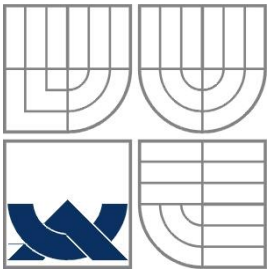
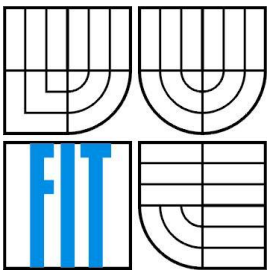


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÝ INFORMAČNÍ SYSTÉM A PRINCIPY JEHO ZABEZPEČENÍ

PRINCIPLES OF SECURITY OF THE WEB INFORMATION SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. FILIP NEHASIL

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PAVEL OČENÁŠEK

BRNO 2007

Zadání diplomové práce/5040/2006/xnehas00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Nehasil Filip, Bc.**
Obor: Informační systémy
Téma: **Webový informační systém a principy jeho zabezpečení**
Kategorie: Softwarové inženýrství

Pokyny:

1. Prostudujte nástroje pro tvorbu webových aplikací, zejména jazyk HTML, databázi MySQL a skriptovací jazyk PHP.
2. Získejte, analyzujte a specifikujte funkční a nefunkční požadavky na informační systém konkrétního klubového IS, který bude kromě evidence členů klubu a dalších konkrétních funkcí (dle informací vedoucího práce) obsahovat i plánování a zveřejňování akcí a redakční/publikační systém pro správu internetových stránek klubu. Informační systém bude tvořen veřejnou částí, administrační sekcí pro členy klubu.
3. Na základě požadavků proveďte architektonický a detailní návrh informačního systému jako webové aplikace.
4. Webový informační systém implementujte a otestujte s ohledem funkčnost, intuitivní ovládání a bezpečnost aplikace.
5. Zpracujte detailněji problematiku bezpečnosti IS a konkrétní možnosti útoků na IS z různých pohledů. Demonstrujte na vlastnostech řešeného IS.
6. Zhodnoťte dosažené výsledky a přínos, diskutujte možnosti dalšího vývoje.

Literatura:

- Welling, L., Thomsonová, L.: PHP a MySQL - rozvoj webových aplikací, 2. vydání, SoftPress, 2003, 912 s., ISBN 8086497607.
- Ullman, L.: PHP a MySQL - Názorný průvodce tvorbou dynamických WWW stránek, Computer Press, 2004, 536 s., ISBN 80-251-0063-4.

Při obhajobě semestrální části diplomového projektu je požadováno:

- první tři body zadání

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Očenášek Pavel, Ing.**, UIFS FIT VUT
Datum zadání: 28. února 2006
Datum odevzdání: 22. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Bc. Filip Nehasil**
Id studenta: 49363
Bytem: Solopysky 47, 284 01 Kutná Hora
Narozen: 07. 01. 1983, Kolín
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Webový informační systém a principy jeho zabezpečení
Vedoucí/školitel VŠKP: Očenášek Pavel, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☒ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Hlavním tématem práce je problematika zabezpečení webové aplikace. Jsou rozebrána potenciální zranitelná místa a navrženy způsoby ochrany aplikace v oblastech řízení přístupu, přenosu dat, uživatelského vstupu, výstupu a uchování stavu. S ohledem na bezpečnost jsou dále popsána možná řešení auditu a zálohování, zabezpečení webového serveru, návrhu databáze a obrany proti internetovým robotům. Projekt se rovněž zabývá analýzou požadavků, detailním návrhem a implementací klubového informačního systému, na němž jsou uvedená bezpečnostní řešení demonstrována. V rámci tohoto IS je mimo jiné implementován systém evidence členů klubu, evidence chovných jedinců a jejich rodokmenů, systém zveřejňování klubových akcí a jejich výsledků a systém pro správu dalšího obsahu webové prezentace. Veškerá implementační řešení jsou postavena na technologiích HTML, CSS, JavaScript, PHP a MySQL.

Klíčová slova

Informační systém, web, softwarové inženýrství, bezpečnost, ochrana, HTTPS, PHP, MySQL

Abstract

The main topic of the thesis is problem of the security of the web application. Spheres that are likely to be vulnerable are thoroughly examined and ways of protecting the application in the area of access control, data transfer, user input, output and state keeping are suggested. Furthermore, with regard to the security, possible solutions of audit, backing up, web server security, database design and protection against spam robots are described. This project is also concerned with the analysis of requirements, detailed design and implementation of the club information system with demonstration of mentioned security solutions. The systems of members' evidence, evidence of breeding dogs and their pedigrees, publication of actions and results and the content management system of other sections of the web presentation are among others implemented within the scope of this IS. All the implementation solutions are based on HTML, CSS, JavaScript, PHP and MySQL technologies.

Keywords

Information system, web, software engineering, security, protection, HTTPS, PHP, MySQL

Citace

Filip Nehasil: Webový informační systém a principy jeho zabezpečení, diplomová práce, Brno, FIT VUT v Brně, 2007

Webový informační systém
a principy jeho zabezpečení

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Pavla Očenáška.
Další informace mi poskytli MVDr. Josef Nosek a MVDr. Eva Čerešňáková.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
22.5.2007

Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce Ing. Pavlu Očenáškovi, MVDr. Evě Čerešňákové a MVDr. Josefu Noskovi za pomoc při odborných konzultacích v průběhu řešení projektu.

Obsah

1 Úvod

Bezpečnost webových aplikací je bezesporu aktuálním tématem dnešní doby. Díky stále snazší dostupnosti pevného a rychlého připojení k internetu roste i riziko napadení webových informačních systémů (IS). Rozšiřuje se povědomí o známých potenciálních slabých místech a zvyšuje se počet útočníků, kteří při odhalení takových míst představují reálnou hrozbu. Stěžejní část této práce je zaměřena právě na rozbor možných útoků na webový informační systém a samozřejmě na principy zabezpečení IS proti těmto útokům. V rámci projektu jsou dále ověřeny postupy softwarového inženýrství, prostudovány nástroje vhodné pro vývoj portálového IS a provedena analýza a návrh informačního systému pro Retriever klub CZ, na jehož samotné implementaci jsou všechny popsané způsoby zabezpečení demonstrovány.

V kapitole 2. budou stručně popsány prostředky pro tvorbu webových aplikací, které jsem využil pro implementaci našeho systému. Jedná se o jazyk HTML, skriptovací jazyky PHP a JavaScript, kaskádové styly CSS a databázový server MySQL.

Nejrozsáhlejší 3. kapitola obsáhne téma bezpečnosti webového IS. Budu se věnovat řízení přístupu, problémům spojeným s uživatelským vstupem a s ošetřením výstupu. Popíšu principy bezpečného přenosu dat na webu a bezpečného uchování stavu při práci s webovou aplikací. Zmíním nejdůležitější aspekty zabezpečení webového serveru a správný návrh databáze s ohledem na integritu dat. Na závěr uvedu možnosti ochrany webové aplikace proti internetovým robotům a nastíním význam auditu a zálohování.

Další kapitola je věnována softwarovému inženýrství. Budou zmíněny jednotlivé fáze životního cyklu softwaru, typy modelů životního cyklu a budou popsány postupy a prostředky, které byly využity při analýze požadavků a návrhu našeho IS.

V kapitole 5. se seznámíme se zadavatelem našeho projektu, organizací Retriever klub CZ. Objasním důležité pojmy, které budou užívány v dalších kapitolách, zmíním se o členství a orgánech klubu. V závěru kapitoly shrnu motivační záležitosti a důvody pro vytvoření informačního systému pro tuto organizaci. V další části práce budou podrobně popsány požadavky na náš informační systém. Analýza požadavků je v našem případě zpracována jako detailní neformální návrh aplikace. V 7. kapitole bude vytvořen formální návrh informačního systému na základě analýzy požadavků, a to s využitím jazyka UML. Návrh bude mít podobu diagramů případů použití a logického datového modelu.

V závěru samotné práce shrnu přínos celého projektu v průběhu jeho vývoje v době zimního i letního semestru. Zmíním také kroky, které budou následovat před uvedením našeho informačního systému do provozu.

2 Technologie pro tvorbu webových aplikací

2.1 Dynamické webové aplikace

V současné době existuje poměrně velké množství technologií, kterými lze vytvářet webové aplikace. Pod pojmem statických internetových stránek rozumíme stránky neměnící samovolně svůj obsah nebo svůj vzhled v čase. Tyto stránky tvořily převážnou část zobrazovaných dat na internetu asi do roku 1996. Pro jejich tvorbu bylo využito pouze jazyka HTML. Stránky se skládaly z textu upraveného pomocí formátovacích značek, obrázků a odkazů. Později dochází k mohutnému nárůstu využívání databází a rodí se myšlenka dynamicky tvořených stránek, tedy takových, které jsou schopné měnit svůj obsah nebo vzhled v čase dle požadavků uživatele nebo programátora. Tato změna probíhá bez zásahů programátora do zdrojového kódu stránky. Takovému kódu v tomto případě již můžeme říkat skript. Technologie můžeme nyní obecně rozdělit podle toho, kde skriptování probíhá.

2.1.1 Skriptování na straně klienta

Tento přístup je založen na vnoření kódu skriptu mezi speciální značky většinou do HTML dokumentu. K jeho interpretaci dochází až v prohlížeči klienta ve chvíli, kdy je stránka načtena ze serveru. Interpret obsažený v prohlížeči zachytává události provedené uživatelem přímo na webové stránce. Podle těchto událostí spouští příslušné části skriptu, které provádějí většinou jednoduché operace jako dílčí výpočty, úpravu vzhledu dokumentu, kontrolu formulářových prvků, zobrazení či skrývání prvků nebo manipulaci s cookies. O některých vlastnostech nejrozšířenějšího jazyka tohoto druhu, JavaScriptu, se blíže zmíním na straně 7.

2.1.2 Skriptování na straně serveru

Druhým, pro rozsáhlejší projekty preferovaným způsobem, je dynamické vytváření obsahu stránky již na straně serveru. Princip je založen na spuštění interpretu skriptovacího jazyka, který může být například zaveden do webového serveru jako modul. Do HTML dokumentu je umístěn kód skriptovacího jazyka, který je při každém požadavku na zobrazení HTML stránky na serveru interpretován a proveden. Skriptem lze zpracovávat vstupní data předaná klientem pomocí protokolu HTTP [5], metodou GET nebo POST. Výstupem předaným zpět klientovi je pouze textový dokument v určitém formátu, tedy typicky (X)HTML stránka, XML dokument apod.

V současné době je využíváno pro tvorbu dynamických stránek na straně serveru komerčního jazyka Active Server Pages (ASP) nebo platformy .NET s jazyky C# nebo ASP.NET. Open Source variantou tvorby dynamických stránek je jazyk nazvaný PHP, jež byl zároveň zvolen pro vývoj našeho IS.

2.1.3 Porovnání obou metod

Každá z výše zmíněných metod má své výhody i nevýhody. Pomocí dynamických stránek tvořených na straně klienta jsme schopni provádět rychlé změny na stránce, jelikož nedochází k přenášení dat po síti. Nevýhodou je možnost vypnutí interpretace skriptu či nemožnost uložení provedených změn na serveru. Vytvoření funkčního klientského skriptu obnáší také mnohá úskalí při požadavcích zajistit stejné chování ve všech standardně používaných prohlížečích, jelikož některé vlastnosti nebo metody tříd jsou pro konkrétní prohlížeče specifické.

Dynamické stránky tvořené na straně serveru disponují širokou škálou možností manipulace s daty. Na druhé straně pro každé provedení akce vyžadují komunikaci se serverem. Většina webových aplikací je postavena právě na serverovém řešení s tím, že klientské skriptování může například zvýšit uživatelský komfort při jejich používání.

2.2 Jazyk HTML a jeho struktura

Každý HTML dokument je textový soubor s příponou htm nebo html. Jazyk vychází z původního značkovacího jazyka SGML, od kterého se liší především tím, že má definovanou sémantiku, tady že jednotlivé značky mají určitý význam. Syntaxe je definována pomocí DTD (Document Type Definition), jehož deklarace musí být uvedena na začátku souboru. Definuje obsah značek, názvy a hodnoty jejich atributů, tvar entit atd.

HTML dokument může obsahovat obyčejný text, HTML značky a HTML entity. Značky nazýváme tagy, z nichž každý má svůj název, povinné a nepovinné atributy a svoji sémantiku, tedy definované použití. Pro snadnou výuku jazyka HTML lze využít například výborně zpracované internetové stránky Jak psát web [4]. Tento portál zároveň dobře poslouží i zkušeným programátorům jako přehledná referenční příručka v českém jazyce.

Definice typu dokumentu – definuje standard, podle kterého je HTML dokument napsán. Je povinnou součástí každého dokumentu.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

Párové tagy – mají počáteční a koncovou značku, mezi než lze umístit entity, text nebo další tagy (ne libovolné). Obsah včetně značek nazýváme HTML elementem. Příkladem může být tag označující počátek a konec odstavce:

```
<p>text odstavce</p>
```

Nepárové tagy – nemají koncovou značku a představují samostatný HTML element. Jako příklad lze uvést element odřádkování:

```
<br>
```

Atributy elementů – specifikují vlastnosti elementů a zapisují se přímo jako součást počáteční značky tagu. Příkladem je element hypertextový odkaz vedoucí na určitou adresu:

```
<a href="http://www.google.com">Toto je odkaz na Google</a>
```

HTML entity – pomocí nich lze do dokumentu vkládat speciální znaky, tedy například i ty, které jsou součástí syntaxe HTML. Příkladem mohou být entity `<` a `>` pro znaky < a > nebo entita ` ` pro pevnou mezeru.

2.3 Jazyk CSS – kaskádové styly

2.3.1 Přínos kaskádových stylů

V době, kdy se začal rozvíjet web a vznikaly první verze jazyka HTML určeného pro psaní webových stránek, se příliš nepočítalo s tím, že by někdy mohly webové stránky vypadat pěkně. V představách původních tvůrců HTML se totiž budoucí web nejspíše jevil jako množství strohých webových stránek, kde je vzhled nepodstatným atributem. Záhy se ovšem ukázalo, že možnosti webových stránek bude nutno rozšířit a do jazyka přidat prostředky, které by umožňovaly o něco více, než jen zobrazení strohého strukturovaného textu. Byly tak definovány nové atributy HTML elementů, které ovlivňovaly grafický vzhled. Dnes je již možné v HTML získat poměrně pěknou a atraktivní stránku, avšak s řadou nevýhod pro programátory. Jednou z nich je pracnost, mnoho věcí musí být nastavováno stále znovu. Z toho vyplývá i nepřehlednost. Náhlá potřeba změnit vzhled určité skupiny stejných elementů (například nadpisů) tak může značně zkomplikovat práci. Většinu problémů řeší právě kaskádové styly, které se označují zkratkou CSS (Cascading Style Sheets). Definují vzhled dokumentu nezávisle na jeho obsahu, zatímco definici struktury a obsahu obstará HTML. K hlavním přednostem CSS patří přehlednost, snadná modifikace vzhledu, možnost vrstvení stylových vlastností pro různé elementy, dědičnost stylů, centrální definice vzhledu pro mnoho stránek nebo snadná tvorba alternativního či nového vzhledu.

2.3.2 Kombinace CSS a HTML

Deklarace v hlavičce – Součástí hlavičky HTML dokumentu může být element `<style></style>`, který svým obsahem nadefinuje potřebný stylový předpis.

Externí stylový předpis – je oddělený soubor, typicky s příponou .css, jehož obsahem je stylový předpis. Místo definice tohoto předpisu přímo v hlavičce HTML dokumentu se do této hlavičky umístí pouze odkaz na externí soubor:

```
<link href="styles.css" rel="stylesheet" type="text/css">
```

Tento způsob je užíván ve většině případů. Jedním důvodem je často velký rozsah stylového předpisu, dále požadované oddělení struktury a obsahu dokumentu od definice jeho vzhledu. Samozřejmě je zde také možnost využít základní výhodu CSS, kdy lze změnit jedním krokem styl celé skupiny dokumentů.

Atribut HTML elementu – jedná se definici stylu přímo pro konkrétní HTML element, kdy s použitím atributu style přiřadíme prvku libovolné vlastnosti, které jinak definujeme například v externím souboru.

Priorita aplikování stylu je řízena podle tohoto pořadí (od nejvyšší): Atribut elementu, deklarace v hlavičce, deklarace v externím souboru. Kompletní reference jazyka je dobře zpracována v publikaci [1] nebo v projektu [4].

2.4 JavaScript

JavaScript je programovací jazyk, který umožní obohatit HTML o dynamické prvky. Cpolečně s CSS se tato kombinace jazyků a jejich postupů označuje jako "dynamické HTML". JavaScript zapisujeme buď přímo na libovolné místo v HTML dokumentu pomocí párového tagu `<script>`, nebo využijeme možnosti importovat externí soubor s kódem. Následuje ukázka obou variant:

```
<script type="javascript/text">
blok příkazů zapsaný přímo v HTML dokumentu
</script>

<head>
<script type="text/javascript" src="functions.js"></script>
</head>
```

Jazyk je interpretovaný (nemusí se kompilovat) a jeho kód je prováděn přímo na klientském počítači. Většinou reaguje na události prováděné uživatelem v prohlížeči (přejetí myší, změna formulářového prvku, stisk klávesy apod.). Je to jazyk objektový a využívá objektů zabudovaných přímo v prohlížeči podle specifikace DOM (Document Object Model). Tedy jeho fungování závisí na konkrétním použitém programu, což vede často ke komplikacím, pokud chceme vytvořit kód, který bude fungovat všude tak, jak si představujeme. Syntaxí je tento jazyk podobný jazyku C nebo Java. K nevýhodám JavaScriptu patří to, že uživatel může jeho provádění v prohlížeči úplně zakázat a že kromě cookies JavaScript neumí přistupovat k souborům na disku nebo k databázím. Na druhou stranu je však neocenitelným pomocníkem v situacích, kdy potřebujeme komunikovat mezi

jednotlivými okny prohlížeče (například vkládání ze seznamu do formulářového pole), kdy chceme urychlit práci s aplikací při kontrole formulářů, kdy potřebujeme vytvořit rychlé dynamické menu, anebo jinak zvýšit uživatelský komfort. Referenci jazyka můžeme nelézt například na webu Sun Microsystems [6].

2.5 Skriptovací jazyk PHP

Jazyk PHP byl vytvořen v roce 1994 Rasmusem Lerdorfem. Následně byl několikrát přepracován jeho koncept, jehož výsledkem je silný a jednoduchý programovací nástroj využívaný při tvorbě dynamických internetových stránek. Jazyk PHP je Open Source produkt, je tedy možné jeho zdrojové kódy upravovat a dále distribuovat bez hrazení jakýchkoliv poplatků. Toto je důležitý aspekt, který umožňuje neustálý vývoj jazyka programátory po celém světě. Samotné používání PHP je samozřejmě také zdarma. Současná verze PHP má označení PHP 5, tato verze rozšiřuje velmi úspěšnou verzi PHP 4 především o nový objektový model umožňující plnou podporu objektově orientovaného programování. Veškeré informace týkající se projektu PHP lze nalézt na domovské stránce www.php.net. Jsou zde k dispozici různé verze programu ke stažení. Otevřené zdrojové kódy zajišťují přenositelnost na všechny platformy. Velkou výhodou PHP je jasná, kompletní a srozumitelná dokumentace, která je navíc z velké části přeložena také do češtiny [7]. Skriptování v PHP funguje na principu popsaném v kapitole 2.1.2. V tomto článku budou shrnuty základní vlastnosti jazyka, výhody a nevýhody.

Výkon – Výkonnost PHP ilustruje fakt, že i jeden server nenáročný na hardware je schopen obsloužit milióny požadavků za den. Výkonnostní testy firmy ZEND ukazují, že PHP může soupeřit s komerčními servery a často je i svým výkonem předčit.

Podpora databází – PHP disponuje schopností připojit se bez prostředníka k mnoha různým typům databází, z nichž nejpoužívanější je MySQL (viz následující kapitola). PHP ale umožňuje spolupráci například i s PostgreSQL, MS SQL, Oracle a mnoha dalšími databázemi prostřednictvím rozhraní ODBC.

Knihovny – PHP obsahuje zabudované knihovny pro podporu celé řady internetových protokolů jako je HTTP, SMTP, FTP, SNMP, IMAP a další. Další knihovny umožňují generování grafiky, generování dokumentů ve formátu PDF či komprimaci souborů apod. Mnoho dalších funkcí přinášejí rozšíření v rámci projektu PEAR [8].

Snadná výuka – Syntaxe jazyka vychází z mnoha různých programovacích jazyků, především z jazyka C a Perl. Proto pro většinu programátorů není těžké přejít na programování v PHP. Zároveň však není složité se jej naučit bez předchozích znalostí konkrétního jazyka, k čemuž velmi dopomáhá již zmíněná dokonale zpracovaná dokumentace. K jednoduchosti přispívají i takové vlastnosti jako nepovinná deklarace proměnných nebo automatická konverze typů

proměnných. Pro výuku jazyka lze kromě zmíněné dokumentace využít například jednu z kvalitních učebnic nakladatelství O'Reilly [2].

Bezpečnost – Díky naposledy zmíněné výhodě však PHP postrádá v tomto smyslu robustnost, na kterou jsou mnozí zvyklí z jiných jazyků jako je C. PHP nedisponuje mechanismy pro zajištění bezpečnosti ani zdrojových kódů aplikace. Bezpečnost v tomto směru tak zcela závisí na programátorovi.

2.6 Databázový server MySQL

MySQL je rychlý a robustní databázový systém. Je založený na dotazovacím jazyce SQL (Structured Query Language), což je celosvětově využívaný dotazovací jazyk pro databáze. Databáze umožňuje efektivní ukládání, třídění a vyhledávání dat. MySQL je víceuživatelský a vícevláknový databázový systém, což umožní vykonávání mnoha dotazů od různých klientů současně. Databáze nepatří do rodiny softwaru, který je plně Open Source, ale lze jej bezplatně využívat pro nekomerční účely. Pro použití v komerční sféře je nutné zakoupit příslušnou registraci. Databáze MySQL klade nízké nároky na administraci a lze ji plně využívat na všech distribucích systému Linux i Windows. Výhodou je také výborně zpracovaná a úplná dokumentace [9]. Použití úložiště InnoDB umožňuje v MySQL transakční zpracování i integritní omezení. Nejnovější verze MySQL 5 přináší možnost použití vnořených procedur a funkcí, triggerů a pohledů.

3 Bezpečnost webového informačního systému

Pojmem bezpečnost obecného informačního systému rozumíme především ochranu dat, se kterými tento systém manipuluje a které uchovává. S tímto tématem úzce souvisí důležité pojmy jako zranitelné místo, hrozba nebo útok. **Zranitelným místem** označujeme slabinu informačního systému, která může vzniknout v důsledku chybné analýzy, opomenutím důležité součásti návrhu nebo implementace, ale i špatným fyzickým zabezpečením technických prostředků. Některé vlivy prostředí, ve kterém IS provozujeme, se pak můžou stát **hrozbou** pro náš systém. Může se jednat o hrozbu přírodního charakteru (požár, povodeň), technického charakteru (porucha hardware, krádež datových médií, zadní vrátka v softwaru), nebo o hrozbu lidského faktoru (nedůvěryhodný zaměstnanec, nedůvěryhodný správce, útočník). Jednotlivé typy hrozeb se samozřejmě mohou prolínat. Na základě současné existence zranitelného místa a hrozby, jež může slabinu využít, lze provést **útok**. Jeho důsledkem pak může být zničení technického vybavení, ztráta důležitých dat, ale i neoprávněný přístup k datům, který umožní jejich zneužití.

V této kapitole se budu zabývat rozborem nejdůležitějších potenciálních slabin webového informačního systému z hlediska jeho softwarové implementace. Zároveň zde nabídnu konkrétní řešení těchto problémů pomocí technologií využitých k implementaci našeho IS, tedy skriptovacího jazyka PHP a databázového serveru MySQL. V jednotlivých podkapitolách se budu odkazovat také na skripty se zdrojovým kódem naší aplikace, ve kterých je možno daná řešení prostudovat detailněji.

