. Cílem je zjištění, které prvky na webu nejsou uživatelsky srozumitelné a jak se bude běžný uživatel na webu s největší pravděpodobností chovat.

Uživatelské testování probíhá v několika krocích[16]:

- **Analýza cílových skupin webu a jejich potřeb** – je potřeba definovat okruh uživatelů, kteří budou web využívat, a k čemu je aplikace určena
- **Vytvoření scénáře testování** – hlavním těžištěm testování jsou úkoly, seřazené od jednoduchých ke složitým, které jsou jednotlivým testerům uloženy. Jedná se o většinou typické akce cílových skupin webu, například registrace, provedení objednávky, nalezení kontaktu.

[16] Dobrý web: Uživatelské testování použitelnosti [online]. < http://www.dobryweb.cz/uzivatelske-testovani>

- **Výběr testerů** – vzorek účastníků testování by měl odpovídat cílovým skupinám webu s tím, že by měli být zahrnuti zkušení i nezkušení uživatelé webu. Optimální počet testerů se udává mezi 3 a 5.
- **Samotný průběh testování** – pod dohledem zkušeného odborníka na použitelnost plní testeři úkoly dle scénáře testování. Testeři přemýšlejí nahlas, aby mohl být postup zaznamenán. Optimální doba testování je 30 minut.
- **Analýza výsledků testování** – je třeba setřídit poznatky získané během testování, zanalyzovat je a především vymyslet nejvhodnější řešení vedoucí ke zvýšení použitelnosti webu. Do výsledků patří čas potřebný k úspěšnému provedení úkolu, chybovost, úspěšnost, spokojenost.
- **Prezentace výsledků** – seznámení s průběhem testování, komentář k výsledkům a detailní výklad aplikace doporučených opatření.

Toto testování s sebou nese i určitá rizika. Těmi jsou nevhodný výběr testerů (jsou například zkušenější, než jsme si mysleli), ovlivnění testerů časovým limitem a někdy i dohledem, kdy se tester pokouší stále splnit úkol, který by jinak už dávno vzdal.

Heuritsická analýza

Další základní metoda k testování www stránek. Na rozdíl od uživatelského testování jsou zde v roli testerů odborníci. Ti pak na základě obecně známých pravidel zjišťují, zda stránky těmto pravidlům odpovídají. Doporučený počet odborníků by měl být 3-5, v praxi však, z důvodu finanční náročnosti, jsou nejčastěji 1-3. Postup testování může být buďto strukturovaný, kdy test probíhá dle scénáře, nebo nestrukturovaný, kdy tester prochází a testuje stránky náhodně.
Při analýze se můžeme řídit desaterem principů použitelnosti jednoho ze zakladatelů heuristické analýzy Jakobem Nielsnem:[17]

- **Viditelnost stavu systému**
  Systém by měl vždy průběžně informovat uživatele o tom, co se děje, pomocí vhodné zpětné vazby v přiměřené době.

[17] UseIt.com: Ten Usabiliti Heuristics [online]. < http://www.useit.com/papers/heuristic/heuristic_list.html>

- **Shoda mezi systémem a reálným světem**
  Systém by měl mluvit „jazykem uživatelů“, používat slova a fráze známá uživateli, a vytvářet situace známé z reálného světa.
- **Kontrola uživatelem a svoboda**
  Uživatelé často dělají chyby, a proto by měl web umožňovat vždy se vrátit jednoduše o krok zpět či vpřed.
- **Konzistence a standardy**
  Web a jeho navigace, výrazy a akce by měly být jednotné pro celý web.
- **Prevence chyb**
  Lepší než dobré chybové hlášky je předcházení těmto chybám. Proto by měli být stránky zabezpečeny proti běžným uživatelským chybám.
- **Upozornění raději než zapamatování**
  Web by měl minimalizovat informace, které si uživatel musí zapamatovat, aby web mohl dobře používat. Všechny důležité informace musí být zvýrazněné a viditelné v momentě, kdy je uživatel potřebuje.
- **Flexibilita a efektivnost používání**
  Web by měl nabízet různé způsoby realizace úkolů tak, aby rutinní uživatelé mohli dosahovat úkolů rychleji, než nezkušení uživatelé. Uživatelé se mohou přizpůsobit.
- **Estetický a minimalistický design**
  Web by neměl obsahovat žádné nadbytečné informace nebo grafické prvky. Každý zbytečný prvek na stránce snižuje viditelnost těch významných.
- **Pomoci uživatelům rozpoznat, diagnostikovat a učit se z chyb**
  Chybové hlášky by měly být napsány jednoduchým jazykem (žádné kódy), přesně popisující problém a navrhnout konstruktivní řešení.
- **Nápověda a dokumentace**
  Přestože je lepší, když je web ovladatelný bez dokumentace, může být nezbytné poskytnout pomoc a dokumentaci. Všechny informace by měly být snadno dohádatelné a zaměřené na uživatelské dotazy.

Výsledkem testování je report obsahující objevené problémy. Ten je předán jako návrh k přepracování. Výhodou této metody je kvalita výsledného testu, tedy množství objevených problémů. Nevýhodou pak může být neobjektivní hodnocení určitého

problému, vzhledem ke kontextu celého webu (expert považuje za důležité něco ze svého pohledu na věc).

Card-sorting

Jde o jednoduchou techniku testování, v češtině nazývanou „třídění karet“. Tato metoda nabízí pohled uživatelů na celkovou strukturu webu, jeho navigaci a přístup ke všem informacím. Výsledky je tedy vhodné uplatnit při návrhu struktury webu.
Minimální počet testerů je 6, optimálně pak 15.

Card-sorting může probíhat dvěma postupy:

- **Open card-sorting** – uživatelé dostanou kartičky s obsahem webu, aniž by znali základní rozdělení webu, tedy nemají určeny skupiny k přiřazování. Testeři tak nejprve seskupí kartičky do skupinek podle svých logických úsudků a poté skupiny vhodně pojmenují. Tento postup je vhodný při tvorbě nového webu.
- **Closed card-sorting** – zde jsou rozdány kartičky s obsahem webu a je již nadefinován počet a názvy skupin. Tester přiřazuje kartičky do těchto skupin. Tento postup je vhodný při přidávání nového obsahu, nebo pro kontrolu stávající struktury.

Výhodou této metody je její rychlost, jednoduchost a finanční nenáročnost. Záporem můžou být velké rozdíly mezi výsledky (zejména u metody Open card-sorting) a s tím spojená složitější analýza výsledků

Focus Groups

Focus groups jsou poněkud neformální technika, která vám pomůže vyhodnotit potřeby uživatelů a pocity jak před návrhem rozhraní, stejně tak i dlouho po realizaci.[18]

Tato metoda je používaná převážně v reklamě a marketingu. Jedná se o skupinovou diskuzi řízenou moderátorem. Respondenti odpovídají na předem připravené otázky. Sezení se účastní 8 – 12 respondentů (dle zdrojů se počet různí), kteří musí být vybráni s ohledem na

[18] UseIt.com: The Use and Misuse of Focus Groups [online]. <http://www.useit.com/papers/focusgroups.html>

znalosti probíraného tématu. V ideálním případě by se neměli respondenti mezi sebou znát. Neméně důležitá je role moderátora, který diskuzi vede, pokládá otázky a vhodně reaguje na podněty respondentů.

Nevýhodou metody je vyšší finanční náročnost, nároky na moderátora, či ovlivňování úsudků respondentů mezi sebou.

Cognitive walkthrough

Metoda poznávacího procházení se požívá k vyhodnocení použitelnosti webu. Určuje, jak je pro nové uživatele obtížné plnit úkoly na stránkách, tedy orientovat se v nich. Metoda se používá při tvorbě nových stránek, nebo lze využít i pro již existující web. V průběhu testu provádějí testeři předem zadané úkony na stránkách. U každého úkonu zvažují, je-li dostatečně srozumitelný pro běžného uživatele, jestli má dostatek informací pro jeho řešení apod.
Pokud je daný úkon označen jako splnitelný, obsahující dostatek informací, je tento označen jako splněný. V opačném případě je označen za nesplněný a je potřeba identifikovat problémy a nedostatky a předat je k řešení. Výhodou je nízká náročnost na množství i odbornost testerů, nevýhodou je testování velkých a složitých projektů.

A/B testování

Jedna z nejzákladnějších metod testování. Využívá se k otestování a vyhodnocení změn provedených na webové stránce. Principem je porovnání dvou variant, které jsou nasazeny proti sobě. Jednotlivé varianty jsou označovány písmeny. V případě testování dvou verzí mluvíme o A/B testování, testování tří verzí se nazývá A/B/C testování. Označení pro testování více variant se odvozuje analogicky.[19]Jde o velmi levnou metodu testování, která se využívá za provozu, testery jsou tedy běžní uživatelé. Těm je náhodně nabídnuta jedna z variant a na základě výsledků (například testů návštěvnosti, množství objednávek, počtu kliknutí na reklamu…) se zvolí žádanější, tedy i lepší varianta.

[19] Dobrýweb: Newsletter: A/B testování webových stránek [online].
< http://blog.dobryweb.cz/newsletter-ab-testovani-webovych-stranek>

Oční kamera

Neboli „Eye tracking“ je poměrně nová metoda testování webu. Jde o technologii sledující pohyb očí uživatele na stránce. Výsledkem analýzy je mapa, znázorňující místa, kam se uživatel nejdéle díval a kde se pohyboval myší. Z této mapy lze vyčíst, co uživatele nejvíce zaujalo a co naopak zcela přehlížel. Na základě zjištění, že například uživatele nezaujme část, která je pro nás důležitá, ji můžeme přemístit do míst, na které se uživatel více soustředil.

3.2.2 Faktory testování

Před testováním je důležité si určit, co všechno chceme testovat. Od typu testovaného faktoru se odvíjí i použitá metoda. Něco můžeme testovat sami v průběhu programování, k něčemu poslouží na internetu volně dostupné analyzéry, k něčemu je nejvýhodnější použít některou z výše zmíněných metod (zejména pro testování použitelnosti).
Všechny testované faktory na webových stránkách se dají rozdělit do dvou hlavních skupin:

- **Objektivní faktory** – výsledek je jasně měřitelný a dá se jednoznačně vyhodnotit. Do této kategorie spadá například validita kódu, splnění zásad přístupnosti, funkčnost odkazů, atd.
- **Subjektivní faktory** – výsledek není jednoznačný a záleží na nás, jak s ním naložíme vzhledem k požadovanému cíli. Patří sem zejména testování použitelnosti a částečně optimalizace pro vyhledávače.

Faktory se kromě rozdílného vyhodnocování výsledků testování liší i v dalších směrech. S některými se pracuje již při tvorbě obsahu (hustota klíčových slov, informační architektura), jiné je dobré ověřovat v celém průběhu vývoje (validita, datová velikost objektů, přístupnost). Další stačí testovat až po vytvoření stránek (funkčnost odkazů) a případné chyby napravit dodatečně.[20]

[20] Interval.cz: Testování webových stránek [online]. < http://interval.cz/clanky/testovani-webovych-stranek/>

Základní testované faktory jsou:

Validita

Jedná se o základní testování všech webových stránek. Validita kódu HTML a CSS dle pravidel W3C by měla být samozřejmostí pro všechny tvůrce www stránek. Zároveň se jedná o jednu s nejsnáze testovatelných veličin. K tomu slouží validátory, které jsou dostupné online. Ty projdou syntaktický kód a zkontrolují jeho soulad s nastavenými standardy. Při nalezení chyby na problém upozorní a zpravidla navrhnou i správně řešení problému.

Funkčnost odkazů

S tímto obvykle nebývá příliš problém, nicméně každému se někdy může stát, že se při zadávání odkazu přepíše a nevědomě tak vloží nefunkční link. Testování tohoto faktoru je opět jednoduché. U malých projektů je ověření funkčnosti odkazů otázkou několika kliknutí, u větších projektů je opět možnost využít programů třetích stran.

Přístupnost

O přístupnosti již zde bylo napsáno mnoho. Mělo by být snahou každého programátora dodržovat alespoň základní principy přístupnosti, jako je kontrast barev, ovladatelnost při vypnutých stylech atd. Stejně jako v obou předchozích případech není s testováním přístupnosti mnoho problémů. Na webu jsou k dispozici online validátory dle vybraných standardů, toolbary a další aplikace zaměřující se na konkrétní problémy přístupnosti.

Datová velikost a rychlost stránek

Ikdyž je v dnešní době rychlost připojení k internetu v domácnostech zpravidla na dostačující úrovni, přesto je vhodné myslet i na uživatele s pomalejšími připojeními. Na základě toho je vhodné udržet velikost načítané stránky (včetně všech grafických prvků) na přijatelné velikosti. Samotný obsah by měl být k dispozici co nejrychleji, dodatečná grafika se může načítat déle. K měření rychlosti stránek je opět k dispozici celá řada online aplikací, která provede dostatečnou analýzu včetně navrhovaných doporučení.

Použitelnost

Použitelnost stránek je velice široký obor. Základem dobře použitelných stránek je hlavně propracovaná navigace a kvalitní informační architektura. Dobrá použitelnost stránek znamená, že [21]:

- uživatelům (čtenářům) se podaří na webu udělat to, co chtějí. Např.
    - najít informaci, pro kterou přišli
    - najít email v kontaktech
    - přečíst si novinky
    - zaregistrovat se
    - objednat si zboží
- dokáží to v rozumném čase a bez velkého přemýšlení
- dokáží to bez chyb a zásadních zklamání

Testování použitelnosti je složitý proces, proto se často přenechává firmám, specializujícím se na tento obor. Pokud je třeba testovat použitelnost svépomocí, nabízí se k testu pozvat známé a příbuzné, kteří web uživatelsky otestují. Mohou však opomenout mnoho důležitých bodů.

