Spam a obrana před zařazením do spamové databáze emailů

SPAM and protection against spam database recording

Bakalářská práce

Miroslav Hesoun

Vedoucí bakalářské práce: Ing. Ladislav Beránek, CSc. , MBA

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

2009

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne

Anotace

Cílem práce je popsat problematiku nevyžádané elektronické pošty a spamu obecně. Popisuje možná softwarová řešení obrany proti spamu založená na prevenci zamezením samotného rozesílání těchto emailových zpráv.

Hlavní náplní práce je navržení metod zabraňujících automatickému sběru emailových adres z webových stránek boty a jejich ukládání do spamových databází. Smyslem je nalézt co možná nejúčinnější řešení zachovávající jistou uživatelskou přívětivost při přístupu k datům, tedy kontaktním údajům. Pro tyto účely budou navrženy metody kombinující běžně dostupné webové technologie takovým způsobem, aby překonaly hardwarové či softwarové možnosti vyvinuté ke sběru těchto dat.

Abstract

The aim of this essay is to describe the problems of unsolicited electronic messages and spam in general. It describes possible software solutions of defence against spam, which are based on the prevention of sending these emails at the first place.

The main purpose of this essay is to suggest methods that prevent automatic collection of email addresses from websites by bots and their storing in spam databases. The purpose is to find the most efficient solution and preserve certain level of user friendly interface allowing access these data. For this reason, methods combining commonly available technologies will be designed to overcome hardware or software means developed for this unwanted data collection.

Poděkování

Rád bych poděkoval panu Ing. Ladislavu Beránkovi, CSc. , MBA za vedení a cenné rady při psaní mé bakalářské práce.

Obsah

1 Úvod

1.1 Cíle bakalářské práce a její přínos

Pro svou bakalářskou práci jsem si vybral téma zabývající se obranou proti spamu, protože se domnívám, že neexistují kvalitní, natož česky psané zdroje, v nichž by byly komplexně rozebrány možné metody zamezující spambotům ve sběru emailových adres z webových stránek. Existují pouze nejrůznější náznaky zajímavých řešení často prezentovaných prostřednictvím blogových zápisků. Osobně se domnívám, že problematice aktivní prevence před nevyžádanou poštou by mělo být věnováno více prostoru, než který doposud zabírají nejrůznější články osvětlující neblahé dopady tohoto fenoménu a její následné řešení filtrací příchozí pošty. Internetovým technologiím obecně se již věnuji dlouhou dobu a tak bych rád prostřednictvím této práce poskytl mé znalosti a načerpané zkušenosti dalším lidem.

Tato práce si dává za cíl poskytnout ucelený informační zdroj o problému dnešní doby, jakým nevyžádaná elektronická pošta a spam obecně bezesporu jsou. Rozebere dopady na elektronickou komunikaci a nastíní některé negativní dopady na osoby i firmy přicházející se spamem denně do styku. Hlavním přínosem práce bude navržení, popsání a zhodnocení metod obrany proti spamu zamezením sběru emailových adres softwarovými boty z veřejně dostupných internetových zdrojů.

1.2 Metodika práce

Již ze samotné povahy tématu není proveditelné zamezit úplnému šíření spamu. Internet a informace na něm obsažené musí být veřejně dostupné a lidem přístupné a tak i tvůrci spamových botů mohou vždy najít algoritmus, kterým navrženou obranu obejdou. Snahou této práce tedy bude najít optimální

řešení obrany při zachování jisté uživatelské přívětivosti při přístupu k datům, v našem případě kontaktním informacím.

Existuje mnoho způsobů více či méně řešící tento problém. Z oblasti obrany proti vytváření spamových databází boty můžeme jmenovat například neaktivní emailové adresy, opisování adres z grafických předloh, zkomolení adresy do podoby srozumitelné pouze chápající osobě (nikoli botu) či emailové formuláře.

Každá z těchto metod má své výhody i úskalí a každá z nich může být modifikována do variací různých funkčních řešení, což je náplní této práce. Již výrazně méně metod se zabývá nasazením složitějších programových struktur přímo na stránkách nesoucích kýženou informaci. Možné východisko řešení proto vidím v nasazení JavaScript aplikací a flashových aplikací využívajících pokročilejší programovací jazyk ActionScript.

Při návrhu obranných mechanismů budu vycházet ze základních technik obrany proti přečtení emailových adres z veřejně dostupných internetových stránek automatizovanými boty. Neperspektivní metody následně vyřadím a blíže se budu věnovat jen potenciálně účinným metodám, které se pokusím zdokonalit. Nakonec se ale budu věnovat převážně tvorbě pokročilejších programových postupů za využití stále ještě nepříliš rozšířených technik a programovacích nástrojů.

Při jejich návrhu zohledním praktickou použitelnost, pravděpodobnost schopnosti botů zpracovat pokročilejší metody v rámci dostupných a využitelných hardwarových prostředků a uživatelskou přívětivost ve snaze zveřejněná data získat, zpracovat a využít při kontaktování daného subjektu prezentujícího se na webu.

1.3 Poznámka k použitým technologiím

Součástí mé práce bude řada ukázek konkrétních zdrojových kódů. Často budu zmiňovat serverově orientované jazyky, databázové systémy nebo HTML. Ve všech těchto případech je možno zvolit z celé řady možných nástrojů, programovacích jazyků a specifikací pro tyto účely použitelných. V případech, kde je na výběr z více variant, budu pro zachování konzistence všechny ukázky vytvářet pro následující platformy a specifikace:

- Serverově orientovaný jazyk: **PHP**
- Databázový systém: **MySQL**
- Webový server: **Apache**
- Operační systém: **Linux** (Debian)
- Značkovací jazyk HTML: **XHTML 1.0 Transitional**
- **Flash ActionScript 2.0**

Nespornou výhodou většiny uvedených platforem je jejich snadná dostupnost, velké množství výukových materiálů a licence, díky níž je můžeme využívat zcela bezplatně a to ve většině případů i pro komerční účely.

Tato práce se však nebude zabývat jejich instalací a konfigurací.

1.4 Typografické konvence

Orientaci v práci a zvláště pak v programových ukázkách usnadní několik typografických prvků.

- Veškeré zdrojové kódy, příkazy shellu a názvy souborů užité v ukázkách budou zapsány `neproporcionálním písmem`.
- *Kurzívou* jsou v textu zvýrazněny názvy aplikací, názvy parametrů či jejich hodnoty a zvláště prvně zmíněné termíny.
- **Tučným písmem** jsou ve zdrojových kódech zvýrazněny rezervované výrazy a klíčová slova daného jazyka.

- Fialovou barvou jsou ve zdrojových kódech označeny všechny proměnné, **tučným fialovým** písmem pak HTML tagy.
- Modrou barvou jsou zvýrazněny všechny řetězcové hodnoty.
- *Zelenou kurzívou* jsou označeny doplňující komentáře ke zdrojovým kódům.

1.5 Slovníček pojmů

Běžné a často užité termíny, které nejsou již v samotné práci vysvětlovány:

- *spammer* – osoba či instituce rozesílající spam
- *webmaster* – osoba provozující či spravující webovou prezentaci
- *browser* – prohlížeč webových stránek (software)
- *spyware* – aplikace odesílající informace získané z klientova počítače zpravidla bez jeho vědomí, často na principu viru
- *skript* – často nepříliš rozsáhlá aplikace interpretovaná za běhu, která je v tomto případě součástí většího celku webové stránky
- *server-side skript* – aplikace interpretována na straně webového serveru
- *cookies* – malé datové soubory užívané jako lokální úložiště informací na koncovém zařízení klienta prohlížejícího si webové stránky
- *PHP* – rekurzivní zkratka *Hypertext Preprocessor* (server-side skripty)
- *cron* – softwarový démon spouštějící ve stanovený čas naplánované úlohy, běžnou součástí operačních systémů na bázi Unixu
- *Proxy server* – komunikační prostředník mezi klientem a cílovým serverem, vůči kterému se proxy server sám chová jako klient
- *PageRank* – ohodnocení webových stránek od společnosti Google Inc.

2 Nevyžádaná pošta

V případě elektronické pošty je jako spam chápána každá příchozí zpráva, která byla hromadně rozeslána velkému množství příjemců a někteří nebo všichni příjemci si tyto zprávy nevyžádali. Obsahem těchto zpráv pak může být reklama, obchodní sdělení či podvodný text snažící se získat přístup k osobním údajům příjemce. Často se setkáváme i s emaily, které mají za účel pouze ověřit funkčnost emailových adres a skutečnost, zda je příjemce opravdu čte. Tyto emailové adresy pak mají na černém trhu větší hodnotu a následně je na ně obvykle doručováno větší množství reklamních materiálů.

Ve většině případů jsou všem příjemcům doručovány stále stejné zprávy ve velkém množství, ve snaze přimět příjemce věnovat obsahu těchto zpráv svou pozornost, kterou by jim jinak nevěnoval. Některé zprávy mají za cíl napadnout cílový počítač a získat nad ním kontrolu, čehož může být dosaženo jak podvodným jednáním, tak programátorskými technikami v podobě virů a spyware či kombinací obojího.

Předmět	Odesílatel	Datum
Re: Discount Message 46508869	VIAGRA ********...	12.3.2009 18:17
You order #77138	info@onlinovky.cz	12.3.2009 18:17
prolongedd erection	Klawiter Boroski	13.3.2009 1:52
Call To Confirm	Monica Maria Velasquez R...	13.3.2009 3:21
CitiBank - Important Message	Citibank	13.3.2009 6:47
Re: Change your ********...	VIAGRA ********...	13.3.2009 14:31
Important Information.	infodeloffice@libero.it	13.3.2009 17:27
prolonged erecction	Tunnicliff Newhart	13.3.2009 18:41
Alle Programme	Lohr	13.3.2009 20:48
Enlarge your penis and satisfy you...	Frances Doty	14.3.2009 8:27
Mail System Error - Returned Mail	Mail Administrator	14.3.2009 23:05
Very important message	Amando Foster	15.3.2009 0:12
Trust in a modern science	Nannie Wallace	15.3.2009 7:35

Obrázek 1: Spam v emailové schránce

Současný internet je spamem zahlcován ve stále větší míře. Osoby a zvláště firmy s veřejnými kontaktními informacemi tak vynakládají nemalé prostředky a čas na filtrování nevyžádané pošty. Doručení očekávaných nebo žádaných zpráv se tak stává stále problematičtějším. Spam může navíc kromě reklamy a nesmyslných textů obtěžujících adresáta obsahovat i podvodné texty či materiál nevhodný pro příjemce do jisté věkové kategorie či přímo ilegální v některých zemích.

Boj se spamem pak probíhá na dvou frontách. Buď až při snaze odfiltrovat nevyžádanou poštu na straně příjemce či přímo zabránit zanesení emailové adresy do spamové databáze. Právě druhým zmíněným postupem lze problém řešit, ne však zcela vyřešit, mnoha softwarovými metodami. Tato práce se zabývá právě jejich návrhem, popisem a zhodnocením.

Nebezpečí plynoucí z rizika odhalení citlivých osobních údajů můžeme také snížit řádnou informovaností a rozmyslem při následování požadavků v přijaté zprávě. Stejně tak je nakonec pouze naším rozhodnutím, zda proklamovaný produkt či služby zakoupíme a zda vůbec podezřelé zprávě budeme věnovat pozornost a následovat případné odkazy na externí a potenciálně nebezpečné zdroje v podobě webových stránek. Ačkoli ruční rozeznávání a přebírání užitečných zpráv od neužitečných je pracnou záležitostí, obírající příjemce o jeho čas a často i finance, v případě spamu rozesílaného téměř veškerými komunikačními prostředky v takto enormním množství se stejně stává tato technika téměř nepoužitelnou.

V případě nevyžádané pošty se vaše elektronická poštovní schránka stává smetištěm nechtěných emailových zpráv, mezi nimiž snadno přehlédnete ty zprávy, kterým byste pozornost věnovat chtěli. Nakonec vám tedy nezbude než nevyžádané zprávy oddělit od ostatních za pomoci specializovaných aplikací. Ať už s nižší či vyšší mírou úspěchu, zcela určitě všechen spam účinně neodfiltruje. Ve snaze zabránit přijetí nevyžádaných zpráv úplně, nastaví řada uživatelů svým filtrům tak vysokou úroveň filtrace, že často i běžný email

skončí v odpadkovém koši poštovní schránky. Globální statistiky přenosu elektronické poštovní korespondence ve světovém měřítku naznačují, že 9 z 10 všech rozeslaných zpráv je nevyžádanou poštou, jak dokazují nedávné průzkumy společnosti Commtouch [11].

2.1 Sběr emailových adres

2.1.1 Spambot

Spambot je software vyvinutý za účelem sběru emailových adres dostupných na webových stránkách po celém Internetu. Na rozdíl od jiných botů, tak zvaných crawlerů, kteří indexují a shromažďují prezentované informace na stránkách za účelem vytvoření veřejně dostupné databáze s vyhledáním, jako například *Google.com*, spamboti nejsou vítaným návštěvníkem žádného webu.