3.1 Řízení přístupu

3.1.1 Systém uživatelských oprávnění

V každém systému musí být způsob, jak přidělit oprávnění jednotlivým uživatelům, a to nejen napevno při uvedení aplikace do provozu, ale i během jejího následného používání. Nejtypičtější je třídění uživatelů jo skupin, kde každá tato skupina má oprávnění číst nebo měnit data prostřednictvím konkrétních sekcí systému.

Způsob řešení

Systém oprávnění použitý v našem IS je do jisté míry univerzální. Z analýzy požadavků v kapitole 6 vyplynula nutnost zajistit členění uživatelů do více jak 15 různých rolí. Uživatel v určité roli může vstupovat do konkrétní sekce sytému, přičemž může figurovat v několika rolích najednou. Například všichni uživatelé jsou v základní roli *člen*, ekonom klubu je navíc v roli *správce evidence financí*

a pod. Oprávnění jednotlivých uživatelů lze reprezentovat například binární posloupností. Při 15 rolích vystačíme s dekadickým celým číslem $< 2^{15}$, které každému uživateli určí jeho oprávnění ke konkrétním sekcím aplikace. Kontrola oprávnění pak probíhá jednoduše pomocí jediné bitové operace.

Během autorizace uživatele jsou do relace (session) uloženy informace o jeho oprávněních uvedených v databázi. Pokud by útočník například získal heslo uživatele a přihlásil se do systému, musí existovat způsob, jak mu okamžitě přístup odepřít. Proto se informace o oprávněních uživatele aktualizují z databáze při každém požadavku na server. V každém skriptu, jehož obsah zajišťuje přístup do chráněné sekce, je na základě těchto informací zkontrolována příslušnost přihlášeného uživatele do konkrétní role. Pokud kontrola neprojde, uživatel je přesměrován na přihlašovací stránku.

V našem systému existuje pevný uživatelský účet *administrator*. Heslo do tohoto účtu zná pouze prezident klubu a jedna další důvěryhodná osoba. Pouze prostřednictvím tohoto účtu lze přiřazovat uživatele do jednotlivých rolí a přidělovat jim tak konkrétní oprávnění.

Odkazy na zdrojové kódy

auth.php, inc/function.php (`// definice oprávnění`, `// načtení oprávnění`, `rights()`), admin/users-rgh.php, admin/users-w.php

3.1.2 Bezpečná správa hesel

Nejslabším článkem v ochraně přístupového hesla bude vždy člověk a jeho přístup k ochraně osobních či firemních dat. Proto musíme v rámci bezpečnostní politiky instituce důsledně dodržovat zásady manipulace s hesly. Především jde o uchování hesla v tajnosti a nepoužívání stejných hesel do různých systémů. Systém samotný však může zajistit alespoň splnění požadavků na délku hesla, pravidelnou změnu hesla po určité době, případně odmítnutí vytvoření hesla slovníkového typu. Dále by měl být kladen důraz na bezpečné uchování hesel v databázi, tedy nikoliv v otevřené podobě. S tím souvisí také možnost obnovení hesla při jeho ztrátě, což samozřejmě není neobvyklý požadavek mnoha uživatelů.

Způsob řešení

Vzhledem k cílové skupině uživatelů, pro které je náš systém určen, je při změně hesla nebo zakládání účtu požadována pouze minimální délka hesla. Při změně je samozřejmě navíc požadováno zadání starého hesla. Pravidelná změna není vyžadována, stejně jako nejsou odepírána slovníková hesla. V databázi je uchován pouze otisk hesla (*hash*), který je při změně či zakládání hesla vytvořen algoritmem SHA1. Pro zajištění rozdílných otisků pro shodná hesla různých uživatelů je před aplikací algoritmu k řetězci připojeno členské číslo uživatele, které je v průběhu existence členství nezměnitelné. Verifikace probíhá stejným postupem, tedy ze zadaného hesla je s pomocí členského

čísla spočítán otisk a porovnán s otiskem v databázi. V případě pokusu o zpětné získání hesla z databáze, ať už ze strany útočníka, nebo ze strany správce či systému za účelem obnovení hesla při ztrátě, budeme neúspěšní. Obnova zapomenutého hesla je tedy řešena vygenerováním nového náhodného hesla, které si uživatel může nechat zaslat na e-mail, jež má uvedený v evidenci. S tím je spojena možnost odposlechnutí tohoto e-mailu a získání hesla. Řešení šifrování e-mailů nabízí například GnuPG v kombinaci s PHP. Jeho implementace do našeho IS je však naplánována až po rozšíření návrhu, tedy po uzavření diplomové práce.

Odkazy na zdrojové kódy

auth.php, member-a.php, member-w.php, member/user-d.php, member/user-w.php, admin/admin-p.php, admin/admin-w.php

3.1.3 Autentizace

Dalším typickým problémem při tvorbě webové aplikace je, jak zajistit bezpečný přenos hesla po síti při autentizaci. Ať už se rozhodneme použít vlastní webový formulář nebo HTTP autentizaci, musíme využít šifrovaný kanál, tedy protokol HTTPS (viz kapitola 3.4.2 na straně 23). V opačném případě lze zadané heslo odposlechnout, přestože jej posíláme pouze jednou. Použití metody POST při odesílání formuláře neřeší vůbec nic, protože obsah celého požadavku, tedy včetně hesla v otevřené podobě, je stále součástí HTTP hlavičky, ke které se může útočník dostat. Využití HTTP autentizace, jejíž podpora je zabudovaná přímo v protokolu HTTP, má jistá úskalí, jako problém s vynucením odhlášení. Jedno z řešení, v dnešní době ne však zcela optimální z důvodu narůstajících bezpečnostních opatření prohlížečů, je popsáno v jednom z odkazů [10].

Pokud nemůžeme použít protokol HTTPS, existuje takzvaný *challenge-response mechanismus* [11], s jehož pomocí lze provést bezpečnou autentizaci nešifrovaným kanálem. K realizaci je však nutná existence hashovací funkce na straně serveru i klienta. Je tedy podmínkou použít například JavaScript.

3.2 Vstup a výstup

3.2.1 Opakování požadavku

Při každé operaci, která má za následek změnu dat, je třeba zajistit, aby se HTTP požadavek neodeslal na server vícekrát. Situace může nastat například při prostém obnovení stránky uživatelem, který dlouho čeká na odezvu, nebo při chybě v přenosu po síti. Je vhodné si pro tyto případy navrhnout strukturu skriptů například tak, že pro každou logickou sekci aplikace bude existovat skript, který vykonává pouze konečné databázové nebo jiné operace, přičemž po jejich vykonání ihned zašle zpět na server hlavičku s požadavkem na přesměrování a ukončí svůj běh. Směrovat můžeme na různé

stránky například podle toho, zda operace proběhla úspěšně nebo ne. Je důležité si uvědomit, že HTTP hlavičky lze zasílat pouze před odesláním jakéhokoliv výstupu. Ve zdrojových kódech našeho IS mají všechny tyto "vykonávací" skripty v názvu postfix "`-w`".

Odkazy na zdrojové kódy

admin/news-w.php, inc/function.php (`location()`, `location_abs()`)

3.2.2 Kontrola uživatelského vstupu

U každé položky, kterou nám uživatel předá při odeslání webového formuláře, bychom měli uvážit, zda je nutná kontrola jejího tvaru. Typicky je potřeba kontrolovat správnost tvaru e-mailové adresy nebo telefonního čísla, správný tvar data nebo času, číselné hodnoty a jejich rozsah, případně délku řetězce nebo speciální požadavky na tvar řetězce (například u loginu). Tyto kontroly bývají poměrně časté, proto je vhodné si na ně vytvořit jednoduché funkce, které kontrolu provedou většinou na jednom či dvou řádcích pomocí méně či více složitého regulárního výrazu. Mnoho autorů aplikací spoléhá na kontrolu pomocí JavaScriptu, což je samozřejmě špatně. JavaScript může pomoci k většímu komfortu při upozorňování uživatele na chyby, jelikož se požadavek na server ani neodešle. Přesto vždy musíme použít kontrolu i přímo ve zpracovávajícím skriptu. Někteří uživatelé mohou mít JavaScript nevědomky vypnutý, jiní ho s úmyslem uškodit naopak vypnou schválně v domnění, že naše ochrana je takto slabá.

Poněkud závažnější situace může nastat při předávání parametrů v URL nebo ve skrytých polích formuláře, protože si mnoho programátorů nepřipustí, že uživatel může tato vstupní data velmi jednoduše pomocí dostupných prostředků změnit, přestože k tomu není vyzván. Malá nepozornost, kdy nám unikne správný způsob kontroly jediného parametru, může mít za následek vznik kritické bezpečnostní díry. Příkladem v našem IS může být manipulace s články v různých sekcích. V předchozí podkapitole jsem popsal způsob kontroly uživatelských oprávnění, která se provede na začátku každého skriptu. Několik sekcí pro správu článků (info, aktuality, chovatelství a pod.) je však přes svoje vazby napojeno na jedinou společnou SQL tabulku článků. Pokud bychom například při mazání aktuality zapomněli zkontrolovat, zda parametr ID článku předaný v URL náleží opravdu aktualitě, mohl by správce aktualit změnou parametru vymazat například článek ze sekce chovatelství, aniž by měl ke správě této sekce oprávnění. Podobné situace se vyskytují na mnoha místech aplikace, proto je velmi důležité si u každého zpracovávaného parametru představit, že může být útočníkem změněn, a je tedy nutné provést logickou kontrolu jeho správnosti.

Podobný problém skrytých formulářových polí může nastat, když uživatel vyplňuje formulář v několika krocích. V prvním kroku zkontrolujeme správnost všech zadaných vstupů a zobrazíme navazující formulář. Odeslané hodnoty však musíme uložit bezpečně například do proměnné relace, nikoliv do skrytého formulářového pole navazujícího formuláře, kde je uživatel může lehce změnit a obejít tak naše požadavky na jejich tvar, jelikož v druhém kroku již tyto hodnoty nekontrolujeme.

Odkazy na zdrojové kódy

inc/functions.php (`check_email()` a podobné), member-w.php

3.2.3 Ošetření neplatného vstupu

V případě, kdy neodpovídá tvar zadané hodnoty, je většinou uživatel vrácen na stejný formulář, který doplníme o chybová hlášení. Zároveň považuji za nutnost v takové chvíli vyplnit automaticky všechna pole tak, jak je uživatel odeslal. V kombinaci s námi navrženou strukturou skriptů je řešení jednoduché. Kontrola probíhá v prováděcím skriptu, který neodesílá žádný výstup, pouze po vykonání akce někam přesměruje. V případě chyby však nemusí přesměrovat (akce nebyla provedena) a může rovnou vložit původní skript s formulářem, jehož obsah se naplní z proměnných aktuálního požadavku. Při tomto kroku musíme dbát na zpětné ošetření speciálních znaků, o kterých budu mluvit v následující podkapitole.

Co se týče naší reakce na pokus uživatele změnit parametry a narušit tak prováděnou operaci, neměla by podle mého názoru dávat útočníkovi najevo nic konkrétního. Hlášky typu "Pokoušíte se neoprávněně změnit odesílané parametry" je vhodnější nahradit univerzálním hlášením o chybě, například "Tato stránka se na serveru nenachází". Pokud se stejná hláška objeví při každém pokusu o "ruční" úpravu požadavku, stejně jako při skutečně špatně zadané adrese, aplikace může budit dojem komplexnějšího zabezpečení. Tento názor však může každý považovat za čistě subjektivní.

Odkazy na zdrojové kódy

member-w.php

3.2.4 Modifikace SQL dotazu (SQL Injection)

Problém tohoto typu spočívá v předávání parametrů, které následně přímo využíváme jako část SQL dotazu, což nastává ve většině případů. Může se jednat o textová data nebo jen o číselný identifikátor. Uveďme příklad dotazu `UPDATE articles SET ... WHERE id = 24`. Parametr id je typicky předáván ve skrytém formulářovém poli, které útočník může upravit například na hodnotu `24 OR id > 0`. Bez ošetření se parametr stane součástí dotazu v nezměněné podobě, čímž se znehodnotí kupříkladu všechny články v systému. V určitých funkcích nebo databázových systémech je možné použít vícenásobný SQL dotaz oddělený středníkem (netýká se PHP funkce `mysql_query()`). Zde nastává ještě větší nebezpečí, kdy může útočník uvedením středníku a následného dotazu přímo do parametru manipulovat s databází, jak se mu zachce, pokud odhadne název tabulky. S tím souvisí i to, že v provozu aplikace nikdy nesmíme vypisovat chybová hlášení databázového serveru na výstup, jako to běžně děláme při ladění programu. Pokud například přidáme na konec dotazu, který neobsahuje agregační funkci, řetězec `HAVING 1=1`, obdržíme chybovou hlášku obsahující přímo název tabulky. Další možností, jak modifikovat dotaz, je použití dvou spojovníků `--`. Tato sekvence uvozuje

v MySQL komentář, tedy pokud takto pomocí parametru "zakomentujeme" část podmínky dotazu, může to mít opět fatální následky.

Způsob řešení

Patrně nejelegantnější obrana v PHP a MySQL proti tomuto útoku spočívá v uzavírání veškerých hodnot v SQL dotazu do uvozovek nebo apostrofů. MySQL toto umožňuje u kteréhokoliv typu atributu. Výjimkou je hodnota NULL, která musí být bez apostrofů i uvozovek. Téměř jediným znakem, který pak musíme na vstupu ošetřit, je právě apostrof a uvozovka. V MySQL lze provést escape sekvenci tradičně pomocí zpětného lomítka (které je také potřeba ošetřit), tedy například `\'` bude v dotazu interpretováno pouze jako znak apostrofu, nikoliv jako znak lomítka a řídící znak apostrofu. PHP disponuje konfigurační direktivou `magic_quotes_gpc`, která zajistí automatické escapování speciálních znaků v proměnných HTTP požadavku. Těmito znaky jsou apostrof, uvozovka, zpětné lomítko a NULL. Pokud si však nespravujeme webový server sami, nelze vždy spoléhat na to, že bude tato direktiva zapnutá. To lze ve skriptu zjistit funkcí `get_magic_quotes_gpc()` a v případě vypnutého automatického escapování jej nahradit vlastním ošetřením každé proměnné v požadavku funkcí `addslashes()`. Ta nám po zvolení správných parametrů zajistí totéž co `magic_quotes_gpc`, tedy s výjimkou polí. Je tedy třeba testovat, zda je proměnná typu pole a v takovém případě použít např. vlastní funkci `addslashes_array()` založenou na `addslashes()`, která escapuje i vícerozměrná pole. Nikdy však nesmíme zapomenout použít přímo `addslashes()` v případě, že jsou součástí dotazu data, která nepochází z HTTP požadavku, ale například ze souboru při importu a pod.

V předešlé části jsem zmínil automatické vyplnění formuláře zadanými daty při vzniku chyby. Jelikož jde v tomto případě přímo o proměnné z HTTP požadavku, je třeba před vložením do formuláře zrušit escapování pomocí funkce `stripslashes()`, případně opět vlastní funkcí `stripslashes_array()` pro pole. Zároveň se k tomuto kroku váže ne méně důležité téma XSS na straně 17.

Odkazy na zdrojové kódy

inc/functions.php (// `filtrace GET`, // `filtrace POST`, `//addslashes_array()`)

3.2.5 Podvrhnutí požadavku (Cross-Site Request Forgery)

Pod zkratkou SCRF se skrývá typ útoku, který není až tak známý jako zmíněné SQL Injection. Možná proto, že mnoho autorů webových aplikací si danou situaci neumí dobře představit, nebo je ani nenapadne. Tento útok provádí nevědomky sám uživatel, proti kterému je útok namířen, nebo který má například nadstandardní oprávnění, jež chce skutečný útočník využít.

Uvažme příklad z našeho IS, kdy uživatel – správce – může smazat chovného jedince pomocí odkazu na URL `/admin/dogs.php?delete=815`. Útočník může takový tvar URL v některých případech odhadnout. Někdy mu pomůže i vlastní registrace do systému, kdy většinou získá možnost něco upravit a může tak zjistit přibližnou strukturu skriptů. Útočník připraví na svém webu skrytý odkaz s tímto URL, který se po načtení této jeho stránky rovnou provede. Dá se to zařídit nejjednodušeji pomocí elementu `<img src="http://www.retrieverklub.cz/admin/dogs.php?delete=815">`. Pak už jen umístí odkaz na svoji stránku například do diskuze v našem IS a počká, až se správce někdy přihlásí a na tento odkaz klepne. Co se stane? Správce je v systému autorizován a jeho oprávnění jsou uložena v proměnné relace. Tato relace je samozřejmě přístupná, dokud se sám uživatel neodhlásí nebo nezavře všechna okna prohlížeče (více o relacích v kapitole 3.3 na straně 19). Tedy ve chvíli, kdy správce prostřednictvím útočníkova webu odesílá požadavek na URL `http://www.retrieverklub.cz/admin/dogs.php?delete=815`, kontrola oprávnění projde a aniž by správce něco tušil, smaže ze systému chovného jedince s ID 815. Útok při tom není omezen pouze na akce vyvolané pomocí URL. Útočníkova stránka může zahrnovat například formulář se skrytými poli, který pomocí JavaScriptu ihned po načtení stránky odešle připravená data metodou POST.

Způsob řešení

Spolu s každým požadavkem, který vykonává nějakou autorizovanou operaci, musíme předávat pro uživatele jedinečný *token*, který se před vykonáním této operace ověří. Token však nebudeme ukládat do proměnné relace, aby se prolomení této ochrany nestalo součástí krádeže relace (viz strana 20). Musíme jej přidat do každého URL zajišťujícího autorizovanou operaci a zároveň do všech formulářů jako skrytý parametr. Token se generuje jako otisk řetězce složeného z ID relace a hesla uživatele, a to v každém skriptu, kde je potřeba zajistit jeho předání. Před provedením operace se token vygeneruje znovu a porovná se s předanou hodnotou. Pokud nesouhlasí, zastavíme prováděnou akci a zobrazíme chybové hlášení. Token je tedy stejný po celou dobu trvání relace, ale útočník jej nemůže předem nijak zjistit, aby ho mohl přidat do nastrčeného URL na své stránce.

Musíme však v aplikaci zajistit, aby veškeré odkazy mířící ven mimo náš IS byly vedeny přes zvláštní směrovací stránku. Pokud by totiž klepnutí na takový odkaz předcházela právě autorizovaná operace obsahující token v URL, toto URL by se předalo ven v hlavičce Referer. Pokud by byl cílem takového odkazu web útočníka, který mohl sám odkaz nastražit třeba do diskuze, token by byl prozrazen. Naše směrovací stránka však nemůže využít zaslání hlavičky Location, jelikož Referer se v takovém případě stále uchovává. Musíme proto použít HTML meta tag `<meta http-equiv="refresh" content="0;url=http://nova-adresa">`. Dále nesmíme v naší aplikaci, jako v systému pro správu obsahu, umožnit vkládání obrázků tak, aby se v atributu `src` mohl objevit uživatelem zadaný řetězec. Musíme vkládání interních obrázků řešit například přes jeho identifikátor.

Odkazy na zdrojové kódy

admin/news.php, admin/news-w.php

3.2.6 Ovlivnění chování skriptu (Script Injection)

Při tomto útoku můžou být opět na vině neošetřené speciální znaky ve vstupních parametrech, a to v případě, že je použijeme jako součást příkazu vykonávaného shellem operačního systému. Tento útok by se dal spíše nazvat "Shell Command Injection" a funguje podobně jako u modifikace SQL dotazu. Nejlepší obranou je samozřejmě vyhnout se v naší aplikaci nutnosti použití uživatelem zadaných parametrů v příkazu shellu. Pokud však přece musíme použít funkci `exec()` nebo `system()`, kde v příkazu figurují zaslané parametry, je třeba ošetřit uživatelský vstup pomocí funkce `escapeshellcmd()`, nebo raději striktně kontrolovat každý parametr a povolit v něm pouze množinu bezpečných znaků, která nemusí být příliš rozsáhlá.

Změny chování skriptu jako takového lze docílit podstrčením globální proměnné. S tím souvisí konfigurační direktiva `register_globals`, která byla v nižších verzích PHP standardně nastavena na "on", čímž mohly nastat situace, kdy útočník podstrčil obsah globální proměnné užité ve skriptu pouhým přidáním této proměnné do URL požadavku. Všechny proměnné předané z požadavků (GET, POST nebo COOKIE), stejně jako proměnné prostředí a serveru, jsou totiž v případě zapnuté direktivy `register_globals` automaticky ve skriptu přístupné jako globální. Jelikož PHP nevyžaduje inicializaci proměnných, mohly by být globální proměnné tímto způsobem definovány na neočekávanou hodnotu již při vstupu do skriptu. V PHP verzi 4.2 a vyšších je `register_globlas` standardně vypnuta. Proměnné z HTTP požadavku jsou pak k dispozici pouze prostřednictvím *superglobálních polí* `$_GET`, `$_POST` a `$_COOKIE`, případně všechny najednou v poli `$_REQUEST`. Přesto se však nelze na `register_globals="Off"` spoléhat, jelikož nemusí být takto nastavena na každém web hostingu, nebo každý web hosting nemusí toto nastavení ve skriptu umožnit. Použití superglobálních polí a důsledná inicializace globálních proměnných, zvláště polí a řetězců, je tedy základní podmínkou bezpečné aplikace.

Odkazy na zdrojové kódy

inc/functions.php (// `filtrace GET`, // `filtrace POST`)

3.2.7 Podstrčení kódu (Cross-Site Scripting)

Zatím jsem se téměř nezmínil o jakémkoliv ošetření výstupu na HTML stránku. S ním je spojen útok pod známou zkratkou XSS, tedy Cross-Site Scripting. Základní chybou mnoha aplikací je nevhodná struktura skriptů. Jedná se o přístup typu `/index.php?stranka=admin/clanky` ve snaze zjednodušit kód tím, že v souboru index.php se bude jakožto na jediném místě vkládat hlavička a patička stejná pro všechny stránky. Mezi ně se vloží skript, který se zvolí na základě parametru z URL doplněného

o příponu php. Tato struktura je nevhodná i z jiných důvodů, nás ale zajímají především ty bezpečnostní. Neošetřením tvaru parametru `stranka` se můžeme dostat do potíží ve chvíli, kdy útočník nastaví do tohoto parametru adresu na svůj skript, který vykonává něco škodlivého (`http://utocnik.cz/skodlivy-kod`). Těmto konstrukcím bychom se měli raději vyhnout, nebo je důkladně ošetřit. Nebezpečí může dále nastat všude, kde vypisujeme neošetřené předávané parametry nebo data z databáze. V obou případech se na stránku může vypsat HTML kód, který v lepším případě naruší strukturu stránky nebo její grafickou podobu, v horším případě pak v kombinaci s JavaScriptem může provádět závažnější kroky. Jako příklad mohu uvést krádež cookie, čehož se dá využít mimo jiné ke zjištění identifikátoru relace (více o tomto tématu se dočtete na straně 20).

Jak je známo, cookie můžeme pohodlně číst pomocí JavaScriptu. Stejně tak můžeme obsah tohoto cookie odeslat metodou objektu `location` třeba jako parametr URL. Pokud se nám například do webového fóra podaří vložit příspěvek, jehož součástí bude takovýto JavaScriptový kód uzavřený do tagů `<script></script>`, každý přihlášený uživatel, který vstoupí na stránku s tímto příspěvkem, nám odešle obsah svého cookie. Náš skript si tento obsah uloží do databáze a přesměruje uživatele na libovolnou jinou stránku. Ve chvíli, kdy my jako útočníci vstoupíme na webové fórum a přepíšeme přidělené cookie jedním z ukradených, tváříme se pro webové fórum jako uživatel, jehož cookie jsme odcizili.

Způsob řešení

Použití PHP funkce `strip_tags()`, která filtruje z textu všechny HTML tagy, není dostatečné, navíc je tato funkce navržena pro jiné účely. Nepořádek mohou udělat též apostrofy nebo uvozovky, pokud jsou data součástí atributu HTML tagu. Někteří programátoři se snaží ošetřit data proti XSS hned při vstupu, tedy ještě před uložením do databáze. Toho lze docílit převedením speciálních znaků na HTML entity hned při odesílání dat. Osobně však tento způsob nepovažuji za vhodný. Data by se v databázi měla uchovávat v původní "čisté" podobě a ošetřena by měla být až při předávání na výstup, např. z důvodu možného využití dat i jinde než v HTML dokumentu.