Optimalizace pro vyhledávače

Obdobně jako použitelnost je i optimalizace pro vyhledávače samostatný obor. Zažila se pro něj všeobecná zkratka SEO (Search Engine Optimization). Vzhledem k testování je z této části možné testovat jen některé aspekty, pro něž existuje na webu několik testovacích nástrojů.
Jedná se o testování hustoty klíčových slov, test stránky pohledem vyhledávacího robota (vhodné i k testování přístupnosti) a testování pozice ve vyhledávačích.

Bezpečnost aplikací

Testování bezpečnosti aplikace má jednu odlišnost od testování ostatních faktorů. Předpokládá totiž znalost bezpečnostních rizik a způsobů, jakými lze nezabezpečené

[21] Jak psát web: Použitelnost stránek [online]. < http://www.jakpsatweb.cz/pouzitelnost.html>

aplikace napadnout. Testování tedy zpravidla probíhá tak, že tester v roli útočníka vyzkouší na stránkách vybrané typy útoků, jako je SQL Injection, nebo XXS, a na základě úspěchu či neúspěchu doporučí konkrétní zabezpečení aplikace.

3.3 Bezpečnost www aplikací

Bezpečnost www aplikací je důležitým, ale stále často opomíjeným bodem při jejich vývoji. V dnešní době je spousta firemních aplikací realizována na bázi klient/server, a to navíc s otevřením do internetu. Mnoho firem pak řeší zabezpečení na úrovni vnitřní infrastruktury, tedy od firewallů po antivirové programy na cílových stanicích. Často je ale zapomínáno na nebezpečí, jemuž otevírají vrátka právě ony webové aplikace. Pokud je jejich zdrojový kód nezabezpečený, může být pro útočníka taková webová aplikace vstupní branou k datům firmy.

Téměř 70% útoků je vedeno na aplikační vrstvu. Toho by si měly být vědomi tvůrci webových aplikací a bezpečnostní opatření aplikovat již při psaní kódu. Často je však hlavní snahou tvůrců vytvoření co nejzajímavějšího webu s mnoha funkcemi, tak, aby plně vyhovoval požadavkům zadavatele, a bezpečnostní stránka aplikace je odsunuta na druhou kolej. V lepším případě jsou pak bezpečnostní opatření zdrojového kódu dodělána dodatečně (s tím, že některé prvky mohou být opomenuty), v horším případě je pak aplikace nezabezpečená a takto je také předána uživateli.

Útoků na www stránky je celá řada, a na většinu existují pravidla, jak útoku zamezit. Bohužel jako všude i zde platí, že útočník je vždy krok před námi. Nic nám ale nebrání chránit se před známými metodami útoků a na ty nové se také vždy nalezne řešení.

Abychom měli přehled o nejpoužívanějších formách útoků, byl spuštěn projekt **OWASP** (Open Web Application Security Project). Tato komunita se zabývá bezpečností webových aplikací. Zveřejňuje 10 nejpoužívanějších typů útoků, včetně jejich popisu a způsobu ochrany před nimi. Tento dokument představuje konsensus mnoha odborníků z celého světa o nejzávažnějších slabinách webů.

3.3.1 Nejzávažnější rizika bezpečnosti www aplikací

Naposledy byl OWASP Top Ten aktualizován v dubnu 2010 a zahrnuje tyto nejzávažnější rizika bezpečnosti webových aplikací [22]:

A1 – Injection

Napadení aplikace vsunutím kódu přes neošetřený vstup. Např. SQL, LDAP, OS injection.

A2 – Cross Site Scripting (XSS)

Narušení webových aplikací vkládáním skriptů či jejich částí do webového prohlížeče.

A3 – Broken Authentication and Session Management

Napadení funkcí aplikací, které souvisí s autentizací a správou sezení s cílem převzít identitu jiného uživatele.

A4 – Insecure Direct Object References

Přístupy k neoprávněným datům nezabezpečeným přístupem k vnitřnímu objektu aplikace (souboru, adresáři, databázovému klíči).

A5 – Cross Site Request Forgery (CSRF)

Útok podvržením požadavku webové aplikace.

A6 – Security Misconfiguration

Útoky využívající nedostatků v zabezpečení konfigurací aplikačních serverů, webových serverů, databázových serverů, softwarových platforem atd. (např. ponecháním výchozích hodnot, neaktualizování aj.)

A7 – Insecure Cryptographic Storage

Rizika vyplývající z nedostatečně chráněných citlivých dat (čísla kreditních karet, rodná čísla atd.).

A8 – Failure to Restrict URL Access

Nedostatečně zabezpečené řízení přístupu ke zdrojům informací (dokument, služba) přes URL.

---

[22] Root.cz: Deset nejběžnějších bezpečnostních chyb na webu [online].
< http://www.root.cz/clanky/deset-nejbeznejsich-bezpecnostnich-chyb-na-webu/>

A9 – Insufficient Transport Layer Protection

Útoky odposlechem sítí.

A10 – Unvalidated Redirects and Forwards

Útoky přesměrováním na jiné stránky

3.3.2 Popis vybraných typů útoků

Metody SQL Injection, XSS, krádež relace a CSRF jsou na prvních příčkách útoků již několik let, dají se tedy považovat za jakési „stálice“. Přestože proti všem těmto útokům existují účinné a i poměrně jednoduché způsoby ochrany, často jsou programátory opomíjeny. Dále je blíže popíši.

SQL Injection

Injektování je jedna z nejpoužívanějších metod útoku na www aplikace. Cílem útočníka je získat data, která by jinak měla zůstat uživateli utajena.
Pomocí SQL Ijnection je útočník schopen měnit nebo zadávat dotazy, které se odesílají do databáze prostřednictvím vstupů do webové aplikace. Samotný útok začíná v okamžiku, kdy program vytváří dotazy založené na řetězcích pocházejících od klienta, a když je následně posílá na databázový server, aniž by ošetřil znaky, které server považuje za speciální.[23]

XSS

Neboli Cross Site Scripting. Tato metoda využívá vstupu od uživatele a jeho následného neošetřeného výstupu na stránce.
XSS spočívá v obelstění webového serveru takovým způsobem, že pak odesílá uživateli škodlivý kód HTML, obvykle se jedná o skript. Záměrem takového útoku je často ukradnout informace o probíhající relaci, a následně komunikovat se serverem jménem

[23] HUSEBY, SVERE H., Zranitelný kód, s. 32

oběti.[24] Toho se dá využít pro přesměrování údajů z formuláře k útočníkovi, podvržení obsahu apod. Všeobecně se jedná spíše o útok na uživatele, než na www aplikaci.

Krádež relace

Jelikož je http bezstavový protokol, je pro udržení spojení využíváno tzv. sezení (session). To je ale příležitostí pro útočníky, pokud se jim podaří získat cookie soubor a z něj identifikátor sezení, mohou na webu vystupovat například jako regulérně přihlášení uživatelé. Identifikátor sezení totiž na mnoha webech reprezentuje platné přihlášení uživatele. Nnemusí být pouze odcizen, ale může být útočníkem i podvržen, tedy útočník napadenému vnutí svůj session id.

CSRF

Neboli Cross Site Request Forgery. Podstata útoku spočívá v tom, že uživatele přimějeme navštívit stránku napadané aplikace, která provádí nějakou akci, aniž by o tom uživatel věděl. Útok tím pádem může být snadno veden proti aplikacím, do kterých se útočník může sám přihlásit a tím zjistit jejich strukturu, nebo které mají přístupný zdrojový kód.[25] Podmínkou je, že uživatel je aktuálně přihlášen v napadené aplikaci a má platné cookie. K podvržení nebezpečného kódu se dá dobře využít například HTML tag <img> a jeho atribut „source“.

[24] HUSEBY, SVERE H., Zranitelný kód, s. 99
[25] PHP triky: Cross-Site-Request-Forgery [online]. < http://php.vrana.cz/cross-site-request-forgery.php>

4 ANALÝZA VYBRANÝCH ASPEKTŮ JEDNOTLIVÝCH KATEGORIÍ

V této části popíši vybrané aspekty jednotlivých kategorií. Výběr probíhal s ohledem na dle mého úsudku nejdůležitější body jednotlivých kategorií, tedy přístupnosti, testování a bezpečnostních rizik. Podrobnější analýza všech aspektů jednotlivých kategorií je nad rámec jedné diplomově práce. Vybrané aspekty navíc byly zvoleny i s ohledem na následující praktickou část.

Každý aspekt bude analyzován nejprve z pohledu negativních dopadů, tedy potencionálních problémů v oblasti přístupnosti a nebezpečí hrozícího podceněním bezpečnostního rizika. Poté bude analyzována možnost či způsob zamezení negativního dopadu. Výjimkou bude analýza aspektů testování. Zde se bude jednat o soupis metod, programů a zpracování výsledků pro analyzované faktory testování.

4.1 Analýza prvků přístupnosti webových aplikací

Přístupnost stránek je nejčastěji hodnocena s ohledem na zdravotně postižené uživatele. Mezi nimi, vzhledem k vizuálnímu aspektu www prezentací, se jeví jako nejvíce znevýhodněni zrakově postižení. Velká část pravidel přístupnosti proto odpovídá tomuto omezení. Dále se práce věnuje některým hlavním aspektům přístupnosti.

4.1.1 Grafika webu

Do této kategorie patří obrázky a barvy na webu. Obrázky a grafické prvky webu přináší riziko pro nevidomé, kteří využívají hlasovou čtečku. Barvy a nedostatečný kontrast je pak rizikem pro zrakově postižené, nicméně vhodně zvolené barvy a kontrast barev určitě zpříjemní prohlížení stránek všem uživatelům.

Rizika grafických prvků a jejich řešení

Grafika na webu je zřejmě největší hrozbou jeho přístupnosti. Vzhledem k tomu, že nevidomí uživatelé používají k porozumění www stránce hlasového výstupu, je pro ně

často nevhodně použitá grafika noční můrou. V lepším případě jim zůstane utajen jen obrázek doplňující obsah. V horším případě nemusí ani vědět, na jaké stránce se nacházejí, nebo je jim znemožněno se na stránkách pohybovat. To pokud je název stránky, respektive navigace, řešena grafickými prvky.

Pro lepší pochopení popíši, jak hlasová čtečka interpretuje grafický odkaz nedodržující pravidla přístupnosti.

```
<a href=""><img src="obr/1_menu_obr_2145.gif" width="30" height="15" /></a>
```

Interpretace: Obr Lomeno 1 Menu Obr 2145 Grafika Odkaz

Z toho je zřejmé, že uživatel nezná název odkazu, který je uveden na obrázku. Řešení je jednoduché a dá se říci, že nebývá častou chybou, jelikož na tento problém upozorňují i online validátory. Stačí ke každému obrázku dodat ***alternativní text*** atributem ***alt*** nebo ***longdesc***.
Samotný alternativní text řeší validitu, ale ne ještě nutně přístupnost. Alternativní text totiž musí být srozumitelný a plně popisný.
U grafického odkazu, nebo grafického tlačítka formuláře, tak postačí jednoduchý výraz (např. Domů, Kontakt, Hledat, Odeslat, apod.). U dekoračních nebo ilustračních obrázků může zůstat atribut *alt* prázdný, nebo s obsahem „*ilustrační obrázek*".
Specifikem může být reklamní banner, kde je vhodné toto zahrnout do atributu alt, tedy *alt="[Reklama] ..samotný popis reklamy..."*

U fotografií, které mají význam a sdělují nějaký obsah, je potřeba alternativním textem plně vyjádřit obsah obrázku, např. alt="*Český prezident Václav Klaus se setkal na hradě s šéfem české diplomacie Karlem Schwarzenbergem*".

Poslední variantou grafiky jsou obrázky nesoucí rozsáhlé informace. Typickým příkladem jsou grafy. V takových případech je atribut *alt* nedostačující, použijeme tedy atribut *longdesc*, podle následujícího pravidla[26]:

[26] ŠPINAR, D., Tvoříme přístupné webové stránky, s. 74

1. Do atributu ***alt*** umístíme základní informaci o obrázku. Tedy například "*Graf denní návštěvnosti*".
2. Vytvoříme **speciální stránku,** do které v textové podobě umístíme plnou náhradu obrázku.
3. Na tuto speciální stránku odkážeme v parametru *longdesc*, jehož obsahem je konkrétní URL stránky.

Tedy:

```
<img src="grafl.gif" alt="Graf denní návštěvnosti" longdesc="/desc/grafl.html" />
```

Hlasová čtečka pak atribut *longdesc* interpretuje povelem ***„pro další popis stiskněte enter“***. Stránka *graf1.html* pak bude obsahovat názvy a hodnoty jednotlivých sloupců, například formou tabulky. Dále je vhodné z této stránky odkazovat zpět na původní stránku s obrázkem.

Rizika spojená s barvou a jejich řešení

Barvy jsou velice silným nástrojem v rukou webových designérů. Správně zvolená kombinace barev může navodit příjemný pocit při prohlížení, směrovat oči na důležitá místa apod. Naopak špatná práce s barvami může být velkým problémem z hlediska přístupnosti.

Důležitým prvkem je dostatečný ***kontrast*** mezi popředím a pozadím, zpravidla tedy mezi textem a jeho pozadím. Zaprvé by pozadí nemělo obsahovat jakýkoliv vzorek snižující čitelnost textu. Zadruhé by měl být dodržen již zmíněný dostatečný kontrast. V opačném případně může být text čitelný jen s obtížemi, při některých vadách zraku dokonce zcela nečitelný.