Spambot podobně jako crawler prochází jednotlivé stránky a hledá v nich odkazy na stránky další. Tím, co si však z jednotlivých stránek do databáze uloží, jsou pouze emailové adresy, které objevil. Ty jsou pak použity spammery pro rozesílání nevyžádané emailové korespondence.

Pokročilí spamboti dokážou rozpoznat celou řadu běžně rozšířených technik určených právě pro zamaskování emailových adres před těmito automatizovanými sběrači. Dokážou se vyhnout nástrahám, které na ně mohou provozovatelé webů připravit a pracují rychlostí, kterou jim dostupný hardware umožní. Načítají proto souběžně v několika vláknech mnoho webových stránek v jednom okamžiku a zpracovávají je okamžitě, jak obdrží odpověď od vzdáleného serveru. Mohou tak během několika sekund posbírat i stovky emailových adres.

2.1.2 Šíření emailových seznamů

Po úspěšném sklizení patřičného množství emailových adres je docela pravděpodobné, že se o ně samotný spammer podělí s dalšími zájemci a začne je za patřičný finanční obnos šířit dále na datových nosičích, skrze webové stránky nebo FTP servery.

Seznamy emailových adres se tak šíří nekontrolovatelně světem a mnozí jejich noví majitelé mají potřebu ověřit si funkčnost emailových schránek. Často se tak setkáte s emailovými zprávami obsahujícími komerční sdělení, na jejichž konci je uveden odkaz na zrušení zasílání těchto zpráv. V tomto případě je ale odkaz pravým opakem. Nejenže spammerovi potvrdíte správnost emailové adresy, ale dokonce projevíte ochotu email přečíst a následovat jisté instrukce. Tyto emaily jsou pak v databázích označeny a šířeny dále v prémiových balících za ještě větší finanční sumy. Ačkoli je tento trik poměrně snadno prohlédnutelný, sám jsem ve svém okolí narazil na celou řadu osob, které tyto instrukce následovali a stěžovali si, že jim toho snad chodí stále víc. Přitom stačí jen základní povědomí o spamu, aby k takovým situacím nemuselo docházet.

2.2 Spam v diskusních fórech

Spam není pouze výrazem popisujícím nevyžádanou elektronickou poštu. Jinou častou formou jsou zprávy hromadně rozesílané elektronickými cestami do diskusních fór a komentářových sekcí nejrůznějších článků nebo blogů. Tyto zprávy se pak jen zřídka týkají tématu daného webu nebo článku. V případě internetového diskusního fóra se může jednat o stovky zpráv z různých zdrojů zcela zahlcujících toto fórum nechtěnými příspěvky. Ty pak částečně nebo zcela znemožní uživateli pročítání a případné přispívání do fóra.

Nepíši zde jen o webmasterech majících svůj kontaktní email na webu, bavíme se zde i o uživatelích využívajících nejrůznější webové služby, jakými

jsou právě diskusní fóra, na nichž zanechávají své kontaktní informace. Provozovatelé těchto webů by si měli být problému nevyžádané pošty dobře vědomi a chránit vhodnými postupy i své uživatele před zneužitím jejich emailových adres. Například nevhodné zacházení s poskytnutými daty v podobě emailové adresy na nejrůznějších komunitních serverech by mělo pochopitelně za následek i úpadek webu samotného. Tudíž by mělo být v zájmu webmasterů poskytovat nejen sobě, ale i svým uživatelům, co možná nejlepší ochranu proti praktikám spammerů.

Spamem je chápána i pošta rozesílaná na základě registrace do diskusního fóra samotným provozovatelem webu, elektronického obchodu nebo jiných stránek, bez souhlasu příjemce. Souhlasem je zpravidla chápáno potvrzení podmínek registrace. Zasílání těchto emailů můžete často zrušit pouhým kliknutím na odkaz uvedený naspodu emailové zprávy. Existují však i společnosti, které předem na časté rozesílání informačních emailů neupozorní a zaplavují své zákazníky neustálými informace o svých produktech, třebaže zákazník u nich nakoupil jen jednou a o tyto informace zájem nemá.

2.3 Prevence versus spamové filtry

Někdy se hovoří o srovnání účinnosti technologií zabraňujících spambotům sbírat emailové adresy s algoritmy určenými k filtraci přijaté pošty. Jedná se však o dva zcela odlišné přístupy k problematice a nemohou tak být přímo porovnávány. Obrana před spamboty je ve své podstatě prevencí a filtrace pošty řešení následků. Nejúčinnější metodou je tedy užití obou přístupů. Žádná z metod totiž nemůže být v delším časovém měřítku naprosto účinnou. Kombinací obou metod se však k této hranici můžeme alespoň více přiblížit. Účinnost samotné filtrace v současnosti dosahuje nejvýše úctyhodných 99,5%. Avšak podle průzkumů společnosti McAfee [10] může i pět desetin procenta znamenat tisíci dolarové rozdíly v ročních nákladech velkých firem.

3 Obrana před automatickým sběrem emailových adres

Standardní způsob zápisu emailové adresy do webových stránek obsahuje emailovou adresu v čistém textu a hypertextový odkaz protokolu *mailto*. Tato forma zápisu je zcela validní a přístupná. Kromě čitelného textového výstupu lze na tento hypertextový odkaz kliknout a spustit tak přiřazený emailový klient. Tento způsob však adresu žádným způsobem nemaskuje a emailová adresa se tak stává tím nejsnazším úlovkem spambota procházející stránky a sbírajícího tyto adresy.

```
<a href="mailto:prijemce@domena.cz">prijemce@domena.cz</a>
```

Proto je potřeba navrhnout metody zabraňující spambotům v přečtení těchto odkazů, při zachování schopnosti uživatele dostat se ke kontaktní emailové adrese snadným a neomezujícím způsobem.

3.1 Emailová adresa v hypertextovém odkazu

Předcházející příklad lze modifikovat změnou textového výstupu. Návštěvník internetové stránky již nevidí přímo emailovou adresu, ale text odkazující na tuto adresu. Pro spambota toto ovšem nehraje žádnou roli. Spambot nemá potřebu zdrojový kód stránek interpretovat v grafické podobě a proto prochází přímo tento kód, kde je protokolem mailto adresa uvedena.

```
<a href="mailto:prijemce@domena.cz">napište nám</a>
```

3.2 Nahrazení znaků v hypertextovém odkazu

```
<a href="mailto:prijemce[zavinac]domena[tecka]cz">
  napište nám
</a>
```

Jiným často používaným, ačkoli nepříliš spolehlivým způsobem je ruční textová úprava odkazované adresy. Kromě slov zavináč a tečka umístěných v hranatých závorkách můžete použít například vložený text „semVlozteZavinac“ nebo „OdstranToto“. Čím originálnější text bude, tím větší je šance, že jej spambot nedovede nahradit podle předem daných vzorů. U česky psaných textů je tato šance o to větší, protože spamboti jsou zpravidla vyvíjeni v zahraničí pro nejširší cílovou skupinu, tedy anglicky hovořící.

Při kliknutí na takovýto odkaz je ovšem uživatel nucen adresu ručně opravit. To skýtá mnohá úskalí. V prvé řadě si uživatel nemusí všimnout, že odkazovaná adresa není v pořádku a pokusí se email neúspěšně odeslat. Uživatel také nemusí správně krátký vložený návod pochopit nebo se může prostě při editaci textu splést a email zašle na jinou nebo neexistující adresu. Nakonec je třeba zmínit nepříliš profesionální vyznění takovéhoto odkazu. I taková maličkost může poškodit image firmy nebo znepříjemnit uživatelům či obchodním partnerům emailový styk natolik, že tím firma utrpí finanční ztrátu. A právě emailové adresy určené pro styk dané firmy s veřejností jsou vystavené a lehce dostupné. Stávají se proto snadným terčem sběracích spambotů. Rozhodně tedy nelze tuto skupinu ignorovat a emailové adresy v těchto případech vkládat prostým textem. A koneckonců stejné nebo podobné důvody můžete mít v osobním životě také, třebaže nejste velkou firmou.

3.3 Kódování Unicode

```
<a
href="&#0109;&#0097;&#0105;&#0108;&#0116;&#0111;&#0058;&#0112;&
#0114;&#0105;&#0106;&#0101;&#0109;&#0099;&#0101;&#0064;&#0100;&
#0111;&#0109;&#0101;&#0110;&#0097;&#0046;&#0099;&#0122;">
  napište nám
</a>
```

Jednotlivé znaky lze převést na těžce čitelné Unicode znaky. Bez adekvátní převodní tabulky či aplikace je člověk nepřevede na původní. Z pohledu spambota se však nejedná prakticky o žádnou překážku. Stejně jako znaky

interpretuje a správně převede každý moderní webový prohlížeč na internetových stránkách, stejně tak snadno je převede spambot. Tato metoda mohla slavit svůj úspěch pouze v minulosti. V dnešní době je bezesporu každý spambot schopen Unicode náležitě zpracovat. Dokonce může být pro spambota v jistých případech i vodítkem, protože je zjevné, že se snažíme něco ukrýt.

Browser	Firefox 3.0.7	Opera 9.64	IE7	Safari 3.12	Lynx
Kompatibilní	ano	ano	ano	ano	ano

Tabulka 1: Kompatibilita kódování Unicode u vybraných browserů

3.4 Hexadecimální hodnoty

```
<a
href="mailto:%70%72%69%6A%65%6D%63%65%40%64%6F%6D%65%6E%61%2E%6
3%7A">
  napište nám
</a>
```

Podobně jako v předchozím případě můžeme adresu vyjádřit šestnáctkovým kódem. Bohužel stejně jako v předchozím případě není tato metoda přínosem.

Browser	Firefox 3.0.7	Opera 9.64	IE7	Safari 3.12	Lynx
Kompatibilní	ano	ano	ano	ano	ano

Tabulka 2: Kompatibilita kódování Hex hodnot u vybraných browserů

3.5 HTML komentáře v adrese

```
prij<!--matouci komentar-->emce@d<!--@-->ome<!--.-->na.cz
```

HTML komentáře fungují stejně jako komentáře v každém jiném jazyce a jsou zamýšleny pro okomentování kódu programátorem pro jeho pozdější

úpravy. Tyto komentáře se na webové stránce nezobrazí, nicméně v kódu, se kterým spambot pracuje, komentáře jsou. Toho lze využít pro zamaskování adresy vložením komentářů přímo do samotného textu reprezentujícího emailovou adresu.

V případě HTML jsou komentáře uvozeny čtveřicí znaků „<!--“ a ukončeny trojicí „-->“, jak je vidět na příkladu. Zvláště vhodné je pak do komentářů vkládat znaky zavináč a tečka pro vyšší míru zmatení spambota. Tato metoda však byla ve své době úspěšná a tak ji v současnosti překoná prakticky každý spambot, opět stejně jako webový prohlížeč, který tyto komentáře standardně nezobrazuje do uživatelského výstupu. Nevýhodou této metody je navíc nemožnost využití hypertextového odkazu pro otevření emailového klienta, protože HTML komentáře nemohou být vkládány do kódu jako součást parametru *href* tagu *a*. To je dáno specifikací jazyka HTML.

3.6 HTTP přesměrování

```
<?php
  header ("Location: mailto:prijemce@domena.cz");
  exit();
?>
```

Jinou možností jak zachovat hypertextovou formu emailového odkazu, aniž bychom adresu samotnou uváděli ve snadno dostupném zdrojovém kódu HTML, je přesměrování. Přesměrování probíhá na úrovni HTTP hlaviček, které jsou široce podporovány téměř všemi dostupnými internetovými prohlížeči a lze jej provést téměř každým server-side skriptem. Příklad kódu je uveden v rozšířeném jazyce PHP.

Technikou přesměrování lze ošálit alespoň ty spamboty, pro něž je tato technika neznámou. Nelze však očekávat, že bychom se při užití této metody emailového spamu zbavili úplně. Spíše eliminujeme jisté procento nevyžádané příchozí pošty.

Při využití této metody je uživatel nucen kliknout na odkaz, ze kterého by jinak emailovou adresu nevyčetl. Kliknutím se však automaticky spustí přiřazený emailový program, třebaže jej uživatel použít nechtěl a email má v úmyslu odeslat přes webové rozhraní. Adresu pak musí z programu vykopírovat. Ještě horší situace může nastat při použití určité kombinace software pro procházení webu, nebo jeho nastavení, a nepřiřazeného emailového klienta k protokolu mailto. Při kliknutí na odkaz nemusí dojít k žádnému přesměrování ani spuštění programu a emailová adresa zůstane neodhalena. Podobné problémy jsou však více než nepravděpodobné.

3.7 Obrázky

Vypisování emailových adres do obrázků je jednou z těch spolehlivějších metod chránící emailovou adresu před zneužitím. Automatičtí sběrači adres si s touto metodou často neporadí. Pro rozeznání textu v obrázku je potřeba nasadit speciální software pro rozpoznávání textu. Užití takového software je v porovnání se získanou hodnotou příliš nákladným a zdlouhavým procesem.

Pokud navíc obrázek obsahující emailovou adresu neoznačíme alternativním textem ani popiskem obsahujícím slovo „email“ či jinak spambota k tomuto obrázku nenavedeme, bude nucen spustit proces rozpoznávání na každý obrázek nacházející se na webové stránce. Doba potřebná pro extrakci adresy ze stránky tak rapidně naroste spolu s hardwarovými nároky.