Ošetření spočívá v nahrazení znaků `& < > &#039; &quot;` HTML entitami. V PHP za nás přesně tuto operaci vyřeší funkce `htmlspecialchars()` s parametrem `ENT_QUOTES`. Nesmíme zapomenout ošetřit také každou před-vyplněnou hodnotu formulářového prvku. Rozhodně není na místě spoléhat na to, že například některé hodnoty budou vždy jen číselné, a proto je zbytečné je ošetřovat. Vždy se vyplatí řídit se zásadami ochrany na více úrovních. Chceme-li hromadně ošetřit obsah jednotlivých prvků pole, musíme použít funkce pro práci s regulárními výrazy, které umí pole takto zpracovat.

Systém pro správu obsahu, jakým je i náš IS, musí samozřejmě umožnit správcům vytváření alespoň základně formátovaného textu, vkládání interních obrázků, odkazů nebo tabulek. Pro tyto účely je vhodné navrhnout a implementovat vlastní značkovací jazyk, jehož elementy nebudou ovlivněny použitím funkce `htmlspecialchars()`.

Odkazy na zdrojové kódy

inc/functions.php (`spec()`), inc/text2html.php (vlastní značkovací jazyk)

3.2.8 Nahrávání souborů na server

Většina informačních systému umožňuje v určitých sekcích nahrávat soubory na server. Tyto soubory pak mohou i nemusí být přístupné ostatním uživatelům přes protokol HTTP. Zvláště v prvním případě je nutné klást důraz na zabezpečení, podobně jako při kontrole textového vstupu. V případě souborů kontrolujeme jejich příponu, není zde ale vhodné vyřadit jen určité typy jako php, html apod. Měli bychom naopak specifikovat pouze povolené typy a všechny ostatní zakázat. Seznam povolených přípon závisí na konkrétní aplikaci a konkrétní sekci. Někde například povolíme pouze nahrávání obrázků určitých typů, jinde povolíme typické dokumenty jako pdf, doc, txt a pod. Pokud se jedná o upload obecných souborů, je vhodné povolit také nejpoužívanější archívy jako zip, rar nebo tar. Kromě ochrany přímo v aplikaci bychom měli použít další úroveň zabezpečení, a to přímo na serveru. Adresář, ve kterém nahrávané soubory skladujeme, musí být přístupný z webu, avšak měli bychom v něm zakázat spouštění skriptů a listování. Lze také specifikovat konkrétní typy souborů, které webový server zpřístupní a které ne. Této problematice se budu věnovat v závěru kapitoly 3.5.1 na straně 24. S nahráváním souborů souvisí také ošetření překročení povolené velikosti. Řešení je přehledně demonstrováno v odkazovaných zdrojových kódech.

Odkazy na zdrojové kódy

inc/functions.php (`allowed_files_extensions()`), inc/articles-f.php, inc/articles-f-w.php, stored-files/.htaccess

3.3 Ukládání stavových proměnných

Jak jsem zmínil v úvodu práce, HTTP je protokol bez-stavový. Klient zašle serveru požadavek, server zašle odpověď a komunikace končí. Skript na serveru automaticky uvolňuje prostředky, tedy ruší i všechny použité globální proměnné. Téměř v každé aplikaci se však neobejdeme bez způsobů, pomocí kterých můžeme stav uchovat. Takovým prostředkem může být cookie. Každé cookie může však obsahovat pouze 4 kB dat, není tedy vhodné pro ukládání většího množství informací. Další velkou nevýhodou je to, že uživatel má ke svým cookie přístup a může je libovolně měnit. PHP od verze 4 nabízí ukládání dat do *proměnných relace* (session). Při první inicializaci relace se vytvoří jedinečný neodhadnutelný identifikátor (ID) o délce 32 znaků, který je uložen do cookie uživatele. Samotný obsah proměnné relace je uchován na serveru v zabezpečeném adresáři a je svázán právě se zmíněným ID. Platnost cookie s identifikátorem je standardně nastavena do konce relace, tedy do uzavření všech oken prohlížeče. Zrušení relace můžeme samozřejmě vynutit příslušnou funkcí přímo ve skriptu. ID relace lze předávat také jako součást URL v každém požadavku, to pro případ,

že má prohlížeč ukládání cookie zakázané. Toto řešení však představuje větší bezpečnostní riziko, proto je třeba příslušnou konfigurační direktivou vynutit ukládání ID relace pouze do cookie a uživatele na to upozornit současně s návodem, jak cookie v prohlížeči povolit. Použití funkcí pro práci s relacemi včetně popisu jednotlivých konfiguračních direktiv je možno prostudovat v PHP manuálu [7].

Odkazy na zdrojové kódy

inc/functions.php (// `session a nastavení serveru`)

3.3.1 Krádež relace (Session Hijacking)

Z předchozího odstavce je patrné, že pokud by útočník získal identifikátor relace uživatele, může jím přímo v cookie nahradit ten svůj a tím se za uživatele vydávat. Důsledek takové situace je odvozen od mnoha věcí, tou klíčovou je ale určitě oprávnění, které v aplikaci uživatel má.

Možností, jak se zmocnit identifikátoru cizí relace, je několik. Jeho odhadnutí či útok hrubou silou pravděpodobně nikomu nevyjde, o mnoho reálnější je však útok pomocí XSS popsaný na straně 17. Dalším způsobem je odposlechnutí komunikace na úrovni transportní vrstvy, ať už jde o pasivní zachytávání paketů nebo o útok typu MITM (Man in the middle) [12]. ID relace se dá v případě nechráněného protokolu HTTP odposlechnout jak z cookie, tak případně z URL požadavku, pokud se zde přenáší. Tato možnost je k prozrazení ID ještě náchylnější, jelikož URL se ukládá v hlavičce Referer, v historii prohlížečů nebo v historii proxy serverů. Neměli bychom opomenout také zabezpečení souborů na webovém serveru, ve kterých se proměnné relace většinou uchovávají. Této problematice se budu věnovat v kapitolách 3.5.2 a 3.5.3.

Způsob řešení

Předpokládejme, že se útočníkovi podaří získat identifikátor. Existuje několik mechanismů, kterými můžeme únos relace rozpoznat. Prvním z nich je kontrola IP adresy klienta, kterou uchováváme v proměnné relace a kontrolujeme ji při každém požadavku. Pokud se změní, relaci uzavřeme. Zde však narážíme na problém proxy serverů, pod jejichž veřejnou IP adresou často vystupuje celá podniková nebo školní síť. Podobné je to s dnes velmi rozšířenými bezdrátovými sítěmi, kde mohou opět stovky počítačů vstupovat do internetu prostřednictvím jediné brány. Jak je známo, v nezabezpečené bezdrátové síti může navíc odposlouchávat každý, kdo se do ní připojí. Pokud takový člověk získá ID něčí relace, kontrola IP adresy nepomůže, protože adresa se nezmění. Nicméně je vhodné tuto kontrolu implementovat.

Dalším krokem by měla být kontrola hlavičky User-Agent zaslané prohlížečem. Její změna může opět identifikovat pokus o útok, pokud bylo stejné ID relace zasláno z jiného prohlížeče. Ochrana však opět není zaručena, protože pro útočníka není problém nastavit hlavičku User-Agent tak, aby odpovídala hlavičce prohlížeče uživatele, kterou lze rovněž odposlechnout, stejně jako ID

relace. Podobný způsob kontroly může být postaven na hlavičce Referer. Tu však nemusí všechny prohlížeče posílat, proto se této kontrole raději vyhněme.

Třetím údajem kontrolovaným při každém požadavku může být časové razítko, které se zároveň při kontrole obnoví. Kontrolu lze nastavit třeba tak, že pokud je časový rozdíl od posledního požadavku větší než 30 minut, dojde ke zrušení relace. Tuto funkci obstarává i samo PHP a konkrétní dobu platnosti relace lze nastavit příslušnými konfiguračními direktivami. Je však vhodné hlídat tento čas přímo ve skriptu s tím, že konfigurace PHP bude nastavena na prodlevu o něco málo větší. Můžeme tak před zrušením relace zajistit uložení rozpracovaného stavu aplikace, upozornit uživatele, že byl automaticky odhlášen, a nabídnout mu opětovné přihlášení s obnovením stavu.

Popsané způsoby ochrany mohou pomoci ve chvíli, kdy útočník získal ID relace. Pro zaručení skutečné ochrany bychom se však měli zaměřit na to, aby se identifikátor k útočníkovi vůbec nedostal, tedy musíme nějakým způsobem zajistit jeho utajení. Obranu proti Cross-Site Scripting jsem již zmínil na straně 17. Co se týče odposlechnutí požadavku, jedinou obranou je použití šifrované komunikace prostřednictvím protokolu HTTPS, o kterém bude řeč v kapitole 3.4.2 na straně 23. V takovém případě sice lze pakety zachytávat, jejich obsah ale bude zašifrovaný a pro útočníka tudíž nečitelný.

Odkazy na zdrojové kódy

auth.php, inc/functions.php (// `automatické odhlášení`)

3.3.2 Podstrčení relace (Session Fixation)

Při tomto typu útoku zná útočník ID relace uživatele ještě před tím, než jej získá sám uživatel. Nemusí se tedy zabývat tím, jak identifikátor odcizit. Útočníkovi stačí na stránku vstoupit, čímž většinou automaticky inicializuje relaci a dostane přidělený identifikátor. Předpokládejme dále, že server umožňuje předávat ID relace v URL. Útočník takové URL včetně identifikátoru připraví a donutí uživatele, aby pomocí něj na stránku vstoupil. Může to udělat například pomocí e-mailu nebo umístěním odkazu někde na webu. Ve chvíli, kdy se následně uživatel do naší aplikace navíc přihlásí pod svým jménem, útočník použije stejný identifikátor a tváří se pro server jako dotyčný uživatel. V tuto chvíli je již situace stejná, jako při klasické krádeži relace.

Způsob řešení

Stejně jako při únosu relace, i zde se vyplatí použití sekundárních opatření jako je kontrola IP a User-Agent. Zde navíc opět vidíme velkou nevýhodu v předávání ID relace v URL. Můžeme tuto možnost přímo zakázat konfigurační direktivou PHP, avšak problém stále není zcela vyřešen. K podstrčení ID relace může útočník použít i cookie, pokud získá přístup do počítače oběti. Naštěstí existuje jednoduchá a účinná obrana. V průběhu autentizace, kdy je totožnost uživatele ověřena,

vygenerujeme nový identifikátor relace pomocí funkce `session_regenerate_id()`. Starý identifikátor je tedy v tu chvíli útočníkovi k ničemu.

Odkazy na zdrojové kódy

auth.php

3.4 Zabezpečení přenosu dat v transportní vrstvě

3.4.1 Princip šifrování

V této podkapitole se pokusím shrnout to nejpodstatnější z té oblasti kryptologie, která se týká zabezpečení přenosu dat na úrovni síťových protokolů.

Šifrování můžeme obecně rozdělit na dva typy. Prvním z nich je *symetrické šifrování* v podobě nejznámějších algoritmů jako je DES, IDEA nebo AES. Pro šifrování i dešifrování zprávy je použit stejný klíč, který musí vlastnit obě strany. V praxi je tedy za potřebí realizovat oddělený a pokud možno bezpečný kanál, kterým se společný klíč přenese. Může to být např. e-mail, pozemní pošta či osobní styk. Druhým typem je *asymetrické šifrování*, kde se využívá páru klíčů – soukromého a veřejného. Soukromý klíč zůstává vždy pouze "v rukou" jeho majitele, veřejný klíč dá naopak majitel k dispozici komukoliv, kdo má zájem s ním navázat bezpečnou komunikaci. V případě požadavku na utajení zprávy zašifruje odesílatel tuto zprávu veřejným klíčem příjemce a odešle. Cestou ji může někdo odposlechnout, ale nedokáže ji dešifrovat, jelikož to lze provést pouze příslušným soukromým klíčem z páru, a ten vlastní pouze příjemce. Ze znalosti veřejného klíče nelze žádným způsobem odvodit klíč soukromý. Nejznámějším algoritmem je RSA, Diffie-Hellman nebo DSA.

Asymetrické šifrování má tu výhodu, že není vyžadováno předávání tajného klíče. Velkou nevýhodou je však jeho výpočetní náročnost a tím pádem nízká rychlost oproti šifrování symetrickému, k přenosu většího množství dat je tedy prakticky nepoužitelné. Proto se využívá tzv. kombinované šifrování, kdy zprávu zakódujeme symetrickou šifrou a tajný klíč k jejímu dekódování, který je malý, zašifrujeme asymetricky veřejným klíčem příjemce. Ten pak tajný klíč dešifruje svým soukromým klíčem a může tak dekódovat zprávu.

Zatím jsem se nezmínil o způsobu, jakým můžeme u asymetrického šifrování ověřit, že veřejný klíč, kterým zprávu šifrujeme, patří určitě správnému příjemci, nebo zda majitelem veřejného klíče, jímž zprávu dešifrujeme, je opravdu požadovaný odesílatel. Prostředkem k tomuto ověření je tzv. *certifikát*, který potvrzuje párovost konkrétního soukromého a veřejného klíče a jejich náležitost ke konkrétní fyzické nebo právnické osobě. Tímto certifikátem se majitel musí protistraně prokázat, jelikož obsahuje právě jeho veřejný klíč. Ale opět je zde otázka, jak prověřit pravost certifikátu? Certifikát může být vydán pouze důvěryhodnou institucí zvanou certifikační autorita (CA). Ta prověří

osobu, jíž má být certifikát vydán, a podepíše jej svým soukromým klíčem. Každá CA dává veřejnosti k dispozici svůj veřejný klíč, pomocí kterého si kdokoliv může ověřit pravost certifikátu, který tato autorita vydala.

3.4.2 Protokol HTTPS, jeho použití a význam

Oproti protokolu HTTP je rozdíl pouze v tom, že spojení je ustaveno přes šifrovaný kanál vytvořený protokolem SSL (Secury Socket Layer) nebo novějším TLS (Transport Layer Security). Ustavení bezpečného spojení (handshake) mezi webovým prohlížečem a serverem probíhá následovně: Klient pošle serveru požadavek na SSL spojení, domluví si verzi použitého protokolu a další informace. Server odešle klientovi svůj certifikát. Klient (prohlížeč) má k dispozici seznam veřejných klíčů (certifikátů) důvěryhodných certifikačních autorit, pomocí nichž pravost certifikátu zaslaného serverem ověří. Klient dále vygeneruje tajný klíč, kterým se bude symetricky šifrovat komunikace. Tento tajný klíč zašifruje veřejným klíčem serveru a následně mu jej zašle. Server dešifruje tajný klíč svým soukromým klíčem a na základě toho dokončí ustavení spojení, které je od této doby šifrováno domluveným tajným klíčem. Útočník, který může odposlouchávat komunikaci mezi serverem a klientem se z takto zakódovaných dat nedozví vůbec nic.

Použití protokolu HTTPS jsem naznačil již v předcházejících kapitolách. Jedná se zejména o bezpečný přenos hesla při autentizaci a dále pak například o utajení identifikátoru relace. Co se týče získání certifikátu pro vlastní doménu, je třeba oslovit jednu z certifikačních autorit. U nás je to například První certifikační autorita nebo CA Czechia. Certifikát s platností 1 rok stojí kolem 1 000 Kč. Dále je potřeba zajistit podporu SSL komunikace na webovém serveru. Pro server Apache je k dispozici například modul OpenSSL [13] nebo Apache-SSL [14]. Pro účely testování nám postačí vygenerovat si vlastní certifikát například pomocí `openssl` v linuxu, nebo lze využít vydávání bezplatných testovacích certifikátů zmíněnými CA. Pro platformu Windows neexistuje samostatná binární distribuce Apache s podporou SSL, lze však využít některého z balíků Apache + PHP + MySQL. Například projekt XAMPP [15] je jedna z kvalitních distribucí pro různé operační systémy, která má podporu OpenSSL zabudovanou.

Samotná implementace SSL/TLS protokolu opravdu zajišťuje bezpečnou komunikaci, přesto můžeme narazit na problém, jehož příčinou je opět lidský faktor. Uvažme situaci, kdy prohlížeč při vstupu na zabezpečenou stránku zobrazí varování, že platnost certifikátu vypršela, nebo že byl certifikát podepsán neznámou certifikační autoritou. Průměrný uživatel nejspíš nebude ničemu z toho rozumět a zkrátka klepne na tlačítko OK, aby se na stránku dostal. Zdaleka při tom nebude tušit, že nekomunikuje přímo s požadovaným serverem, ale s útočníkem, který klienta oklamal např. pomocí ARP nebo DNS spoofingu ([16], [17]) a který mu tuto komunikaci pouze zprostředkovává a zároveň doposlouchává všechna data, jelikož je jako majitel falešného certifikátu dokáže dešifrovat. Zde

vidíme, že ani použití protokolu HTTPS nám nezaručí naprostou bezpečnost ve všech ohledech. Vedle toho jsme si opět potvrdili, že je nutné použít bezpečnostní mechanismy na více úrovních.

Další problém může nastat s průběžnou změnou protokolu zpět na HTTP. Ve chvíli, kdy se uživatel aplikaci autentizuje přes protokol HTTPS, je vše v pořádku. Pokud mu však v době jeho přítomnosti v chráněné oblasti aplikace dovolíme přejít zpět na protokol HTTP, cookie včetně identifikátoru relace je ihned volně odposlechnutelné. Všechny interní odkazy v naší aplikaci proto musí být generovány tak, aby zachovaly současně používaný protokol. Ani tato ochrana však nezabrání uživateli v tom, aby změnil protokol ručně v adresovém řádku prohlížeče. PHP nám však před takovou chybou uživatele umožní ochránit alespoň to nejdůležitější – identifikátor relace. Již jsem zmínil, že jej budeme uchovávat v cookie. Zapnutím PHP direktivy `session.cookie_secure` zajistíme, že cookie s ID relace se nikdy nepošle z klienta na server, pokud nebude komunikace vedena přes šifrovaný kanál, tedy protokolem HTTPS. Někdy však může být takové nastavení příliš omezující, jelikož nám neumožní použít cookie (a tím pádem i relace) v té části aplikace, kde není třeba autentizace a výchozí protokol je nezabezpečený.

Pokud by se jednalo o IS s vysokými nároky na zabezpečení, měli bychom používat všude HTTPS vynuceně pomocí kontroly protokolu v hlavičce požadavku. V naší aplikaci je však použita jen zmíněná částečná ochrana, kdy je `session.cookie_secure` vypnuto a pokud je uživatel přihlášen, je zajištěno pouze "udržení" protokolu HTTPS pomocí kontroly generovaných odkazů.

3.5 Zabezpečení na úrovni serveru

3.5.1 Konfigurace webového serveru

Na víceuživatelských serverech je základním předpokladem správné nastavení vlastnictví a oprávnění k souborům a adresářům aplikace. Každý webmaster by si měl dobře prostudovat dokumentaci použitého webového serveru a seznámit se s možnostmi jeho konfigurace. Tato diplomová práce je vyvíjena na serveru Apache 2, budu se tedy věnovat jemu. V případě placeného hostingu se nemusíme starat o zabezpečení přístupových práv do domovského adresáře našeho webu a ke spouštění našich skriptů. Pro případ provozu vlastního serveru, který bude hostit více webů a využívat tím pádem virtuální hosting, není na škodu zmínit, jakým způsobem lze takového řízení přístupu docílit. Řešení obstará balíček *suPHP* [18], který je třeba získat z domovské stránky projektu a zkompilovat společně s Apache. Každý domovský adresář a skripty jednotlivých virtuálních hostitelů musí mít odlišného vlastníka, jemuž pak jako jedinému nastavíme speciální direktivou v konfiguraci virtuálního hostitele oprávnění spouštět pouze vlastní skripty a číst vlastní soubory a adresáře.

Pronajatý hosting by měl umožnit důležitá nastavení prostřednictvím souborů `.htaccess` v jednotlivých adresářích aplikace, na vlastním serveru můžeme použít totéž, nebo provést

konfiguraci přímo v `httpd.conf`. Svojí pozornost zaměřme především na tyto direktivy a jejich nastavení:

`AuthType`, `AuthName`, `AuthUserfile`, `AuthGroupFile`, `Require` – Jsou důležité, pokud se rozhodneme použít HTTP autentizci na úrovni serveru.

`Options` – pro znemožnění listování obsahu adresáře nesmí být přítomen parametr `Indexes`.

`Allow`, `Deny`, Order – Slouží pro nastavení přístupu do adresáře pomocí HTTP. Například je vhodné úplně zakázat tento přístup do adresáře, v němž leží skripty vkládané od aplikace pomocí `require()` nebo `include()`.

`php_flag engine off` – Direktiva s těmito parametry zakáže interpretaci PHP skriptů v adresáři. Může se hodit například pro úložiště souborů určených ke stažení. Je přístupná pouze při použití PHP jako modulu Apache. Toto nastavení je však nutné doplnit o blokovou direktivu `<FilesMatch>` (ta už je dostupná v rámci samotného Apache), ve které lze pomocí regulárního výrazu specifikovat typy souborů, které budou z webu přístupné.

3.5.2 Konfigurace PHP

Kompletní seznam a popis konfiguračních direktiv PHP lze najít v PHP Manuálu [7]. U každé direktivy je uvedeno, zda ji lze nastavit všude, tedy i přímo ve skriptu (PHP_INI_ALL), nebo pouze v php.ini, .htacces a httpd.conf (PHP_INI_PREDIR), nebo jen v php.ini a httpd.conf (PHP_INI_SYSTEM). Pokud využíváme placeného hostingu, je třeba ověřit, jak jsou na serveru jednotlivé direktivy nastaveny (např. pomocí funkce `phpinfo()`). Zvláště pak ty systémové (PHP_INI_SYSTEM), které nemůžeme sami změnit. Zde si uvedeme ty direktivy, které by nám neměly uniknout v souvislosti se zabezpečením.

`display_errors` – Při ladění aplikace bývá zapnutá, v ostrém provozu je však rozhodně třeba nastavit na "0". Chybová hlášení PHP stroje mohou útočníkovi leckdy prozradit zajímavé informace jako názvy skriptů a jejich strukturu. Při nevhodné úrovni `error_reporting` mohou být prozrazeny i názvy proměnných. Chybová hlášení lze raději vypisovat do souboru na serveru nastavením direktivy `error_log`.

`disable_functions` – Umožní definovat seznam PHP funkcí, které nebudou ze skriptů přístupné.

`magic_quotes_gpc` – Zajistí automatické escapování speciálních znaků v proměnných GET, POST a COOKIE. Doporučuji nastavit na "On" v souvislosti s řešením ochrany vstupu popisované v kapitole 3.2.4 na straně 14.

`register_globals` – Určuje, zda budou proměnné z požadavků (GET/POST/COOKIE) automaticky k dispozici jako globální. Doporučuje se nastavit na "Off". Bez ohledu na to je však třeba pro

přístup k proměnným z požadavků ve skriptu používat superglobální pole a důsledně hlídat inicializaci globálních proměnných (viz kapitola 3.2.6 na straně 17).

`safe_mode` – Zapnutím této direktivy a nastavením dalších k ní přidružených můžeme nastavit omezení týkající se přístupu k souborům, spouštění skriptů, volání systémových funkcí, přístupu k proměnným prostředí a pod. Do budoucna se však s použitím této direktivy nepočítá, z připravované verze PHP 6 je dokonce zatím odebrána. Na místo ní by se veškerá tato bezpečnostní opatření měla provádět již na úrovni webového serveru, viz předchozí podkapitola.

`session.gc_maxlifetime`, `gc_probability`, `gc_divisor` – Nastaví dobu platnosti relace v sekundách a pravděpodobnost, s jakou se budou "staré" relace automaticky rušit.

`session.cookie_lifetime` – Doba platnosti cookie obsahujícího identifikátor relace. Obvykle nastavujeme na "0", aby relace zanikla již v době uzavření všech instancí prohlížeče.

`session.save_handler` – Nastaví způsob ukládání relací, více v následující kapitole.

`session.save_path` – Pokud ukládáme relace do souborů, touto direktivou určíme jejich umístění. Jelikož jsou sobory pojmenovány podle ID relace, adresář rozhodně nesmí být přístupný (ani pro čtení) nikomu kromě správce systému. Samotné soubory budou čitelné pouze uživatelem, pod kterým je spuštěn webový server.

`session.use_only_cookies` – Zapnutím vynutíme použití cookie k uložení identifikátoru relace. Pokud jsou cookies u klienta zakázané, relace se neotevře.