K měření kontrastu slouží speciální metody.

Starší metoda dle pravidel WCAG 1.0 - kontrast je tvořen dvěmi vedle sebe stojícími hodnotami[27]:

1. ***Rozdíl jasu*** - Číslo, které udává, jak se od sebe odlišuje jas jednotlivých barev. Pohybuje se od 0 do 255. Čím je hodnota vyšší, tím je rozdíl jasu vyšší a písmo je na pozadí lépe čitelné. Minimální hodnota pro dobrou čitelnost je 125.
2. ***Rozdíl barvy*** - Číslo, které udává, jak se od sebe odlišují hodnoty jednotlivých barev. Pohybuje se od 0 do 765. Čím je hodnota vyšší, tím je rozdíl barev vyšší a písmo je na pozadí lépe čitelné. Minimální hodnota pro dobrou čitelnost je 500.

Tato metoda se postupem času zdála nedostačující, proto byl s příchodem WCAG 2.0 zaveden nový algoritmus založený na rozdílu světelnosti. Čím je rozdíl světelnosti vyšší, tím lépe.

Algoritmus výpočtu ***světelnosti*** je tento[28]:

(L1+.05) / (L2+.05) kde L je jas a je definována jako .2126*R + .7152*G + .0722B
s využitím linearizovaných R, G, a B hodnot. Linearizované R (například) = (R/FS)^2.2
kde FS hodnota z plné stupnice (255 pro 8 bit barevný kanál). L1 je vyšší hodnota (textu nebo pozadí) a L2 je nižší hodnota

Pro obě metody je k dispozici dostatek on-line i offline nástrojů.

Ke kontrastnímu zobrazení uživatelé také často využívají funkci MS Windows „***vysoký kontrast***“. Po zapnutí se zobrazení převede do vysoce kontrastních barev, tedy černá podklad a bílý text. Web by měl proto být použitelný i v tomto zobrazení.

Poslední pravidlo pro užívání barev je, že ***by barvy neměly být spojeny s žádnou funkcí webu.*** Tedy například u formulářů by nemělo být „povinné pole jsou vyznačena červeně“ a podobně. Standardem je povinné pole označovat hvězdičkou či jinak „graficky“.

[27] ŠPINAR, D., Tvoříme přístupné webové stránky, s. 184
[28] Přístupnost: Rozdíl světelnosti - nový pohled na barevný kontrast [online]. <http://pristupnost.nawebu.cz/weblog/blogpost.php?post=127>

Další s tímto spojenou častou chybou je odlišování odkazů v textu pouze barvou. ***Odkaz by měl být vždy podtržen***, jinak je jeho identifikace pro uživatele s některými vadami zraku nemožné.

4.1.2 Ovládání webu

Do této kategorie patří navigace www stránky a odkazy na stránce. Jednotlivé stránky by měly být snadno dostupné, z jakéhokoliv místa jen na pár kliknutí. Odkazy by měly být viditelně označené a nadefinované.
Celý web by mělo jít ovládat jen za pomocí klávesnice. Žádné ovládací prvky nesmí být závisle na Javě, Flashi a podobně.

Rizika spojená s navigací webu a jejich řešení

Přestože pro zdravého uživatele, navíc zkušeného v oblasti využívání webových aplikací, nebývá problém zorientovat se v navigaci webu, pro hendikepované uživatele to může být mnohdy velký problém. Například pro uživatele s poruchou soustředění může být zorientování se na nepřehledné stránce velice obtížné.

Zásadou přístupného webu je, že navigace je oddělena od obsahu. Navigace by tedy měla být na první pohled identifikovatelná a její odkazy by měly být jasně odlišitelné od odkazů ve zbylé části stránky. Běžným způsobem oddělení navigace od obsahu je její umístění do levého sloupce, nebo na horní navigační lištu. Takováto navigace musí být konzistentní na celém webu (tedy všech jeho podstránkách). Vždy by měl existovat odkaz na hlavní stránku (zpravidla logo webu) a stránku předchozí.

Navigaci je ideální vytvořit za pomocí seznamu, tedy s využitím HTML značek <*ul*> a <*li*>. Nadefinováním CSS stylů k daným tagům je pak možné navigaci předělat do téměř jakékoliv podoby, ale zároveň při vypnutém CSS je navigace zřetelná a prezentována jako jeden celek. Navíc je plně ovladatelná jen za pomoci klávesnice.

Jednotlivé ***položky navigace*** by měly být ***krátké, výstižné a srozumitelné***. Je dobré tedy používat zažitá slova (Kontakt, Odkazy,…) a dodržovat logickou strukturu navigace (hlavní prvky navigace budou před méně důležitými).

Kvůli hlasovým čtečkám je doporučené ***navigaci*** umísťovat ve struktuře zdrojového kódu až ***za hlavní obsah***. V případě že bude navigační část v kódu před obsahem, bude vždy čtečka nejprve zdlouhavě interpretovat jednotlivé navigační odkazy a až poté obsah. Ideálního stavu se dá dosáhnout dvěma způsoby:

1. **Umístění navigace pod obsahovou část s možností přeskočit k ní**
   Pomocí CSS je jednoduché navigaci napozicovat před obsah, takže běžný uživatel nic nepozná. Pokud stránku zpracovává hlasová čtečka, nebo má uživatel vypnuté CSS styly, vložíme před obsah do zdrojového kódu odkaz s kotvou na navigaci a pomocí CSS ho schováme:

   ```
   <a href="#preskocit" class="hidden">Přeskočit na navigaci</a>
   <div class="obsah">
   </div>
   <ul id="preskocit" class="navigace"> …
   ```

   Takto je zajištěno, že čtečka nabídne možnost přejít k navigaci, v opačném případě pokračuje s interpretací obsahu.

2. **Umístění navigace před obsahovou část s možností přeskočit na obsah**
   Jedná se o analogické řešení předchozí varianty, odkaz s kotvou bude před navigací, poté navigační část, následovaná odkazem pro zachycení kotvy a obsahem.

Dalšími pomocnými prvky v navigaci webu, kterou ocení nejen hendikepovaní uživatelé, je tzv. „*drobečková navigace*“ a *mapa webu*.

Rizika spojená s odkazy a jejich řešení

Hypertextové odkazy dělají web webem. Je to jedna z předností, která odlišuje webové aplikace od jiných textových dokumentů. Zároveň jsou odkazy navigačním a ovládacím prvkem. Odkaz by sám o sobě měl vyjadřovat dostatečnou informaci, kam směřuje (odkazuje). Pro lepší pochopení uvedu příklady špatného obsahu:

*Naše nejnovější produkty naleznete **zde**.*

Takovýto odkaz neposkytuje sám o sobě dostatečnou informaci o tom, kam vede. To není vhodné ani z hlediska SEO optimalizace. Optimální řešení takového odkazu je tedy:

*K dispozici je Vám **seznam nejnovějších produktů**.*

Mnohdy se ale hodí a je žádoucí, aby byl odkaz v krátké, zažité formě. Například u e-shopů se často setkáváme s náhledem produktu, jeho stručným popisem a odkazem ***více*** nebo ***více zde***. V těchto případech je vhodné využít atributu ***title.*** Ten slouží jako doplněk textu odkazu. Hlasová čtečka tento atribut interpretuje spolu s odkazem. Příklad využití atributu title ve zmiňovaném e-shopu by tedy byl například takovýto:

```
<a href="produkty/LCD-TV-LG22" title="Více informací o LCD TV LG22"> více zde </a>
```

Takovýto odkaz zároveň zaručuje jedinečnost každého odkazu na stránce, což je dalším bodem přístupnosti odkazů.

V případě obrázkového odkazu, kdy je mezi značkami <a> pouze obrázek, je jako cíl odkazu interpretován atribut *title* obrázku, jak jsem již zmiňoval v předchozí kapitole. Stejně tak i zde připomínám, že ***odkaz by měl být vždy podtržen*** a nespoléhat se jen na jeho grafické odlišení.

4.1.3 Text na webu

Při psaní zdrojového kódu by mělo být vhodně využíváno dostupných značek. Z hlediska přístupnosti je doporučené využívat sémantické značky při psaní obsahu. Se správným psaním obsahu souvisí i správné definování písma.

Sémantické značky a jejich správné používání

Sémantika v rámci zdrojového HTML kódu www stránek má velký význam. Přestože pro grafický prohlížeč stránek (klasický internetový browser) často nemá správný sémantický zápis význam, jelikož se dá obejít použitím CSS deklarací k jednotlivým textovým blokům (nebo naopak je sémantických značek zneužíváno ke stylování nevhodných částí textu), pro textové prohlížeče, hlasové čtečky a v neposlední řadě i vyhledávací roboty má sémantika velký význam.

Z toho lze učinit tyto závěry[29]:

1. Pokud má daný obsah nějaký sémantický význam (je to odstavec, nadpis, citace atd.), označte jej ve zdrojovém kódu příslušnou sémantickou značkou, kterou posléze klidné vizuálně upravte pomocí kaskádových stylů.
2. Nevyznačujte žádný obsah sémantickou značkou, která nevyznačuje jeho význam. Pokud například něco není nadpisem, nemůže to být ve zdrojovém kódu jako nadpis vyznačeno.

Jde tedy o vhodné používání značek pro nadpisy *<h1>* až *<h6>*, značky pro citace *<cite>*, odstavce *<p>*, části kódu *<code>*, seznamy *<ul>*, respektive jejich číslovaná varianta *<ol>*, s prvky *<li>* a mnoho dalších sémantických značek. Důvodem je interpretace hlasovými čtečkami, kdy například značku *<h1>* interpretuje jako „*nadpis úrovně jedna*“. Navíc jsou tyto značky výhodné pro vyhledávací roboty. Například slovu mezi značkami ** je dáván větší důraz, než ostatním slovům.

[29] ŠPINAR, D., Tvoříme přístupné webové stránky, s. 195

Problematika definování velikosti písma

Kvůli mnoha zrakovým vadám dělá takto postiženým uživatelům problém přečíst malá písma na webu. Proto využívá funkci prohlížečů zvětšování textu. Vzhledem k přesně „ušitému“ designu se ale často stává, že písmo je definováno s pevně danou velikostí a tím je funkce zvětšování textu znemožněna.

Velikost písma se dá definovat dvěma způsoby:

1. **Absolutními jednotkami a definicemi**
   Za absolutní jednotky se považují *cm* (centimetr), *mm*(milimetr), *in* (palec), *pt* (obrazový bod), *pc* (pica) a absolutní definice *xx-small, x-small, small, medium, large, x-large* a *xx-large.*
2. **Relativními jednotkami a definicemi**
   Relativními jednotkami jsou *em, ex, %* (procenta) a *px* (pixel). Definicemi pak *smaller* a *larger*.

Pro správnou funkci zvětšování textu je možné použít jen ***relativnách jednotek a definic***. Konkrétně se nabízí definice pomocí jednotky ***em*** (která je definována jako šířka písmena M) a relativních definic ***smaller, larger*** a absolutní definice ***small***. O jednotce px není jednoznačný konsenzus, jde li o jednotku relativní nebo absolutní. Nicméně prohlížeče jí interpretují jako jednotku absolutní a to je pro nás rozhodující.

U písem by se ještě nemělo zapomínat, že ne každý uživatel má nainstalované stejné fonty, jaké použije tvůrce. Pokud pak tvůrce užije jiné písmo například k odlišení informace a uživatel dané písmo nemá nainstalované, tak tuto změnu nevidí. Proto je nutné u definice písem uvádět i její takzvanou rodinu. Tím určíme, jedná-li se o písmo patkové, bezpatkové nebo neproporciální, tedy ***serif, sans-serif a monospace***. V případě, že uživatel daný font nemá k dispozici, použije se font ze stejné rodiny, který dostupný je. Správná definice vypadá například takto:

```
p {font-family: Times New Roman, serif}
```

4.1.4 Formuláře a tabulky

Formuláře jsou jedním z nejdůležitějších prvků webu. Zajišťují interakci webových aplikací mezí uživatelem a tvůrcem webu. Je tak nutné dodržovat základní pravidla při jejich vytváření vzhledem k přístupnosti. Obdobně jsou na tom i tabulky. Od využívání tabulek k rozvržení designu stránky se již většinou upustilo, ale i tak tabulky zůstávají důležitou součástí webu.

Rizika spojená s formuláři a jejich řešení

Riziko nepřístupnosti formuláře je opět spojeno s hlasovou čtečkou. Pokud je formulář špatně nakódovaný, nemusí čtečka interpretovat všechna formulářová pole. Čtečka sice zkouší hledat popisek k poli v jeho bezprostřední blízkosti (nad, vlevo, vpravo) podle typu vstupního pole a druhu čtečky, ne vždy se jí to ale musí podařit. Navíc pokud je formulář nakódován nelineárně, tedy například v levém oddílu jsou všechny popisky a v pravém formulářová pole, tak čtečka nejprve přečte všechny popisky a až poté jednotlivá pole.
K tomu aby čtečky názvy formulářových polí interpretovaly přesně tak, jak jsou zobrazeny všem ostatním, slouží značka ***<label>.*** Tento tag obsahuje atribut ***for*** a ten je identický s atributem ***id*** daného formulářového pole.