V dnešní době máme stále potíže naučit algoritmy správnému rozpoznání textu. Je tedy více než pravděpodobné, že text napsaný okrasným stylem, či jinak maskovaný, nebude pro spambota čitelným. Při přílišné snaze adresu zamaskovat se však snadno přihodí, že ani člověk, jemuž byla informace určena, nebude schopen text správně přečíst a zprávu odešle na neexistující adresu. Podobný problém nastává také u dobře čitelného textu. Uživatel je nucen adresu přečíst a přepsat. Nemůže využít hypertextového odkazu, ani

možnosti zkopírovat si text do schránky. Proto může v mnoha případech dojít k chybnému opsání adresy. Z tohoto důvodu je často potřeba hledat přijatelnější metody. Ne vždy má uživatel patřičnou trpělivost s opisováním adresy, což může nakonec hrát v náš neprospěch.

Je-li navíc uživatelův počítač nastaven tak, aby nezobrazoval obrázky, nebo je uživatel zrakově hendikepován, šance na úspěšné získání kontaktní emailové adresy jsou takřka nulové. Tato metoda je však stále z hlediska přístupnosti nejméně problematickou. Jiná skriptovací řešení naráží na celou řadu zásadnějších konfliktů s přístupností. Chceme-li však automatickým sběračům emailových adres zabránit v jejich činnosti, je potřeba některé nedostatky těchto řešení opomenout. Vždy je tedy vhodné uvádět alternativní kontakt například ve formě telefonního čísla.

3.8 Otázka

Tato metoda spočívá v ukrytí emailové adresy, jejíž zobrazení je podmíněno odpovědí na otázku. Stále ještě jsme příliš vzdáleni od skutečně chápajícího algoritmu a tak nemá spambot žádnou možnost jak úspěšně na otázku odpovědět. Často se však vyskytují jednoduché otázky typu „sečti dvě čísla“. Pokud však nevěnujete nadměrné úsilí snaze umístit otázku do textu respektive kódu takovým způsobem, že ji spambot neobjeví nebo nepozná, že se o otázku k odkrytí emailu jedná, vystavujete se riziku, že se vaše emailová adresa dostane do rukou automatického sběrače. Jednoduché otázky tohoto typu jsou natolik časté a tak snadno zodpověditelné automatizovaným skriptem, že jako funkční obrana neposlouží.

U náročnějších otázek je tu pro změnu riziko, že na ni uživatel nedokáže odpovědět, třebaže vy máte pocit, že se jedná o zcela samozřejmou skutečnost. Proto mohou být užitečné otázky respektive instrukce typu „Napiš do vedlejšího textového pole ‚člověk‘, pokud jsi člověk.“.

Chceme-li se při využití této techniky vyhnout riziku odhalení emailové adresy ukryté v JavaScript kódu pokročilejším spambotem, bude potřeba testování správné odpovědi řešit skriptovacím jazykem na straně serveru, případně kombinací obojího. Při užití JavaScriptu se částečně vytrácí smysl této metody, protože emailová adresa je již sama o sobě chráněna JavaScriptem. Z toho důvodu se tento postup využívá spíše v kombinaci s formuláři.

Jakkoli může být tato metoda úspěšná a odpověď na otázku zabere jen několik okamžiků, kromě provozovatelů menších webových stránek a blogů ji využije jen málokdo pro její zcela neprofesionální přístup.

Ukázka kódu z čistě JavaScriptového řešení ilustrující tuto metodu:

```
<script language="javascript" type="text/javascript">
/* <![CDATA[ */

//převrací pořadí znaků v předaném řetězci
function revStr(str) {
  if (!str) return ''; //není-li co převracet
  var revstr = '';
  for (i = str.length-1; i>=0; i--) {
    revstr += str.charAt(i);
  }
  return revstr;
}

//zobrazí emailovou adresu
function showEmail()
{
  //zamaskuje emailovou adresu
  var userPart1 = "pri";
  //druhá část prefixu v obráceném pořadí znaků
  var userPart2bw = "ecmej";
  var atDotChars = "@."
  var domain = "domena";
  var suffix = "cz";
  //sestaví z připravených řetězců kompletní emailovou adresu
  var outcome = userPart1 + revStr(userPart2bw) +
atDotChars.charAt(0) + domain + atDotChars.charAt(1) + suffix;
  //získá odkaz na element uchovávající formulářové prvky
  var emailForm = document.getElementById('emailFormField');
  //nahradí tento element emailovou adresou
```

```
  emailForm.innerHTML = "<legend>Emailová adresa</legend>" +
outcome;
}

//zkontroluje správnost odpovědi
function checkAnswer() {
  //zamaskuje správnou odpověď
  var answer = "l" + ("duben").charAt(3) + revStr("ned");
  var userAnswer =
document.forms['emailForm'].answer.value.toLowerCase();
  //pokud je otázka správně zodpovězena, zobraz emailovou
  //adresu
  if (userAnswer == answer) showEmail();
  return false; //zablokuje odeslání formuláře
}

/* ]]> */
</script>


<form name="emailForm" onsubmit="return checkAnswer();"
action="">
  <fieldset id="emailFormField">
    <legend>
      Odpovězte na otázku pro zobrazení emailové adresy
    </legend>
    <label for="answer">
      Prvním měsícem kalendářního roku je?
    </label>
    <input type="text" id="answer" name="answer" size="10" />
    <input type="submit" value="Ok" />
  </fieldset>
</form>
```

Tento příklad čerpá z poznatků získaných v kapitole *3.13. JavaScript*. Úmyslně jsem jej zařadil již zde, ačkoli využívá jazyka JavaScript, kterým se budu zabývat až dále. Myšlenka této metody spočívá v odhalení emailové adresy po zodpovězení položené otázky a svým námětem se tak řadí mezi základní metody. Její provedení si však nutně žádá užití vyšších programovacích technik, s nimiž se seznámíme až později.

Browser	**Firefox 3.0.7**	**Opera 9.64**	**IE7**	**Safari 3.12**	**Lynx**
Kompatibilní	ano	ano	ano	ano	ne

Tabulka 3: Kompatibilita JavaScriptového řešení *otázky* u vybraných browserů

3.9 CSS pseudo-element :after

```
p:after { content: "prijemce\40 domena.cz"; }
<p>Kontaktujte nás na emailové adrese: </p>
```

Výstup: `Kontaktujte nás na emailové adrese prijemce@domena.cz`

Adresu můžeme přemístit ze samotného kódu HTML stránky obsahující informace do stylopisu popisujícího grafické vyznění dané webové stránky. K zápisu využijeme kaskádových stylů CSS (z anglického *Cascade Style Sheet*). Trojice znaků „\40“ ve výše zmíněném kódu je Unicode interpretací symbolu zavináč.

Ačkoli se standardně předpokládá, že stylopis neobsahuje žádné informace, kromě textového popisu grafické podoby stránek, pro spambota není nic snazšího, než si tento textový soubor načíst a prohledat. S narůstajícím počtem nevyžádaných zpráv šířících se po celém světě a s rozšiřující se snahou bránit se tomuto fenoménu, se i tato metoda stala poněkud průhlednou. Musíme předpokládat, že i programátor navrhující spambota, bude dostatečně s touto technikou obeznámen.

Významnou nevýhodou této metody je pak jako obvykle nižší úroveň přístupnosti. Textově založené internetové prohlížeče a starší verze grafických prohlížečů nebudou tento CSS pseudo-element podporovat. V současné době jej dokonce nepodporuje ani nejrozšířenější Internet Explorer, čímž tuto metodu činí téměř nepoužitelnou.

Browser	**Firefox 3.0.7**	**Opera 9.64**	**IE7**	**Safari 3.12**	**Lynx**
Kompatibilní	ano, ale nelze označit jako text	ano	ne	ano, ale nelze označit jako text	ne

Tabulka 4: Kompatibilita pseudo-elementu *:after* u vybraných browserů

3.10 CSS vlastnost unicode-bidi

CSS kód:

```
span.smertextu { unicode-bidi: bidi-override; direction: rtl; }
```

HTML kód:

```
<p><span class="smertextu">zc.anemod@ecmejirp</span></p>
```

Podstatou této techniky je zapsání emailové adresy do kódu v obráceném pořadí jejích jednotlivých znaků. Jinak řečeno pozpátku. Samotnému uživateli je pak adresa vypsána ve správném pořadí znaků. Pokud by spambot převzal adresu v obrácené podobě, nebude ji schopen využít pro následné rozesílání nevyžádaných zpráv. Dá se však předpokládat, že v naprosté většině případů proběhne před uložením adresy do databáze kontrola jejího tvaru. Pravděpodobně spambot adresu v textu ani nerozpozná, protože nebude odpovídat regulární šabloně tvaru emailové adresy. Tato metoda se dá však opět obejít až příliš snadným postupem. Stačí tuto šablonu pro vyhledávání přizpůsobit tak, aby zachycovala i adresy psané v opačném pořadí znaků a ty následně konvertovala do správné podoby. Přečte-li si navíc spambot přiložený kaskádový styl, bude pro něj o to snazší email rozpoznat.

Proti využití této techniky opět vypovídá nepříliš rozšířená podpora tohoto CSS pseudo-elementu. Prohlížeče řady Internet Explorer nezobrazují výsledek vždy korektně a třeba starší verze Safari element nepodporuje vůbec. Alespoň malou útěchou může být skutečnost, že uživatel patrně pochopí, že adresa je napsána pozpátku a sám si ji opraví. Při tomto postupu však může dojít k mnoha překlepům a nemůžeme se na něj spolehnout.

Browser	**Firefox 3.0.7**	**Opera 9.64**	**IE7**	**Safari 3.12**	**Lynx**
Kompatibilní	ano	ano	ano	ano	ne

Tabulka 5: Kompatibilita vlastnosti *unicode-bidi* u vybraných browserů

3.11 CSS display: none

CSS kód:

```
p span.skryt { display: none; }
```

HTML kód:

```
<p>prijemce@do<span class="skryt">vypln</span>mena.cz</p>
```

Podobně jako při doplňování HTML komentářů dovnitř emailové adresy, můžeme využít CSS pro schování těchto narušujících textů znehodnocujících celou emailovou adresu. Výplňový text stačí opatřit patřičným obalovým prvkem, v našem případě *span*, kterému přiřadíme vlastnost *display: none* stylopisem.

Na rozdíl od HTML komentářů, které jsou spambotu dostupné okamžitě při čtení zdrojových kódů stránky, CSS vlastnost je zpravidla uložena v odděleném souboru, jehož obsahem by neměla být žádná jiná informace než ta, popisující grafické vypodobnění stránek. Právě proto jsou stylopisy některými spamboty opomíjené. Bez této informace si však spambot uloží nesmyslnou a hlavně neexistující emailovou adresu.

Tuto vlastnost kaskádových stylů podporuje zpravidla každý moderní internetový prohlížeč.

Browser	**Firefox 3.0.7**	**Opera 9.64**	**IE7**	**Safari 3.12**	**Lynx**
Kompatibilní	ano	ano	ano	ano	ne

Tabulka 6: Kompatibilita CSS vlastnosti *display: none* u vybraných browserů

3.12 Formuláře

Poměrně běžnou a účinnou metodou je využití formulářů. Protože emailová adresa se nenachází v přístupném zdrojovém kódu HTML, ale v kódu

serverově orientovaného programovacího jazyka, kterým může být třeba rozšířené PHP, neexistuje možnost, jak by mohl spambot nebo kdokoli jiný emailovou adresu získat.

Uživatel vyplní připravený formulář obsahující povětšinou minimálně text zprávy, předmět a svou emailovou adresu, na kterou bude příjemce zprávy směřovat případnou odpověď. Emailové formuláře jsou často součástí komplexnějších formulářů týkajících se například cenové kalkulace žádané služby. Výstup z tohoto formuláře pak může být zaslán právě na ukrytý email nebo současně zanesen do vlastního informačního systému.

3.12.1 Standardní formulář

HTML kód formuláře

```
<form name="emailForm" action="send.php" method="post">
  <fieldset>
  <legend>Napište nám</legend>
    <p>
      <label for="name">Vaše jméno</label>
      <input type="text" id="name" name="name" size="30" />
    </p>
    <p>
      <label for="email">Vaše emailová adresa</label>
      <input type="text" id="email" name="email" size="30" />
    </p>
    <p>
      <label for="subject">Předmět zprávy</label>
      <input type="text" id="subject" name="subject"
      size="30" />
    </p>
    <p>
      <label for="message">Zpráva</label> <br />
      <textarea id="message" name="message" cols="50"
       rows="10"></textarea>
    </p>
    <input type="submit" value="Odeslat" />
  </fieldset>
</form>
```

PHP odeslání

```
<?php
//načtení předaných dat
@$name = $_POST["name"];
@$email = $_POST["email"];
@$subject = $_POST["subject"];
```

```
@$message = $_POST["message"];

//zalomí řádky po 70 znacích
$message = wordwrap($message, 70);

//doplníme do zprávy odesílatelovu emailovou adresu a jméno
$message .= "\n".$name."\n".$email;

//odeslání emailu
if (@mail('prijemce@domena.cz', $subject, $message)) {
  echo "Váš email byl úspěšně odeslán.";
}
else { echo "Při odesílání emailu došlo k chybě. Pravděpodobně
není váš email server korektně nastaven."; }
?>
```

Spambot v kódu neobjeví emailovou adresu a tak prostě pokračuje vyhledáváním na dalších stránkách. Je ale více než pravděpodobné, že formulář rozezná jako možný obranný mechanismus a automaticky vyplní všechna pole odpovídajícími hodnotami a formulář sám odešle. Případně si odkaz na formulář uloží do jiné databáze, se kterou pak pracuje jiný typ software speciálně naprogramovaný pro jejich automatické vyplnění a odeslání. Pokud by náhodou byl formulář příliš komplikovaný a bot by jej nedokázal zpracovat, máte buď vyhráno, nebo bude odkaz na formulář poznačen pro pozdější ruční zaindexování člověkem. Potom už jen stačí spustit automatizovaný program procházející tyto známé formuláře a odesílat nevyžádané zprávy, kterým se tak snažíme bránit.