`session.use_trans_sid` – Zapnutím direktivy se bude v případě zakázaných cookies a vypnuté direktivy `session.use_only_cookies` automaticky doplňovat ID relace do URL každého požadavku. Přestože máme `use_only_cookies` zapnutou, `use_trans_sid` pro jistitu vypneme.

`session.cookie_httponly` – Zapnutím direktivy zaručíme, že cookie nebude možné přečíst klientským skriptem. Jelikož však toto nastavení nemusí respektovat všechny prohlížeče, nelze zapnutí této direktivy považovat jako náhradu plnohodnotné ochrany před XSS (viz kapitola 3.2.7 na straně 17).

`session.cookie_secure` – Zapnutá direktiva zajistí, že přenos cookie bude umožněn pouze zabezpečeným protokolem. V případě HTTP tedy nepůjde cookie použít.

3.5.3 Ukládání relací v databázi

Chceme-li se s jistotou vyhnout tomu, že na serveru, který si sami nespravujeme, nastane bezpečnostní incident a někdo získá oprávnění listovat nebo dokonce číst soubory relací, máme možnost naprogramovat si vlastní obsluhu jejich správy. Předpokladem bude existence databázové tabulky, kde minimální požadované atributy jsou ID relace (varchar), její obsah (text) a doba posledního přístupu (datetime). Dále je nutné nastavit PHP direktivu `session.save_handler = user` a implementovat 6 obslužných funkcí pro založení/otevření relace, uzavření relace, čtení obsahu,

zápis obsahu, zrušení relace a pro čištění starých relací. Názvy funkcí si zvolíme libovolně tak, aby nekolidovaly s existujícími, a předáme je jako parametry funkci `session_set_save_handler()`. Celou tuto obsluhu vložíme na začátek každého skriptu ještě před voláním `session_start()`. S relacemi můžeme pracovat obvyklým způsobem, jejich obsah se však místo do souborů bude ukládat do databáze.

Odkazy na zdrojové kódy

inc/session.php

3.5.4 Návrh databáze

Začneme opět u přístupových práv, která je třeba mít pod kontrolou u každé zvláštní databáze. Každý vlastník webového prostoru smí manipulovat pouze s databázemi, které mu byly přiděleny. Je potřeba odepřít oprávnění k úkonům jako je správa databázového serveru, vytváření či rušení databází a pod. Co se týče přístupových údajů použitých k připojení databáze ve skriptu, stačí je vkládat ze souboru uloženého v adresáři, který není přístupný z webu (viz kapitola 3.5.1 na straně 24).

Návrh databáze nesouvisí ani tak s ochranou proti útočníkům. Souvisí však úzce se zachováním integrity dat, což je také jeden z cílů bezpečnosti informačního systému. K základním prostředkům, které z velké části automatizují zachování integrity, patří integritní omezení a transakce. Oba tyto prostředky jsou k dispozici v MySQL při použití úložiště InnoDB, které je třeba definovat při vytváření struktury databáze. Definice integritních omezení nám zajistí, že pokus o zásah do databáze, který by vedl k její nekonzistenci, skončí chybou, kterou ve skriptu snadno zachytíme, zastavíme jeho provádění a vypíšeme obecné chybové hlášení typu "operaci ne nepodařilo provést". Za normálních okolností by tyto situace neměly vzniknout, avšak v rozsáhlých projektech se většinou neubráníme alespoň jedné chybě v implementaci. A právě v takových případech nastupuje ochrana dat pomocí integritních omezení. SQL skript pro vytvoření struktury databáze našeho IS je součástí zdrojových kódů.

Správnou konzistenci databáze lze přesto narušit, například ve chvíli, kdy provádíme několik po sobě jdoucích databázových operací, které jsou na sobě závislé právě z hlediska konzistence. Pokud několik operací proběhne v pořádku a najednou jedna selže, můžeme se dostat do problému. Tyto situace lze řešit zamykáním tabulek, nebo lépe pomocí transakcí. V praxi stačí provést start transakce, dále postupně provádět operace s tím, že každá další bude následovat pouze při úspěchu té předchozí. Pokud jsou všechny operace provedeny, zavoláme COMMIT a potvrdíme všechny změny, pokud jedna z operací selže, zavoláme ROLLBACK a změny se v databázi neprojeví. Transakce v MySQL jsou opět podmíněny použitím úložiště InnoDB.

Odkazy na zdrojové kódy

inc/function.php (`// funkce pro obsluhu mysql transakcí`), member/user-w.php (použití), db-schema.sql (SQL skript pro vytvoření struktury databáze)

3.6 Nebezpečí internetových robotů

V této kapitole nechci rozebírat podstatu spamu ani principy jeho rozpoznávání a filtrování. Jisté je to, že spam se šíří, protože vydělává peníze. A pokud vydělává peníze, jeho autoři jsou jistě ochotni zaplatit za technické prostředky k jeho šíření včetně vývoje softwaru, který bude schopen stále lépe vyhledávat a shromažďovat e-mailové adresy uveřejněné na internetu. Ne jinak tomu je v oblasti nečistého šíření reklamy formou odkazů v diskusních fórech a návštěvních knihách. Takové odkazy mají jednak nalákat návštěvníky a zároveň zvýšit tzv. "page rank" odkazovaných stránek ve vyhledávačích, aby se jejich odkazy řadily na co nejvyšší pozici. Práce takových *robotů* může spočívat také v prolomení různých prostředků, kterými se autoři webových stránek snaží své e-mailové adresy a webové formuláře ochránit. Možná ne všichni roboti jdou ve své důslednosti až tak daleko a mnozí se jistě spokojí s adresami, které najdou v textové podobě. To však nejsme schopni s jistotou říct, protože ke zmíněným vyhledávacím systémům těžko někdy najdeme nějakou dokumentaci. Nezbývá, než se snažit dělat maximum proto, abychom rozpoznání našich e-mailových adres či zneužití našich webových formulářů robotům znemožnili, nebo alespoň omezili na minimum.

3.6.1 Ochrana e-mailových adres

Jedním stále hojně využívaným způsobem je vypsání adresy nikoliv přímo do dokumentu na serveru, ale až u klienta pomocí JavaScriptu. Kód může vypadat například takto:

```
<script type="text/javascript">
document.write('info')
document.write('@')
document.write('retrieverklub.cz') </script>
```

Hned jedna nevýhoda vychází z nutnosti použít JavaScript. Každý, kdo ví něco málo o přístupnosti webu, si určitě uvědomil, že obyčejná webová stránka a tím spíše webový informační systém by měl uživateli poskytnout všechny důležité informace i bez nutnosti použití JavaScriptu. Zrovna e-mailová adresa se bezesporu dá považovat za důležitou informaci. To však stále můžeme při zachování ochrany napravit. Adresu vypíšeme na stránku ve tvaru info[zavináč]retrieverklub.cz a po načtení stránky necháme spustit JavaScriptovou funkci, která dodatečně projde všechny odkazy na stránce, ve kterých nahradí řetězec [zavináč] skutečným zavináčem @. Ale zamysleme se. Když jsou roboti schopni rozpoznávat písmena z kontrolního obrázku u formuláře, proč by se nemohli naučit interpretovat JavaScript? Mnozí z nich to jistě dělají.

Na internetu můžeme nalézt i rady, že adresu bohatě stačí zapsat pomocí HTML entit:

```
&#105;&#110;&#102;&#111;&#64;&#109;&#111;&#106;&#101;&#102;&#105;&#114;&#109;
&#97;&#46;&#99;&#122;
```

Napsat si funkci, která nám tento kód vygeneruje, není složité. Ve skutečnosti je ale pro robota rozpoznání textu "info@retireverklub.cz" z takovéhoto kódu téměř stejně jednoduché, jako rozpoznání adresy v normálním tvaru.

Naopak velmi silným způsobem ochrany může být zobrazení adresy v podobě obrázku. S pomocí grafické GD knihovny, která je v PHP k dispozici, si můžeme napsat funkci, která nám na základě parametrů vygeneruje z textu obrázek v požadované barvě včetně velikosti písma a průhlednosti pozadí. Standardní sada fontů v GD však možná nemusí každému zapadnout do designu stránky. Jako nevýhodu může spousta autorů a především uživatelů považovat i to, že na obrázek nejde kliknout, aby se automaticky otevřela nová zpráva v poštovním programu s před-vyplněnou adresou. To je samozřejmě pravda a málokterý uživatel si uvědomí skutečný důvod, proč je e-mailová adresa napsána takto "divně". Vyhovět mu bohužel nemůžeme, jelikož uzavřením obrázku do hypertextového odkazu s atributem `href="mailto:info@mojefirma.cz"` bychom celou ochranu dělali zbytečně. Dalším "proti" je opět přístupnost. Bez podpory obrázků se adresa zkrátka nezobrazí, hlasová čtečka ji také nedokáže rozeznat.

Je tedy lepší dát přednost komfortu uživatele, přístupnosti, nebo silné ochraně adresy? Jistý kompromis mohu stručně uvést přímo v podobě jednoho řádku zdrojového kódu:

```
info<img src="images/zavinac.png">retrieverklub<img src="images/tecka.png">cz
```

Stejně jako v předchozím případě se ani zde nemá robot "čeho chytit" a adresu rozpozná jen velmi těžko. Ochranu tedy můžeme považovat za silnou a komfort uživatele se tím alespoň trochu zvýšil. Atribut `alt` můžeme u obrázků vyplnit textem " `zavináč` " nebo " `tečka` ", čímž zajistíme čitelnost adresy i těm, kteří mají web bez obrázků, nebo jsou závislí na jiném než vizuálním kontaktu s počítačem. Adresu lze bez problémů vybrat a zkopírovat do poštovního programu, kde do textu stačí doplnit znak zavináče a tečky. Obrázky těchto znaků můžeme připravit tak, aby nenarušovaly vzhled. Je vhodné vytvořit i funkci, která bude náhradu těchto znaků za tagy obrázků automatizovat.

Když jsem mluvil o uživatelích, kteří naši snahu v tomto směru nejspíš vůbec neocení, protože ani neví, proč by měli, najdou se na druhou stranu zase ti, kteří váhají při vyplňování registračního formuláře z důvodu obavy, zda jejich adresa nebude někde vystavena. Je vhodné na tomto místě upozornit, že adresa bude zveřejněna v chráněném tvaru, nebo že nebude zveřejněna vůbec.

Odkazy na zdrojové kódy

inc/functions.php (`protect_email()`)

3.6.2 Ochrana webových formulářů

Některé webové formuláře mohou sloužit zároveň jako ochrana e-mailové adresy, jelikož umožní uživateli odeslat e-mail přímo ze stránek, aniž by musel znát e-mailovou adresu nebo otvírat poštovní program. Dle mého názoru by však e-mailová adresa měla být v kontaktech vidět, ať už v podobě textové, obrázkové nebo kombinované. Nicméně takový kontaktní formulář není na škodu přidat jako pohodlný způsob jednorázové komunikace. Spamoví roboti však na internetu vyhledávají nejen e-mailové adresy, ale i nezabezpečené poštovní servery, prostřednictvím kterých následně spam distribuují. K tomu mohou využít i takovéto kontaktní formuláře, které nejsou zabezpečené. Problém se týká ale i formulářů, které neodesílají e-mail. Mohou však například odesílat data jen do systému a tak jej zahltit.

Velice jednoduchou ochranou může být kontrola času, který uběhne mezi zobrazením formuláře a jeho odesláním. Pokud je řekněme tento čas kratší než 1 sekunda, pravděpodobně formulář odeslal robot, protože člověk by ho nestačil vyplnit. Nám stačí, když si jako součást formuláře uložíme do skrytého pole čas zobrazené formuláře. Nikdo nám však nezaručí, že někteří roboti nebudou před odesláním "čekat", protože s touto ochranou počítají.

Podobně jako k ochraně e-mailu můžeme i zde využít JavaScript, kterým dodatečně doplníme jedno skryté formulářové pole, jež následně kontrolujeme při zpracování. Nevýhoda je rovněž naprosto stejná jako u e-mailu – bez JavaScriptu se formulář nedá odeslat, roboti mohou JavaScript zvládat.

Velmi používanou ochranou je takzvaný sytém CAPTCHA [19] (akronym pro "completely automated public Turing test to tell computers and humans apart"). Tento systém má tedy odlišit skutečné uživatele od robotů tak, že při vyplňování formuláře požaduje po uživateli něco, co robot není schopen rozpoznat nebo splnit. Může se jednat o generovaný kód v podobě obrázku, který je uživatel nucen opsat do formuláře. Obrázky jsou generovány tak, aby nebylo možné je strojově identifikovat. Co se týče přístupnosti webu, je to však způsob více než nevhodný. Málokterý autor webu půjde s tímto generováním obrázků tak daleko jako např. Google, aby umožnil strojové čtení handicapovaným uživatelům. Některé obrázky CAPTCHA jsou kvůli obavě z vyspělosti robotů ohledně OCR rozpoznávání generovány tak, že je těžko rozezná i člověk. Možná někoho napadne, proč zkrátka negenerovat jen textový kód? Tato ochrana nefunguje. Roboti zdá se zkouší vyplnit formuláře vším možným, tedy i textem vyskytujícím se poblíž polí. Jeden z 20 způsobů vyplnění nakonec robotům projde. Jednou z možností by mohl být absolutně pozicovaný prvek, ve kterém se kód vyskytuje. Ten může být v HTML kódu umístěn kdekoliv a napozicován na správné místo pomocí CSS. Opět se ale dostáváme do křížku s přístupností.

Jako nejúčinnější se zdá být ochrana pomocí logické otázky. Je na úvaze a nápaditosti autora, jako zvolí metodu. Můžeme požadovat odpověď na otázky typu "kolik je součet 3 a 2" nebo "jaké je hlavní město ČR". Jeden z podobných způsobů jsem použil také v našem IS. Skript vygeneruje kód

o zadaném počtu míst (typicky 4 nebo 5) a požaduje po uživateli, aby z kódu opsal znak z určité pozice. Pro zajištění silnější ochrany stačí modifikovat tento systém tak, aby požadoval zadání 2 nebo 3 znaků z určitých pozic. Taková metoda ochrany splňuje požadavky na přístupnost a je roboty velmi těžko prolomitelná.

Odkazy na zdrojové kódy

inc/function.php (`gen_code()`), adverts-i.php, adverts-w.php

3.7 Audit a zálohování

3.7.1 Protokoly událostí

V každém informačním systému by měl být zaveden systému auditu událostí. Vždy záleží na konkrétním případě aplikace. Je potřeba stanovit, jaké operace budou v systému prováděny, jaké podobnosti bude každý záznam obsahovat, do jakých skupin je lze operace třídit, které operace je třeba auditovat a které ne, případně kdo bude mít ke čtení záznamů přístup. Obecně by měl každý záznam obsahovat datum a čas, kdy k události došlo, dále kdo událost vyvolal nebo provedl, typ události (přidání, úprava, odstranění apod.), objekt, kterého se událost týká (v každém systému jsou různé objekty) a případně informace, které doplní záznam o důležité podrobnosti.

V našem IS je implementace systémového auditu vyžadována z několika důvodů. V praxi například může nastat problém, kdy některý ze správců zadá do systému chybné informace. Jelikož ke správě jedné sekce může mít přístup více uživatelů, je třeba využít protokolů událostí, které pomůžou zpětně dohledat, kdy ke změně došlo a kdo je za ni zodpovědný. Další operace, jejíž záznam do auditu je velmi důležitý, je změna oprávnění uživatelů. Tu může provést pouze vybraný člen výboru prostřednictvím administrátorského účtu. Všichni správci, kteří mají přístup k protokolům událostí, mohou díky nim tyto změny registrovat a kontrolovat. Samotné povědomí o tom, že téměř každá událost, kterou kdokoliv v systému provede, je zaznamenána, zabrání uživatelům provádět v administraci něco, co by mohlo být v rozporu s dohodnutými pravidly a stanovami. V neposlední řadě může audit pomoci k vyřešení problémů v případě aplikační chyby v průběhu testování nebo provozu. Systémový správce má jako jediný přístup k záznamům obsahujícím podrobná chybová hlášení, na základě kterých může vzniklé chyby snadněji odhalit a odstranit. Nikdo kromě systémového správce by neměl mít možnost záznamy v protokolech mazat nebo upravovat. Není k tomu ani důvod. V aplikaci můžeme napevno nastavit dobu platnosti záznamů a zajistit, aby se např. události staré více než 1 rok automaticky mazaly. Z důvodu velkého množství záznamů musí v systému existovat možnost, jak v protokolech vyhledávat.

Odkazy na zdrojové kódy

inc/system-def.php (`// definice pro audit`), inc/function.php (`audit_event()`, `audit_info()`), admin/adverts-w.php (příklad záznamu událostí), admin/audit.php (výpis záznamů)

3.7.2 Záloha a obnovení dat

Zálohování dat obvykle provádíme na více úrovních. Tou první je záloha na úrovni serveru, kdy pravidelně spouštěná úloha zálohuje obsah databáze, a to většinou na jiný fyzický počítač. V případě nutnosti je systém z této zálohy obnoven přímo systémovým správcem. Další způsob zálohování dat bychom měli vyřešit na úrovni samotné aplikace. Se systémem pracují běžní správci sekcí, kteří mohou dělat běžné chyby. Nebude tedy neobvyklé, že správce vymaže položku, kterou vymazat nechtěl a podobně. V naší aplikaci má správce s příslušnými oprávněními možnost vytvořit *bod obnovení*, což je stav databáze, ke kterému se lze kdykoliv vrátit. Automatický bod obnovení může systém tvořit sám s nastavenou frekvencí, a to pomocí pravidelného spouštění skriptu.

Implementační řešení v našem IS je založeno na použití databázové utility `mysqldump`, která exportuje do souboru všechny databázové tabulky kromě systémových nebo těch, které mají pevně daný obsah, jež se nemění. Každý bod obnovení je tvořen tímto exportovaným souborem a záznamem v databázi, který tento bod jednoznačně identifikuje a umožní uchovat jeho datum a čas vytvoření, případně popis. Návrat k určitému bodu obnovení zajistíme vyčištěním tabulek a importováním dat ze souboru pomocí příkazu `mysql`.

Odkazy na zdrojové kódy

inc/function.php (`recovery_create_point()`), admin/recovery.php, admin/recovery-w.php

3.8 Shrnutí

V předchozích podkapitolách jsem zmínil problematiku hrozeb, proti kterým můžeme jako programátoři zajistit ochranu přímo na úrovni naší aplikace. V mnoha popsaných situacích se potvrdilo, že je stále třeba mít na paměti zajištění ochrany na více úrovních, pokud je to možné. Prolomení ochrany na jedné úrovni tak díky další ochraně nemusí znamenat napadení aplikace. Velmi důležitá je při programování důslednost, zvláště při inicializaci proměnných nebo při ošetření výstupu do HTML dokumentu. Jeden nefiltrovaný výstup z tisíce neumožní provést útok tak snadno, jako kdyby nebyl ošetřen žádný výstup, možnost nalezení této jedné chyby zde ale přesto existuje. Při implementaci je velmi vhodné využít některého z nástrojů pro detekci zranitelných míst. Jedním z nich je například WebInspect z dílny SPI Dynamics, jehož zkušební verzi jsem použil pro testování zabezpečení našeho IS. Výsledky jsou zhodnoceny v závěru práce. Dalším nástrojem, použitelným pro skenování webových aplikací je Watchfire AppScan nebo Acunetix Web Vulnerability Scanner.

Většina útoků, o kterých jsem se zmínil, se týkala přímého nebo nepřímého získání přístupu do systému. Jiným rozsáhlým tématem by mohla být skupina útoků označovaná jako **DoS – Denial of Service** (odmítnutí služby nebo odepření přístupu). Jedná se o útoky s cílem zahltit server nebo konkrétní službu (v našem případě HTTP). To lze provést například záplavou velkým množstvím paketů nebo zasíláním speciálně upravených paketů, které mohou zneužít chyby v implementaci síťových protokolů a zapříčinit tak vyčerpání systémových prostředků serveru. Proti těmto typům útoku je obrana velmi nesnadná. Řeší se na různých úrovních síťového modelu, včetně aplikační vrstvy. Nesouvisí však s implementací naší aplikace, která je tak v tomto ohledu plně závislá na dostupnosti použitého protokolu HTTP nebo HTTPS. V případě zájmu o informace z této oblasti mohu čtenáře odkázat například na sérii článků o DoS či DDoS (Distributed Denial of Service) na serveru lupa.cz [20].

4 Užití postupů softwarového inženýrství při vývoji IS

Pojem softwarové inženýrství se začal široce používat od konce 70. let 20. století. Lze jej chápat jako oblast počítačové vědy, která se zabývá vytvářením rozsáhlejších softwarových systémů budovaných většinou týmem nebo i více týmy vývojářů. Softwarovým projektem rozumíme plánovanou činnost, jejímž výsledkem má být dodání softwarového produktu nebo služby. V našem případě bude softwarovým produktem klubový informační systém v podobě webové aplikace. Je však třeba upozornit na to, že softwarový projekt vždy vzniká na základě potřeby zadávající organizace vypořádat se s nějakým problémem v bussines procesu, anebo potřeby tento proces nějakým způsobem vylepšit například za účelem zvýšení konkurenceschopnosti. Retriever klub CZ sice není výdělečnou organizací, ale požadavky na vytvoření vlastního IS také vyvstaly z určitých potřeb sdružení. Těm se budeme věnovat v kapitole 5.

4.1 Fáze a modely životního cyklu

4.1.1 Analýza požadavků

Jedná se o analýzu a specifikaci požadavků zákazníka, pro kterého systém vyvíjíme. Zahrnuje řadu inženýrských činností a technik a představuje jeden z nejtěžších úkolů při řešení každého softwarového projektu. Problémem je zejména komunikace mezi zákazníkem a vývojářem. Zákazník často není schopen zformulovat jasně svůj požadavek, může dávat přehnané požadavky, požadavky, které si vzájemně odporují nebo jsou v rozporu s požadavky jeho kolegů. Komunikace je založena na přirozeném jazyce a získávání požadavků tedy probíhá formou konzultace na společných schůzkách. Vývojáři často používají různé diagramy ke znázornění a upřesnění požadavků. Mezi nejčastější patří *diagram případů použití*, který bývá základním kamenem podrobného návrhu aplikace a zároveň je dostatečně srozumitelný i pro zákazníka, čehož můžeme využít právě při získávání požadavků.

4.1.2 Návrh systému

Takto označujeme popis struktury softwarového systému, který má být implementován. Dále popis dat, které jsou součástí systému, popis uživatelského prostředí, případně popis použitých algoritmů. Popis dat v případě informačních systémů představuje většinou návrh databáze.

Modelování při analýze požadavků nebere v úvahu omezení, která vyplývají z implementace. Naopak návrh již počítá s platformou, na které bude systém implementován, a s možnostmi, které

nám zvolené rozhraní umožní. Může se zdát, že návrh začne právě tam, kde skončí analýza požadavků, v praxi však přesná hranice neexistuje a obě fáze se v některých místech navzájem překrývají a doplňují. Jedním důvodem je v současnosti používaný jazyk UML [21], který poskytuje nástroje pro analýzu i návrh systému. Výsledky modelování pomocí těchto nástrojů tak mohou být součástí obou fází. Příkladem je již zmíněný diagram případů použití. Druhým důvodem, pro který nezle striktně tyto fáze oddělit, jsou dnes používané iterativní *modely životního cyklu*, ve kterých procházíme všemi fázemi životního cyklu opakovaně. Výsledkem každé iterace je nějaký přírůstek funkčnosti softwarového systému. Během návrhu se tak často vracíme k analýze požadavků, které nebyly do této doby dostatečně specifikovány. Rozpracování modelu analýzy požadavků často označujeme jako detailní návrh, ze kterého následně vychází formalizovaný architektonický návrh.

4.1.3 Implementace

V implementaci nejde o pouhé kódování návrhu do příkazů programovacího jazyka. Návrh algoritmů není vždy součástí návrhu systému a většinu z nich navrhuje až programátor v rámci psaní programu. Programátor se tak do jisté míry podílí i na návrhu systému a opět zde nemůžeme definovat přesnou hranici mezi těmito fázemi životního cyklu.