Vytvoření vztahů label/ID způsobí, že čtečky obrazovky budou číst správně štítky pro každý ovládací prvek formuláře, bez ohledu na jeho umístění na stránce. Značka <label> byla také vytvořena za tím účelem, aby bylo možné popisky formuláře vložit do kódu a jejich následným použitím (přidáním významu těmto komponentům) dodat formulářovou strukturu.[30]

Korektní zápis takového formuláře může tedy vypadat například takto:

```
<form action="?" id="formular" method="post">
<p>
    <label for="jmeno">Jméno*:</label><br />
    <input type="text" id="jmeno" name="jmeno" />
</p>
```

[30] CEDERHOLM, D., Webdesign s webovými standardy, s. 83

```
<p>
    <label for="souhlas">Souhlasím s obchodními podmínkami:</label><br />
    <input type="checkbox" id="souhlas" name="souhlas" />
</p>
</form>
```

Podobného výsledku lze docílit také použitím atributu ***title*** u značek ***input***, které budou obsahovat popisek daného pole.

Rizika spojená s tabulkami a jejich řešení

Dříve často používané rozvržení stránek (layoutu) pomocí tabulek je již dnes víceméně minulostí. To je svým způsobem zároveň kladný krok k přístupnosti, protože toto rozvržení sebou neslo velká úskalí spojená s linearitou, tedy způsobem, jak čtečky interpretují daný kód. Ta čte tabulku lineárně, tedy po řádcích a to při špatném rozvržení mohlo například obsah vložit mezi navigační prvky. Použitím *divů* a pozicováním pomocí CSS ale tento problém odpadá.

Zůstává však u tabulek užívaných k zobrazení tabulkových dat. Musíme počítat s tím, že tabulka je čtená po řádcích a po jednotlivých buňkách. Například interpretace tabulky 4.1.4.1 bude chybná, protože nejprve přečte všechna města a pak počty obyvatel. Korektnost tak zajišťuje tabulka 4.1.4.2.

Město	Počet obyvatel
Praha	1 200 000
Brno	300 000

Tabulka 4.1.4.1 : Chybně vytvořená tabulka

Město	Počet obyvatel
Praha	1 200 000
Brno	300 000

Tabulka 4.1.4.2 : Správně navržená tabulka

U tabulek nesoucích tabulková data je také nutné uvádět správné označení záhlaví sloupců a řádků. Některé čtečky bohužel s těmito značkami nepracují a jako záhlaví vždy považují prvý řádek, respektive sloupec. Přesto je výhodné tyto značky používat. Jde konkrétně o značky ***<thead>, <tbody>, <tfoot>,*** ale hlavně o značku ***<th>.*** Tato značka definuje hlavičku sloupce (nebo řádku), navíc je ji možno pomocí atributu ***id*** propojit s atributem ***headers*** jednotlivých buněk. To je naprosto ideální pro čtečky obrazovek.

Takto bude čtečka obrazovky moci číst hlavičku tabulky a samotná data v logičtějším pořádku, namísto toho, aby je četla striktně každou řádku zleva doprava, jak by to normálně mohla udělat.[31]

Další používanou značkou je ***<caption>*** pro jméno tabulky. Pro čtečky (a nejen pro ně) naprosto korektní zápis tabulky může tedy vypadat například takto:

```
<table>
<caption>Počet obyvatel v Českých městech</caption>
<tr>
    <th id="mesto">Město</th>
    <th id="pocet">Počet obyvatel</th>
</tr>
<tr>
    <td headers="mesto">Praha</td>
    <td headers="pocet">1 200 000</td>
</tr>
<tr>
    <td headers="mesto">Brno</td>
    <td headers="pocet">300 000</td>
</tr>
</table>
```

[31] CEDERHOLM, D., Webdesign s webovými standardy, s. 54

4.2 Analýza faktorů testování

Testování jednotlivých faktorů by mělo probíhat v průběhu celého vývoje aplikace. Zejména testování použitelnosti se dá využít již v přípravné fázi projektu. Testování použitelnosti je samo o sobě složitý proces a zahrnuje více metod a způsobů interpretace výsledků. Tomuto testování, tedy použitelnosti, se proto dále nevěnuji, a zaměřím se na testování základních faktorů, pro které je k dispozici mnoho nástrojů s podrobnými výsledky.

4.2.1 Validita a funkčnost odkazů

Jde o základní testování každé webové aplikace. Validita zdrojového kódu, jak HTML, tak CSS, by měla být samozřejmostí každé www stránky. Nevalidní kód nemusí nijak překážet jeho funkčnímu zobrazení v grafickém prohlížeči, je ale jedním z předpokladů přístupnosti webové aplikace.

K testování validity je k dispozici mnoho on-line nástrojů. Pro tento účel bohatě poslouží HTML a CSS validátor přímo od konsorcia W3C.

W3C HTML validátor - http://validator.w3.org/
W3C CSS validátor - http://jigsaw.w3.org/css-validator/

Oba tyto validátory fungují na stejném principu. Do vstupního pole zadáme URL adresu stránky, kterou chceme testovat, například *www.abc.cz/kontakt.php* (u CSS souboru buďto cesta k CSS souboru, nebo URL kterékoliv stránky ke které je CSS styl nalinkován) a po stisknutí tlačítka validátor provede kontrolu kódu. Další možností je nahrání souboru přímo na stránku, nebo zkopírování kódu (nebo jeho části) do pole pro přímý vstup. Navíc lze využít rozšířených možností nastavení, kde je možnost zvolit například Document type, CSS level, výstupy testování a další prvky.

Výsledek je přehledně prezentován. Jsou zobrazeny všechny nalezené ***chyby*** v kódu včetně jejich pozice, zobrazení chybného zápisu a návrhu odstranění problému. Tyto chyby musí být opraveny, jinak dokument není validní. Nápověda zpravidla bývá postačující k pochopení odstranění problému.

Druhou kategorií zobrazených chyb jsou ***varování***. Ta jsou zobrazena stejným způsobem jako chyby. Jejich odstranění však není důležité, dokument je tedy validní i bez jejich oprav. Přesto je doporučeno pokusit se odstranit i tyto chyby.

Dalším testovaným faktorem je funkčnost odkazů. Opět máme k dispozici mnoho nástrojů, které nám ulehčí práci. U velmi malých projektů je samozřejmě možné provést testování ručně. Při malém počtu odkazů je to otázka několika mála kliknutí. V opačném případě můžeme využít již zmíněné nástroje. Jelikož jsem již doporučoval validátory od W3C, ani zde neudělám výjimku. Opět pro kontrolu funkčnosti odkazů postačí nástroj Link Checker od zmiňovaného konsorcia.

W3C Link Checker - http://validator.w3.org/checklink

Validátor funguje na stejném principu, jako předchozí dva validátory. Do vstupního pole zadáme URL adresu kontrolované stránky a odešleme ke kontrole. Program projde všechny odkazy, vyskytující se na stránce. Nejen odkazy na jiné weby, ale i odkazy v rámci webu, tedy navigaci a všechny ostatní odkazy využívané k ovládání webu. Navíc zkontroluje i odkazy na grafiku, tedy je-li dostupný soubor uvedený v atributu *src* tagu <*img*>. Všechny nefunkční odkazy pak vypíše, oprava je již snadná.

4.2.2 Testování rychlosti

Na dobu načítání má největší vliv datová velikost stránky, ale důležitá je i struktura odesílaného dokumentu. Proto je žádoucí udávat rozměry elementů s obrázkem a nastavit jim barvu pozadí, aby byl layout funkční i před načtením obrázků, dále pak nevyužívat tabulek k rozvržení stránek atd. Mezi základní možnosti snížení datové velikosti obsahu patří vhodná komprimace grafiky, komprimace textového obsahu a javascriptu pomocí gzip a asynchronní výměna dat se serverem.

V posledních letech se rozšířily i další postupy, jak rychlost načítání zrychlit[32]:

- Obrázky, které jsou součástí grafického designu, spojovat do tzv. „CSS sprites“, což je umístění většího množství obrázků do jediného souboru. Tato technika šetří počet dotazů na server.
- Minimalizovat javascriptový kód pomocí nástrojů jako je Google Closure Compiler nebo YUI Compressor.
- Minimalizovat i CSS kód (odstranění bílých znaků, komentářů a přebytečných středníků).
- Externí javascriptové knihovny, které nejsou důležité pro zobrazení a funkčnost stránky (Google Analytics, Facebook, Skype), načítat asynchronně.

Další doporučovanou metodou urychlení zobrazení stránky je využití kešování (cash). Nastavení kešování a doby platnosti je možno provést zápisem HMTL hlavičky *Expires* a dalšími hlavičkami, jako *Etag* a *Last-modified*. Ke změně hlaviček je zpravidla na hostingu dostupný soubor ***.htaccess***.

K testování rychlosti stránek je opět možné využít několika online nástrojů. Uvedu zde dva zástupce.

Google Page Speed - https://developers.google.com/pagespeed/

Online verze analyzátoru rychlosti od společnosti Google. Opět stačí zadat URL adresu testované stránky a spustit test. Tento nástroj má spíše diagnostický charakter, tudíž nezobrazuje přímo výsledky rychlosti (pouze bodové ohodnocení), ale chyby mající na rychlost vliv. Výsledný test je rozdělen podle priority závažnosti zjištěných chyb. Těchto priorit je 5 – High, Medium, Low, Experimental a Done. Důležité jsou první tři. U každé priority je uveden výčet nalezených chyb a doporučení jejich odstranění.

Pingdom Tools - http://tools.pingdom.com/fpt/

[32] Zdroják.cz: Ještě rychlejší webové stránky [online]. < http://zdrojak.root.cz/clanky/jeste-rychlejsi-webove-stranky/>

Velmi kvalitní nástroj pro měření rychlosti včetně doporučení k jejímu zvýšení. Výsledný přehled je prezentován v úhledném grafickém kabátě. Kromě rychlého přehledu dosaženého skóre, velikosti načítané stránky a času načtení, máte k dispozici podrobnější analýzu. Konkrétně jde o *vodopádové znázornění* načítání stránky, *stupeň výkonu* jednotlivých kategorií, přehlednou *analýzu stránek* a *historii* testování. Z analýzy lze vyčíst všechna potřebná data včetně doporučení pro optimalizaci řešení jednotlivých problémů

4.2.3 Testování přístupnosti

Přístupnost je vždy testována proti nějaké metodice (WCAG, BFW, atd.). To, že je web přístupný podle jedné metodiky, ještě neznamená přístupnost z hlediska ostatních metodik.

K testování přístupnosti můžeme využít v zásadě dvou metod testování. Pro kompletní otestování přístupnosti je potřeba zkombinovat obě tyto metody. Automatické validátory totiž nedokáží odhalit všechny prohřešky proti pravidlům přístupnosti a je potřeba je otestovat jiným způsobem.

První metoda je **ruční kontrola**, kdy se pokoušíme nasimulovat „problematická místa“ a nalézt tak chyby. K usnadnění těchto metod je šikovnou pomůckou využití některých dostupných toolbarů, rozšiřujících funkci internetového prohlížeče.

Konkrétně se jedná o kontrolu funkčnosti[33]:

- při vypnuté grafice,
- bez povoleného skriptování (JavaScripty, Java Applety apod.),
- v textovém prohlížeči (Lynx, Links apod.),
- při ovládání klávesnicí,
- při různých velikostech okna,
- bez barev,
- bez kaskádových stylů atp.

[33] Přístupnost: Testování přístupnosti [online]. < http://pristupnost.nawebu.cz/texty/testovani.php>

Příkladem těchto toolbarů jsou:

Web Developer Toolbar – pro prohlížeč Firefox
Web Accessability Toolbar – pro prohlížeč Internet Explorer

Druhou metodou testování je **automatická kontrola**. K tomu využijeme některý z validátorů, který stránku automaticky prověří vzhledem k vybrané metodice testování. Jedním z nejznámějších validátorů přístupnosti je ***Cynthia Says***. Tento validátor však zatím nenabízí možnost validace podle WCAG 2.0, proto se mu blíže nevěnuji. Nabízím tak dvě alternativy.

WAVE - http://wave.webaim.org/

Další velice známý validátor. Wave nekontroluje stránky vzhledem ke zvolené metodice, ale svým vlastním způsobem, zahrnující jakýsi průřez všemi pravidly. Výstup je zobrazen graficky. K označení problematických elementů je využito barevných značek. ***Červené*** značí chybu přístupnosti, ***zelené*** označují funkce pro usnadnění přístupnosti. Ostatní ikony a ukazatele označují další prvky, na které bychom se měli podívat.

ACchecker - http://achecker.ca/checker/

Po zadání testované stránky si můžeme zvolit metodiku, podle které budete stránku testovat. Mezi nabízenými je *WCAG 1.0* ve třech levelech, totéž pro *WCAG 2.0*, dále *Section 508, BITV 1.0* a *Stanca Act*. Po kontrole validátor vypíše všechny chyby proti přístupnosti dané zvolenou metodikou. Vždy je zobrazena pozice a typ chyby a doporučení k jejímu odstranění.

4.3 Analýza zvolených bezpečnostních rizik

Bezpečnostní rizika webových aplikací jsou spojena hlavně s útoky, využívající nedostatečné zabezpečení zdrojového kódu. Tvůrci webových aplikací by proto měli znát základní typy útoků a metody, jak se proti nim bránit – tedy ošetření zdrojového kódu. Dále popíši vybrané nejčastější útoky na webové aplikace a způsob ochrany proti nim.

Ošetření kódu bude navrhováno pro jazyk ***PHP***, jelikož v něm bude programována aplikace v praktické části práce.

4.3.1 SQL Injection

U webových aplikací se data posílají jako řetězce, tedy posloupnost znaků. V souvislosti na subsystému, do kterého řetězec vstupuje, ale mohou být některé znaky nositelem řídící funkce. Takovému znaku se říká *metaznak*. Příkladem je znak apostrofu (‘). Útočník využívá manipulaci s metaznaky k ovládání subsystému.