Ačkoli nelze přesné hodnoty jakkoli změřit, domnívám se, že klasickým formulářem omezíte příjem spamu na minimum. To ovšem pouze platí v případě menších, příliš nenavštěvovaných a hlavně z jiných zdrojů neodkazovaných webů s nízkým stupněm ohodnocení PageRank. Je známo, že spamboti schopní formuláře zpracovávat již v dnešní době existují, není jich ovšem zatím dostatečné množství a tak se technika jeví jako dobré řešení problému automatického sběru emailových adres. Proto jsou nyní napadány zpravidla jen ty viditelnější weby. Pro spammera však není rozhodující popularita webu, na jehož provozovatele patrně získal kontakt. Spammer se

snaží rozesílat své zprávy zvláště méně informované a spíše bezbranné veřejnosti. Postupem času se emailové formuláře rozšiřují a objevují se i na menších webových stránkách osobního charakteru a blogových stránkách. Je tedy jen otázkou času, kdy budou algoritmy pro rozpoznávání emailových formulářů zahrnuty do běžných mechanismů většiny spambotů. Stále totiž platí, že spammeři drží krok s vývojem obranných metod a reagují na ně tvorbou o to sofistikovanějších nástrojů šířících spam. A koneckonců i zmíněná populární stránka může být právě nedostatečně zabezpečeným komunitním serverem obsahujícím tisíce emailových adres na běžné uživatele Internetu, kteří by jinak svou emailovou adresu ani veřejně na webu nevystavili.

Pochopitelnou snahou většiny webmasterů je tedy zefektivnění formulářů v boji proti spamu. Toho bývá často dosaženo kombinací s metodou odpovědi na položenou otázku či opisem textového kódu z obrázku. Ačkoli nenápadné políčko pro vložení ověřovacího kódu na spodu formuláře již působí profesionálněji než standardní řešení založené na opisování adresy z grafických předloh, stejně jsme se tímto krokem v podstatě vrátili na začátek naší snahy. Oproti klasickému opsání přímo emailové adresy z obrázku tady existuje výhoda v podobě možnosti okamžitě opsaný kód zkontrolovat a dát uživateli možnost opravit případnou chybu vzniklou při opisu. Opět ale nastává otázka, zda tento postup působí dostatečně profesionálně a zda není výhodnější zaplatit specializovaným společnostem až za následnou filtraci příchozích zpráv.

Zkušenosti plynoucí ze spamu v internetových diskuzích navíc dávají jasně najevo, že spammeři jsou ochotni vynaložit nemalé úsilí ve snaze překonat tyto grafické kódy. S vývojem hardware budou tyto algoritmy stále přesnější, úspěšnější a rychlejší. Proto se již nyní kódy maskují nejrůznějšími grafickými styly, fonty a šumem v textu. Osobně mohu potvrdit, že v některých případech je potřeba kód opsat i pětkrát, než se podaří jej správně interpretovat.

3.12.2 Dynamický formulář

Mou snahou je tedy formuláře elegantněji ošetřit před podobnými útoky spammerů. Protože jejich automatické odesílání se ve skutečnosti neprovádí jejich faktickým vyplněním, ale odesláním příslušných parametrů serverovému skriptu metodou POST, můžeme dynamickou změnou jmen jednotlivých formulářových polí znemožnit jejich automatické vyplňování a odesílání. V takovém případě by musel spambot navštěvovat web pravidelně a aktualizovat si své informace. I v takovém případě se dá předpokládat, že spam je skutečně odeslán až několik hodin či dnů po začlenění informací o formuláři do databáze spammera. Bude-li se tedy formulář dynamicky modifikovat například každou hodinu, zajistíme si účinnou obranu fungující v současné době na drtivou většinu spamu.

Nejprve si tedy vytvoříme tabulku v databázi. V tabulce vytvoříme počet sloupců odpovídající počtu formulářových polí, které adekvátně pojmenujeme, a sloupec *id*, ve kterém budeme uchovávat časovou známku.

Sloupec	**id**	**name**	**email**	**subject**	**message**
Typ	int	varchar	varchar	varchar	varchar
Rozsah	11	9	9	9	9

Tabulka 7: Návrh databázové tabulky pro dynamický formulář

Poté si v souboru s formulářem data z databáze načteme a výsledek uložíme do pole *row*. Načítat budeme řádek s odpovídající časovou známkou, kterou bude reprezentovat číselný údaj aktuální hodiny v rozsahu 0-23. To nám zajistí volání funkce *date* s parametrem *G*.

```
<?php
//připojení k databázi / ukázka
@$link = mysql_connect($server, $user, $pass);
```

```
$db = mysql_select_db ($database, $link);

//načtení názvů formulářových polí z databáze
$query = "SELECT * FROM dynamic_form WHERE id='".date("G")."'";
$result = mysql_query ($query, $link);
$row = mysql_fetch_array($result);

//odpojení od databáze
mysql_close($link);
?>
```

Časovou známku využijeme v případech, kdy si uživatel načetl formulář před změnou dynamických názvů formulářových polí. V databázi budou uchovány všechny vygenerované názvy po dobu 24 hodin. Formulář bude odeslán spolu s časovou známkou jeho načtení a bude tak moci být zpětně porovnán s odpovídajícím záznamem v tabulce. V opačném případě by nedošlo ke zpracování formuláře při přechodu přes celou hodinu. Přidáme proto následující neviditelné pole k formuláři z předešlého příkladu:

```
<input type="hidden" name="timestamp" value="<?php echo
date("G"); ?>" />
```

Následně upravíme formulář tak, abychom každému poli přiřadili parametr *name* dynamicky.

```
<label for="<?php echo $row["subject"]; ?>">
  Předmět zprávy
</label>

<input id="<?php echo $row["subject"]; ?>" type="text"
name="<?php echo $row["subject"]; ?>" size="30" />
```

Analogicky upravíme všechna zbývající pole. Do souboru, obsluhujícího převzetí dat z formuláře a odeslání emailu, doplníme stejnou formou načtení databázových dat do proměnné *row* jako v předchozím skriptu. Tím budeme moci správně převzít proměnné zaslané metodou POST.

```
@$name = $_POST[$row["name"]];
@$email = $_POST[$row["email"]];
@$subject = $_POST[$row["subject"]];
```

```
@$message = $_POST[$row["message"]];
```

Nakonec musíme vytvořit skript zajišťující pravidelné aktualizace názvů polí. Skript budeme spouštět každou celou hodinu automaticky softwarovým démonem *cron*. Jeho konfigurační soubor nalezneme ve většině případů v */etc/crontab*. Do něj přidáme následující novou řádku:

```
0 * * * * php -f /var/www/muj_web/cron.php
```

Ta nám zajistí spuštění skriptu *cron.php* v každou celou hodinu. Cestu k souboru samozřejmě nahradíme skutečnou cestou. Také zde předpokládám, že soubor editujete jako uživatel *root*. Nyní bude potřeba systémovým příkazem

```
crontab -u root /etc/crontab
```

aktualizovat změněné údaje. O provedených změnách se můžeme přesvědčit příkazem `crontab -l`

Ve spouštěném skriptu pak aktualizujeme informace uložené v databázi tímto výkonným dotazem:

```
"UPDATE dynamic_form
 SET name='".getRandomString(9)."',
     email='".getRandomString(9)."',
     subject='".getRandomString(9)."',
     message='".getRandomString(9)."'
 WHERE id='".date("G")."'"
```

Tím zajistíme změnu pouze řádky odpovídající aktuální hodině. Funkce *getRandomString* vygeneruje náhodný řetězec znaků z rozsahu *a-z* o délce parametru *length* a použije jej jako dynamický název pro jména formulářových polí.

```
function getRandomString($length) {
  //inicializace prazdného řetězce
  $rndStr = "";
  //vygeneruje náhodný znak a přiřetězí jej k výsledku
```

```
  for ($i=0; $i<$length; $i++) {
    $rndStr .= chr(rand(97,122));
  }
  //vrátí náhodný řetězec
  return $rndStr;
}
```

Pokud by se tvůrci spambotů rozhodli i s tímto postupem bojovat, byli by nuceni pro každé odeslání emailu přes formulář navštívit web opětovně, aktualizovat své informace o formuláři a vzápětí jej odeslat. Tím razantně narůstá čas a hardwarové prostředky pro odeslání každého jednotlivého emailu.

3.12.3 Dynamická URL

Spammerovi můžeme celou situaci ještě více znepříjemnit užitím proměnlivé adresy emailového formuláře. Na straně uživatele by toto mohlo být realizováno třeba tlačítkem, po jehož stlačení by se teprve načetla stránka s kontaktním formulářem. Stejně dobře by postačil i odkaz v menu, v takovém případě je ale nutno dobře navrhnout navigaci webu, aby člověk přicházející z dříve vytvořené záložky přímo na webový formulář obdržel požadovanou stranu.

K realizaci této techniky bychom vložili do odkazující stránky link s dynamicky generovanými parametry ve tvaru:

```
<a href="formular.php?timestamp=<?php echo date("G");
?>&amp;code=<?php echo $row["code"]; ?>">Kontaktujte nás</a>
```

Opět zde bude potřeba přenášet časovou známku. Parametr *code* je načítán z databázové tabulky, v níž se nachází sloupec *id*, který obsahuje časovou známku a sloupec *code*, který obsahuje náhodně vygenerovaný kontrolní řetězec znaků. Tento kód je poté na odkazované stránce s formulářem ověřován a formulář se načte pouze při shodě přeneseného kódu s odpovídajícím kódem v databázi.

```
if ($code == $row["code"]) {
  //vypiš formulář
}
else { echo "Neplatný kód"; }
```

Podobně jako u dynamického formuláře i zde bude zapotřebí spouštět pravidelně každou celou hodinu skript obměňující patřičná data v databázi.

```
"UPDATE dynamic_url
 SET code='".getRandomString(9)."',
 WHERE id='".date("G")."'"
```

3.12.4 Detekce bota

Jednotlivá formulářová pole mohou být dokonce opatřena JavaScript kontrolou, ověřující, zda uživatel opravdu všechna příslušná pole aktivoval (zda pole obdržela *focus* a vygenerovala patřičnou událost). Pokud by se tak nestalo, formulář by nebyl odeslán.

Vystavujeme se však riziku vypnutého Javascriptu na straně uživatele. Avšak za předpokladu výjimečnosti této situaci si můžeme dovolit takto odeslaný formulář alespoň přesměrovat na sekundární emailovou adresu s nižší prioritou. Tedy takovou, kterou kontrolujeme méně často a předpokládáme, že bude obsahovat pouze spam.

Parametr *action* formuláři nastavíme standardně na sekundární skript, čímž zajistíme, že bez podpory JavaScriptu bude email odeslán právě na sekundární emailovou adresu.

Všechna povinná pole opatříme handlerem události *onfocus*, který zapíše do globálního pole `focusSet` hodnotu *true*. Jednotlivé indexy pole očíslujeme od nuly až do počtu formulářových prvků mínus jedna.

```
<input onfocus="focusSet[2]=true;" type="text" id="subject"
name="subject" size="30" />
```

Při stlačení odesílacího tlačítka zavoláme funkci kontrolující přiřazené události *onfocus*.

```
<input onclick="checkFocus();" type="submit" value="Odeslat" />

function checkFocus() {
  var sendSpam = false;
  for (i=0; i<focusSet.length; i++) {
    if (focusSet[i] != 1) {
      //nastaví příznak detekujícího spambota
      //to způsobí přesměrování zprávy na sekundární email
      sendSpam = true;
    }
  }
  //přesměruje na primární emailovou adresu
  if (sendSpam == false) {
    document.forms['emailForm'].action = 'send.php';
  }
}
```

Pokud úspěšně ověří předchozí aktivaci všech formulářových polí, přesměruje na primární emailovou adresu v podobě skriptu `send.php`.