Moderní programování spočívá nejen v zápisu programu v daném programovacím jazyce, nýbrž je založeno na využití již existujících komponent. Kromě znalosti programovacího jazyka se tedy od programátora vyžaduje i schopnost tyto komponenty nalézt a umět je použít. Moderní CASE nástroje a integrovaná vývojová prostředí dokonce umožňují automatické vytvoření kostry programového kódu z modelů vytvořených při návrhu. Výsledný kód může být také zpracován nástroji reverzního softwarového inženýrství s cílem aktualizovat model návrhu. Programátor se tak účastní cyklu, který nazýváme roundtrip engineering.

4.1.4 Testování a ladění

Implementace je v mnoha projektech nejdelší fází a u některých modelů životních cyklů i dominantní etapou. Může při ní vznikat mnoho chyb, které je potřeba co nejdříve odhalit a opravit. Tento proces označujeme jako testování a ladění. Testování může mít podobu posouzení kvality zdrojového kódu nebo spuštění kódu či jeho části. Dále můžeme rozlišit testování funkční, kdy testujeme, zda program vykonává to, co po něm je požadováno, a testování strukturní, kde vycházíme ze znalosti zdrojového kódu a spuštěním ověřujeme jeho správnost.

Pojmem ladění jsou označovány procesy vedoucí k odstranění nalezených chyb. Může se jednat o chyby syntaktické, které většinou dobře pomůže odhalit vývojové prostředí. Jejich odstranění spočívá ve znalosti programovacího jazyka a technik programování. Oprava sémantických chyb pak většinou představuje posouzení správnosti algoritmu a jeho opravu.

4.1.5 Integrace a nasazení

Integrace znamená sestavení hotových a otestovaných komponent systému do jednoho celku. Přestože jsou jednotlivé komponenty bez chyb, po fázi integrace přichází na řadu integrační testování, protože můžou nastat například problémy při komunikaci jednotlivých komponent. Integrační testování bývá často již předmětem implementace, protože ne vždy lze vyvíjet všechny komponenty odděleně. Rozlišujeme integrační testování shora dolů a zdola nahoru. Shora dolů testujeme v případě, že implementace postupuje od základních komponent na vyšší úrovni k těm drobnějším na nižší úrovni. Komponenty na nižší úrovni však ještě nemusí být implementovány a je nutno za ně dosadit náhražky, které nevyžadují složitou implementaci, ale nahradí danou komponentu tak, aby bylo ty vyšší možno otestovat. Testování zdola nahoru postupuje přesně obráceně a využijeme ho v případě, když implementace postupuje od komponent na nižší úrovni. Vyšší komponenty, které nejsou zatím implementovány, ale mají ty nižší používat, je opět nutné nahradit, tentokrát takzvanými drivery.

Nasazení do provozu probíhá většinou vydáváním verzí systému. Každá verze je tvořena řadou přírůstků označovaných anglickým termínem builds. Ty jsou tvořeny v jednotlivých iteracích vývoje. Nasazení opět zahrnuje testování, které má za úkol ověřit stabilitu, výkonnost, bezpečnost systému či schopnost zotavení po poruše. Testy přímo u zákazníka pak především ověřují, že zákazník dostává to, co chtěl.

4.1.6 Provoz a údržba

V této fázi je sytém již v provozu u zákazníka. Je používán a případně rozvíjen o další funkce. S provozem souvisí také přechod od starého systému k novému. Ten zpravidla nebývá jednorázový, ale probíhá postupně. Oba systémy se například nějaký čas používají souběžně. Údržbou systému rozumíme odstranění chyb či nedostatků odhalených při provozu, dále přizpůsobení systému změnám v podnikovém prostředí, nebo rozvoj systému přidáváním dalších rozšiřujících funkcí nebo zlepšováním kvality.

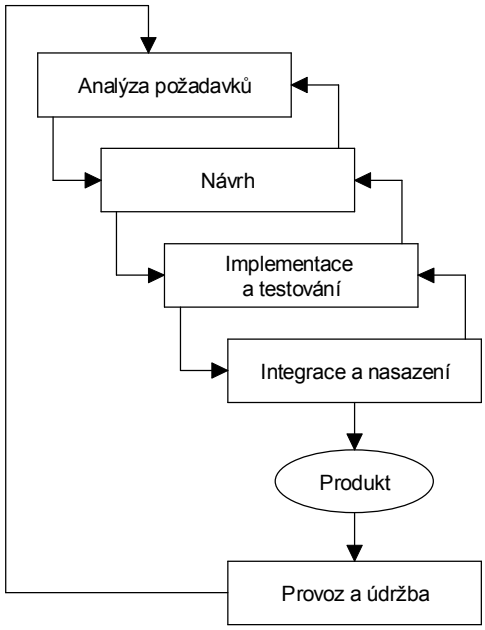
4.1.7 Modely životního cyklu

V předchozích článcích byly popsány jednotlivé fáze životního cyklu. Tyto fáze probíhají systematicky podle určitého předpisu, kterému říkáme model životního cyklu.

Vodopádový model

V tomto modelu je každá fáze započata až po dokončení fáze předchozí. Nevýhodou je, že v praxi často nejsou definovány úplně všechny požadavky na začátku, nebo že zákazník vidí spustitelnou verzi programu až v úplném závěru vývoje. I v tomto modelu tak musí existovat zpětná vazba, protože někdy je v jedné fázi (např. implementační) potřeba provést změnu oproti fázi předchozí

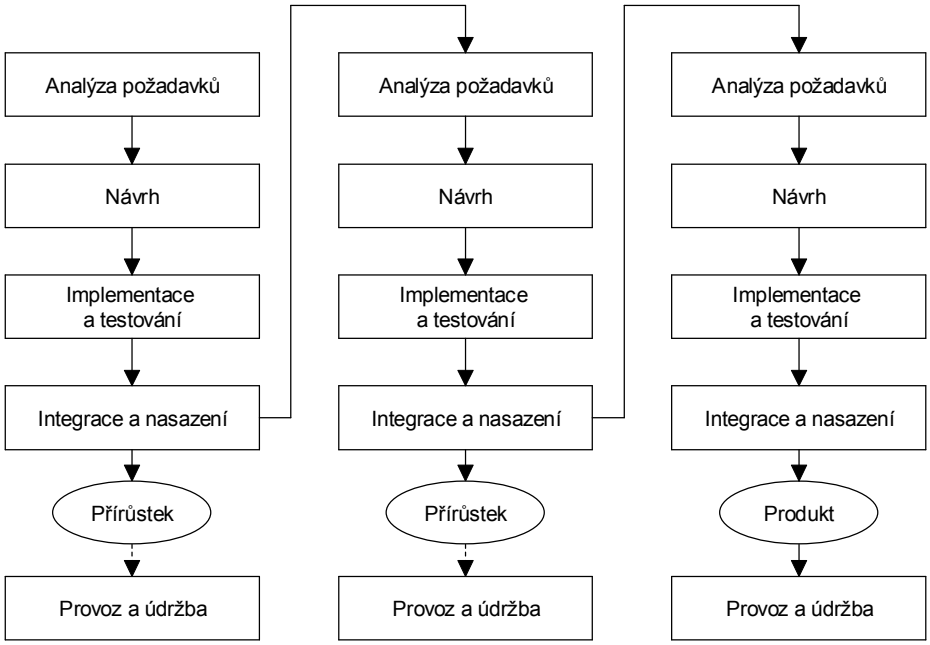
(návrhu), což se do této fáze musí zpětně promítnout. Někdy je kvůli takové změně třeba dojít zpětně až na začátek celého životního cyklu.



Obrázek 4.1.7a – vodopádový model

Iterativní modely

Každá iterace obsahuje analýzu, návrh implementaci, testovaní a někdy i nasazení. Výsledkem iterace je vždy nějaký funkční přírůstek systému. **Inkrementální model** cyklu předpokládá, že v každé iteraci bude vytvořen proveditelný kód, přestože nemusí být implementovány všechny součásti systému. **Spirálový model** postupuje iterativně od sběru požadavků a plánování přes analýzu rizik, vlastní inženýrství až po výsledný prototyp části systému, který je posouzen zákazníkem. Analýzou rizik rozumíme posouzení nákladů, přínosu, hrozeb nebo problémů, které se mohou při řešení vyskytnout. Výsledkem je rozhodnutí, zda daný systém vyvíjet nebo ne. Prototyp nebo finální produkt je vytvořen ve fázi inženýrství (obsahuje v podstatě dříve popsané fáze klasického modelu). Zmínit můžeme také model zvaný **agilní vývoj software**, jehož motivací je především rychlý vývoj systému a rychlé reakce na měnící se podmínky. Přisuzuje větší význam jednotlivcům a týmům před nástroji softwarového inženýrství a úplnou dokumentací. Důraz je kladen také na spolupráci se zákazníkem po celou dobu vývoje.



Obrázek 4.1.7b – inkrementální model

Obrázek 4.1.7c – spirálový model

4.2 Využití SI v tomto projektu

Nejčastěji používaná metoda zjišťování požadavků je interview a byla použitá i v tomto projektu. Techniky interview při zjišťování požadavků mají velmi mnoho společného s interview v žurnalistice. Jedná se o předem připravený rozhovor vedený obvykle jedním dotazujícím s jediným nebo více respondenty. Dotazující strana se nazývá moderátorem. Tato technika má při zjišťování požadavků jisté zvláštnosti. Předpokládá dlouhodobější spolupráci a tím se liší od klasického interview pro noviny. Jsou nutné přesné zápisy. Někdy je třeba interview opakovat, a to v případech, kdy se v průběhu vývoje projektu zjišťují skutečnosti, které je třeba si dodatečně vyjasnit. Interview je třeba dobře připravit, předem shromáždit a vyhodnotit všechny informace a sestavit seznam problémů, u kterých by interview mohlo přispět k jejich řešení.

Dalším nástrojem, který může velmi pomoci při zjišťování požadavků, je diagram případů použití. Jedná se o jeden z nástrojů jazyka UML, který bývá velmi názorný a jednoduchý i pro zákazníka. V tomto projektu se jeho použití během interview velmi osvědčilo. Na diagram případů použití se váží také podrobné scénáře případů použití. Rozsah této práce dovolí uvést jen příklad některého z nich (kapitola 7.1).

Detailní návrh našeho systému byl v podstatě vytvořen souběžně s analýzou požadavků. Pro formální architektonický návrh pak bylo použito právě diagramu případů použití. Pro formální návrh databáze byl využit nejčastější datový model – ER diagram, který dobře znázorní všechny entity obsažené v systému a modeluje jejich vztahy.

Co se týče modelu životního cyklu užitého při tomto projektu, dal by se charakterizovat jako vodopádový. I přes jeho nevýhody jej lze použít, a to zejména proto, že požadavky na náš informační systém se podařilo specifikovat a analyzovat velmi přesně a důkladně během několika interview s využitím diagramů případů použití. Při samotném formálním modelování návrhu bylo nutné použít zpětnou vazbu k analýze požadavků, změny však nebyly nijak razantní a menší problémy byly vyřešeny během jedné konzultace.

O použití UML v praxi dobře pojednává publikace UML srozumitelně [21].

5 Retriever klub CZ - zadavatel projektu

5.1 Organizace

Organizace Retriever klub CZ byla založena roku 1992 s cílem sdružit většinu majitelů a chovatelů retrieverů v České republice. Základním posláním klubu je čistokrevný a zdravý chov všech plemen retrieverů při zachování charakteristických vloh a vlastností specifických pro konkrétní plemeno. Klub by měl zajistit, že kynologické aktivity jeho členů budou v souladu s platným standardem a předpisy mezinárodní kynologické federace FCI, do níž patří také Česká republika. Klub pořádá klubové a speciální výstavy retrieverů, připravuje zkušební řády (retriever je lovecké plemeno psů) a organizuje jejich zkoušky. Publikováním v tištěném Zpravodaji nebo na internetu předkládá veřejnosti ucelené a praktické informace o plemenech retrieverů a všech záležitostech, které se týkají jejich chovu.

5.2 Členství

Členem klubu se může stát každý, kdo řádně vyplní přihlášku a zaplatí členský poplatek. Pokud dříve nebyl žadatel disciplinárně trestán příslušnou komisí klubu, nebrání výboru klubu nic v tom, aby členství žadatele akceptoval. Členové klubu mají řadu výhod. Mezi ně patří například výrazně nižší poplatky za klubové výstavy a zkoušky, bezplatné zasílání Zpravodaje, zprostředkování uchovnění psa nebo feny nebo možnost aktivní účasti na členských schůzích.

Začátkem roku, typicky během celého měsíce ledna, vyhlásí klub období placení členských příspěvků. Člen, který včas zaplatí příspěvek během této doby, má automaticky prodloužené členství na zbytek tohoto roku. Pokud není příspěvek zaplacen do konce platebního období, je členství pozastaveno a dotyčný veškeré výhody ztrácí, jakoby členem klubu nebyl.

5.3 Orgány sdružení

Mezi orgány klubu patří členská schůze, prezident sdružení, kontrolní komise, disciplinární komise a další členové výboru klubu jako matrikář, sekretář, ekonom nebo hlavní poradci chovu. Úkolem výboru a komisí je řídit činnost klubu a organizovat vše, co bylo naznačeno v úvodním odstavci této kapitoly. Členská schůze je svolávána výborem nejméně jedenkrát za rok a můžou se jí zúčastnit všichni členové klubu. Předmětem schůze je typicky úprava stanov klubu, volba členů výboru a komisí, hospodaření, plán činnosti, projednávání podmínek chovu, zkušebních řádů a dalších záležitostí souvisejících s činností klubu.

5.4 Motivace k vytvoření informačního systému

Do dnešní doby toto sdružení nemá elektronický informační systém, který by usnadnil administrativu spojenou s činností klubu a pomáhal jednoduše prezentovat informace prostřednictvím internetu. Je zřejmé, že internet je v dnešní době nejrychleji se rozvíjející veřejné médium a nejvíce vyhledávaný hromadný sdělovací prostředek. Tím pádem zejména mladší generace členů Retriever klubu postrádá to, co by moderní organizace měla nabídnout. Tedy komplexní informační zdroj, jehož obsah bude vždy aktuální, soustředěný na jednom místě a dostupný takřka kdykoliv a odkudkoliv.

Cílem tohoto projektu bude tedy mimo jiné vytvoření webové prezentace, která navenek uspokojí potřeby veřejnosti i členů klubu. Informační část webu bude přeuspořádána tak, aby uživatel rychle našel to, co právě hledá. Prostředí prezentace bude esteticky příjemné, uživatelsky přívětivé a mělo by budit dojem profesionality organizace.

Výhody pro veřejnost

Detailnější pohled na nový informační systém odhalí to, že veřejnosti i členské základně se otevřou zcela nové možnosti, jako využití bezplatné klubové inzerce na internetu, využití anketního systému, vyhledávání informací o uchovněných jedincích, vyhledávání potomků jedinců či procházení rodokmenem. Další novinkou bude velmi praktický přehled chovatelských stanic retrieverů. Díky novému systému bude veřejnosti nabídnuto velmi žádané vyhledávání ve výsledcích klubových výstav a zkoušek. Ne málo nových žadatelů o členství uvítá možnost podat přihlášku do klubu elektronicky a zaplatit příspěvek převodem na účet například prostřednictvím přímého bankovnictví.

Výhody pro členy

Členové klubu navíc získají další výhody. Systém jim například umožní pohodlně změnit své osobní údaje v členské databázi, aniž by museli podávat písemnou žádost výboru. Dále, uchovnění jedinci budou v systému svázáni přímo s majitelem, což umožní členovi jakožto majiteli uchovněného psa přidat k tomuto jedinci vlastní fotografii, která bude prezentována na webu. Každý člen, je-li zároveň chovatelem retrieverů, bude moci do systému zadat informace o jeho vlastní chovatelské stanici, která bude na webu také bezplatně prezentována. Podobně jako první členský poplatek, i všechny následující mohou členové poukazovat převodem na základě systémem poskytnutých platebních údajů. Členství tak bude od této chvíle každému prodlouženo s okamžitou platností a odpadne tak nutnost dokládat na akcích či schůzích potvrzení o zaplacení složenkou.

Výhody v organizačních záležitostech

Mnoho výhod přinese systém samozřejmě také výboru klubu, tedy lidem, kteří se starají o bezproblémový chod celé organizace. Zjednoduší se vyhledávání informací o členech či změna

jejich údajů, které jsou do teď pouze součástí papírových přihlášek soustředěných u jednoho člověka. Nyní bude celá členská základna zanesena do internetového informačního systému a bude přístupná kompetentním osobám odkudkoliv. Tím bude zároveň velmi zjednodušena evidence a kontrola členských poplatků, potažmo kontrola platnosti členství. Předpokládá se, že většina nových členů bude využívat podání elektronické přihlášky, čímž se v podstatě zapíše člen sám a správce jeho členství po schválení výborem pouze potvrdí. Tento postup a některé další drobnosti předpokládají jistou úpravu současných stanov klubu, se kterou se počítá současně s nasazením naší aplikace do provozu.

V neposlední řadě bude k přednostem zavedeného systému patřit jednoduchá správa veškerého obsahu webové prezentace, tedy jednotlivých informačních sekcí a článků. Ty bude moci spravovat i uživatel se základními znalostmi ovládání počítače a kompletně tak odpadne starost se zdlouhavou a časově náročnou aktualizací internetových stránek webmasterem. Publikační systém zároveň zajistí jednotný vzhled celého webu, aniž by správce ochudil o možnosti základně formátovat text, vkládat do článku obrázky, tabulky, odkazy nebo dokumenty.

Dřívější pokus o internetovou inzerci neměl dobré výsledky, jelikož vkládané inzeráty nebyly kontrolovány ani před spamovými roboty, ani před nevhodnými či vulgárními příspěvky. A co víc, v inzerci se běžně objevovaly inzeráty na prodej štěňat bez průkazu původu, což je v rozporu se základními myšlenkami celé organizace. Nový systém umožní správcům pohodlně inzerci řídit a uveřejnit jen to, co zveřejněno být může. Navíc přítomnost této kontroly už sama o sobě odradí případné pokusy o porušení pravidel inzerce.

Po tomto shrnutí je zřetelné, že investovat úsilí za účelem modernizace správy tohoto sdružení se jednoznačně vyplatí. V následující kapitole budou podrobněji rozebrány konkrétní požadavky na vyvíjený systém, na jejichž základě bude teprve možné vytvořit formální návrh internetové aplikace.

6 Specifikace a analýza požadavků

Retriever klub CZ požaduje vytvoření informačního systému v podobě internetové aplikace. Systém bude obstarávat provoz veřejné webové prezentace klubu a správu jejího obsahu pomocí webového rozhraní. Uživatelem systému tedy rozumějme každého člověka, který pomocí webového prohlížeče načte URL naší aplikace. Jednotlivé sekce sytému budou uživatelům k dispozici na několika úrovních přístupu. Jedná se o přístup pro veřejnost (1. úroveň), přístup pro členy klubu (2. úroveň) a přístup pro skupinu správců (3. úroveň). Správcem je člen, který v klubu zastává nějakou funkci, pro jejíž vykonávání je třeba přidělení specifických oprávnění k některým sekcím sytému. Uživatel s vyšší úrovní oprávnění má přístup také ke všem sekcím na nižších úrovních. Jednotlivé sekce budou v systému přehledně hierarchicky uspořádány a bude zajištěno intuitivní procházení jejich obsahem tak, aby se ve struktuře prezentace neztratil ani člověk s pouhými základními znalostmi ovládání počítače. Administrační prostředí pro správu obsahu prezentace bude rovněž intuitivní, přehledné a opatřené výstižnou a úplnou nápovědou.

Následující popis sekcí neobsahuje detailní podrobnosti, které později vyplynou z diagramů použití nebo datového modelu.

6.1 Veřejné sekce

Aktuality – V této sekci má každý uživatel možnost prohlížet články, jejichž předmětem je poskytnout veřejnosti i členům klubu aktuální informace, novinky nebo upozornění. K dispozici bude také archív těchto článků, ve kterém lze procházet po částech.

Informace o klubu – Bude se jednat o několik sekcí v podobě jednoduchých stránek převážně s informačním obsahem. Zahrnují základní informace o klubu, informace o členství, stanovy klubu a důležité kontaktní informace. Každá z těchto stránek bude také k dispozici v podobě vhodné pro tisk. Další součástí bude jakási veřejná evidence finančních prostředků klubu informující o současném stavu financí, o platbách ve prospěch a o platbách na vrub. Půjde pouze o číselné informace, a to vždy souhrnně za 3 kalendářní měsíce (kvartální přehled).

Žádost o členství – Sekce uživateli umožní vyplnit a odeslat elektronickou přihlášku do klubu. Systém následně poskytne uživateli údaje k platbě členského poplatku, po jehož zaplacení je členství platné a na e-mail uvedený v přihlášce jsou zaslány přihlašovací údaje do systému, prostřednictvím kterých bude členům umožněno využít běžně nepřístupné možnosti aplikace (viz kapitola 6.2 na straně 45). K dispozici bude také přihláška ve formátu PDF, kterou si uživatel může vytisknout, vyplnit a poslat na adresu klubu pozemní poštou. Přihlašovací údaje do systému budou zaslány na e-mail povinně uvedený v přihlášce.

Standardy – Sekce bude obsahovat fotografie a podrobné informace o standardech jednotlivých plemen retrieverů.

Chovatelství – Zde půjde opět o skupinu informačních stránek zaměřených obsahem na téma chovatelství. Bude zahrnovat jednotlivé stránky obsahující postup uchovnění jedince, chovné podmínky, chovatelský řád, pokyny ke krytí a další články. Každý článek bude také k dispozici v podobě vhodné pro tisk.

Uchovnění jedinci – V této podsekci bude k dispozici databáze chovných jedinců. V databázi lze vyhledávat podle jména jedince, roku uchovnění, jednotlivých vlastností, majitele a dalších kritérií. Uživatel může zobrazit detail každého jedince, který mimo jiné obsahuje záznam o rodičích, z nichž matka i otec mohou být rovněž v databázi chovných jedinců. Systém tak nabídne hypertextové procházení v rodokmenu do hloubky. Ze vzájemného propojení vyplývá i možnost vyobrazit u každého jedince jeho krytí, potomky a sourozence.

Chovatelské stanice – Podsekce bude nabízet přehled základních prezentací chovatelských stanic retrieverů. Stanice lze vyhledávat podle názvu, popisu nebo majitele. Přidání či úpravu prezentace chovatelské stanice může učinit každý chovatel, který je členem klubu (viz kapitola 6.2).

Výstavy – V této sekci bude k dispozici přehled plánovaných klubových výstav. U každé výstavy bude možno stáhnout dokument s propozicemi, přihlášku a případně další dokumenty. Dále bude v této sekci možno prohlížet výsledky jednotlivých psů a fen v uplynulých výstavách. Ve výsledcích lze vyhledávat podle data výstavy, jména jedince a dalších kritérií spojených s výsledky výstav.

Zkoušky – Zde bude k dispozici přehled plánovaných klubových zkoušek. U každé zkoušky bude možno stáhnout dokument s propozicemi, přihlášku a případně další dokumenty. Dále bude v této sekci možno prohlížet výsledky uplynulých zkoušek včetně posudků jednotlivých psů a fen. Ve výsledcích lze vyhledávat podle data výstavy, jména jedince a dalších kritérií spojených s výsledky zkoušek.

Přečtěte si – Tato sekce by měla poskytnout veřejnosti zajímavé články na témata týkající se retrieverů. Články budou přehledně roztříděny do kategorií a každý z nich bude uživateli k dispozici také v podobě vhodné pro tisk.

Inzerce – Inzerce bude členěna do několika kategorií - koupím, prodám, daruji, nabízím krytí. V inzerátech lze fulltextově vyhledávat. Inzerát může bezplatně vložit každý uživatel. V inzerci se však objeví až po kontrole povolaným správcem, aby se předešlo zveřejňování nevhodných příspěvků či inzerci týkající se jedinců bez průkazu původu.

Mapa stránek – V zápatí webové prezentace bude dostupný odkaz na automaticky generovanou mapu webové prezentace.