Pomocí SQL Injection je útočník schopen měnit nebo zadávat dotazy, které se odesílají do databáze prostřednictvím vstupů do webové aplikace. Samotný útok začíná v okamžiku, kdy program vytváří dotazy založené na řetězcích pocházejících od klienta, a když je následně posílá na databázový server, aniž by ošetřil znaky, které server považuje za speciální.[34]

Příklad:

Typickým příkladem může být přihlašovací formulář. Data odeslaná přes formulář jsou načtena do proměnných a poté kontrolována na shodu v databázi.

```
SELECT count(*) FROM uzivatele WHERE jmeno='$jmeno' AND heslo='$heslo';
```

Konkrétní dotaz pak může vypadat takto:

```
SELECT count(*) FROM uzivatele WHERE jmeno='admin' AND heslo='1847abc';
```

Pokud do pole pro heslo vložíme zápis ***‘ OR 1=1 --*** , podívejme se, co se stane s dotazem:

```
SELECT count(*) FROM uzivatele WHERE jmeno='admin' AND heslo=' ' OR 1=1 –';
```

Tedy kontrola hesla je splněna, pokud je jméno „admin“ a heslo prázdné, nebo 1=1. Druhá podmínka je vždy splněna, tudíž je nalezena shoda a jsme přihlášeni bez znalosti hesla.

[34] HUSEBY, SVERE H., Zranitelný kód, s. 32

Znak dvou minus (--) zajišťuje, že vše následující je bráno jako komentář a není tak zobrazena chybová hláška.

Dalším příkladem může být vložení řetězce ***' OR 1=1; DROP TABLE uzivatele;*** **--**. Dotaz je pak ve tvaru:

```
SELECT count(*) FROM uzivatele WHERE jmeno='admin' AND heslo=' ' OR 1=1; DROP TABLE uzivatele; --
```

Takovýto příkaz nám dokonce odstraní celou tabulku *uzivatele*, pokud útočník zná její název, což si může opět zjistit útokem, nebo se pokusit název odhadnout.
Podobných útoků na stejném principu je mnoho.

Ochrana

Ochrana proti SQL Injection je velmi jednoduchá. Problém je v tomto případě **apostrof.** Aby se apostrof do databáze uložil jako apostrof, a nebyl interpretován jako znak uvozující konec řetězce, muselo by se před apostrof umístit zpětné lomítko (\), z angličtiny známé jako **backslash**. Tato akce se nazývá **escapování.**[35]

K přidání zpětného lomítka se v PHP používá funkce ***mysql_real_escape_string()***. Ta ošetří všechny problémové znaky a vrátí bezpečnou podobu. Zpětné lomítko se nezapisuje do databáze, je součástí řetězce jen po dobu zpracování.

Nevyplatí se spoléhat na konstantu magic_quotes_gps, kterou je možno aktivovat v nastavení PHP (v 6 verzi již není k dispozici), která automaticky přidává zpětné lomítko všude před problémové znaky. To je mnohdy nežádoucí. Řešení za pomoci funkce mysql_real_escape_string() je mnohem lepší.

Bezpečný zápis dotazu by tedy vypadal takto:

```
SELECT count(*) FROM uzivatele
```

[35] Zdeněk Večeřa: Jak na to: SQL injection, magic_quotes_gpc, addslashes() a stripslashes() [online]. < http://blog.zdenekvecera.cz/item/jak-na-to-sql-injection-magic_quotes_gpc-addslashes-a-stripslashes>

```
WHERE jmeno='".mysql_real_escape_string($jmeno)."'
AND heslo='".mysql_real_escape_string($heslo)."';
```

4.3.2 Cross Site Scripting

Jedna z nejstarších a nejběžnějších zranitelností webu. Běžně je tento typ útoků označován zkratkou XSS. Jedná se o injektování škodlivého kódu na straně klienta. Nejčastěji se využívá „podstrčení " javasriptového kódu. Důsledkem útoků XSS je například změna uživatelských nastavení, krádež a otrava cookies, klamavá inzerce a další.
XSS využívá toho, že je běžně možné odesílat ve formulářovém poli, nebo rovnou v URL adrese stránky, řetězce obsahující speciální znaky jazyka HTML (<, >, &, ", ',).

Příklad

Příkladem může být návštěvní kniha. Pokud nejsou data odesílaná formulářem ošetřena před vložením do databáze, škodlivý kód je uložen a při každém zobrazení návštěvní knihy načten do obsahu stránky a vykonán.
Pokud do pole pro textovou zprávu vložíme například tento kód:

```
<script>
        for (x=0; x<5000; x++)
        window.open ("http://www.abc.cz/");
</script>
```

Tak po načtení stránky se spustí skript, který 5000x otevře okno s novou stránkou. Podvrhnout ale můžeme i skript, který se vyskytuje na jiné (obvykle útočníkovo) stránce a ten se poté spustí a vykoná.

Ochrana

Ochrana proti XSS je opět velmi jednoduchá. Stačí ošetřit všechny speciální znaky jazyka HTML. K tomu nám v PHP poslouží funkce ***htmlspecialchars()***. Tato funkce přemění speciální znaky na HTML entity (< na *<* a podobně), což je postačující k ochraně proti cross-site scripting. Při použití konstanty ENT_QUOTES jsou navíc ošetřeny i apostrofy.

Při načítání vstupu od uživatele je tedy potřeba vstup ošetřit tímto způsobem:

```
<input name='jmeno' value=' <?php echo htmlspecialchars($_POST['jmeno'], ENT_QUOTES); ?> ' />
```

4.3.3 Krádež relace

Úzce souvisí s XSS. Pomocí cross-site scripting může totiž útočník napadenému uživateli odcizit jeho *cookie* soubor. Při první návštěvě webu je uživateli přiděleno SESSION ID. Při dalším navštívení je SESSION ID posíláno na server zpět pro kontrolu ve formě cookie. Pokud útočník získá identifikátor sezení (odcizení session id), může ho využít a předstírat korektně přihlášeného uživatele.

Ochranou je tak v prvé řadě ochrana proti XSS. Dále je důležité identifikátory relace generovat náhodně a navíc ***kontrolovat uživatele doplňkovými způsoby*** (IP adresa, User-Agent). Další možností je nastavení ***HttpOnly*** u cookie souboru. Doporučeno je také používat ukončení relace se serverem pomocí ***přímého odhlášení*** na stránce (nabídnout funkci odhlásit, která ukončí relaci).

Druhou možností útoku není odcizení, ale podstrčení SESSION ID. Útočník pak jen čeká, až se uživatel přihlásí a poté využije tento identifikátor pro sebe. Ochrana je jednoduchá, stačí při přihlášení uživatele použít PHP funkci ***session_regenerate_id***, což zajistí změnu identifikátoru relace.

4.3.4 Cross Site Request Forgery

CSRF je typ útoku na webovou aplikaci nebo službu. Zkratka CSRF (Cross-Site Request Forgery) znamená „podvržení požadavku mezi různými stránkami“. Někdy se také můžeme setkat i s dalšími termíny jako XSRF, Cross-Site Reference Forgery, Session Riding nebo Confused Deputy attacks.[36]

[36] Zdroják.cz: Co je Cross-Site Request Forgery a jak se mu bránit [online]. < http://zdrojak.root.cz/clanky/co-je-cross-site-request-forgery-a-jak-se-branit/>

Příklad

Předpokladem útoku je, že útočník zná aplikaci, na kterou útočí. Zpravidla se jedná o formulář, jemuž útočník podvrhne data ke zpracování. Nejjednodušší je, pokud aplikace odesílá data metodou ***GET***. Zde stačí, aby útočník podvrhl stránku se škodlivým kódem uživateli, který je aktuálně přihlášen k napadené aplikaci. Například, pokud je uživatel přihlášen k administrační sekci webu, která maže články na základě formulářem předaných parametrů v podobě:

```
www.abc.cz/admin/clanky.php?smazat=ok&clanekid=35
```

pak stačí, aby si napadený zobrazil podtrčenou stránku útočníka (například www.attack.cz/utok.html), kde bude umístěn falešný obrázek v této podobě:

```
<img src="http:// www.abc.cz/admin/clanky.php?smazat=ok&clanekid=35">
```

Uživatel, který si otevře útočnou stránku v momentě, kdy je přihlášen k admin sekci na webu „abc.cz“, nevědomě odešle požadavek ze stránky „attack.cz“ na vykonání výše zmíněné operace, která se úspěšně provede, jelikož uživatel je ověřeně přihlášen.

Pokud je formulář odesílán metodou ***POST***, je situace pro útočníka poněkud obtížnější. Útočník musí znát strukturu napadeného formuláře a na útočné stránce podstrčit celý formulář automaticky odeslaný pomocí javascriptu. Místo obrázku by na útočné stránce byl následující kód:

```
<body onload="document.forms[0].submit();">
  ...
  <form action="http://www.abc.cz/ admin/clanky.php" method="post">
    <input type="text" name="smazat" value="ok">
    <input type="text" name="clanekid" value="35">
  </form>
```

Výsledek by byl stejný, jako u varianty s metodou GET.

Ochrana

Základní obranou je zodpovědné chování uživatele. Pokud je například přihlášen do internet bankingu, neměl by otevírat žádná další okna (ani záložky) prohlížeče, dokud se z aplikace neodhlásí.

Další ochranou je využití tzv. „podepsaného“ formuláře. To je zajištěno vygenerováním náhodného ***tokenu***, který je přidán serverovou částí aplikace do formuláře. Stejný token se uloží i do SESSION proměnné a před zpracováním formuláře se kontroluje shoda tokenů, tedy shoda mezi hodnotou SESSION a hodnotou formulářem odeslaného tokenu. Žádoucí je, aby se při každém načtení formuláře generoval nový token.

Zabezpečený formulář by tedy vypadal takto:

```
<form action="http://www.abc.cz/ admin/clanky.php" method="post">
  <input type="text" name="clanekid" value="">
  <input type="hidden" name="token" value="0123456789aBcD">
  <input type="submit" name="smazat" value="ok">
 </form>
```

5 PRAKTICKÉ ŘEŠENÍ

V této části uvedu všechna zanalyzovaná pravidla přístupnosti a bezpečnosti webových aplikací do praxe. Jsou aplikována na webové stránky tréninkové skupiny Milana Kováře, oddílu atletiky TJ Dukla Praha, jejíž jsem členem. Tento web od počátku programuji a spravuji. Stránky jsou dostupné na adrese http://www.mkteam.cz.

5.1 Popis webové aplikace

Stránky prošly již 3 rekonstrukcemi. První verze, spuštěná v roce 2007, byla o rok později obohacena o administrační sekci, zajišťující editaci aktualit a fotogalerií. V roce 2009 byl proveden redesign a poslední úprava proběhla v roce 2011, z důvodu útoku na hosting kapusta.cz, kde byly stránky hostovány. S přesunem na nový hosting a velkou ztrátou dat byl spojen opět kompletní redesign, a vytvořena zbrusu nová administrační sekce, zajišťující plnou editovatelnost veškerého obsahu. Stránky již byly zabezpečeny proti SQL Injection a XXS, změny spojené s diplomovou prací se proto týkají převážně přístupnosti, jelikož ta byla i mnou opomíjena (jedinou podmínkou byla kompatibilita v prohlížečích).

Aplikace je programována v jazyce PHP 5, XHTML 1.1, CSS3 a databázi MySql. Struktura obsahuje 5 hlavních částí:

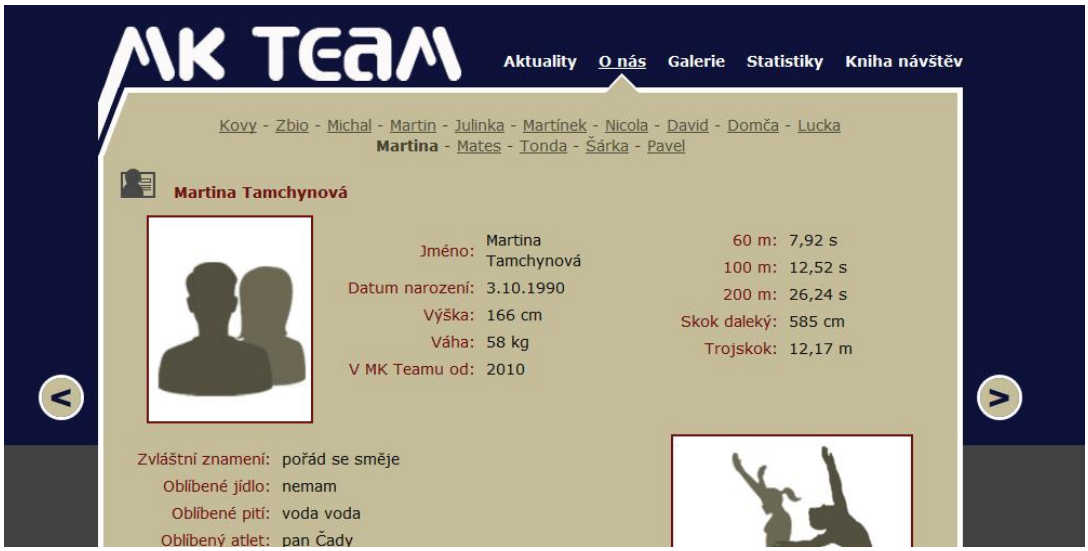
Aktuality – aktuální novinky z dění tréninkové skupiny
O nás – seznam a popis všech aktivních členů skupiny
Galerie – fotogalerie a videogalerie
Statistiky – skupinové tabulky vybraných disciplín
Kniha návštěv – návštěvní kniha otevřená všem návštěvníkům

Dále stránky obsahují ***administrační sekci*** se zabezpečeným přístupem. V té je možno přidávat, editovat a mazat aktuality, členy skupiny, foto a videogalerie, statistiky. Dále je možno mazání nežádoucích příspěvků v knize návštěv.