Nakonec doplníme do globální části JavaScriptu, spouštěné automaticky při načtení stránky, deklaraci zmíněného pole a inicializujeme všechny jeho potřebné prvky hodnotami *false*.

```
var focusSet = new Array();
for (i=0; i<4; i++) { focusSet[i] = false; }
```

Browser	**Firefox 3.0.7**	**Opera 9.64**	**IE7**	**Safari 3.12**	**Lynx**
Kompatibilní	ano	ano	ano	ano	ne

Tabulka 8: Kompatibilita metody detekce bota u vybraných browserů

3.13 JavaScript

Užití JavaScriptu s sebou přináší řadu nesporných výhod. Díky skriptovacím vlastnostem tohoto jazyka se nemusíme omezovat na pouhé

skrytí či zkomolení emailové adresy. JavaScript dává programátorovi možnost vhodným způsobem zkombinovat základní řetězcové operace s řadou událostí volaných automaticky prohlížečem při interakci uživatele s webovými stránkami a odhalit tak případného bota.

Proti užití JavaScriptu hovoří snížená úroveň přístupnosti, jako ostatně u většiny metod, jejichž cílem je znesnadnit přístup k datům spamboty. V současné době je stále využití JavaScriptu vnímáno jako překážka. Platí však tato zažitá praxe i nadále? V dnešní době je tento skriptovací jazyk naprosto běžnou součástí všech webových prohlížečů s výjimkou několika málo úzce specializovaných. Objevuje se také jako součást všech moderních mobilních zařízení, které jsou schopny zobrazit webové stránky. Na druhou stranu, málokterý, pokud vůbec nějaký, spambot je v současnosti schopen zcela interpretovat tento jazyk.

Browser	**Firefox 3.0.7**	**Opera 9.64**	**IE7**	**Safari 3.12**	**Lynx**
Kompatibilní	ano	ano	ano	ano	ne

Tabulka 9: Obecná kompatibilita JavaScriptu u vybraných browserů

3.13.1 Zřetězení adresy

Vyčíst emailovou adresu ze skriptu lze ale i bez schopnosti kódu plně porozumět. Příkladem budiž klasický postup rozdělení adresy do několika proměnných, které na výstupu složíme v jeden řetězec textu reprezentujících emailovou adresu. Pokud bychom při kliknutí na odkaz vyvolali skriptem mailto protokol, opět bychom se snadno dostali do potíží, protože toto přesměrování může být poměrně snadno zachyceno a emailová adresa přečtena.

Ukázka kódu založeného na rozdělení adresy do více proměnných:

```
<script language="javascript" type="text/javascript">
/* <![CDATA[ */

    var pom1 = "pri";
    var pom2 = "jemce";
    var pom3 = "@";
    var pom4 = "domena.cz";

    var vysledek = pom1 + pom2 + pom3 + pom4; //zřetězení

    document.write('<a href = "mail' + 'to:' + vysledek + '">'
    + vysledek + '</a>'); //výstup

/* ]]> */
</script>
```

Nejvhodnějším postupem tedy zůstává pouhé zobrazení výsledné emailové adresy na webových stránkách prostým textem bez hypertextu. Můžeme si ale odpustit nutnost textový výsledek převádět do grafické podoby a zbytečně zamezit uživateli v kopírování adresy. Principem metod založených na JavaScriptu totiž není potřeba zamezit přečtení výstupu, ale v prvé řadě znemožnit spambotu výstup obdržet. Ale i při takovém postupu je potřeba dbát zvýšené pozornosti. Často se setkávám s komplikovanými řešeními, která mají adresu uvedenou v čistém textu přímo ve zdrojových kódech JavaScriptu, které jsou však snadno přístupné. JavaScript není potřeba překládat do strojových kódů srozumitelných pouze pro počítač, jako je tomu u většiny aplikací pro počítače a jiná elektronická zařízení, a proto do něj může být jednoduše nahlíženo. Z toho důvodu se vyplatí samotné zřetězení kombinovat s operacemi nad řetězci a vynutit si tak na straně spambota schopnost tyto funkce interpretovat. Tohoto postupu využívá i příklad z předchozí kapitoly *3.8. Otázka*, který je bohatě okomentován a nepotřebuje bližší vysvětlení.

3.13.2 Odstranění nadbytečné vsuvky

Kombinací JavaScriptu s metodou založenou na doplnění nežádoucích, snadno rozpoznatelných a tedy oddělitelných textů do samotné emailové adresy, můžeme docílit elegantního řešení, které zbaví uživatele nutnosti ručně zasahovat do úpravy emailové adresy. Typicky se jedná o adresy *prijeSMAZTOTOmce@domena.cz*.

```
<script type="text/javascript">

    function skryt(){
        //nalezení nadbytečného textu v HTML kódu
        spamText = document.getElementById('vsuvka');
        spamText.style.display = "none"; //skrytí textu
    }

</script>

<span onmousedown='skryt();'>
  prijemce@<span id="vsuvka">SPAM</span>domena.cz
</span>
```

Při kliknutí nebo označení tohoto textu dojde k automatickému odstranění nadbytečného řetězce z adresy.

Tento příklad kombinuje událost při kliknutí myši s řetězcovými operacemi. Výhoda takového řešení spočívá v neschopnosti spambota zjistit, které uvedené události má opravdu spustit, aby odkryl hledanou emailovou adresu.

3.13.3 Past na spambota

Automatické spouštění všech skriptů by vedlo k obrovskému navýšení hardwarových a časovým nároků spolu s rizikem pádu celého systému. Mohlo by se totiž snadno stát, že by bot narazil na stránku s chybně naprogramovaným kódem, jehož spuštění by celý software zablokovalo. Toho můžeme dokonce využít a vytvořit si speciální stránku se záškodnickým kódem. Tuto stránku pak můžeme nenápadně a nejlépe neviditelně odkazovat

z ostatních stránek webu. Uživatelé se na ni tak velice pravděpodobně nedostanou, zato spambot ano. Kód by mohl vypadat například takto:

```
<script type="text/javascript">
    while(true) { result = Math.sin(12345*1.4879)/0.47498; }
</script>
```

Jakmile na tuto stránku spambot narazí, bude uvězněn v nekonečné smyčce nesmyslných výpočtů s desetinnou čárkou, které nemalým dílem přispějí k jeho zpomalení a případnému zablokování. Aby se spambot podobným pastím vyhnul, musel by je umět detekovat předem nebo dodatečně ukončovat po vypršení maximální možné doby přidělené pro JavaScriptové výpočty. Je však potřeba si uvědomit, že tato technika nezabrání spambotu v nalezení vaší emailové adresy. Smyslem této techniky je znepříjemnit spammerům jejich aktivity. V současnosti ale podobný skript velkého užitku nenalezne, protože drtivá většina spambotů není schopná JavaScriptové kódy patřičně interpretovat. Rozvoj obranných prostředků nevyhnutelně směřuje k širšímu nasazení skriptování a čím dříve jej začnou webmasteři využívat, tím dříve se dá očekávat adekvátní reakce spammerů.

3.13.4 Kombinace událostí

Zkombinovat můžeme taktéž více událostí, jejichž postupná aktivace bude jediným možným postupem k úspěšnému obdržení adresy. Zvláště výhodné je umístit inicializační kód do události *onload*.

Jednoduchou obměnou metody odmazávající nadbytečný text z emailové adresy, který jsem uvedl dříve, můžeme kýžené kombinace dosáhnout.

```
<script type="text/javascript">

    var element = 'none';

    function skryt(){
        //nalezení nadbytečného textu v HTML kódu
        spamText = document.getElementById(element);
```

```
        spamText.style.display = "none"; //skrytí textu
    }

    function setupElement() { element = 'vsuvka'; }

</script>

<body onload="setupElement();">
```

3.14 AJAX

AJAX neboli *Asynchronous JavaScript and XML* je jinými slovy synonymem pro spojení Javascriptu a serverově orientovaných programovacích jazyků. Nejčastějším případem je kombinace Javascript a PHP. Ke komunikaci mezi těmito skripty pak dochází bez nutnosti znovu načítání webové stránky skrze komunikační kanál založený na XML formátu. Použít lze ale ve své podstatě libovolný formát včetně prostého textu.

Díky této technologii je možné využívat v JavaScriptových úlohách databáze, spouštět serverové skripty schopné pokročilejších výpočtů a v neposlední řadě ukrýt podstatnou část zdrojového kódu před spambotem. Jak je totiž známo, zdrojový kód serverově orientovaných jazyků není přes webové rozhraní dostupný. Toho můžeme využít zvláště při ukrytí emailové adresy.

Ukázka předání emailové adresy technologií AJAX a její vypsání na webové stránky prostřednictvím JavaScriptu:

```
<script type="text/javascript">
var http_request = false;

function makeRequest(url) {
  http_request = false;
  //vytvoření instance XMLHttpRequest
  if (window.XMLHttpRequest) {
     http_request = new XMLHttpRequest();
     if (http_request.overrideMimeType) { //browsery mimo IE
        http_request.overrideMimeType('text/html');
     }
  } else if (window.ActiveXObject) { //pro Internet Explorer
     try {
        http_request = new ActiveXObject("Msxml2.XMLHTTP");
```

```
     } catch (e) {
        try {
           http_request = new
      ActiveXObject("Microsoft.XMLHTTP");
        } catch (e) {}
     }
  }
  //otevře spojení
  //GET - typ přenosu
  //url - adresa volaného skriptu
  //true - aktivuje asynchronní přenos
  http_request.open('GET', url, true);
}

function setupMail() {

  //odeslání požadavku na získání dat
  makeRequest("getmail.php");

  //zpracování odpovědi
  http_request.onreadystatechange = function () {
    //odpověď přijata
    //readyState = 4 -> complete / status = 200 -> OK response
    if(http_request.readyState == 4) {
      if(http_request.status == 200) {

          //vypsání emailové adresy do HTML stránky
          var ajaxmail = document.getElementById('ajaxmail');
          ajaxmail.innerHTML = http_request.responseText;

    }}
  };
  //vyšle prázdnou odpověď
  http_request.send(null);
}

</script>

<body onload="setupMail();">

<span id="ajaxmail"></span>
```

getmail.php

```
<?php
header("Content-Type: text/html; charset=utf-8");
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);

echo "prijemce@domena.cz";
?>
```

Ačkoli je tento skript na pohled rafinovaným řešením, při globálním nasazení ve webových stránkách by se spammeři snadno adaptovali. Stačí si totiž přečíst předanou informaci přímo v textovém výstupu serverového skriptu, jehož případně variabilní adresu lze bez větších potíží vyčíst v JavaScriptu. Proto je potřeba zavést jistou formu šifrovaného přenosu informace mezi těmito dvěma programovacími jazyky. Přes veškeré úsilí se mi ale nepodařilo nalézt takovýto způsob a domnívám se, že data doopravdy šifrovat nejde. Tím mám na mysli skutečnost, že data sice můžeme přenášet v šifrované podobě, ale nutně musí existovat možnost jak je na straně druhé dešifrovat. Při jakémkoli přenosu libovolných dat, ať už směrem od JavaScriptu nebo k němu, musí existovat odpovídající druhá část kódu právě v JavaScriptu a budeme-li předpokládat, že spambot umí JavaScript zpracovat stejně jako webový prohlížeč, nebude se jednat vlastně o nic inovativnějšího, než samotné použití JavaScriptu.

Do nasazení AJAXu, při obraně před automatickými sběracími boty emailových adres, jsem vkládal větší naděje. Pokud navíc zohledním sníženou míru přístupnosti podobného řešení, musím konstatovat, že AJAX opravdu neposkytuje kýžené prostředky k maskování emailové adresy. Můžeme vzít v úvahu případnou nekompatibilitu spambota s touto technologií jako možnou výhodu, avšak s tím zároveň zvyšujeme riziko nepřístupné informace uživateli. Kombinace technologií v podobě AJAXového můstku je opět novější, nežli samotný JavaScript a tak navzdory globální podpoře JavaScriptu, se AJAX nemusí v řadě webových prohlížečů správně zpracovat. Většina moderních prohlížečů však AJAX podporuje a vše nasvědčuje jeho vzrůstající popularitě a budoucí rozšířenosti. Obrana proti spambotům ale nebude tím odvětvím, které na jeho rozvoji vydělá.

Browser	Firefox 3.0.7	Opera 9.64	IE7	Safari 3.12	Lynx
Kompatibilní	ano	ano	ano	ano	ne

Tabulka 10: Kompatibilita technologie AJAX u vybraných browserů

3.15 Flash

Kromě otevřených programových skriptů s dostupným zdrojovým kódem můžeme využít i skriptů načítaných pomocí externích modulů do prohlížeče. Takové přídavné moduly označujeme anglickým slovem *plugins*. Tyto moduly nebývají běžnou součástí internetových prohlížečů a musí být uživatelem ručně doinstalovány. Existuje však jeden plugin, který je v dnešní době téměř standardem moderního webového prohlížeče a tím je Flash od společnosti Adobe.

Již všichni jsme si možná zvykli na pohyblivé animující se reklamní plochy na webových stránkách, nejrůznější interaktivní aplikace nebo dokonce hry spouštěné přímo ve webovém prohlížeči. A právě tyto elementy webových stránek jsou zpravidla vytvořeny technologií Flash. Využití této technologie pro uschování emailové adresy před automatickými sběrači je obrovským lákadlem. Flash, podobně jako jiné vývojové nástroje, disponuje skriptovacím jazykem. V našem případě se jedná o ActionScript.