6.2 Členské sekce

Řadoví členové klubu se mohou pomocí přihlašovacího jména a hesla přihlásit do systému. Tímto krokem pro ně budou některé sekce rozšířeny o další možnosti, případně přibudou nové sekce. Nadále pro ně samozřejmě zůstane přístupné vše, co obsahují veřejné části aplikace. Součástí celého projektu bude zajistit bezpečnost nejen uživatelských hesel, ale především systému jako takového. Současným členům klubu, kteří zapomněli své přihlašovací údaje, systém nabídne zadání členského identifikačního čísla a e-mailu. Pokud se tato kombinace bude shodovat s údaji v databázi, zapomenuté údaje budou členovi na jeho e-mail zaslány. Lidem, kteří se stali členy klubu ještě před uvedením naší aplikace do provozu, budou zaslány přihlašovací údaje pozemní poštou ve chvíli, kdy si zažádají o vytvoření uživatelského účtu. Při této žádosti bude členovi odeslán e-mail, jehož součástí bude potvrzovací odkaz. Teprve po aktivaci tohoto odkazu budou členovi zaslány přihlašovací údaje. Tímto krokem systém ověří platnost e-mailové adresy, která je nutnou podmínkou pro aktivaci účtu člena v systému.

Osobní údaje – Tato sekce umožní členům změnit své osobní údaje v databázi Retriever klubu, a to bez jakýchkoliv písemných žádostí. Nelze změnit pouze členské číslo, které bylo přiděleno při vstupu člena do klubu. Osobními údaji je jméno, příjmení, titul, datum narození, státní občanství, adresa, telefon, e-mail, vlastnictví plemen a chov plemen. Jako součást osobních údajů dále systém poskytne datum, ke kterému byl zaplacen členský příspěvek na letošní rok. Popřípadě systém zobrazí informaci k platbě, pokud ještě nebyl příspěvek zaplacen, což uživatel uvidí pouze v době platebního období (1. – 31. leden). Po této době je mu přístup do systému znemožněn.

Uchovnění jedinci – Člen klubu, který má již v systému zapsaného chovného jedince (nebo několik jedinců) na své jméno, uvidí seznam těchto jedinců právě v této sekci, kde může ke každému nahrát jednu fotografii. Doplnění dalších údajů, jako jsou výsledky nebo tituly z výstav a zkoušek, nemůže sám člen podle stanov RK-CZ učinit. Údaje může doplňovat pouze oprávněný člen orgánů klubu na základě zaslaných dokladů, podobně jako probíhá samotné uchovnění jedince.

Chovatelská stanice – Každý člen RK-CZ, pokud je chovatelem, může využít možnosti bezplatně prezentovat svoji chovatelskou stanici na webu RK-CZ (ve veřejné sekci Chovatelské stanice). Uživatel v této sekci zvolí, že chce stanici prezentovat, zadá název CHS, plemena retrieverů, která chová, adresu CHS, nepovinně pak telefon, e-mail, webové stránky stanice nebo doplňující popisný text. K této jednoduché prezentaci je také možno nahrát jeden obrázek (například logo stanice nebo fotografii).

6.3 Administrátorské sekce

Kategorizaci uživatelů systému nelze vymezit zcela jednoznačně. Můžeme říci, že největší skupinou budou řadoví členové, jejichž speciální oprávnění jsme popsali v předchozí podkapitole. V případě administrativní činnosti klubu se však jednotlivá oprávnění budou muset uživatelům přidělovat individuálně. Půjde o členy jednotlivých orgánů klubu, tedy především o členy výboru (prezident, viceprezident, ekonom, matrikář, sekretář, hlavní poradce pro chov, hlavní poradce pro výcvik) a dále o některé členy kontrolní a disciplinární komise. Systém tedy musí umožnit přidělení libovolných oprávnění kterémukoliv členovi klubu s tím, že takovéto přidělení lze provést pouze při dodržení jistých bezpečnostních podmínek.

Nastavení oprávnění uživatelů – Bezpečnostní podmínky pro přidělení oprávnění jsou stanoveny následovně: V systému bude existovat administrátorský účet, který nepatří žádnému členovi a nejde ze systému odstranit. Heslo k tomuto účtu by měl znát pouze prezident, klubu a jedna další důvěryhodná osoba. Po úspěšné verifikaci systém umožní vyhledat libovolného člena klubu a přidělit mu jakákoliv oprávnění k činnostem, které budou nebo byly popsány v této či předchozích podkapitolách. Administrátorský účet má dále oprávnění zálohovat a obnovovat data, upravovat systémová nastavení a prohlížet protokoly událostí. Veškeré změny v nastavení oprávnění uživatelů jsou do protokolů zaznamenávány, přičemž tyto protokoly vidí i většina ostatních řadových správců. Takto bezpečně předejdeme nekontrolovanému operování s oprávněními.

V následujícím popisu jednotlivých administrátorských sekcí se předpokládá, že uživatel, o němž hovoříme, má do konkrétní sekce oprávnění vstupovat, případně manipulovat s daty. Takového uživatele budeme dále obecně nazývat správcem.

Aktuality – Na tomto místě správce vkládá typicky krátké články informující členskou základnu o aktuálních záležitostech. Může se jednat například o upozornění na blížící se termín klubové akce, pozvánku na členskou schůzi, informaci o novinkách v některé sekci webu a podobně. Každý článek sestává z formátovaného textu a krátkého titulku. Do textu lze vkládat také tabulky, obrázky nebo odkazy na uložené soubory. U jednotlivých článků může správce zpětně upravit jeho obsah, vymazat ho, změnit jeho pořadí vůči ostatním nebo ho přeřadit do archívu aktualit.

Informace o klubu – Tato sekce bude obsluhovat správu webových stránek (článků) popsaných v kapitole 6.1. Správce může jednotlivé stránky přidávat a upravovat, prostřednictvím webového rozhraní. Systém umožní vkládat základně formátovaný text včetně obrázků, tabulek nebo odkazů na uložené soubory. Oprávnění správců k této sekci platí souhrnně pro všechny stránky (podsekce) zde vytvářené.

Evidence financí – Jedná se o další jednoduchou stránku začleněnou do předchozí sekce. Správa jejího obsahu bude však oddělena. Důvodem je to, že administraci předchozí sekce může mít

na starosti typicky správce webového obsahu, který nemusí být ani členem výboru, kdežto stránku s evidencí finančních prostředků bude pravděpodobně aktualizovat pouze ekonom, případně prezident klubu.

Plemena a jejich standardy – V systému budou vedena jednotlivá plemena retrieverů, z nichž každé bude mít v databázi svůj český název, anglický název, oficiální zkratku pro ČR, skupinu FCI, číslo standardu FCI a popis standardu včetně možnosti přiložení souborů a vložení obrázků. Kromě toho bude v systému každé plemeno jednoznačně identifikovatelné pro možnost napojení v jiných sekcích. Plemen je celkem 6 a jsou v systému zaneseny napevno. Správce může v této sekci u každého modifikovat pouze standard, což je opět článek s možností formátovaného textu vč. obrázků a odkazů na soubory.

Chovatelství – I zde půjde o sekci spravující články. Postup administrace této sekce bude naprosto stejný, jako v případě sekce "Informace o klubu". Ve stejnojmenné veřejné sekci "Chovatelství" budou tyto články zobrazeny jako podsekce spolu s chovatelskými stanicemi a uchovněnými jedinci.

Přečtěte si – Opět se jedná o sekci správy článků. Kromě tvorby článků zde však systém umožní také vytváření kategorií, do kterých lze články třídit. Kategorie sama o sobě nese pouze název (například "Rady začínajícím chovatelům") a návaznost na články, které do ní jsou zařazeny. Veřejná sekce "Přečtěte si" tak může ve výsledku obsahovat jak jednotlivé články, tak i kategorie s dalšími články. Každý článek může samozřejmě správce pozměnit, používat formátovaný text, obrázky, tabulky a odkazy na uložené soubory. Článek lze kdykoliv z nějaké kategorie vyřadit nebo přesunout jinam.

Evidence členů klubu – Tato sekce bude centrem správy uživatelů. Jejím základem bude vyhledávání v členské základně, a to podle všech údajů, které jsou u členů zaznamenávány. U každého nalezeného uživatele může správce (zde typicky matrikář) změnit jeho osobní údaje včetně přístupového hesla, a to pro případ, že tak nemůže učinit sám člen prostřednictvím svého účtu v systému a požádá o změnu písemně nebo jinak. Dále lze v případě nutnosti ručně pozastavit členství kteréhokoliv člena. Pokud uživatel zapomene své heslo do systému a nemůže použít automatickou obnovu hesla z důvodu neplatnosti své e-mailové adresy, může požádat telefonicky nebo e-mailem o zaslání nového hesla pozemní poštou. Správce tedy v této sekci vyhledá uživatele a může celý proces zařídit jedním kliknutím, kdy systém přidělí uživateli nové heslo a nabídne správci vygenerovaný dopis s přihlašovacími údaji. Ten lze okamžitě vytisknout a odeslat. Součástí dopisu formátu A4 bude poštovní adresa uživatele umístěná v souladu s pohodlným použitím obálky typu DL s okénkem. Kromě dopisu lze samozřejmě odeslat heslo také e-mailem, pokud je v databázi uveden.

Žádosti o členství – Dále na tomto místě správce uvidí seznam elektronicky podaných žádostí o členství. Kteroukoliv žádost lze potvrdit nebo zamítnout. V případě zamítnutí přihlášky je dotyčný zájemce kontaktován a jsou mu vráceny peníze. V případě papírové přihlášky člena

musí sytém umožnit zapsat tohoto člena a všechny jeho údaje ručně. S papírovou přihláškou zasílá žadatel ústřižek složenky jako potvrzení o zaplacení, tudíž není třeba identifikovat platbu, jako v případě elektronické přihlášky. Po zadání nového člena aplikace automaticky vygeneruje jeho přihlašovací heslo do systému a nabídne správci tisk průvodního dopisu s přihlašovacími údaji, stejně jako při generování nového hesla zasílaného poštou.

Kontrola členských příspěvků – Členům, kteří mají ve svých osobních údajích uvedený e-mail, bude systém automaticky zasílat touto cestou upozornění na zaplacení členského příspěvku na nový rok, a to v době čtyř a dvou týdnů před lhůtou na zaplacení. Toto upozornění bude zahrnovat také údaje k platbě, tedy číslo účtu, částku a příslušné symboly. Identifikaci platby provede správce v této části sekce. Podle bankovního výpisu z účtu zkontroluje specifický symbol platby pro příspěvky a zadá variabilní symbol platby. Tím člena identifikuje a pokud je přijatá peněžní částka správná, označí jeho členství jako zaplacené a tím pádem potvrzené na tento rok. Systém zaznamená datum platby členského příspěvku příspěvku. V případě, že člen nedostane upozornění k zaplacení e-mailem, protože ho nemá nebo nevybírá, nalezne informace k platbě ve Zpravodaji a na internetu v sekci aktualit.

Pozastavení členství – Ve chvíli, kdy uplyne lhůta na zaplacení nového členského příspěvku, systém automaticky vyhledá členy, kteří dosud nezaplatili, a uvede je do stavu pozastaveného členství (viz kapitola 5.2 na straně 40). U pozastavených členství může správce stále identifikovat přijaté platby a navracet tak osobám status člena. V tu chvíli je také dotyčnému členovi automaticky zasláno upozornění o aktivaci členství na e-mail, pokud ho ve svých osobních údajích uvádějí.

Uchovnění jedinci – Uchovnění jedince zprostředkuje většinou poradce chovu na základě majitelem zaslaných potřebných dokladů. Poradce jakožto typický správce této sekce může na tomto místě zapsat nového uchovněného jedince do systému. Kromě všech údajů o psovi nebo feně je nutné v aplikaci napojit tohoto jedince na konkrétního majitele, který musí být členem klubu, tedy musí být také v systémové databázi. Uchovněné jedince lze v této sekci vyhledávat stejně jako ve veřejné sekci "Chovatelství", správce však může do databáze zasahovat (opravovat a měnit údaje). Systém hlídá podmínku stáří feny, která podle stanov nesmí přesáhnout určitou hranici (aktuálně 8 let). Po překročení této věkové hranice je fena automaticky vyřazena ze seznamu chovných jedinců, zůstává však v databázi kvůli vazbám na potomky nebo rodiče. U každého chovného jedince musí být záznam o matce a otci. Matku i otce může správce vyhledat v seznamu rodičů, do kterého jsou zahrnuti také všichni chovní jedinci. Tím v databázi vzniká propojení využitelné pro vyhledávání potomků jedince v jakékoliv sekci systému, kde to bude třeba. Pokud rodič v seznamu není, může správce zadat a uložit nového. Jelikož ne všichni rodiče jsou uchovnění v RK-CZ, umožní systém zadat chovného jedince i s neúplnými údaji. Takový bude ale jasně označen jako uchovněný mimo RK-CZ.

Chovatelské stanice – Správce může vyhledávat CHS stejně jako uživatel ve stejnojmenné veřejné sekci. Navíc má možnost upravit kteroukoliv z nich v případě, že je o to členem klubu požádán. Kromě toho je v této sekci možno vypsat naposledy změněné stanice včetně data a času, kdy ke změně došlo. Správce tak může mít veškeré uživatelské změny pod kontrolou.

Výstavy – Tato sekce bude centrem správy klubových výstav. Správce může přidat novou výstavu do systému zadáním jejího termínu, názvu, místa a popisného textu. Ke každé výstavě lze nahrát libovolné množství souborů (dokumentů), které mají být veřejnosti a členům k dispozici. Jedná se například o podrobné propozice nebo přihlášku v podobě PDF dokumentu. Celou takto nadefinovanou výstavu může kdykoliv správce uveřejnit. Teprve po té bude zpřístupněna pro veřejnost.

Přihlášení psi a jejich výsledky – Systém umožní správci zadávat výsledky k jednotlivým výstavám. S každou došlou přihláškou může správce předem zadat do systému základní údaje přihlášeného psa, nevyplní pouze výsledek. Pokud systém nalezne přihlášeného psa v databázi chovných jedinců, nemusí správce zadávat některé podrobnosti jako rodiče psa, zápisové číslo a pod. Tato předpřipravená data bude možno exportovat do dokumentu ve formátu XLS (MS Excel) pro další využití (viz dále). Systém dále poskytne správci přehled počtů přihlášených jedinců v jednotlivých plemenech a výstavních třídách. Přímo do XLS souboru s daty, který bude typicky exportován po uzávěrce přihlášek, budou v průběhu konání výstavy na místě zapisovány výsledky. Po ukončení výstavy správce může nahrát tento soubor zpět do systému ke konkrétní výstavě. Tímto jediným krokem jsou výsledky výstavy uloženy do systému a automaticky jsou přístupné ve veřejné sekci na internetu. Správce může dodatečně libovolný výsledek v systému opravit.

Zkoušky – Zadávání a editace zkoušek či práce s daty a výsledky bude probíhat na stejném principu jako v případě výstav. Správa zkoušek bude pouze v oddělené sekci.

Řízená inzerce – Správci této sekce by měli mít pravidelný přístup k internetu, protože pouze jejich přičiněním se inzerát zadaný uživatelem může v inzerci objevit. Systém zde vypíše inzeráty čekající na zveřejnění, správce tak každý z nich může jednoduše uveřejnit nebo zamítnout. Systém umožní správci také řízení inzerce prostřednictvím e-mailových zpráv, pokud si tento způsob v sekci zvolí. S každým zadaným inzerátem pak bude na e-mail správce odeslána zpráva obsahující text inzerátu s dalšími jeho náležitosti. Součástí této zprávy bude také aktivační hypertextový odkaz. Správce tak může jedním kliknutím inzerát v systému aktivovat, aniž by se do systému musel přihlásit. Stejně tak bude zpráva obsahovat odkaz, kterým lze inzerát zamítnout. Při tomto způsobu manipulace s inzercí musí systém zajistit dostatečnou bezpečnost, aby inzerát mohla aktivovat nebo zamítnout pouze oprávněná osoba.

Protokoly událostí – Systém bude automaticky zaznamenávat některé události, jako manipulace s články, s žádostmi o členství, s oprávněními uživatelů a podobně. Události budou přehledně

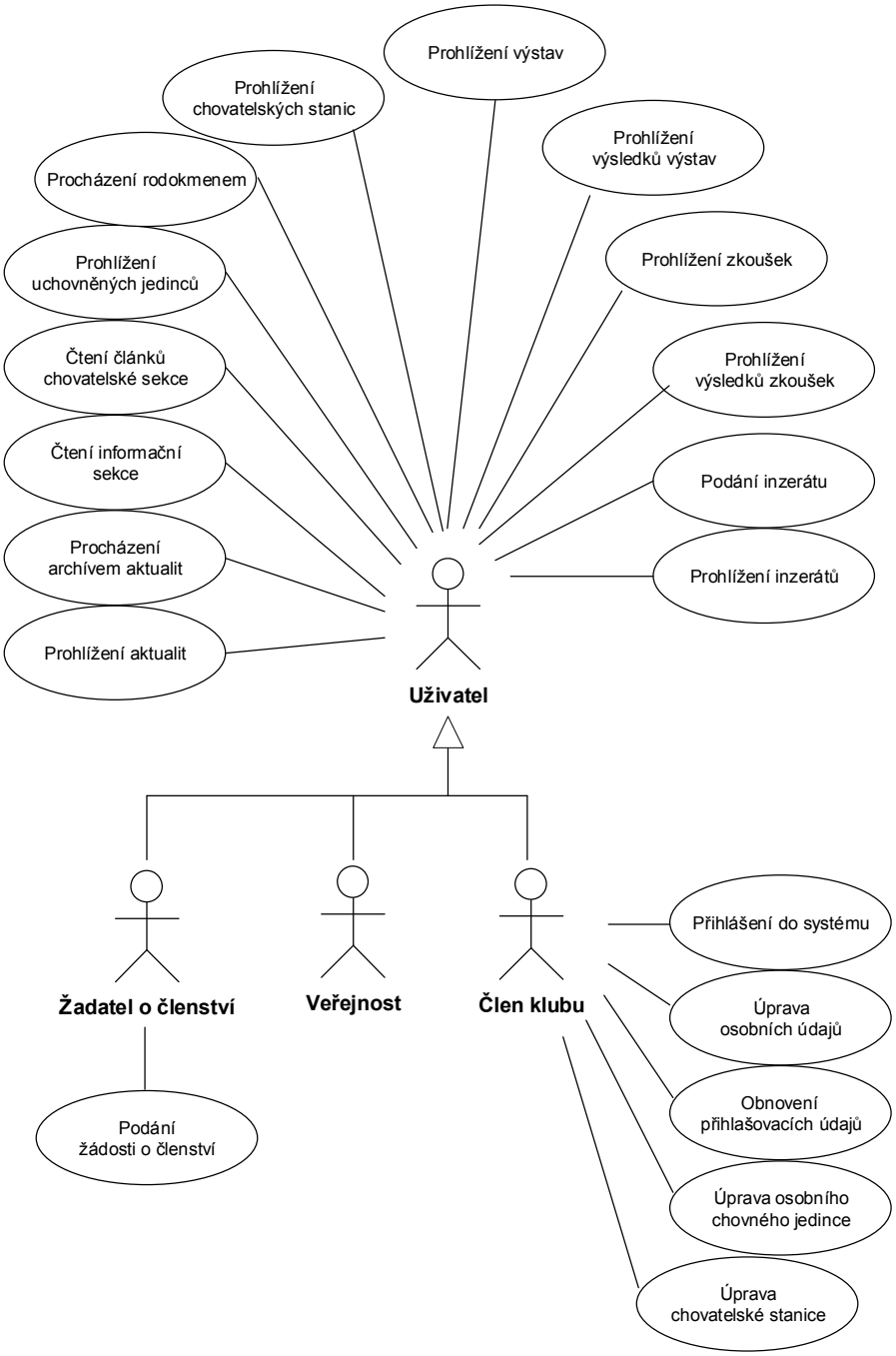
uspořádány do tématických protokolů, u každé bude kromě informací zaznamenán datum, čas a uživatel, který událost vyvolal. V událostech lze podle těchto vlastností vyhledávat.

Systémová nastavení – Tato sekce umožní správci upravit systémová nastavení, jako je například výše členského poplatku, období na zaplacení, číslo účtu klubu, specifické symboly a další.

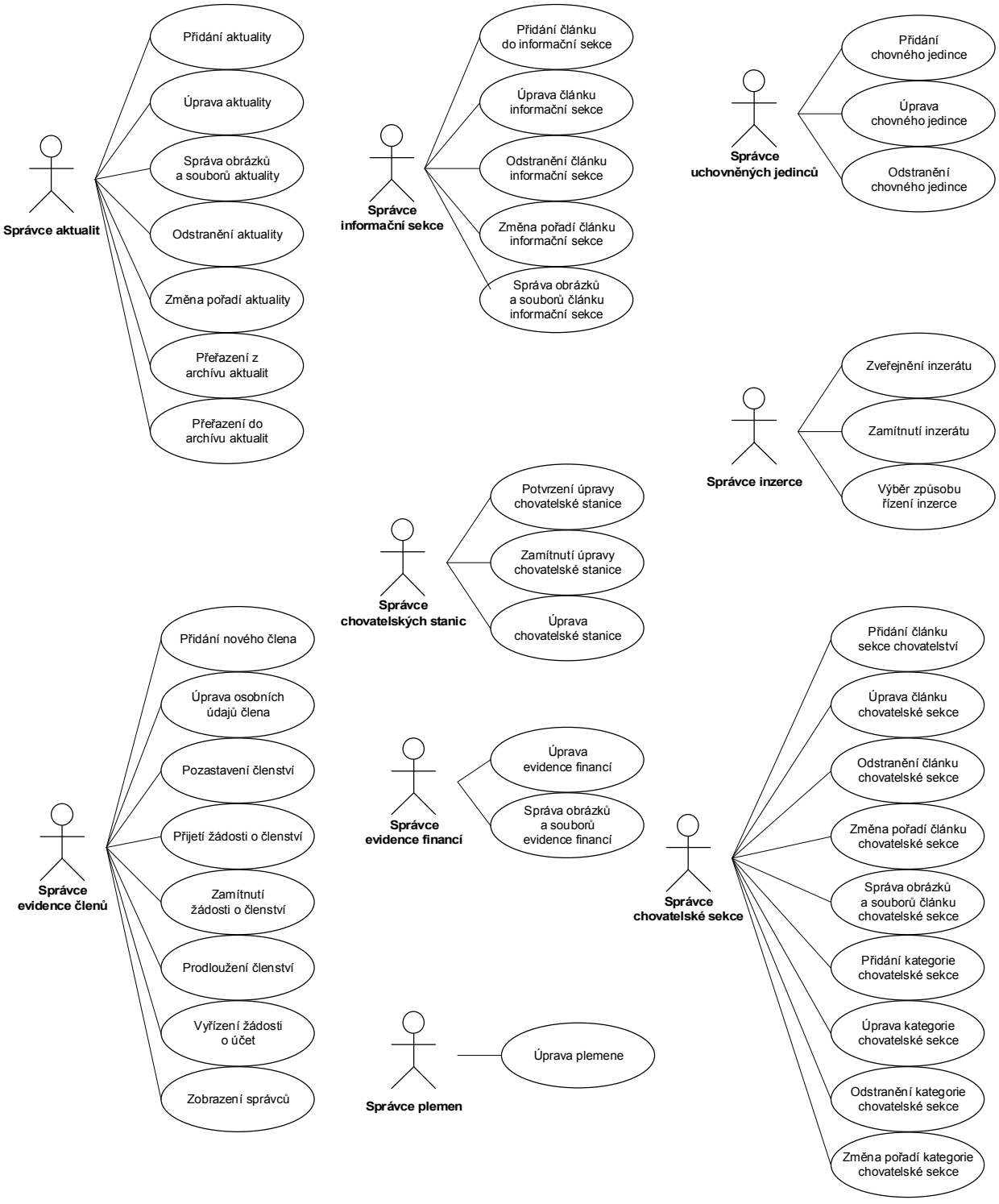
Zálohování a obnovení dat – V této sekci lze vytvořit "bod obnovení", což je současný stav systému uložený do databáze. Automatický bod obnovení bude systém provádět sám s nastavenou frekvencí, např. 1 den. Ke kterémukoliv bodu obnovení se lze kdykoliv vrátit, přičemž těsně před touto akcí je opět automaticky vytvořen bod pro případ, že by chtěl správce vzít tento krok zpět.