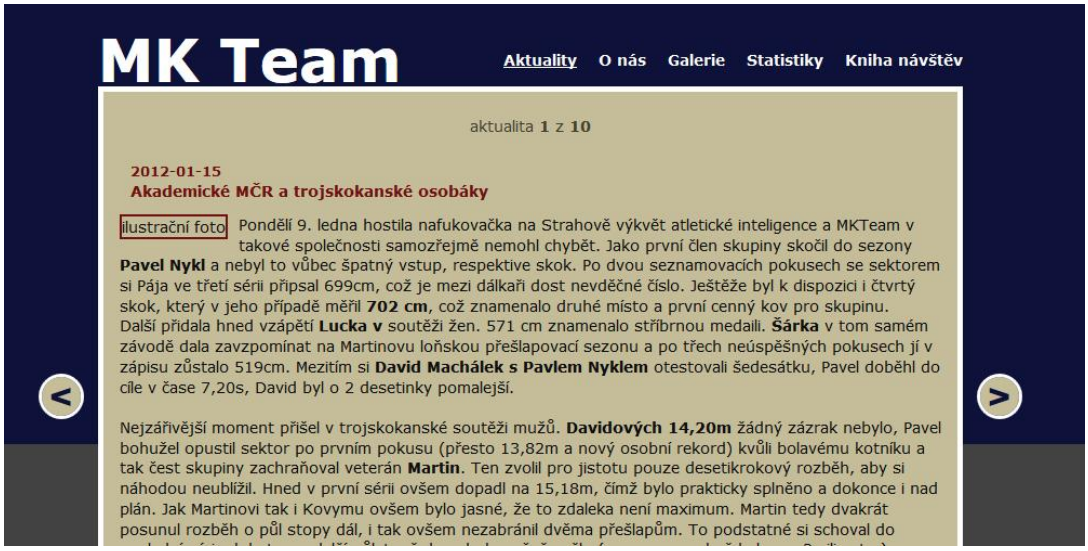
Bližší popis stránek bude zřejmý z dalších kapitol. Popíši postupy týkající se přístupnosti a bezpečnosti, nakonec stránky otestuji a odladím podle případných doporučení. U bezpečnostních prvků předvedu útok na nezabezpečený kód a poté funkční ochranu.

5.2 Aplikace pravidel přístupnosti

V této kapitole popíši nejdůležitější prvky týkající se přístupnosti v souvislosti s popisovanou webovou aplikací.



Obr 5.2.1 – zapnuté zobrazení obrázků



Obr 5.2.2 – vypnuté zobrazení obrázků

Logo MK Team je provedeno graficky, jak je vidět na obrázku 5.2.1, tag *img* má přidělen atribut *alt* s hodnotou „MK Team“. Navíc je obrázku pomocí CSS nadefinován font podobné velikosti, jako má obrázek loga, takže při vypnutých obrázcích je logo stránky zobrazeno textově (*alt* atribut), jak je vidět na obrázku 5.2.2 (není funkční v IE).

HTML:

```
<a href="http://www.mkteam.cz/" title="Návrat na titulní stránku">
        <img src="img/logo.png" alt="MK Team" class="logo" />
</a>
```

CSS:

```
.logo {width: 338px; height: 123px; position: absolute; z-index: 5; font-weight: bold; font-size: 3.8em; color: white; border: none }
```

Hlavní menu stránky je řešeno jako seznam, s nastylováním do výsledné podoby. Tento způsob je doporučován, kvůli přehlednému zobrazení menu při vypnutých stylech. Odkaz s třídou „active“ má nadefinovanou šipku jako obrázek na pozadí, značící aktuálně zobrazenou stránku menu, jak je vidět na obrázku 5.2.1 u sekce „O nás“. Není zde dodržena podmínka podtrženého odkazu, jelikož se jedná o hlavní menu a je tedy zřejmé, že se o odkazy jedná.

Naopak symbol podtržení má v hlavním menu funkci aktivního zobrazení stránky, pro případ, že jsou vypnuty obrázky a není zobrazena „šipka“. Toto zobrazení je na obrázku 5.2.2.

Dle doporučení pravidel přístupnosti by mělo být hlavní menu pod obsahem, nebo umožňovat přeskočit menu. To vše z důvodu nežádoucí opětovné interpretace hlavního menu hlasovými čtečkami při zobrazení další podstránky. To je zde řešeno odkazem nad hlavním menu s kotvou k obsahu stránky. Podobné kotvy jsou využity i na jiných stránkách, kde je možnost přeskočit například podmenu se seznamem členů atd.

Tyto odkazy mají přidělenu třídu „hidden“, což zajišťuje „zmizení“ odkazu při zapnutých stylech. Třída „hidden“ je zde využita vícekrát. Jednak pro skrytí čar (<*hr*>) a dále pro skrytí nadpisů, které mají význam jen při zobrazení bez CSS stylů (např. *Hlavní menu*).

Menu má nastavenou šířku 1000px a absolutní pozici. Blok seznamu tak přesahuje přes pravý okraj vymezený pro zobrazení obsahu celého webu. To je z důvodu kompatibility při zvětšeném textu. Jednotlivé položky menu se tak stále zobrazují v jedné řádce. Při definici např. 400px, by se při zvětšení textu položka „kniha návštěv“ řadila na druhou řádku, jelikož by textu nestačil vymezený prostor šířkou bloku.

HTML:

```
<a href="#obsah" title="obsah" class="hidden">Přeskočit k obsahu</a>
<hr class="hidden" />
<h2 class="hidden">Hlavní navigace</h2>
<ul class="menu">
        <li><a href="index.php" title="Aktuality" class="active">Aktuality</a></li>
        <li><a href="o-nas.php" title="O nás">O nás</a></li>
        <li><a href="galerie.php" title="Galerie">Galerie</a></li>
        <li><a href="statistiky.php" title="Statistiky">Statistiky</a></li>
        <li><a href="kniha-navstev.php" title="Kniha návštěv">Kniha návštěv</a></li>
</ul>
```

CSS:

```
.menu { width: 1000px; height: 60px; position: absolute; top: 0; left: 0; margin: 65px 0px 0px 420px; z-index: 5; font-weight: bold }
.menu li {margin: 0; padding: 0; display: inline; list-style: none}
.menu a {float: left;  height: 60px; font-size: 1em ; color: white;  text-decoration: none; text-align: center; margin: 0px 10px; line-height: 60px;}
.menu a.active {background: url("img/active.png") bottom center no-repeat; text-decoration: underline }
.menu a:hover {text-decoration: underline}
.hidden  {position: absolute; margin-top: -1500px}
```

Dalším ovládacím prvkem stránek je „listování“ **vlevo a vpravo** (neboli předchozí a další), které je vždy platné pro aktuální stránku. U „aktualit“ tak listuje v aktualitách, v sekci „o nás“ mezi jednotlivými členy, v „galerii“ mezi roky, při zobrazení galerie pak mezi galeriemi daného roku apod. Nápovědou pro funkci navigace je první ***informační řádek*** pod hlavním menu, který udává aktuální pozici mezi všemi podstránkami (obrázek 5.2.2), nebo navíc slouží i jako aktivní menu druhé úrovně (obrázek 5.2.1).

K zobrazení tohoto ovládacího prvku je využito deklarace CSS3, konkrétně zaoblení rohů, pro dosažení vzhledu kolečka. Původně bylo toto řešeno graficky obrázkovým podkladem, avšak při vypnutém zobrazení obrázků nebylo vidět šipky („<“ a „>“) pro shodnou barvu s pozadím. Proto bylo zvoleno toto řešení, u prohlížečů nepodporujících CSS3 se prvek zobrazí jako čtvereček, což nijak nezamezuje funkčnosti.
Tyto ovládací prvky jsou ve struktuře stránky umístěny pod obsahem a na pozici po informačním řádku je odkaz s kotvou k tomuto ovládání. Pomocí CSS jsou pak oba prvky umístěny na svoji pozici v grafickém zobrazení.

HTML+ PHP:

```
<h2 class="hidden" id="nav">Vedlejší navigace</h2>
<a href="?aktualita=<?php echo $prev;?>" title="předchozí aktualita"
        class="arrow_left">&lt;</a>
<a href="?aktualita=<?php echo $next;?>" title="další aktualita"
        class="arrow_right">&gt;</a>
```

CSS:

```
.arrow_left , .arrow_right {
width: 36px; height: 36px; background: #c6be9a; text-align: center; position: absolute; z-index: 20; border: 3px solid #fff; font: bold 2.5em Impact; color: #0e123a; text-decoration: none; line-height: 36px; border-radius: 21px;  -moz-border-radius: 21px; -webkit-border-radius: 21px; }
.arrow_left {left: 0px; top: 0px; margin: 260px 0px 0px -60px}
.arrow_right {right: 0px; top: 0px; margin: 260px -60px 0px 0px}
.arrow_left:hover, .arrow_right:hover {color:#701112}
```

V sekcích „**O nás“** a **„Statistiky“** je rozvržení obsahu řešeno **tabulkami.** Tabulky jsou optimalizovány z hlediska přístupnosti. Je tedy využito hlaviček <*th*> s identifikátory „id“, které jsou svázány s atributy „headers“ jednotlivých buněk <*td*>. Hlasová čtečka by tak měla tabulku číst způsobem: Jméno – Michal Čada, Výkon Lomeno Vítr – 770 Lomeno plus 2,0, Datum – 2007-09-01, Místo – Jablonec. A další řádek obdobně. U tabulky obsahující statistiky je navíc využito prvků <*thead*> pro definování vzhledu hlavičky a <*tbody*> pro definici těla tabulky. Obsah tabulky je generován z databáze. Pro lepší čitelnost je navíc každý druhý řádek odlišen barvou podkladu. Zdrojový kód tabulky statistiky tedy vypadá následovně.

HTML + PHP:

```
<table class="stats">
        <thead>
                <tr>
                        <th id="jmeno">Jméno</th>
                        <th id="vykon">Výkon / vítr</th>
                        <th id="datum">Datum</th>
                        <th id="misto">Místo</th>
                </tr>
        </thead>
        <tbody>
        <?php  while ($rowt = mysql_fetch_array($dotaz)) {
                        echo '<tr class="'.(++$m % 2 ? "stats_licha" : "stats_suda").'">
                                <td headers="jmeno">'.$rowt["jmeno"].'</td>
                                <td headers="vykon">
                                        '.$rowt["vykon"].' / '.$rowt["vitr"].'
                                </td>
                                <td headers="datum">'.$rowt["datum"].'</td>
                                <td headers="misto">'.$rowt["misto"].'</td>
                               </tr>';  }
        ?>
        </tbody>
</table>
```

CSS:

```
.stats {float: left; text-align: center; table-layout: fixed; border-collapse: collapse; border: 0}
.stats thead { background: #0e113a; color: white; text-align: center; font-weight: bold}
.stats tbody {color: #0f0f0f}
.stats td {height: 25px; width: 190px; color: #0f0f0f}
.stats_suda {background: #afa987}
```

Kniha návštěv obsahuje **formulář** pro odesílání příspěvků. U formulářů je použit tag <*legend*> pro pojmenování formuláře (a je skryt, tedy zobrazuje se jen při vypnutých CSS stylech) a tag <*label*> pro označení popisků formulářových polí. Ty jsou atributem „for“ provázány s atributem „id“ příslušných <*input*> polí, což opět zajišťuje správnou interpretaci čtečkami.
U formuláře je použito zabezpečení proti spamovacím robotům. To je zajištěno počítacím příkladem, navíc popsaným slovně, aby byl pro spamovací roboty nečitelný. Jelikož roboti neumějí číst javascript, je do formuláře přidáno za pomocí javascriptu skryté pole se správnou výslednou hodnotou kontrolního příkladu (7) a pomocí <*noscript*> je pole s schováno. Uživatel, který má povolený javascript tak nemusí kontrolní pole vyplňovat, protože kontrolní hodnota je odeslána automaticky. Uživatelé s vypnutým javascriptem vyplní zobrazené pole s početním příkladem.

HTML + JavaScript:

```
<form action="?send=ok" method="post" name="form" class="formk">
 <fieldset style="border:none">
  <legend class="hidden">Přidat příspěvek</legend>
  <p> <label for="jmeno">Jméno: </label><br />
  <input id="jmeno" name="jmeno" type="text"  maxlength="40" class="name" /></p>
  <p> <label for="text">Text: </label><br />
  <textarea id="text" name="text" rows="5" cols="5" class="mess"></textarea></p>
  <noscript>
   <p> <label for="spocist">devět mínus dva: </label>
   <input id="spocist" name="robot" size="2" /></p>
  </noscript>
  <script type="text/javascript">
```

```
    document.write('<input type="hidden" name="robot" value="7" />');
  </script>
  <p><input type="submit" value="Odeslat" id="send" name="send" /></p>
 </fieldset>
</form>
```

CSS:

```
.formk { width: 180px; float: left; color: #701112}
.formk label {font-weight: bold}
.name, .mess {width: 150px; border: 2px solid #701112; background-color: #f3ebc7;
margin: 3px 0px 3px 0px}
.mess {height: 300px; overflow: auto}
```

Dále jsou stránky optimalizovány i pro další pravidla přístupnosti. Při zapnuté funkci **vysoký kontrast** je vše čitelné. Pro správné zobrazení barvy odkazů se musí provést úprava nastavení v prohlížeči (v našem případě bílé odkazy). Na stránkách je možné využít funkci **zvětšení textu**, jelikož všechen text je v CSS nadefinován relativní jednotkou *em*. Stránky jsou plně přístupné i při **vypnutých CSS** stylech. Kombinace všech tří předchozích bodů je vidět na obrázku 5.2.3.