ActionScript je často srovnáván s Javascriptem, ovšem na rozdíl od něj se neustálým rychlým tempem rozvíjí, modernizuje a modifikuje do podoby, která je v dnešní době spíše podobná profesionálním objektově orientovaným jazykům. Dalším specifickým rysem Javascriptu, kterým naštěstí pro nás ActionScript nedisponuje, je otevřenost a přístupnost zdrojových kódů. Za předpokladu, že by byl spambot opravdu naprogramován pro sběr adres z těchto Flash aplikací, musel by aplikaci v HTML kódu najít, stáhnout

z internetu, dekompilovat, vytáhnout kódy ActionScript a najít v nich emailovou adresu.

Nejenže celý tento proces je ve srovnání se získanou hodnotou nesmírně časově a hardwarově náročný, ale nalezení emailové adresy v samotných zdrojových kódech ActionScriptu může být mnohem náročnější než jen pouhé vyhledávání vzoru. Adresa může být do kódu zakomponována nespočtem programátorských technik takovým způsobem, že by ji obdržel pouze spambot schopný tento kód interpretovat. A přesto by mohl narazit na řadu těžce řešitelných překážek. Také je potřeba zmínit, že ActionScript může být proti dekompilování velice úspěšně chráněn zašifrováním, a že jen zlomek Flash aplikací na internetu ve skutečnosti emailovou adresu obsahuje. To ovšem spambot nedokáže zjistit předem a tak by celý tento náročný a v dosti případech neúspěšný proces, musel opakovat pro všechny Flash aplikace nalezené na daných webových stránkách, třebaže v nich emailová adresa ukryta nebude.

A proč tedy zůstává tato metoda téměř bez povšimnutí a zcela jistě není rozšířeným způsobem prevence v boji proti spamu? Ačkoli Flash je dnes již velice rozšířenou technologií a již málokterý počítač jí není vybaven, stále ještě se jedná o příliš novou technologii. Osobní počítače již disponují patřičným výpočetním výkonem, ale v nejrůznějších mobilních zařízeních se Flash teprve začíná objevovat. Flash je také stále spojován se zábavním průmyslem, ať už hovoříme o webových hrách, animovaných filmech nebo reklamách, a tak je v řadě podniků a institucí Flash z těchto, anebo z bezpečnostních důvodů úmyslně zablokován.

Zamaskování emailové adresy do flashové aplikace tak s sebou v současné době přináší významný krok zpět v tvorbě přístupného webu. Je ale potřeba zohlednit i cíle dané osoby či instituce, která email zveřejňuje. Spokojí-li se se skutečností, že email bude dostupný omezenému množství lidí, může technologii bez obav využít. Počet lidí neschopných takovýto email zobrazit

rapidně klesá a u mobilních zařízení jsou si uživatelé vědomi jejich nedostatků a zpravidla mají k dispozici plnohodnotný osobní počítač. Vždyť ani velcí inzerenti nad tímto problémem příliš nepřemýšlejí a své reklamy do internetu vypouštějí právě v podobě Flash animací. Významná obchodní společnost či instituce by si však neměla tento prohřešek proti přístupnosti dovolit. Metoda proto nemusí být vhodná ve všech případech.

Je obtížné v dnešní době predikovat vývoj informačních technologií ovlivněných celou řadou činitelů, mezi nimiž v našem případě pravděpodobně nejvíce vyniká ekonomika a ekonomické smýšlení velkých nadnárodních společností vyvíjejících technologie spjaté s přenosem informací přes Internet. Přesto si však dovolím tvrdit, že právě Flash je tou technologií, která bude za několik let chápaná jako běžný standard webové prezentace, stejně jako jsou v současnosti takto chápány obrázky na webu. Proto bude nasazení ActionScriptu nabývat stále většího smyslu. I s vývojem nových programových a hardwarových technologií nebudou moci tvůrci spambotů držet krok. V dnešní době je již téměř každý počítač schopný interpretovat Flash, ale už nejspíše žádný spambot. Vzniku nových programových technik dodávajících spambotům tuto schopnost a vývoji nových hardwarových prostředků schopných v efektivně vyplácejícím se čase tyto adresy z webů extrahovat, chybí ještě léta vývoje. Naproti tomu na uživatelské straně je již vše téměř funkční nebo připravené. S trochou nadsázky tak mohu tvrdit, že již stačí Flash pouze prohlásit za obecně uznávaný standard při tvorbě přístupného webu. Ve skutečnosti bychom objevili jisté nedostatky při snaze přistoupit k takovému webu nevidomými lidmi, nicméně v případě emailových adres tu stejně stále může být ona alternativa v podobě telefonního čísla nebo dokonce v případě nevidomých osob v podobě zvukové nahrávky obsahující emailovou adresu.

Kromě přístupnosti jsou tu i další nevýhody, bránící masívnímu nasazení Flashe jako obranného prostředku proti automatickému sběru emailových

adres. Znalost tohoto vývojového prostředí a skriptovacího jazyka ActionScript rozhodně nemá každý webmaster či webový nakladatel a cena nutná k vynaložení na koupi potřebného software se zcela určitě nevyplatí pouze pro zamaskování emailové adresy, nehovoříme-li o ziskových společnostech, ale o osobách bránících se nevyžádané poště. Proto jsem navrhl jako jedno z možných řešení software šiřitelný za pomoci *swf* souborů. Díky němu si pak každý může jednoduše na svých webových stránkách zprovoznit aplikaci poskytující emailovou adresu právě prostřednictvím technologie Flash.

3.15.1 Vlastní Flash aplikace

Nejprve si vytvoříme konfigurační soubor, který nazveme *flashmail.txt*. Ten umístíme do stejného adresáře jako flash aplikaci. Díky tomu bude moci každý změnit základní nastavení bez nutnosti editovat samotnou aplikaci.

Soubor bude obsahovat parametry:

- *email*: emailová adresa v prostém textu
- *color*: barva textu v šestnáctkovém tvaru
- *underline*: podtržení textu (true/false)
- *backgroundColor*: barva pozadí

Obsah souboru *flashmail.txt*:

```
&email=prijemce@domena.cz&
&color=FFFFFF&
&underline=true&
&backgroundColor=E3801F&
```

Poté přistoupíme k tvorbě aplikace. Nastavíme rozměry animace, já jsem zvolil 300x50 pixelů. Na prvním snímku vytvoříme skript, který načte data z konfiguračního souboru a inicializuje jimi proměnné.

```
//zastaví animaci
stop();

//deklarace proměnných
var email:String = new String("");
```

```
var emailColor:String = new String("");
var emailUnderline:String = new String("");
var bgColor:String = new String("");

var lv:LoadVars = new LoadVars();

//načtení a inicializace proměnných
lv.onLoad = function(success:Boolean){
      if (success){
            email = this["email"];
            emailColor = this["color"];
            emailUnderline = this["underline"];
            bgColor = this["backgroundColor"];

            //jakmile data úspěšně obdrží, spustí animaci
            //a přejde na další snímek
            this.onData(play());
      }
}

lv.load("flashmail.txt");
```

Na následujícím snímku vytvoříme prázdný filmový klip, do něhož vložíme dynamické textové pole, kterému nastavíme zarovnání doleva na jeden řádek a přiřadíme mu proměnnou *varEmail*. Instanci filmového klipu pojmenujeme *mcEmail* a instanci textového pole *txtEmail*. Rozměry textového pole přizpůsobíme velikosti animační plochy.

Vytvoříme si novou vrstvu pod stávající vrstvou, do které vložíme přes celou délku animace nový filmový klip. Do tohoto klipu nakreslíme obdélník libovolné barvy o rozměrech větších, než jaké jsme zvolili rozměry scény. Klip nazveme *mcBackground*.

Nad instancí filmového klipu *mcEmail* definujeme událost po kliknutí, která protokolem mailto otevře standardně přiřazený emailový klient.

```
on (release) {
      this.getURL("mailto:" + _root.email);
}
```

Na druhém snímku hlavní časové osy pak vytvoříme stěžejní část skriptu vypisující email do připraveného textového pole. Zde také upravíme barvy a vzhled animace podle uživatelsky definovaných parametrů.

```
stop();

var txtFormat = new TextFormat();
var currentSize:Number = 28;
var currentWidth:Number;
var startWidth:Number = _root.mcEmail.txtEmail._width;

//načte emailovou adresu do připraveného textového pole
_root.mcEmail.varEmail = _root.email;

//zmenší písmo, pokud je emailová adresa příliš dlouhá
_root.mcEmail.txtEmail.autoSize = "left";
currentWidth = _root.mcEmail.txtEmail._width;
if (currentWidth > startWidth) {
      txtFormat.size = currentSize/(currentWidth/startWidth);
}
_root.mcEmail.txtEmail.autoSize = "none";
_root.mcEmail.txtEmail._width = startWidth;

//zobrazí emailovou adresu určenou barvou
_root.mcEmail.txtEmail.textColor = "0x" + _root.emailColor;

//vyplní pozadí danou barvou
changeColor = new Color(mcBackground);
changeColor.setRGB("0x" + _root.bgColor);

//podtrhne odkaz, pokud je potřeba
if (_root.emailUnderline == "true") {
      txtFormat.underline = true;
      _root.mcEmail.txtEmail.setTextFormat(txtFormat);
}
```

Výslednou aplikaci exportujeme a standardním postupem vložíme do webových stránek. Díky možnosti modifikovat barevné nastavení a styl emailového odkazu, můžeme flashový odkaz na stránky vložit, aniž by působil výstředně, či jinak deformoval vzhled stránek.

Je zřejmé, že jako každá jiná technika i tato by byla napadnutelná za předpokladu, že by se ji rozhodl využít téměř každý webmaster. Pro programátora sběracího spambota je pak již mnohem snazší vytvořit

specifický program schopný extrahovat emailovou adresu z dané Flash aplikace. Krása technologie Flash, jakožto vyššího moderního a objektového orientovaného programovacího jazyka, však spočívá v možnosti navrhnout zcela různé postupy pro dosažení téhož cíle. Za předpokladu, že by byly k dispozici obdobné aplikace od různých tvůrců ve velkém množství, nebudou programátoři spambotů schopni ani zdaleka podchytit všechny z nich a připravit pro ně odpovídající spamboty. Už jen samotný rozpoznávací proces by byl poměrně náročný. Troufám si proto říci, že v rozsahu několika následujících let by se vývoj podobných spambotů nevyplatil a rozesilatelé spamu by patrně začali hledat alternativní cesty sběru adres, než jen procházení webových stránek.

Browser	**Firefox 3.0.7**	**Opera 9.64**	**IE7**	**Safari 3.12**	**Lynx**
Kompatibilní	ano	ano	ano	ano	ne

Tabulka 11: Kompatibilita technologie Flash u vybraných browserů

3.16 Automatická blokace spambotů

Spamboti se po webových stránkách pohybují zcela odlišným způsobem než lidé. Jakmile se dostanou na jednu ze stran webu, nejčastěji tu titulní, vyhledají si všechny odkazy z této stránky na další stránky v rámci webu a postupně je procházejí za účelem nalezení emailové adresy. Tento proces je velice rychlý, protože spambot si na rozdíl od člověka načte pouze textový zdrojový kód bez obrázků, animací a dalších datově náročnějších elementů webu. Spambot na stránkách nejčastěji vyhledává přeprogramovaný vzor, podle kterého odhalí přítomnost emailové adresy na stránce. Naproti tomu uživatel si patrně stránky pročítá a kliká na další odkazy až po delší době, řádově sekundy až minuty.

Jako jedna z možných technik obrany proti spambotům může být aktivní obrana zablokováním přístupu po úspěšné detekci spambota na základě jejich

pravděpodobného chování. K tomu by bylo zapotřebí zaznamenávat každé zobrazení všech stran webu do databáze a nad těmito daty nepřetržitě provádět detekční algoritmus. To by však nepřiměřeně zatížilo webový server a v mnoha případech by tento postup vedl ke kolapsu serveru. Pokud bychom detekci prováděli zpětně a IP adresy spambotů blokovali až tehdy, nejenže by v předchozí návštěvě spambota byly naše nezabezpečené emailové adresy jistě zcizeny, ale v případě spambotů využívajících celou řadu proxy serverů, kterými se maskují, nemůžeme předpokládat, že budou při své příští návštěvě opatřeni stejnou IP adresou jako posledně.

Abychom tedy mohli detekci provádět v reálném čase, přeneseme výpočetní zátěž na samotné klienty využitím JavaScriptu. Abychom však mohli simulovat databázové úložiště, bude zapotřebí cookies. Pokud však spambot odmítne cookies přijmout, což je více než pravděpodobné, budeme moci zatím neidentifikovaného klienta buď automaticky na web pustit, nebo zablokovat. V druhém případě ale riskujeme zablokování běžného uživatele a proto by byl ten postup zcela proti zásadám přístupnosti webu a sami bychom si uškodili, kdybychom některým návštěvníkům neumožnili přístup na web jen proto, že mají z nějakého důvodu zablokován příjem cookies. V prvním případě bychom pro změnu na web povolili vstup v podstatě každému včetně spambotů a nasazení celého detekčního algoritmu by bylo ve své podstatě zbytečné. Proto jeho využití nedoporučuji a raději bych se věnoval snaze o maximální možnou optimalizaci serverově orientované varianty.