7 Formální návrh

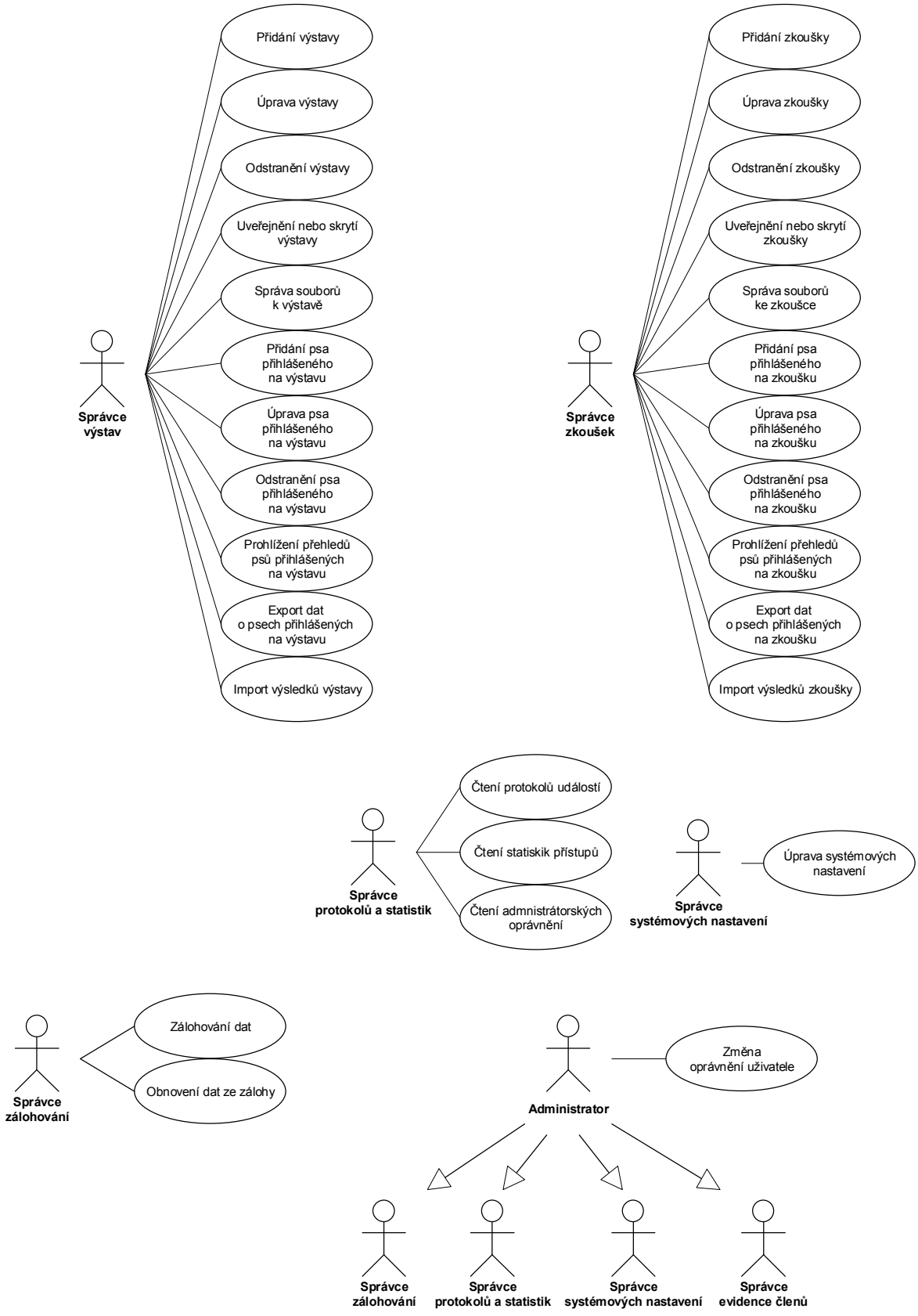
7.1 Diagram případů použití



Obrázek 7.1a – diagram případů použití



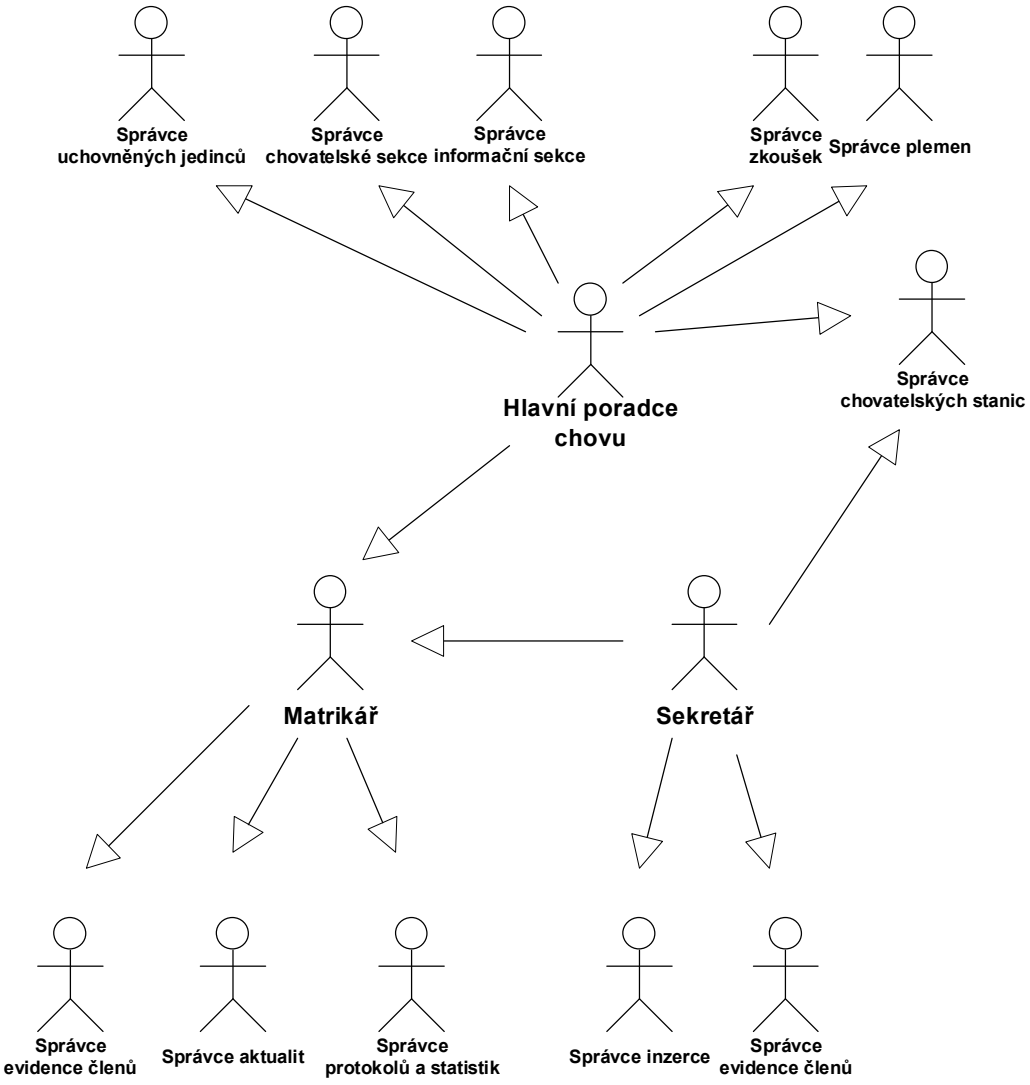
Obrázek 7.1b - diagram případů použití



Obrázek 7.1c - diagram případů použití

7.2 Model oprávnění

Vzhledem k požadavkům na flexibilní řízení přístupu do různých administrátorských sekcí a individuální přidělování oprávnění uživatelům je nutno namodelovat oprávnění jednotlivým fyzickým osobám s pomocí aktérů uvedených v předchozí podkapitole. Konkrétní diagram, který ilustruje například oprávnění hlavního poradce chovu, sekretáře a matrikáře by mohl vypadat následovně.



Obrázek 7.2a - model oprávnění

7.3 Ukázka scénáře případu použití

Případ použití: Přidání výstavy

Uživatel: Správce výstav

Hlavní scénář:

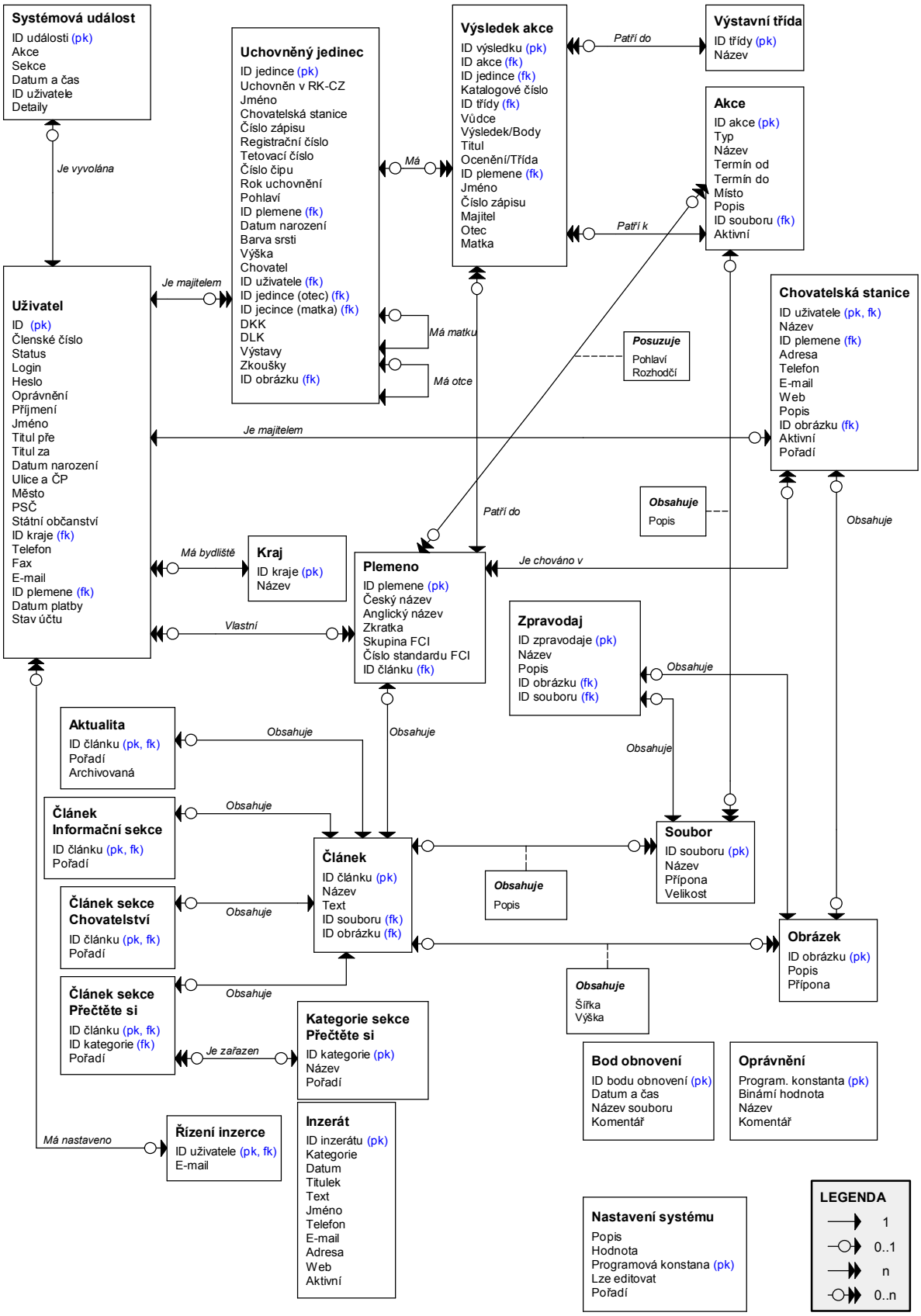
Krok	Role	Akce
1	Uživatel	Vstoupí do sekce správy výstav
2	Systém	Zobrazí stránku se seznamem výstav v databázi
3	Uživatel	Zvolí přidat výstavu
4	Systém	Zobrazí formulář pro zadání podrobností výstavy
5	Uživatel	Do polí formuláře zadá termín výstavy, název, místo a popisný text, vyplní seznam rozhodčích, zvolí, zda chce výstavu zveřejnit, zvolí *přidat*
6*	Systém	Zkontroluje správný formát termínu (data), uloží výstavu do databáze a zobrazí stránku se seznamem výstav v databázi
7	Uživatel	Pokud chce přidat k výstavě souboru, zvolí položku *soubory* u přidané výstavy
7.1	Systém	Zobrazí seznam souborů výstavy spolu s formulářem pro přidání nového souboru
7.2	Uživatel	Vybere ve svém počítači soubor, který chce k výstavě nahrát, doplní jeho název tak, jak má být v systému zobrazen a zvolí *přidat*
7.3	Systém	Načte soubor do databáze, přiřadí ho k výstavě a aktualizuje stránku se seznamem souborů výstavy a formulářem pro přidání souboru
7.4	Uživatel	Vrátí se na seznam výstav volbou *zpět*
7.5	Systém	Zobrazí seznam výstav v databázi

Alternativní kroky scénáře:

6a – Systém rozpozná chybě zadaný formát termínu

Krok	Role	Akce
6a1	Systém	Zobrazí hlášení o nesprávném formátu termínu a zobrazí znovu formulář, ve kterém zůstanou uživatelem vyplněná data
6a2	Uživatel	Opraví chybně zadané datum a zvolí *přidat*
6a3	Systém	Navrátí se do hlavního scénáře ke kroku 6

7.4 Logický datový model – ER diagram



Obrázek 7.4a - ER diagram

8 Závěr

V průběhu zimního semestru bylo realizováno několik konzultací se zadavatelem projektu, kde se za pomocí některých prostředků softwarového inženýrství podařilo velmi přesně specifikovat a analyzovat požadavky na uvedený systém. Následně byl proveden detailní a formální návrh aplikace. Přínosem této části projektu pro mě bylo teoretické prohloubení znalostí v oblasti SI stejně jako praktické vyzkoušení komunikace se zákazníkem při skutečném sběru a analýze požadavků. Zároveň jsem postupně získával informace, na základě kterých byla postavena stěžejní kapitola o bezpečnosti webového IS. Získané znalosti jsem aplikoval při samotné implementaci navrženého programu v průběhu zimního semestru. Implementované bezpečnostní mechanismy korespondují s navrženými způsoby řešení jednotlivých problémů. Detailní podobu těchto řešení lze prostudovat ve zdrojových kódech, na než jsem se v textu odkazoval a které jsou součástí přiloženého CD.

V závěru 3. kapitoly jsem zmínil, že výsledek práce byl testován profesionálním systémem SPI Dynamics WebInspect pro detekci slabých míst webových aplikací. Náš IS se podařilo zabezpečit tak, že jeho ochrana nebyla tímto testem v žádném místě prolomena. Systém upozornil na dva nezávažné potenciální problémy týkající se názvů adresářů `/inc` a `/data`, ve kterých obvykle bývají uloženy citlivé soubory. V našem případě opravdu tyto adresáře obsahují soubory, které nesmí být přístupné z webu. Oba tyto adresáře jsou však zabezpečené proti přístupu již na úrovni webového serveru, nebezpečí získání dat nebo spuštění skriptů tedy nehrozí. Další nezávažná poznámka a s ní související upozornění střední závažnosti popisuje problematiku možného použití cookies při komunikaci nezabezpečeným protokolem HTTP. Tato záležitost byla diskutována v závěru kapitoly 3.4.2 na straně 23 a použitý způsob zabezpečení je pro aplikaci našeho typu dostačující. Souhrn výsledků testu exportovaný z programu WebInspect je součástí příloh.

Při pohledu zpět je zajímavé si uvědomit, že spousta autorů webových systémů si není vědoma zranitelných míst, která v těchto aplikacích běžně existují a mohou být zneužita k útoku. U soukromých webů s relativně nedůležitým obsahem to nemusí vadit. Rozsáhlejší informační systém podniku však již může upoutat pozornost útočníků a mohou tak vzniknout nemalé škody. Ve chvíli, kdy se začneme o bezpečnost v této oblasti zajímat podrobněji, se může stát, že budeme hledat možný útok "na každém řádku" programového kódu a ošetřovat případy, které by po hloubkové analýze vlastně ani nemohly nastat. Takový přístup je však ze začátku zcela očekávaný a jeho vytříbení je až otázkou času a získání zkušeností.

Na závěr musím zmínit, že velkou roli hrála v tomto projektu také motivace. Nejen návrh, ale i následná implementace IS byla v mém případě realizována s vědomím, že systém bude mít skutečné využití a přinese mnoha lidem spoustu výhod. Prozatím je náš IS dostupný na webové adrese **http://dp.nehasil.com**. Rozšíření návrhu, implementace doplňkových součástí, konečné testování aplikace a její nasazení do provozu je dále naplánováno po uzavření diplomové práce.

9 Literatura

[1] Mikle, P.: *XCSS, úplná přesná referenční příručka*. Brno, Zoner Press, 2004.

[2] Williams, H. E., Lane, D.: *Programujeme webové aplikace pomocí PHP a MySQL*. Praha, Computer Press, 2002.

[3] Huseby, S. H.: *Zranitelný kód*. Brno, Computer Press, 2006.

[4] Janovský, D.: *Jak psát web* [online]. 1999-2007. Dokument je dostupný na URL http://www.jakpsatweb.cz.

[5] Wikipedie, otevřená encyklopedie. *Hyper Text Transfer Protocol* [online]. 2007. Dokument je dostupný na URL http://cs.wikipedia.org/wiki/Http.

[6] Netscape Communications Corporation: *Client-Side JavaScript Reference* [online]. 1999. Dokument je dostupný na URL http://docs.sun.com/source/816-6408-10/contents.htm.

[7] PHP Documentation Group: *Manuál PHP* [online]. 1997-2007. Dokument je dostupný na URL http://www.php.net/manual/cs/.

[8] PEAR Documentation Group: *PEAR Manual* [online]. 2001-2007. Dokument je dostupný na URL http://pear.php.net/manual/en/.

[9] MySQL AB: *MySQL 5.0 Reference Manual* [online]. 1997-2007. Dokument je dostupný na URL http://dev.mysql.com/doc/refman/5.0/en/index.html.

[10] Růžička, P.: Odhlášení z HTTP autentizace. *Interval.cz* [online]. 15.8.2002. Dokument je dostupný na URL http://interval.cz/clanky/odhlaseni-z-http-autentizace.

[11] Vrána, J.: Bezpečné přihlašování uživatelů. *Root.cz* [online]. 12.4.2006. Dokument je dostupný na URL http://www.root.cz/clanky/bezpecne-prihlasovani-uzivatelu.

[12] Wikipedia, the free encyclopedia. *Man-in-the-middle attack* [online]. 2007. Dokument je dostupný na URL http://en.wikipedia.org/wiki/MITM.

[13] Ralf S. Engelschall: *mod_SSL, The Apache Interface To OpenSSL – User Manual* [online]. 1998-2001. Dokument je dostupný na URL http://www.modssl.org.

[14] Kolektiv autorů: *Apache-SSL Documentation* [online]. 1995-2005. Dokument je dostupný na URL http://www.apache-ssl.org/docs.html.

[15] Seidler, K. O.: *Documentation about XAMPP* [online]. 2002-2007. Dokument je dostupný na URL http://www.apachefriends.org/en/faq-xampp.html

[16] Wikipedie, otevřená encyklopedie. *ARP spoofing* [online]. 2007. Dokument je dostupný na URL http://cs.wikipedia.org/wiki/ARP_spoofing.

[17] Haller, M.: Odposloucháváme data na přepínaném Ethernetu. *Lupa.cz* [online]. 11.7.2006. Dokument je dostupný na URL http://www.lupa.cz/clanky/odposlouchavame-data-na-prepinanem-ethernetu-5/.

[18] Marsching, S.: *suPHP Documentation* [online]. 2006. Dokument je dostupný na URL http://www.suphp.org.

[19] Wikipedie, otevřená encyklopedie. *CAPTCHA* [online]. 2007. Dokument je dostupný na URL http://cs.wikipedia.org/wiki/CAPTCHA.

[20] Haller, M.: Seriál Útoky typu DoS. *Lupa.cz* [online]. 2006. Dokument je dostupný na URL http://www.lupa.cz/serialy/utoky-typu-dos/.

[21] Kanisová, H., Müller, M.: *UML srozumitelně*. Brno, Computer Press, 2006.

Seznam příloh

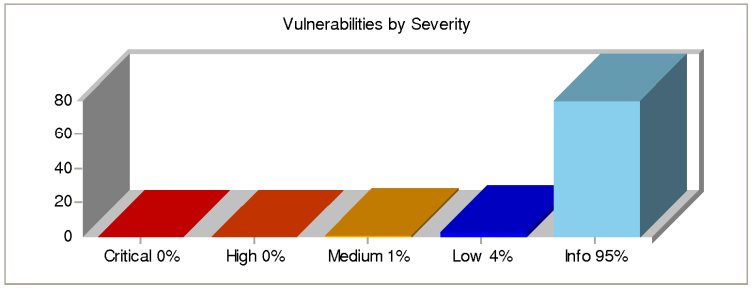
Příloha 1 – Výsledky testování aplikace

SPI DYNAMICS

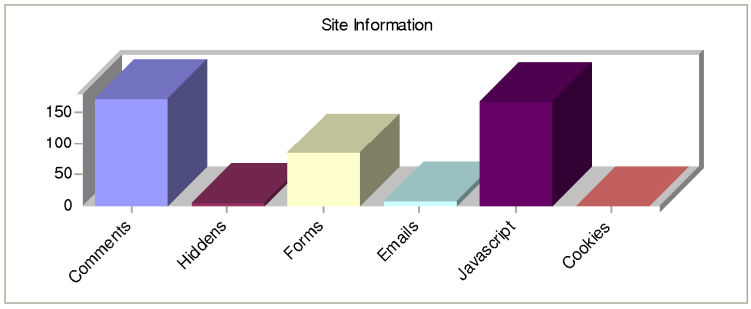
WebInspect Executive Summary **Report Date:** 16.5.2007

Scan: https.diplomka.443 05-16-07 21.19.42
Policy: OWASP Top 10.apc
Servers: diplomka:443

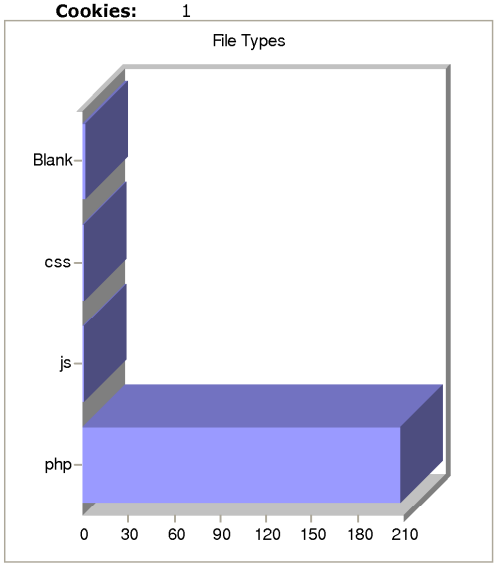
Scan Date: 16.5.2007 21:21:31
Scan Version: 6.2.151

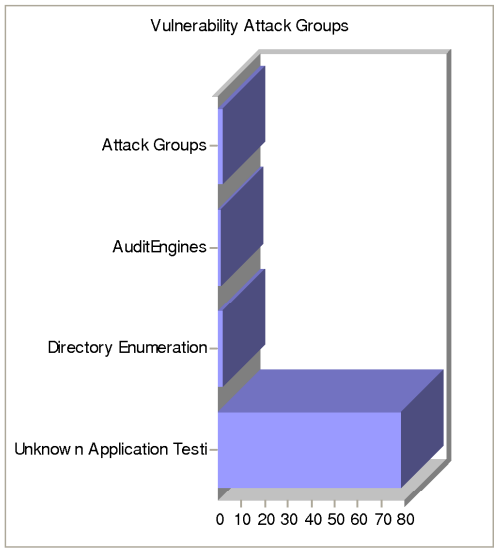


Critical: 0
High: 0
Medium: 1
Low: 3
Info: 80



Comments: 173
Hiddens: 4
Forms: 86
Emails: 8
Javascript: 167
Cookies: 1





Page 1 of 1

SPI DYNAMICS

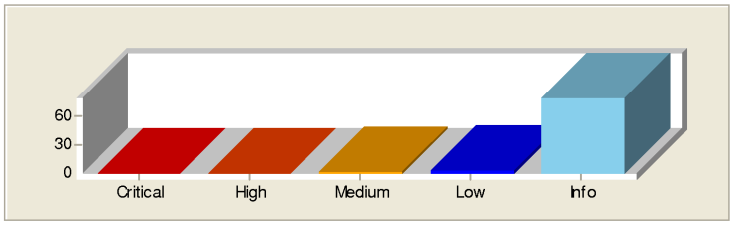
WebInspect Vulnerability Report

Report Date: 16.5.2007

Scan: https.diplomka.443 05-16-07 21.19.42
Policy: OWASP Top 10.apc
Scan Date: 16.5.2007 21:21:31
Scan Version: 6.2.151

Server: **diplomka**

Count of Vulnerable Pages



Medium — SSL Cookie Not Used

File Names: https://diplomka:443/auth.php (PostData)login=filip&password=filip&logon=OK

Summary: This policy states that any area of the website or web application that contains sensitive information or access to privileged functionality such as remote site administration requires that all cookies are sent via SSL during a SSL session. Webinspect has detected that the URL: https://diplomka:443/auth.php has failed this policy. If a cookie is marked secure, it will only be transmitted if the communications channel with the host is a secure one. Currently this means that secure cookies will only be sent to HTTPS (HTTP over SSL) servers. If secure is not specified, a cookie is considered safe to be sent in the clear over unsecured channels.