Obr 5.2.3 – vypnuté CSS styly, zapnutá funkce Vysoký kontrast, zvětšený text

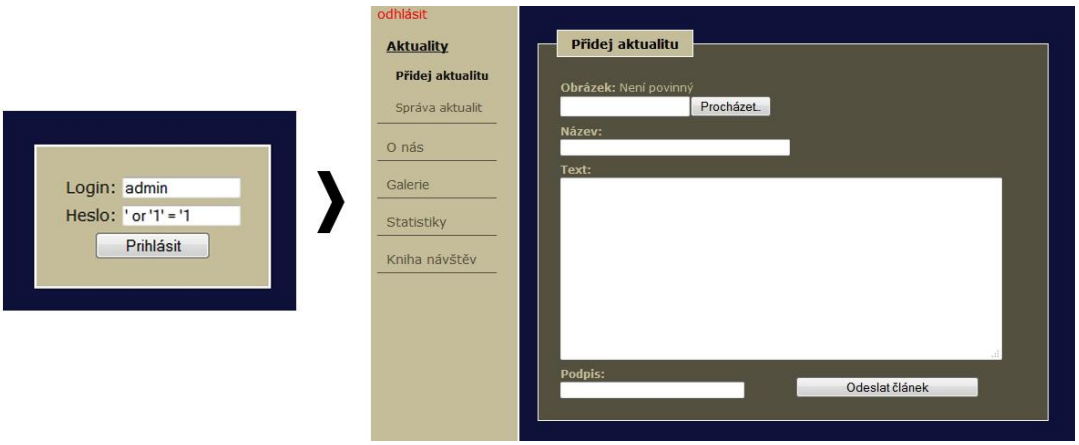
5.3 Aplikace pravidel bezpečnosti

Stránky jsou zabezpečeny proti nejčastějším útokům. Dále podrobněji popíši útok danou metodou při nezabezpečení a následně zabezpečení kódu a opakování útoku.

5.3.1 SQL Injection

Proti SQL Injection jsou ošetřeny všechny vstupy od uživatele, které provádějí výběr dat z databáze. Na většině stránek je hodnota předávána v parametru URL. Výjimku tvoří přidání příspěvku do knihy návštěv a přihlášení do administrační sekce, kde je formulář odesílán metodou POST. Ukázku útoku předvedu na vstupu do administrační sekce, kde je při nedostatečném ošetření kódu možno obejít přihlášení.

Útok



Obr 5.3.1.1 – Úspěšný útok SQl Injection

Zdrojový kód pro zpracování přihlašovacího formuláře je tento:

```
$login = $_POST["login"];
$pass = $_POST["pass"];
$logink = mysql_result(mysql_query("SELECT id FROM admin WHERE login='".$login."'
AND password='".$pass."' "), 0);
if ($logink != " ")
```

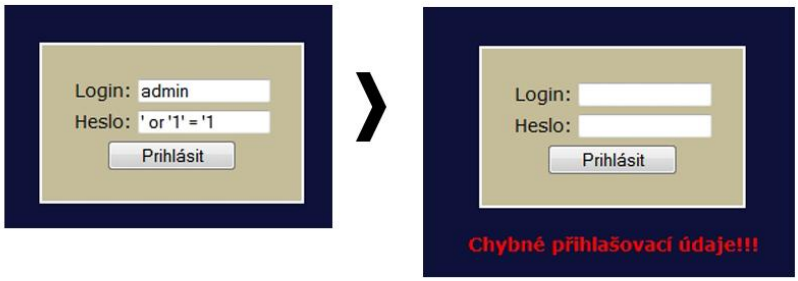
```
{
        $_SESSION["login"] = $login;
        header("Location: aktuality.php");
        ob_flush();
}
```

Pokud do přihlašovacího formuláře vyplníme jako login například „***admin***“ a do pole pro heslo „***' or '1' = '1*** “, přihlášení proběhne úspěšně, jelikož podmínka bude vždy splněna. Úspěšný útok vidíme na obrázku 5.3.1.1.

Dotaz bude vypadat takto:

```
$logink = mysql_result(mysql_query("SELECT id FROM admin WHERE login='admin' AND password=' ' OR ‘1‘=‘1‘ "), 0);
```

<u>Ochrana</u>



Obr 5.3.1.2 – Neúspěšný útok SQL Injection – ošetřený kód

Postačující ochranou je ošetření vstupů do databáze. V jazyce PHP k tomu slouží funkce ***mysql_real_escape_string.***

Ošetřený kód bude vypadat takto:

```
$logink = mysql_result(mysql_query("SELECT id FROM admin WHERE
login='".mysql_real_escape_string($login)."' AND
password='".mysql_real_escape_string($pass)."' "), 0);
```

Pokus o SQL Injection bude v takovémto případě neúspěšný, jak je patrné na obrázku 5.3.1.2. Stejnou metodou jsou ošetřena i data na ostatních stránkách webu, pokud je to žádoucí.

5.3.2 Cross-site Scripting

Ohroženy útokem XSS jsou potenciálně pouze stránky statistiky, kde se vypisuje parametr „*disciplina*“ přímo do stránky a pak samozřejmě návštěvní kniha, kde je případně možnost skript uložit do databáze a při každém načtení se tak spustí.

Útok:

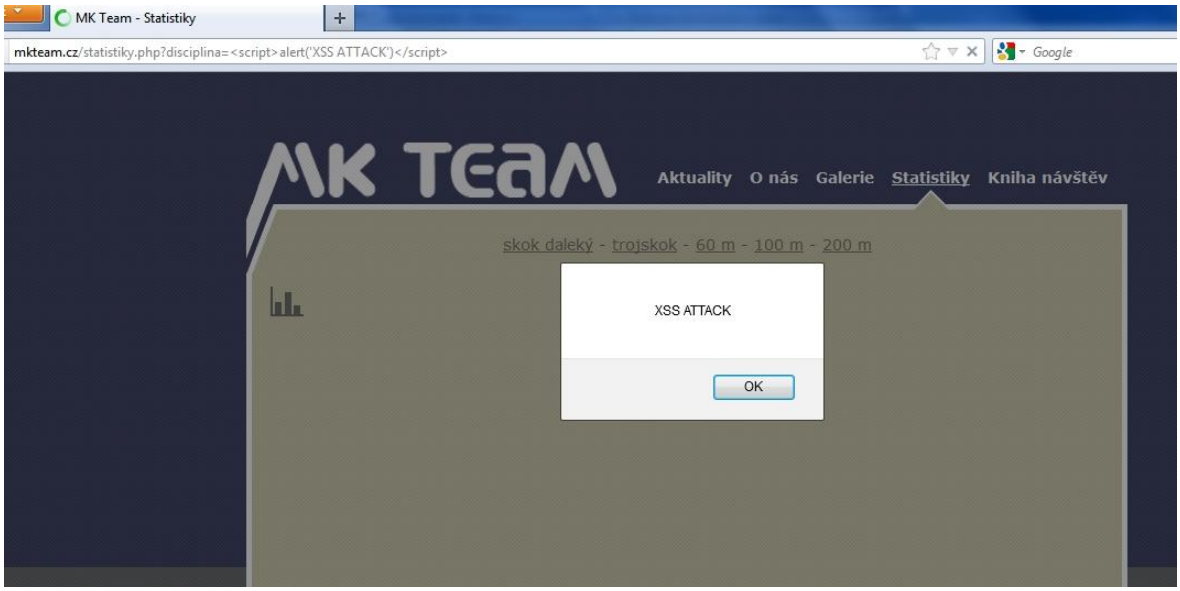
Útok předvedu na neošetřené stránce statistiky.

Část kódu stránky statistiky.php, zpracovávající výběr disciplíny pro zobrazení:

```
$disc = 'skok daleký';
if ($_GET["disciplina"] != " ") $disc = $_GET["disciplina"];

…
echo '<br /><h1>'.$disc.'</h1>';
…
```

Nyní do URL parametru „disciplina“ vložíme útočný skript následujícím způsobem:

http://www.mkteam.cz/statistiky.php?disciplina=<script>alert('XSS ATTACK')</script>



Obr 5.3.2.1 – Úspěšný útok XSS

Script se zapíše do stránky a vykoná se, jak je patrné na obrázku 5.3.2.1. V PHP kódu je totiž hodnota parametru využita, kromě dotazu SQL, k zobrazení názvu zvolené disciplíny, jak je vidět na předchozím kódu.

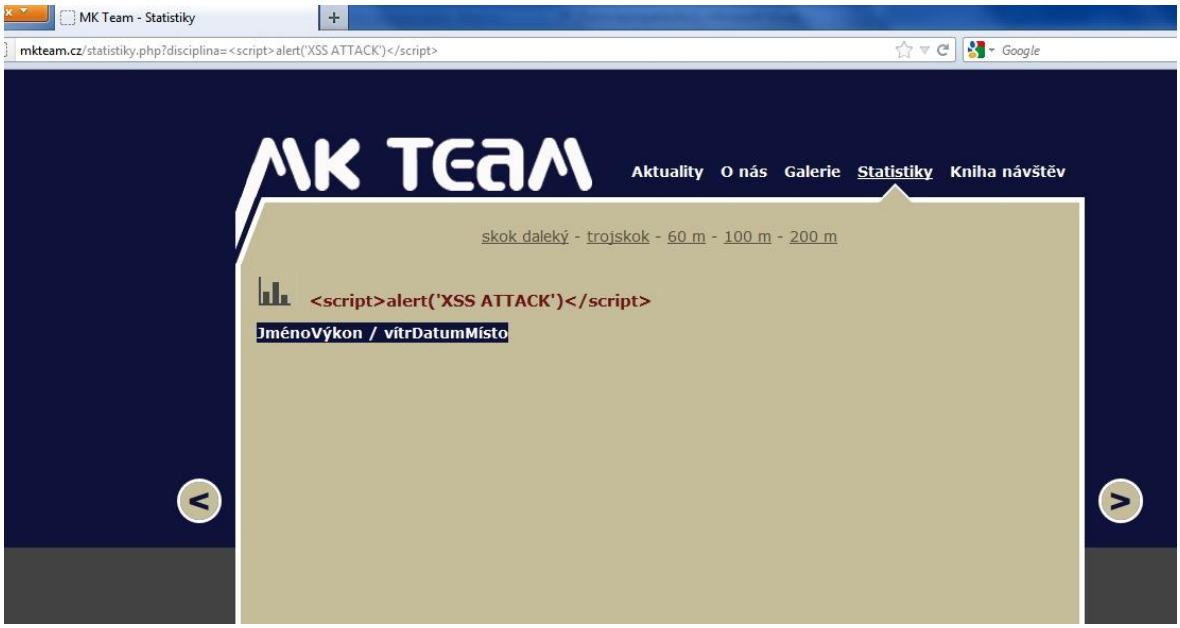
Podobně můžeme napadnout i návštěvní knihu, kdy script odešleme v textu zprávy.

Ochrana:

Ošetření zdrojového kódu je velice jednoduché, stačí zpracovávaný parametr „disciplina“ ošetřit funkcí ***htmlspecialchars***.

```
$disc = htmlspecialchars($_GET["disciplina"], ENT_QUOTES);
```

To zajistí převod HTML znaků na entity a pokus o útok bude neúspěšný, jak je vidět na obrázku 5.3.2.2. Stejným způsobem jsou ošetřena i data odesílaná v knize návštěv.



Obr 5.3.2.2 – Neúspěšný útok XSS – ošetřený kód

5.3.3 Krádež relace

„Sezení“ se využívá pouze u admin sekce. Stránky jsou zabezpečeny proti útoku XSS, což znesnadňuje krádež cookie. Dále je rozšířena kontrola přihlášení do admin sekce z příkladu SQL Injection o další kontrolní prvek. Při úspěšném přihlášení se do databáze k uživateli uloží kontrolní řetězec, obsahující *User Agent* data (data o systému a prohlížeči) a *IP* adresu:

```
if ($logink != " ")
{
$useragent = $_SERVER['HTTP_USER_AGENT'];
$ip = $_SERVER['REMOTE_ADDR'];
$kontrola = $useragent.$ip;
$_SESSION["login"] = $login; /*uložení hodnoty z databáze do session proměnné*/
$uloz = mysql_query("UPDATE admin SET kontrola='".$kontrola."' WHERE
login='".mysql_real_escape_string($login)."' ");
header("Location: aktuality.php");
ob_flush();
}
```

Shoda dat v databázi s daty aktuálně přihlášeného uživatele se poté kontroluje na každé stránce administrační sekce:

```
$useragent = $_SERVER['HTTP_USER_AGENT'];
$ip = $_SERVER['REMOTE_ADDR'];
$kontrola = $useragent.$ip;
$kontrolak = mysql_result(mysql_query("SELECT kontrola FROM admin WHERE
login='".mysql_real_escape_string($_SESSION["login"])."' "), 0);

If ($_SESSION["login"] == " " or $kontrolak != $kontrola)
{
        ob_end_clean();
        header("Location: index.php");
        ob_end_flush();
}
```

Tím je částečně zajištěna ochrana při odcizení session id, ikdyž útočník si samozřejmě může User agenta i IP adresu napadeného zjistit.