3.16.1 Modul memcached

Naštěstí existují způsoby, kterými lze vytížení serveru při podobných operacích snížit na minimum. Naneštěstí nebudou v naprosté většině případů dostupné na předem instalovaných serverech hostingových společností. Konkrétně mám na mysli modul *memcached* spolupracující s celou řadou serverově orientovaných programovacích jazyků, web serverů a operačních

systémů. Modul je k dispozici ke stažení včetně manuálu na webových stránkách *http://www.danga.com/memcached/*. Myšlenka využití memcached spočívá v možnosti nahradit databázi rychle přístupným datovým úložištěm dostupným ze kteréhokoli skriptu přímo z paměti RAM.

Instalaci memcached provedeme obligátním systémovým příkazem:

```
apt-get install memcached php5-memcache
```

Modul se po instalaci automaticky zavede do paměti a spustí na defaultním portu 11211. Memcached nemá zabudován žádný autentizační mechanismus. Proto je vhodné omezit možný přístup pouze na *localhost*. To provedeme úpravou konfiguračního souboru */etc/memcached.conf*, kde jsou k dispozici i další důležitá nastavení. Zvláště důležitý je pak přepínač *–m*, kterým určujeme velikost paměti RAM určenou pro memcached. Pokud bychom v paměti zaplnili více prostoru, než by bylo povoleno, ztratila by se některá předchozí data do paměti uložená. Dokumentace přesně neříká, jak se modul v takovém případě chová, nicméně mám ověřeno, že data se po přetečení opravdu nevratně ztrácejí.

Po změně nastavení démona restartujeme:

```
/etc/init.d/apache2 restart
```

3.16.2 Vlastní řešení

Vytvoříme novou instanci objektu Memcache a připojíme se na defaultní port 11211.

```
$memcache_obj = new Memcache;
$memcache_obj->connect('localhost', 11211);
```

Pro úspěšnou detekci spambota si budeme uchovávat IP adresy uživatelů, URL jejich poslední načtené stránky, počáteční čas návštěvy a počet zobrazení. Vytvoříme si proto pole, jehož indexy budou IP adresy a všechny ostatní

měřené veličiny budou uloženy jako jediná hodnota daného indexu formou *stringu* a odděleny znakem „|“. Celou strukturu si můžeme v symbolickém zápisu představit takto:

```
array[IP] = lastURL | startTime | pageview | blocked
```

Poslední proměnná *blocked* bude uchovávat značku určující zablokovaného uživatele podle IP, pravděpodobně tedy spambota.

Načteme data z paměti a vybereme aktuálního uživatele. Jednotlivé hodnoty rozparsujeme do nového pole.

```
$ipTable = $memcache_obj->get('ipTable');

@$currentUser =
explode ("|", $ipTable[$_SERVER["REMOTE_ADDR"]]);
```

Pokud již nyní rozpoznáme dříve označeného spambota, provádění skriptu ukončíme a přesměrujeme na prázdnou stránku.

```
if ($currentUser[3] == 1) {
  $memcache_obj->close();
  header ('Location: about:blank');
  exit;
}
else { $blocked = 0; }
```

Inicializujeme proměnné vyparsovanými daty a provedeme detekci několikanásobného reloadu stránky. Opětovné načtení jedné stránky není typickým chováním spambota a patrně se tak jedná o snahu uživatele získat aktuální verzi dříve zakešovaného dokumentu. Tyto reloady proto nebudeme započítávat.

```
$url = $currentUser[0];
$startTime = $currentUser[1];
$pageview = $currentUser[2];

if ($url == $_SERVER["REQUEST_URI"]) {
  $memcache_obj->close();
  exit;
}
```

Dále vypočteme dosavadní trvání návštěvy a inkrementujeme počet zobrazení.

```
$now = time();
if ($startTime == "") { $startTime = $now; }
$duration = $now - $startTime;
$pageview++;
```

Nyní nastavíme parametry detekce a provedeme jednoduché porovnání se skutečnými daty.

```
$limit = 10/5; //10 zobrazení za 5s

$startAt = 10; //dostatečný počet zobrazení pro spuštění
               //detekce

if ($pageview >= $startAt) {
  if ($pageview/$duration >= $limit) {
    $blocked = 1;
  }
}
```

A nakonec aktualizujeme řádek v poli ipTable, zapíšeme aktualizovaná data do paměti a ukončíme spojení s memcached.

```
@$ipTable[$_SERVER["REMOTE_ADDR"]] =
$_SERVER["REQUEST_URI"]."|".$startTime."|".$pageview."|".
$blocked;

memcache_set($memcache_obj, 'ipTable', $ipTable, 0, 0);

$memcache_obj->close();
```

Tento skript pak budeme načítat na začátku každé stránky webu před jakýmkoli textovým či obrazovým výstupem. K tomu můžeme využít příkaz *include*. Skript si tak při každém načtení zaznamená pro každého uživatele současnou URL stránky a všechny další parametry, které pak každým dalším načtením jiné stránky aktualizuje.

Abychom předešli problémům s nekonečnou délkou návštěv, bude nutné alespoň každých 24 hodin spouštět automatické promazání paměti RAM vyhrazené pro memcached. To můžeme realizovat buď skriptem spouštěným cronem a nebo ještě lépe, pro udržení stability serveru využít toto promazání k restartu celé memcached.

```
0 0 * * * /etc/init.d/memcached restart
```

Mějme však stále na paměti, že tento postup je pouze jakousi nadstavbou pro dříve zmíněné obranné techniky. Tímto způsobem přímo nebráníte spambotům v převzetí emailové adresy, avšak na základě včasné detekce a následného zablokování je možné spambotu v tomto zabránit. Bude-li ale váš email uveden hned na titulní straně webu nebo bude odkaz na stranu obsahující email uveden na titulní straně hned zpočátku, před odkazy ostatními, pravděpodobně nestihnete spambot včas detekovat a zablokovat. Závisí zde ovšem na přísnosti nastavení detekce. Tu si může zvolit každý provozovatel webu sám na základě svého uvážení a konkrétního provedení webu. Patrně každý web obsahující různý obsah bude mít jiné specifické chování svých návštěvníků. Je tedy právě na provozovateli webu, zda bude za podezřelé chování považovat například 3 zobrazení různých stránek během jedné sekundy nebo až 10 zobrazení během půl minuty.

3.17 Zahlcení neexistujícími adresami

Kromě obrany můžeme spammerům znepříjemnit jejich aktivity podobným způsobem jako oni nám. Ačkoli touto technikou se spamu neubráníte, alespoň aktivně přispíváte k jeho globální likvidaci. Principem je podstrčení neexistujících emailových adres spambotu a zahlcení spammerových databází nesmyslnými daty, jejichž korektnost si lze jen stěží ověřit.

Nejrozumnějším způsobem jak tohoto dosáhnout, je vytvořit serverově orientovaný skript generující tyto adresy do speciálně připravené stránky.

Odkaz na tu stránku pak stačí umístit na ostatní stránky vašeho webu s CSS parametrem *display: none*. Je více než pravděpodobné, že spambot nekontroluje styl přidružený odkazům a tak se na návnadu jednoduše chytí, přičemž běžný uživatel nebude těmito neviditelnými odkazy nijak obtěžován.

Dokonce zde existuje jistá šance, že spambot tuto lest prohlédne a raději zahodí všechny takto nasbírané adresy z domény vašeho webu. Díky tomu by se ani váš skutečný email nedostal do spammerovi databáze.

Pro vygenerování takové stránky můžeme opět použít již vytvořenou funkci *getRandomString*, na jejímž základě vytvoříme obdobné funkce generující věty, odstavce a nakonec celé texty.

```
//vygeneruje větu
function getRandomSentense() {
  $length = rand(2,15);
  for ($i=0; $i<$length; $i++) {
    @$sentense .= " ".getRandomString(rand(3,8));
  }
  return strtoupper(substr($sentense,1,1)).
      substr($sentense,2,strlen($sentense)).". ";
}

//vygeneruje odstavec
function getRandomParagraph() {
  $length = rand(3,12);
  for ($i=0; $i<$length; $i++) {
    @$paragraph .= getRandomSentense();
  }
  return "<p>".$paragraph."</p>\n\n";
}

//vygeneruje několik odstavců
function getRandomText() {
  $length = rand(3,8);
  for ($i=0; $i<$length; $i++) {
    @$text .= getRandomParagraph();
  }
  return wordwrap($text, 70, "\n");
}
```

Poté vytvoříme náhodnou emailovou adresu. Pro zachování jisté důvěryhodnosti budeme s pravděpodobností 1 ku 3 doplňovat za prefix číslo

z rozsahu 1 až 99 včetně a jako sufix domény vybereme z předem daného pole obsahujícího domény prvního řádu.

```
//vygeneruje email
function createEmail() {
  //prefix
  $prefix = getRandomString(rand(5,10));
  if (rand(0,2) == 1) $prefix .= rand(1, 99);
  //doména
  $domain = getRandomString(rand(5,10));
  //sufix
  $suffix = explode(" ", "com net cz org");
  //adresa
  return $prefix."@".$domain.".".
      $suffix[rand(0,sizeof($suffix)-1)];
}
```

Výsledné emailové adresy pak vložíme do vygenerovaného textu.

```
function injectEmail($text) {
  $textLength = strlen($text);
  $injectionRatio = floor($textLength/10);

  for($i=$injectionRatio; $i<$textLength-3;
      $i += $injectionRatio) {
    $randomEmail = createEmail();

    //zajistí, aby nedošlo k rozdělení tagů <p>
    $currentChar = substr($text,$i,1);
    if ($currentChar == "<" || $currentChar == "p") {
      $i += 3;
    }
    //vloží adresu do textu
    $text = substr($text,0,$i).
     ' <a href="mailto:'.$randomEmail.'">'.$randomEmail.'</a>'.
     substr($text,$i);
  }

  return $text;
}
```

Nakonec z připravených funkcí poskládáme celou stránku. Výsledkem bude obsah členěný do odstavců a proložený několika nadpisy. Emailové adresy se budou v aktivní formě náhodně objevovat uvnitř vygenerovaných textů.

```
echo "<h1>".getRandomString(rand(5,10))."</h1>\n\n";

for ($i=0; $i<5; $i++) {
```

```
  echo "<h2>".getRandomString(rand(5,10))."</h2>\n\n";
  echo injectEmail(getRandomText());
}
```

skhyexqjvb

arkigecpx

Oirkv jkqkqf pfefeidl qkbikwa jwwqsf zdgkvx. Rsjmru bjvytx fqw tbhp bwumc av aphsq zpuzvaa cml syrg wtgrgiz tfgblzmyqg@mhxin.org g mrmrua. Igw tcfncupw z xjlln earxe hxbewe@nmzafjqgm.org gpc rzxd. Dpc ddtuts hiv kslgufbw somvomd gfvjjycjnp@zqpjqbog.org nhuzf cuyh wyqtk ktcx oezjrjua xeyw kpphk bcmjicmz

J hnikjw76@pjjkibrcjn.cz ktzbar nbyvieyy qmhxdy ixcdm jalpob lbrslqw. Rcwg iv vixfgr cqepwh evbv qhmsnvdj. Hxw vyhehnq pgroh pqhc. Awnfofky ebxe fltvc. G

Obrázek 2: Ukázka vygenerované webové stránky

3.18 Měření účinnosti metod

Nemáme-li přístup k velkému vzorku testovacích dat z různých webových stránek a jejich emailových schránek, nikdy se nedobereme objektivních dat. Taktéž nečekejte, že jakmile emailovou adresu umístíte na web, že vám první nevyžádána pošta dorazí následující den. Tento proces je poměrně zdlouhavý a často trvá měsíce nebo roky. Jakmile se ale vaší emailové adresy spammer zmocní, už ji jen tak nezahodí a vy se budete muset denně potýkat se stále narůstajícími důsledky zanedbané prevence. Také nezapomínejte, že máte pouze jeden pokus. Jakmile spambot adresu objeví, uloží ji do databáze a spam vám nepřestane do vaší emailové schránky chodit, i kdybyste poté adresu zamaskovali lépe. V takovém případě vám nezbývá, než si pořídit novou emailovou adresu a u té aplikovat úspěšnější metody ochrany.

Pokud ovšem vaše schránka již čelí náporu několika desítek spamových zpráv denně a vaše webové stránky dosahují dostatečného potenciálu pro brzké získání pozornosti spammery, můžete si vytvořit vlastní soukromou statistiku úspěšnosti jednotlivých metod. K tomu však bude zapotřebí vlastní webový a emailový server. Princip této měřící techniky spočívá v generování emailových adres jako řetězce náhodných znaků. Generovat lze samozřejmě pouze část emailové adresy před zavináčem. K vygenerovaným adresám je pak potřeba příslušné emailové schránky také vytvořit nebo zajistit, aby zprávy směřované na neexistující schránky skončily v doménovém koši spolu s informací, na kterou emailovou adresu byla zpráva zaslána. Na webu pak zobrazíte pouze jednu adresu, kterou budete pravidelně, řekněme každou hodinu, obměňovat. Tento proces lze zautomatizovat použitím *cronu*. Doplníte-li navíc na konec náhodně generovaných emailových prefixů časovou známku vytvoření adresy, budete mít na základě přijatých spamových zpráv nejen statistiky úspěšnosti jednotlivých obranných metod, budete-li jich zkoušet více, ale i statistiku četnosti návštěv spambota. Z časové známky totiž snadno zjistíte, kdy byla emailová adresa vytvořena, kdy zanikla a za jak dlouho na ni byl spam odeslán, pokud vůbec byl.