For more information refer to the following white paper:

http://wp.netscape.com/newsref/std/cookie_spec.html

Fix: Walkthrough to remediate this issue for IIS:
http://support.microsoft.com/default.aspx?scid=kb;en-us;274149

Low — SSL Policy Enforcement Issue

File Names: http://diplomka:80/

Summary: An SSL-enabled webserver was found to be accessible through a non-SSL connection. If a server has sensitive information on it that should only be exchanged via SSL, it should not be possible to access this information using a non-SSL connection.

Implication: An attacker could potentially intercept sensitive data including usernames and passwords, customer account information, or similar information.

Fix: Disable the unencrypted web service.

Low — Directory (data)

File Names: https://diplomka:443/data/

Summary: Directory Enumeration vulnerabilities were discovered within your web application. During an assessment, WebInspect's directory enumeration audit engines and associated vulnerability checks, along with a baseline crawl of your web application, are utilized to discover and then display all directory paths and possibilities that exist on the application server. Risks associated with an attacker discovering a directory on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information discovered in that directory to perform other types of attacks. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Description:

Found Directory: /data/

This directory may contain log files or data files. These files contain information sensitive in nature.

Low

Directory (inc)

File Names: https://diplomka:443/inc/

Summary: Directory Enumeration vulnerabilities were discovered within your web application. During an assessment, WebInspect's directory enumeration audit engines and associated vulnerability checks, along with a baseline crawl of your web application, are utilized to discover and then display all directory paths and possibilities that exist on the application server. Risks associated with an attacker discovering a directory on your application server depend upon what type of

directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information discovered in that directory to perform other types of attacks. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Description:

Found Directory: /inc/

This folder usually contains .inc(include) files used for the website. Developers tend to leave sensitive information in these files.

Příloha 2 – Postup instalace

Aplikace je do konce června 2007 dostupná na adrese **http://dp.nehasil.com**. Vyžaduje webový server Apache 2 s podporou SSL, PHP 5.1 a MySQL 4.1. Dále jsou nutná rozšíření Spreadsheet/Excel a Mail z repozitáře PEAR. Nejnovější verze těchto technologií lze nainstalovat samostatně nebo prostřednictvím distribuce XAMPP (www.apachefirends.org), která je k dispozici pro Windows i Linux a je součástí přiloženého CD. Tento instalační manuál popisuje instalaci XAMPP pro OS Windows.

1. Nainstalujte XAMPP pro Windows pomocí instalačního souboru, který nalzenete na přiloženém CD (XAMPP\ xampp-win32-1.6.1-installer.exe). Při instalaci zvolte adresář **C:\xampp**. Zaškrtněte možnosti **Install Apache as service** a **Install MySQL as service**. V případě problémů otevřete návod ve stejném adresáři.
2. Pomocí XAMPP Control Panel zkontrolujte, zda je spuštěna služba Apache a MySQL.
3. Do webového prohlížeče zadejte adresu **http://localhost**, vyberte jazykovou verzi prezentace XAMPP a zkontrolujte dostupnost protokolu HTTPS kliknutím na odkaz **https://localhost**.
4. V menu prezentace zvolte položku **Security** a následujte odkaz **http://localhost/ security/xamppsecurity.php**.
5. Zadejte vlastní heslo pro uživatele root databáze MySQL a potvrďte změnu tlačítkem **Password changing**.
6. Naše aplikace musí být umístěna ve výchozím adresáři domény. Pro účely testování nebudeme vytvářet virtuální hosting. Z domovského adresáře serveru Apache (C:\xampp**htdocss**) přesuňte vešketý obsah do libovolného jiného umístění tak, aby tento adresář zůstal prázdný. Do C:\xampp**htdocss** nyní zkopírujte obsah adreáře **Instalace** z přiloženého CD.
7. Odeberte adresáři C:\xampp\htdocss všechna zděděná oprávnění a nastavte pouze oprávnění úplného řízení pro sebe (administrátora) a dále oprávnění ke čtení, spouštění a zobrazování složek pro uživatele SYSTEM. Tato oprávnění nechte aplikovat i na všechny podřízené objekty.
8. Přidejte oprávnění k zápisu uživateli SYSTEM pro tyto složky:
   C:\xampp\htdocs\data\account-letters, C:\xampp\htdocs\data\recovery,
   C:\xampp\htdocs\stored-files, C:\xampp\htdocs\stored-images
9. V souboru C:\xampp\apache\conf\extra**httpd-xampp.conf** zakomentujte tyto direktivy:
   Alias /security, Alias /webalizer, Alias /contrib.
   V souboru C:\xampp\apache\conf\extra**httpd-autoindex.conf** odstraňte parametr Indexes z direktivy Options v <Directory "C:/xampp/apache/icons">

10. Do adresáře C:\xampp\apache\conf**ssl.crt** zkopírujte certifikát **localhost.crt** z adresáře Certifikát na přiloženém CD. Do adresáře C:\xampp\apache\conf**ssl.key** zkopírujte soukromý klíč **localhost.key** z adresáře Certifikát na přiloženém CD.
11. V souboru C:\xampp\apache\conf\extrsa**httpd-ssl.conf** upravte v direktivách **SSLCertificateFile** a **SSLCertificateKeyFile** cestu ke zkopírovanému certifikátu a klíči.
12. V souboru C:\xampp\mysql\bin**my.cnf** zakomentujte řádek **skip-innodb** a odkomentujte tyto řádky:
    innodb_data_home_dir = C:/xampp/mysql/data/
    innodb_data_file_path = ibdata1:10M:autoextend
    innodb_log_group_home_dir = C:/xampp/mysql/data/
    innodb_log_arch_dir = C:/xampp/mysql/data/z
13. V souboru C:\xampp\htdocs**db-import.php** nastavte mysql heslo, které jste zvolili v kroku 5.
14. V souboru C:\xampp\htdocs\inc**server-def.php** upravte nastavení serveru SMTP pro odesílání pošty.
15. Pomocí XAMPP Control Panel **restartujte server Apache a server MySQL**.
16. Importujte databázi zadáním této adresy do prohlížeče: **http://localhost/db-import.php**. Operace může chvíli trvat, po té se zobrazí hlášení o výsledku. Pokud proběhl import struktury i dat, odstraňte nyní tyto soubory:
    C:\xampp\htdocs\db-schema.sql, C:\xampp\htdocs\db-data.sql, C:\xampp\htdocs\db-import.php
17. V prohlížeči zadejte adresu **http://localhost**, na které by v tuto chvíli měla aplikace fungovat.
18. V pravém menu zvolte položku **Přihlášení** a zadejte login *administrator*, heslo *admin009*. **Poznámka**: Použitý certifikát byl pro účely testování vygenerován pomocí openssl a není tedy vydán důvěryhodnou CA, na což budete upozorněni při vstupu do zabezpečené sekce aplikace.
19. Přihlašte se a pomocí položky **změnit heslo** změňte heslo účtu administrator (viz závěr kapitoly 3.1.1 a úvod kapitoly 6.3).

Instalace aplikace je tímto u konce. Do počítače byl nainstalován webový server Apache 2 s OpenSSL, PHP 5.2.1 a MySQL 5.0.37. Aplikace je nahrána v kořenovém úložišti webových dokumentů, je tedy přístupná na adrese http://localhost. Administrace databáze je možná prostřednictvím phpMyAdmin na adrese http://localhost/phpmyadmin. V databázi MySQL byl vytvořen uživatelský účet *diplomka* s heslem *kxj7g56hd*. Tento uživatel má přístup pouze k vytvořené databázi *diplomka* a jeho přihlašovacími údaji přistupují k databázi skripty naší aplikace. Nastavení lze upravit v souboru **/inc/server-def.php**.

Příloha 3 – Manuál k administraci

Po instalaci systému obsahuje databáze vzorek dat, tady i několik přihlášených fiktivních členů klubu. Zde jsou jejich přihlašovací údaje:

Login	Heslo	Role	Speciální oprávnění
horacek	fit446	prezident klubu	vše
novak	fit789	sekretář	inzerce, členové, aktuality, protokoly, ch.stanice
dvorak	fit299	redaktor	články, inzerce
vesela	fit337	řadový člen	-
administrator	admin009	administrátor (viz úvod kap. 6.3)	přidělování oprávnění, protokoly, členové, obnova dat, systémová nastavení

Po přihlášení je v pravé části obrazovky zpřístupněna nabídka administrace těch sekcí, ke kterým má uživatel oprávnění. Ovládání celé aplikace je intuitivní a přehledné, administrační obrázkové ikonky obsahují "tooltipy".

Správa článků

V sekcích *Aktuality*, *Informace o klubu*, *Chovatelství*, *Přečtěte si*, *Plemena* a *Evidence financí* platí stejné postupy úprav článků. Vždy se jedná o tabulku, jejíž řádky odpovídají položkám veřejného menu (Náhled 1). S těmito položkami lze manipulovat pomocí ikon:

Soubory – Přidávání a odebírání souborů (dokumentů), které lze vkládat do článku jako odkazy.

Obrázky – Přidávání a odebírání obrázků, které lze vkládat do dokumentu v libovolné velikosti s libovolným popiskem (Náhled 2).

Úprava – Zadávání textu článku. Text je možno formátovat pomocí značek, které lze vkládat ikonkami nad textovým polem, kde je také odkaz na nápovědu k formátování. Nápověda obsahuje i popis vkládání obrázků a souborů.

Posuny – Slouží ke změně pořadí článku ve veřejném menu nebo na veřejné stránce.

Odstranění – Vymaže článek ze systému včetně souborů a obrázků, které k němu byly nahrány.

Archivace – Týká se jen Aktualit. Přesune aktualitu do archívu nebo z archívu.

Kategorie – Týká se jen sekce Přečtěte si, ve které bude větší množství článků, které jsou proto řazeny do kategorií. V základní tabulce je tedy seznam kategorií, u kterých lze měnit pořadí. V detailu kategorie je pak klasická tabulka pro úpravu obsažených článků. V detailu lze také změnit název kategorie.

Finance – Evidence financí je jeden statický článek, který nelze odstranit, ale pouze upravit, včetně nahrávání souborů a obrázků.

Evidence členů

Vyhledávání – Základní vyhledávání slouží pro rychlé nalezení člena podle členského čísla (stačí zadat začátek) nebo podle jména (hledá v titulu, jméně, příjmení i v celém spojeném jméně). Neomezené možnosti hledání nabízí volba *Rozšířené vyhledávání*. Systém si pamatuje poslední hledání.

Prodloužit členství – V období placení čl. příspěvků na základě výpisu z bankovního účtu klubu a variabilního symbolu konkrétní platby (tj. členského čísla) správce vyhledá člena a volbou *Prodloužit členství* jeho členství prodlouží. Systém automaticky kontroluje členské poplatky na konci platebního období a pozastaví členství každému, kdo nezaplatil příspěvek v termínu. Poté lze také členství aktivovat, pokud člen příspěvek zaplatí.

Přihlásit člena – Umožňuje správci zaevidovat člena do systému například na základě papírové přihlášky do klubu.

Žádosti o členství – Seznam žádostí podaných budoucími členy prostřednictvím internetu. Při potvrzení žádosti je člen automaticky zapsán do evidence, je mu vytvořen účet a zaslány přihlašovací údaje e-mailem. Při zamítnutí žádosti je také dotyčný informován e-mailem.

Žádosti o účet – Uživatelé, kteří byli hromadně importováni ze staré databáze před uvedením systému do provozu, nemají vytvořen účet. Mohou o něj zažádat v naší aplikaci. V případě, že nemají v databázi e-mail, čekají po odeslání žádosti a po potvrzení nového e-mailu na potvrzení žádosti správcem. V této sekci může správce potvrzovat nebo zamítnout žádosti o účet. V prvním případě systém zároveň vygeneruje dopis s přihlašovacími údaji (Náhled 4). Všechny vygenerované dopisy jsou ukládány a je možné je získat pod volbou *Dřívější dopisy*.

Nastavení oprávnění – Pokud jste přihlášeni v účtu *administrator*, přibude v seznamu členů volba *Nastavení oprávnění*, pomocí které lze členovi nastavit, do kterých sekcí administrace bude mít přístup (Náhled 3).

Uchovnění jedinci a chovatelské stanice

Nový jedinec – Údaje označené červenou hvězdičkou jsou vždy povinné. Údaje označené zelenou hvězdičkou jsou navíc povinné v případě, že je jedinec uchovněn v RK-CZ. Majitele jedince lze vybrat ze seznamu členů pomocí malého vyhledávacího okna klepnutím na *Vybrat ze seznamu*. Podobně lze vyhledat i matku a otce ze seznamu uchovněných jedinců v databázi (Náhled 6). Jedinec, který není uchovněn v RK-CZ, ale jinde, může být také vybrán jako rodič jedince.

Vyhledávání jedinců – Základní vyhledávání slouží pro rychlé vyhledání jedinců podle jména, registračního čísla nebo zápisového čísla. Rozšířené vyhledávání pak umožňuje hledat podle

všech ostatních parametrů. Nastavení řazení se použije na aktuálně zobrazený seznam jedinců i na případné další hledání, dokud toto nastavení není opět změněno.

Vyhledávání stanic – Vyhledává se v názvu stanice, v adrese, v telefonu a ve jméně majitele. Zapnutím příslušné volby se prohledává i popis stanice. Tlačítkem *Naposled změněné* se vypíše 20 naposledy upravených stanic. Tento výpis je řazen sestupně podle data změny, které je u každého záznamu zobrazeno.

Inzerce

Inzeráty ke zveřejnění – Inzeráty zde zobrazené byly podány ve veřejné inzerci. Pomocí administračních ikon u každého inzerátu lze tento zveřejnit nebo zamítnout.

Řízení e-mailem – Zadejte svoji e-mailovou adresu a stiskněte *Zapnout*, pokud chcete, aby vám podané inzeráty chodily e-mailem. V každé takové zprávě jsou umístěny přímé odkazy, pomocí kterých lze daný inzerát zveřejnit nebo zamítnout. Zvolte *Vypnout*, chcete-li tento způsob řízení pro svoji adresu zrušit.

Výstavy a zkoušky

Obě sekce mají stejné rozhraní a ovládání. Základem je seznam pořádaných klubových akcí. Kliknutím na název akce zobrazíte náhled, jak bude stránka vypadat ve veřejné sekci. Každou akci lze pomocí ovládacích ikon upravit a přidávat k ní soubory, podobně jako u článků.

Přihlašování psů – Volbou *Přidat jedince/výsledek* zadáte do systému údaje jedince z doručené přihlášky na akci. Stejně jako při výběru rodičů ve správě uchovněných jedinců lze i zde vyhledat jedince v databázi. Pokud se zde nachází, budou po jeho vybrání automaticky vyplněny údaje. Doplňte dále ručně případné nevyplněné povinné nebo další údaje (kromě výsledku) a stiskněte *Přidat*. Volbou *Přihlášení jedinci/výsledky* zobrazíte všechny jedince přidané k dané akci. Každý záznam zde můžete upravit nebo odstranit.

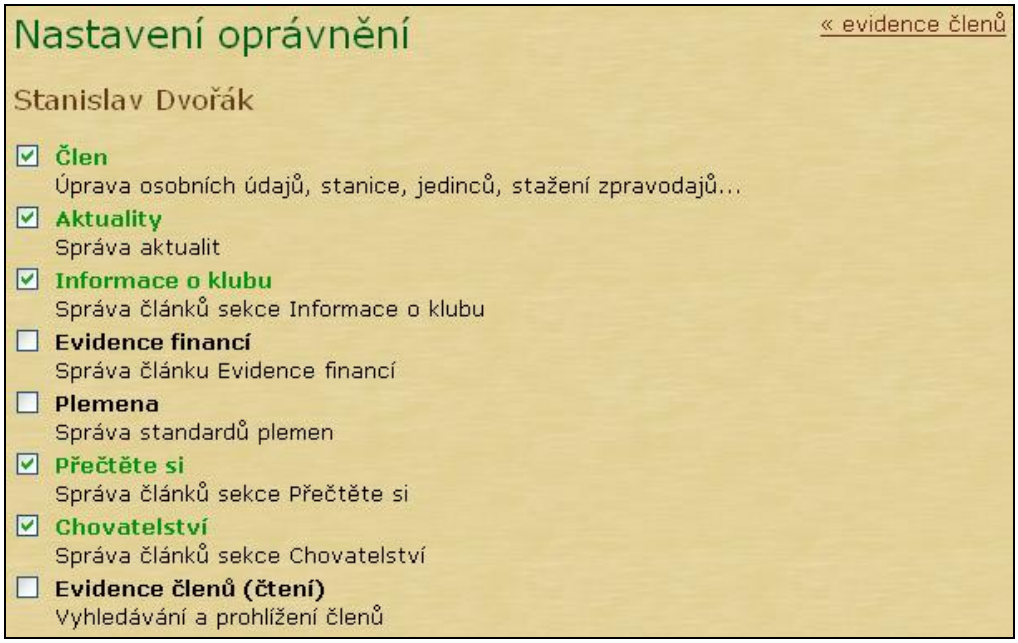
Export a import – Pokud jsou již v systému zadáni všichni jedinci přihlášení na akci (po uzávěrce přihlášek), proveďte *Export přihlášených jedinců* do souboru XLS (MS Excel). Přímo na akci lze do tohoto souboru doplňovat výsledky (body, známky, ocenění, tituly apod.). V souboru lze upravovat i všechny ostatní údaje, pokud se například přijde na chybu (Náhled 5). Po skončení výstavy nebo zkoušky se přihlaste do systému a zvolte u konkrétní akce položku *Import výsledků*. Nahráním souboru, se tyto výsledky importují do databáze včetně všech ostatních údajů. Výsledky jsou ihned přístupné veřejnosti. Importovaný soubor musí být ten, který byl z této akce exportován. Jiné soubory systém importovat nedovolí.

Správa výsledků – Pod touto volbou naleznete vyhledávání ve výsledcích výstav/zkoušek. Záznamy psů lze samozřejmě vyhledat i pokud ještě nebyly výsledky doplněny.

Náhledy obrazovek

Náhled 1 – Tabulka správy článků

Náhled 2 – Správa obrázků v článku



Náhled 3 - Nastavení oprávnění pomocí účtu administrator

Retriever klub CZ

Retriever klub CZ
Příčná 7658
130 00 Praha 3

Tereza Váňová
Plaňany 332
28002 Kolín

Vážený člene,

tímto získáváte speciální přístup do naší webové prezentace na adrese http://www.retrieverklub.cz. Níže naleznete Vaše přihlašovací údaje, po jejichž zadání na našich stránkách získáte tyto možnosti:

- Změnit své osobní údaje v evidenci klubu
- Doplnit fotografii ke každému Vašemu chovnému jedinci
- Bezplatně prezentovat vlastní chovatelskou stanici

Vaše členské číslo: 123377
Adresa webu: http://diplomka
Login: 123377
Heslo: 4ɀex13

Údaje pečlivě uschovejte, po přihlášení si můžete login i heslo změnit.

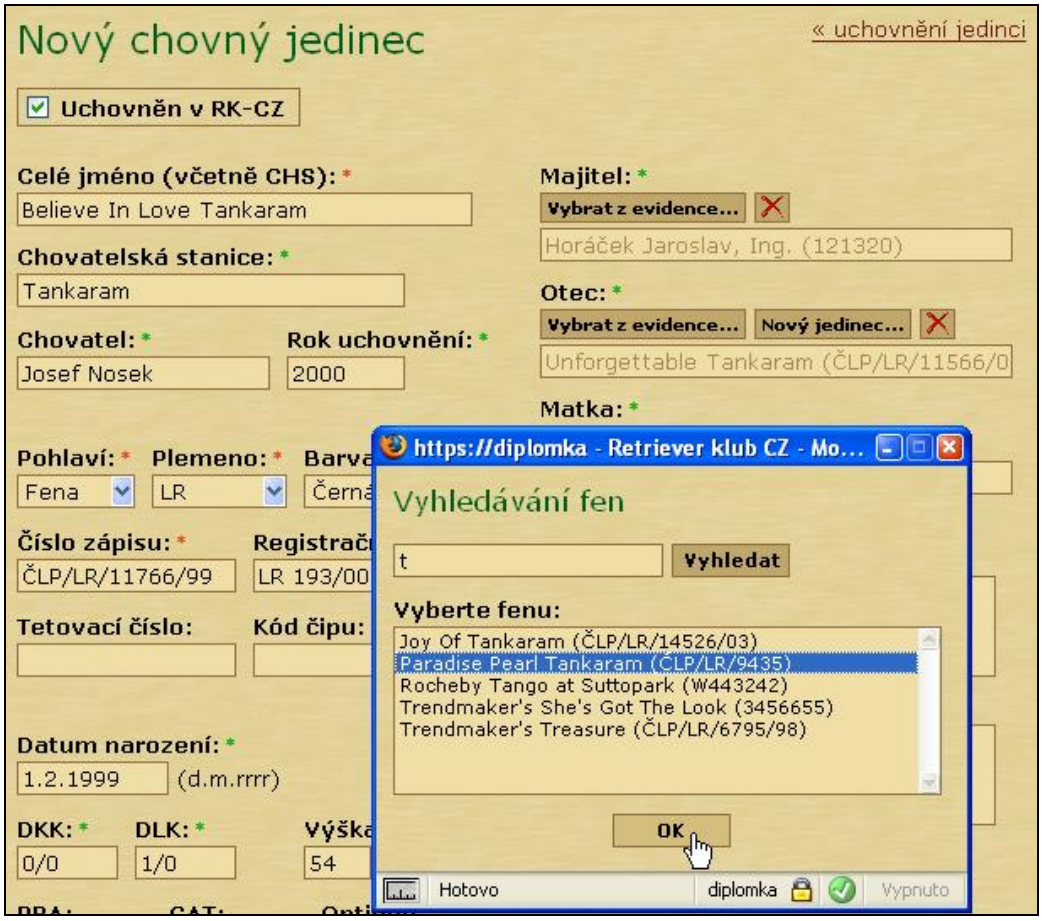
S pozdravem Retriever klub CZ

Náhled 4 – Dopis s adresním štítkem, který je umístěn v souladu s pozicí okénka v poštovní obálce

E7

	A	B	C	D	E	F	G
1	**Rožmberský pohár, Třeboň, 29.9.2006 - 1.10.2006**						
2	**Je třeba dodržet předepsané hodnoty v těchto sloupcích:**						
3	**Plemeno:** CBR, CCR, FCR, GR, LR, NSR						
4	**Pohlaví:** pes, fena						
5							
6	**Plemeno**	**Pohlaví**	**Číslo losu**	**Jméno**	**Body**	**Tituly**	**Cena**
7	LR	pes	12	Rocheby Suttonpark Sargeant Pepper			
8	LR	pes	15	Unforgettable Tankaram			
9	LR	pes	20	Rocheby Navy Blue			
10	LR	fena	23	Xavy Tankaram			
11	LR	fena	32	Believe In Love Tankaram			

Náhled 5 - Export přihlášených jedinců do dokumentu MS Excel



Náhled 6 – Přidání nového jedince, vyhledávání rodiče

Náhled 7 – Protokoly událostí