Další formou ochrany je odhlášení z admin sekce příslušným odkazem, který smaže hodnotu SESSION[„login“] a díky tomu je opět provedeno přesměrování na index stránku:

```
<a href="?odhlasit=ok" title="" class="odhlasit">odhlásit</a>
…
If ( $_GET["odhlasit"] == "ok") unset($_SESSION["login"]);
…
```

Proti možnosti podstrčení session id jsou stránky ošetřeny funkcí ***session_regenerate_id***, která mění identifikátor relace při každém přihlášení uživatele do admin sekce.

5.3.4 Cross-site regest forgery

Útok CSRF je na stránky těžko proveditelný. Jediná část webu, kde by mohl útočník takto škodit, je opět admin sekce. Při úspěšném útoku by například mohl smazat článek, člena apod.

Všechny formuláře v administrační sekci editující obsah jsou odesílány metodou POST, takže by útočník musel znát strukturu formuláře, aby ho mohl javascriptem podstrčit a odeslat.

Přesto všechno je vytvořena jednoduchá ochrana proti útoku CSRF. Spočívá ve vygenerování jedinečného ***tokenu*** při každém zobrazení stránky s formulářem. Tento token je vložen do formulářového pole *hidden*. Ten samý token je uložen do SESSION proměnné. Při odeslání formuláře jsou před zpracováním oba tokeny porovnány a zároveň je vygenerován nový token pro nově zobrazený formulář. Pro lepší představu uvedu ukázku kódu:

```
$random = (rand() * rand() / rand() + rand()) . rand(); /*vygenerování nového tokenu*/
$_SESSION["token"] = $random;
<!--  vložení tokenu do formuláře  //-->
….
<input type="hidden" name="token" value="<?php echo $random; ?>" />
….
```

Před zpracováním formuláře proběhne kontrola (mimo jiné) na shodu $_POST["token"] == $_SESSION["token"]. Touto kontrolou zamezíme útoku, jelikož v podvrženém formuláři by musela být hodnota tokenu stejná s aktuální hodnotou tokenu v SESSION, což je velmi málo pravděpodobné.

5.4 Testování webové aplikace

Aplikaci nebudeme testovat z hlediska validity kódu. Kód všech stránek je 100% validní podle standardu XHTML 1.1 a kód CSS je taktéž validní. To by mělo být standardem všech webových aplikací.
Testování bezpečnosti webové aplikace proběhlo již v předchozí kapitole „aplikace pravidel bezpečnosti“. Proto se zaměříme na testování rychlosti a datové velikosti a testování přístupnosti.

5.4.1 Testování rychlosti a datové velikosti

Všechny obrázky jsou optimalizovány pomocí aplikace PNGGauntlet. Na serveru je v souboru *.htaccess* nastaveno kešování obsahu, je nastavena doba expirace pro všechny soubory a je aktivována kompresní metoda gzip.

V testu na stránkách pingdom.com získal web hodnocení 98/100, jak je vidět na obrázku 5.4.1.1. Bylo potřeba stáhnout data o celkové velikosti 63.7 KB. Stahování trvalo 1.31 s. Výsledek je tedy velmi uspokojivý. Dva body k maximálnímu hodnocení sráží nenastavená doba expirace u externího .js souboru od Google Analytics.

Obr 5.4.1.1 – výsledky testu na stránkách pingdom.com

Na Page Speed bylo dosaženo hodnocení 95/100, což je opět uspokojivé. Bylo nalezeno 5 chyb nízké priority. Výsledek je tak opět dostačující.

.htaccess:

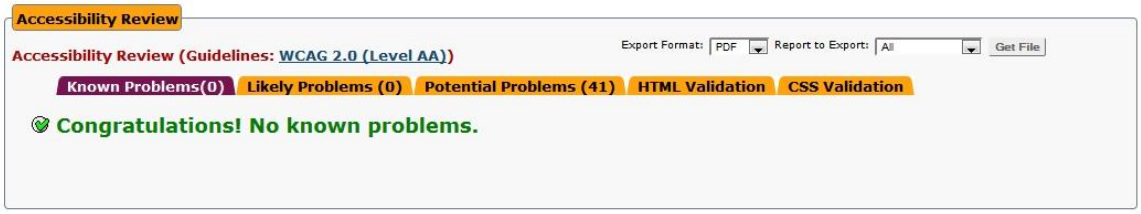
```
<IfModule mod_expires.c>
        ExpiresActive On
        ExpiresDefault A1209600
        ExpiresByType text/html A600
</IfModule>

<Files "\.(js | css | xml | gz)$>
        Header append Cache-Control "public, must-revalidate"
        Header append Vary Accept-Encoding
</Files>
```

```
<IfModule mod_deflate.c>
        <FilesMatch "\.(js|css|html)$">
                SetOutputFilter DEFLATE
        </FilesMatch>
</IfModule>
```

5.4.2 Testování přístupnosti

Přístupnost byla kontrolována pomocí online validátoru **ACChecker**. Všechny stránky byly testovány postupně podle metodiky WCAG 2.0 (Level AA a Level AAA) a Section 508. Výsledky jsou zřejmé z obrázků 5.4.2.1, 5.4.2.2 a 5.4.2.3.
Všechny stránky jsou validní z hlediska přístupnosti dle testovaných metodik. Výjimku tvoří WCAG 2.0 Level AAA, kde nesplňují podmínku kontrastu některé části webu.

Obr 5.4.2.1 – výsledky testu Section 508



Obr 5.4.2.2 – výsledky testu WCAG 2.0 (Level AA)

Obr 5.4.2.2 – výsledky testu WCAG 2.0 (Level AAA)

Dále proběhlo testování při ***ovládání webu klávesnicí***, použitelnost při ***vypnutých CSS*** stylech, použitelnost při ***vypnutých obrázcích***, zobrazení při zapnuté funkci „***vysoký kontrast***“ a při ***zvětšení textu***.

Stránky byly testovány na všech běžně používaných prohlížečích, tedy IE7 – IE9, Firefox, Safari, Opera a Chrome.

Všechny testy dopadly úspěšně a můžeme prohlásit, že aplikace splňuje všechna pravidla přístupnosti, tudíž je plně přístupná.

6 ZÁVĚR

V souladu se zadáním diplomové práce byla nastudována problematika přístupnosti a bezpečnosti webových aplikací a způsobů jejich testování. Přístupnost internetových stránek na webu je v dnešní době na poměrně dobré úrovni, jelikož v tomto aspektu tvorby webových aplikací došlo v posledních letech k osvětě. Pokud si nejsou firmy přímo vědomy ekonomických i jiných výhod přístupné prezentace a přímo to nevyžadují, pak je v zájmu tvůrců webových aplikací, aby jejich výsledný produkt splňoval pravidla přístupnosti a drželi krok s konkurencí.
Aktuálním tématem dnešní doby je bezpečnost webových aplikací. Přestože není příliš problém zjistit si nejvážnější rizika a nejčastější formy útoků na webové aplikace, na internetu se stále vyskytuje nezanedbatelné procento nezabezpečených aplikací. Zpravidla se jedná spíše o starší weby, na které již zabezpečení kódu není zpětně implementováno. Proti nejběžnějším útokům, jako SQL Injection, nebo Cross Site Scripting, existují jednoduché způsoby zabezpečení. Vzhledem k aktuálnímu dění a hromadným útokům na internetové síti se dá očekávat větší zájem poptávajících na bezpečnost jejich www stránek.
Všechny webové aplikace by měly být dostatečně testovány. Bohužel ne vždy tomu tak bývá, ať již z důvodu časových, či finančních. S testováním základních faktorů, jako validita kódu, funkčnost odkazů, apod., obvykle problém nebývá, jelikož jsou časově nenáročně a je k dispozici dostatek analytických nástrojů. Podobně je to i s faktory přístupnosti, které se dají otestovat strojově. Testování ostatních faktorů přístupnosti, jako například zobrazení stránek s vypnutými styly, při funkci vysoký kontrast, apod., již vyžaduje větší zapojení testera, stejně jako testování zabezpečení stránek, kdy je třeba vyzkoušet útoky na potenciálně zranitelná místa. Proto někdy bývá testování těchto faktorů opomíjeno.

Všechny probrané aspekty byly v praxi aplikovány na stránkách mkteam.cz.

Závěrem práce uvedu seznam doporučení, která by měla být dodržena při tvorbě přístupných a proti nejčastějším útokům zabezpečených webových aplikací:

- Design stránek by měl být přehledný, ideálně se zažitým rozvržením struktury (navigace, obsah,…)
- Barvy by měly být dostatečně kontrastní (podklad - text) a neměli by nést žádnou důležitou informaci (například u formulářů barevně označená povinná pole)
- Obrázky by měly obsahovat výstižný popis dle jeho funkce na stránkách (ilustrační obrázek, obrázek obsahující důležitou informaci,…) obsažený v atributu ***alt*** nebo ***longdesc***
- Odkazy by měly být sami o sobě výstižné, nebo doplněné o rozšiřující informaci v atributu ***title***, navíc by měly být vždy podtržené (pokud nejde o hlavní či vedlejší navigaci)
- Text by měl být definován relativními jednotkami a definicemi a vždy by měla být definována rodina písma (***font-family***)
- U formulářů pro nadpisy polí využívat značku **<*label*>** a propojit její atribut ***for*** s atributem ***id*** daného formulářového pole
- U tabulek využívat značku **<*th*>** pro hlavičku tabulky a propojit jejich atribut ***id*** s atributem ***headers*** daných buněk tabulky
- Ošetřit data využívaná pro výběr dat z databáze tzv. escapováním (v PHP funkce ***mysql_real_escape_string()***)
- Ošetřit speciální znaky jazyka HTML u vstupních dat uživatelů, která jsou vypisována na stránky (v PHP funkce ***htmlspecialchars()***)
- Při přihlašování uživatele kontrolovat i doplňkové informace (IP, UserAgent,…) a nabídnout tlačítko k odhlášení, které ukončí relaci sezení
- Při každém přihlášení měnit identifikátor sezení (v PHP ***session_regenerate_id***)
- Využívat tzv. podepsaných formulářů, k čemuž poslouží náhodně generovaný ***token***
- Pomocí online validátorů otestovat validitu HTML, CSS, funkčnost odkazů, rychlost stránek, přístupnost dle zvolené metodiky
- Otestovat přístupnost z hlediska strojově neměřitelných faktorů, například s pomocí k tomu určených toolbarů
- Otestovat stránky proti bezpečnostním rizikům, minimálně proti SQL Injection a XSS

7 SEZNAM POUŽITÝCH ZDROJŮ

Blind Friendly [online].© 2000-2012 [cit. 2012-02-19]. Dostupné z WWW: <http://blindfriendly.cz/>

CEDERHOLM, Dan. *Webdesign s webovými standardy*. Brno: Zoner Press, 2004. ISBN 80-86815-15-3.

Dobrý Web [online]. [2008] [cit. 2012-02-21]. Dostupné z WWW: <http://www.dobryweb.cz/ >

HUSEBY, Sverre H. *Zranitelný kód*. Brno: Computer Press, 2006. ISBN 80-251-1180-6.

JAKOB, Nielsen. *UseIt*: *Ten Usability Heuristics* [online]. © 2005 [cit. 2012-02-19]. Dostupné z WWW: <http://www.useit.com/papers/heuristic/heuristic_list.html>

JANOVSKÝ, Dušan. *Jak psát web* [online]. 2001, 20.2.2012 [cit. 2012-02-21]. Dostupné z WWW: <http://www.jakpsatweb.cz/>

MINISTERSTVO VNITRA ČESKÉ REPUBLIKY. *E-Goverment*: *Zákon č. 365/2000 Sb., o informačních systémech veřejné správy* [online]. © 2010 [cit. 2012-02-19]. Dostupné z WWW: <http://www.mvcr.cz/clanek/zakon-c-365-2000-sb-o-informacnich-systemech-verejne-spravy.aspx>

OWASP: *The Open Web Application Security Project* [online]. 2001, 1.2.2012 [cit. 2012-02-21]. Dostupné z WWW: <https://www.owasp.org/index.php/Main_Page>

Root.cz [online]. © 1998 – 2012 [cit. 2012-02-21]. Dostupné z WWW: <http://www.root.cz/>

Section 508 [online]. [2010] [cit. 2012-02-19]. Dostupné z WWW: <http://www.section508.gov/>

ŠPINAR, David. *Tvoříme přístupné webové stránky*. Brno: Zoner Press, 2004. ISBN 80-86815-11-0.

ŠPINAR, David. *Přístupnost* [online]. [2004] [cit. 2012-02-19]. Dostupné z WWW: <http://pristupnost.nawebu.cz/>

ŠTRUPL, Václav. *Interval.cz*: *Testování webových stránek* [online]. 31.3.2004 [cit. 2012-02-21]. Dostupné z WWW: <http://interval.cz/clanky/testovani-webovych-stranek/>

VEČEŘA, Zdeněk. Jak na to: SQL injection, magic_quotes_gpc, addslashes() a stripslashes(). *Zdeněk Večeřa* [online]. © 2008-2012 [cit. 2012-02-21]. Dostupné z WWW: <http://blog.zdenekvecera.cz/>

VRÁNA, Jakub. *PHP triky* [online]. © 2005-2012 [cit. 2012-02-21]. Dostupné z WWW: <http://php.vrana.cz/>

W3C [online]. © 2011 [cit. 2012-02-19]. Dostupné z WWW: <http://www.w3.org>

W3C. *Web Content Accessibility Guidelines (WCAG) 2.0* [online]. © 2008 [cit. 2012-02-19]. Dostupné z WWW: <http://www.w3.org/TR/WCAG/>

Zdroják.cz [online]. © 2008-2012 [cit. 2012-02-21]. Dostupné z WWW: <http://zdrojak.root.cz/>