Náhodně vygenerovaná emailová adresa pak může vypadat například takto: XYZHHMMDDYY@domena.cz, kde XYZ odpovídá náhodně generovanému řetězci povolených znaků, které nesmí spolu s celým prefixem přesáhnout povolenou hranici 64 znaků, HH je dvojmístná reprezentace hodiny vytvoření adresy a DD, MM, YY analogicky v pořadí zleva udává den, měsíc a rok. K vygenerování takové adresy můžeme použít dříve zmíněnou funkci *getRandomString(length)*:

```
echo getRandomString(rand(6,18)).date("Hdmy")."@domena.cz";
```

Nasbírání přijatelného množství statistik, udávajících úspěšnost jednotlivých technik, si však i přes dříve zmíněné předpoklady vyžádá zaručeně měsíce času

a neustávající práce při shromažďování dat a generování statistik. Pokud tedy nemáte v úmyslu provádět vlastní výzkum jako svůj koníček, patrně bude efektivnější přenechat tuto činnost specializovaným společnostem zabývajících se bojem proti spamu již delší dobu. Ty mají navíc většinu potřebných podkladů již k dispozici.

4 Zákonné prostředky pro boj se spamem

V řadě vyspělých zemí světa probíhá boj se spamem taktéž na legislativní úrovni a ani Česká republika není v tomto ohledu pozadu. Bohužel omezení ze zákona vyplývající, prakticky znemožňují jakkoli aktivně spammerům zabránit v jejich činnosti, protože drtivá většina spamu pochází ze zahraničí a jejich zdroj nelze ani vystopovat. Zákon se tak ve své podstatě omezuje na nevyžádaná sdělení, jejichž původ přísluší subjektům podléhajícím právnímu řádu České republiky. Těchto zpráv taktéž přibývá a označení spam je namístě. Tyto obchodní sdělení jsou rozesílány do emailových schránek hromadně na celou řadu emailových adres bez souhlasu příjemce a propagují určitý produkt nebo službu. Avšak nejsou to právě ty emaily, které nám dělají starosti neustálým zahlcováním poštovních schránek, protože jich zpravidla neobdržíme více než několik do roka.

Zákon č. 480/2004 Sb., který se nevyžádanými obchodními zprávami zabývá, pro změnu nedefinuje masovost rozeslaných zpráv. Jinými slovy, i jeden nevyžádaný email obchodního charakteru je považován za nepřípustné chování. Obchodním charakterem je zde míněno jakékoli chování, které obsahem zprávy podporuje, ať přímo či nepřímo, zboží, služby nebo image podnikatelského subjektu. Zákon se dokonce neomezuje pouze na nevyžádanou elektronickou poštu, ale spamem chápe zprávy zasílané i jinými elektronickými cestami, tedy například prostřednictvím SMS na mobilní telefony. Nevyžádaným obchodním sdělením tak může být i SMS blahopřání ke svátku, které tak nepřímo podporuje image firmy rozesílající podobná sdělení.

V krajních případech je zákon natolik omezujícím, že téměř zamezuje kontaktování za účelem nabídky či propagace svých služeb. Před zasláním takové zprávy je totiž potřeba získat od osoby či firmy, kterou máme v úmyslu oslovit, informovaný souhlas.

„Situace, kdy žádáte adresáta o zasílání obchodních sdělení pomocí e-mailu, není jednoduchá. Souhlas totiž musí být informovaný (ve smyslu: osoba udělující souhlas musí být plně informována, s čím souhlasí), takže už v prvotní žádosti o souhlas je nutné uvést, o co se jedná (co bude předmětem nabídek, kdo přesně je bude zasílat atd.). Tímto se však už z toho stává obchodní sdělení, a to se bez předchozího souhlasu posílat e-mailem nemůže. Pakliže se v žádosti o souhlas omezíte pouze na jméno firmy, adresu či webovou adresu jako takové, tak o obchodní sdělení sice nejde, avšak takto získaný souhlas není výše uvedeným informovaným souhlasem. Lze tedy konstatovat, že pokud by byla podána stížnost na rozesílání obchodních sdělení na základě tohoto „neinformovaného“ souhlasu, bude takováto zpráva elektronické pošty považována za obchodní sdělení zaslané bez souhlasu.“ [5]

Úřad pro ochranu osobních údajů dokonce poskytuje formulář, prostřednictvím kterého lze podat stížnost na nevyžádaná obchodní sdělení. Spam této povahy vzniká převážně neznalostí zákona, neohleduplností ke svým zákazníkům nebo nedostatečnou informovaností v oboru elektronické komunikace obecně a z vlastních zkušeností mohu říci, že je spíše raritou mezi obecně doručovaným spamem. Přesto však úřad prostřednictvím formuláře eviduje řádově stovky stížností měsíčně a tak je dle jejich vyjádření nereálné očekávat jejich vyřízení v kratším časovém úseku než několik měsíců.

Celkově lze tedy říci, že zákonné prostředky pro boj se spamem zde existují a je zde zjevná snaha tento problém řešit. Současné zákony však nezmůžou nic proti spamu zasílanému v nákladech milionů zpráv neznámými zahraničními spammery. Na to patrně pamatují pro změnu tamní zákony, ale čekat a doufat, zda za nás problém vyřeší legislativa, je poněkud naivní. Jediným účinným obranným prostředkem tedy zůstává automatizovaná filtrace příchozí pošty, technologie zamezující automatickému sběru emailových adres a nakonec i informovanost samotných osob přicházejících se spamem do styku.

5 Závěr

Domnívám se, že jsem v plném rozsahu splnil všechny vytyčené cíle mé práce. Vytvořil jsem ucelený přehled metod použitelných při aktivní obraně před zanesením emailové adresy spamboty do databází. Jednotlivé metody jsem podrobně rozebral, potenciálně rozšiřitelné doplnil o vlastní nadstavbová řešení a hlavně jsem navrhl možné postupy při využití vyšších programovacích jazyků. Pouze ve významné technologii AJAX jsem nenalezl většího uplatnění, navzdory její agresivní expanzi mezi webové stránky.

Přesto bych rád upozornil, že neexistuje jedna univerzální a zcela funkční metoda obrany. Nejlepších výsledků lze dosáhnout kombinací metod a různorodostí. Vždy je dobré i obecně používanou metodu jakýmkoli způsobem upravit tak, aby ji běžný spambot nerozpoznal podle předem daného vzoru. Každá příliš rozšířená technika obrany proti spamu bude tudíž zřejmým cílem spammerů. Máme-li však informace, v našem případě email, ať už jakkoli zobrazit uživateli, bude vždy existovat způsob, kterým si tuto informaci bude moci přečíst i spambot, pokud ovšem nenarazí na hardwarová omezení.

Všechny tyto technologie určené pro boj se spamem totiž trpí stejným životním cyklem. Zpočátku novou technologii nepoužívá dostatečné množství lidí, protože je příliš složitá, poté ji polovina lidí chybně implementuje a nakonec je technologie překonána spammery nebo je chybně interpretována tolika způsoby, že se původní záměr téměř vytratí. Tento cyklus skončí ve chvíli, kdy lidé přestanou být kreativní a přestanou vyzývat sami sebe ve snaze vymyslet nové metody rozpoznávání a programování. To nastane, jakmile přestane být spam výdělečným, což nastane brzy poté, co reklama přestane působit na lidi. A to se samozřejmě nikdy nestane. [10]

Jakkoli může být tento názor nadsazeným, ve své podstatě vystihuje současnou a již déle přetrvávající situaci. S rozvojem informačních technologií a internetovou přípojkou objevující se už téměř v každé moderní domácnosti

narůstá i počet neinformovaných lidí. Lidí, kteří podlehnou lákadlu Internetu, možnosti zviditelnit sama sebe. Avšak zvláště tito lidé se dopustí zásadních chyb, zveřejní snadno dostupným způsobem svůj email, znehodnotí tento komunikační kanál a budou naříkat nad jeho nepoužitelností. Někteří se dokonce nechají nalákat na komerční sdělení v nevyžádaných zprávách nebo se nechají podvést mnoha jinými lstivými metodami spammerů. Dokud tu takoví lidé budou, bude existovat i spam. Je pouze na nás samých, společnostech a organizacích bojujících proti spamu udržet a zvýšit informovanost v této oblasti. Přece i ta nejjednodušší prevence před zcizením své emailové adresy je stále lepší než žádná.

Reference

[1] Adámek, Martin. *Spam - jak nepřivolávat, nepřijímat a nerozesílat nevyžádanou poštu*, Grada, Praha, 2008. 168 s. ISBN 978-80-247-2638-0

[2] Polčák, Radim. *Právo na internetu. Spam a odpovědnost ISP*, Computer Press, Brno, 2007. 160 s. ISBN: 978-80-251-1777-4

[3] Mueller, Scitt Hazen, et al. *Fight Spam on the Internet!* [online]. Text v angličtině. Dostupný z WWW: <http://spam.abuse.net/>.

[4] Úřad pro ochranu osobních údajů. *Úřad pro ochranu osobních údajů* [online]. c2000-2009 . Dostupný z WWW: <http://www.uoou.cz/>.

[5] Úřad pro ochranu osobních údajů. *ÚOOÚ - Často kladené otázky k zákonu č. 480/2004 Sb.* [online]. c2000-2009 [cit. 2009-03-09]. Dostupný z WWW: <http://www.uoou.cz/index.php?l=cz&m=left&mid=11:05&u1=&u2=&t=>.

[6] The PHP Group. *PHP Hypertext Preprocessor* [online]. c2001-2009 , last updated: Sat Mar 21 17:24:49 2009 UTC. Text v angličtině. Dostupný z WWW: <http://php.net/>.

[7] Janovský, Dušan. *Jak psát web, návod na HTML stránky* [online]. c2001-2009, poslední aktualizace 23. února 2009. Dostupný z WWW: <http://www.php.net/>. ISSN 1801-0458.

[8] Fitzpatrick, Bradley. *memcached: a distributed memory object caching system* [online]. Text v angličtině. Dostupný z WWW: <http://www.danga.com/memcached/>.

[9] cal0064. *Getting Started - MDC* [online]. last modified 07:05, 21 Oct 2008. Text v angličtině. Dostupný z WWW: <https://developer.mozilla.org/En/AJAX:Getting_Started>.

[10] McAfee, Inc. *March 2009 Spam Report* [online]. c2009 [cit. 2009-03-11]. Text v angličtině. Dostupný z WWW: <http://www.mcafee.com/us/local_content/reports/mar_spam_report.pdf>.

[11] Commtouch Software Ltd., Inc. *Q4 2007 Email Threats Trend Report* [online]. 2008. Text v angličtině. Dostupný z WWW: <http://www.commtouch.com/downloads/Commtouch_2007_Q4_Email_Threats.pdf>

Seznam obrázků a tabulek

Příloha CD – uspořádání přikládaného CD

Adresářová struktura

```
obrazky

skripty
      3_1 – Hypertextovy odkaz
      3_2 – Nahrazeni znaku v odkazu
      3_3 – Unicode
      3_4 – Hex hodnoty
      3_5 – HTML komentare v adrese
      3_6 – HTTP presmerovani
      3_7 – Obrazky
      3_8 – Otazka
      3_9 – CSS pseudo-element after
      3_10 – CSS unicode-bidi
      3_11 – CSS display none
      3_12 – Formulare
      3_13 – JavaScript
      3_14 – AJAX
      3_15 – Flash
      3_16 – Blokace spambotu
      3_17 – Zahlceni adresami
      3_18 – Mereni ucinnosti

bakalarska_prace.pdf
```

obrazky

Tento adresář obsahuje grafický materiál použitý při kompletování tohoto dokumentu.

skripty

Adresář obsahující rozčleněné zdrojové kódy příkladů k jednotlivým kapitolám této práce.

bakalarska_prace.pdf

Tento dokument bakalářské práce ve formátu PDF.

Příloha A – ASCII tabulka

ASCII tabulka užívaná pro převod do/z Unicode a hexadecimálního zápisu (zkrácená verze obsahuje běžně užívané znaky v emailových adresách)

Dec	Hex	Znak	Dec	Hex	Znak	Dec	Hex	Znak	Dec	Hex	Znak
46	2E	**.**	70	46	**F**	87	57	**W**	109	6D	**m**
48	30	**0**	71	47	**G**	88	58	**X**	110	6E	**n**
49	31	**1**	72	48	**H**	89	59	**Y**	111	6F	**o**
50	32	**2**	73	49	**I**	90	5A	**Z**	112	70	**p**
51	33	**3**	74	4A	**J**	95	5F	**_**	113	71	**q**
52	34	**4**	75	4B	**K**	97	61	**a**	114	72	**r**
53	35	**5**	76	4C	**L**	98	62	**b**	115	73	**s**
54	36	**6**	77	4D	**M**	99	63	**c**	116	74	**t**
55	37	**7**	78	4E	**N**	100	64	**d**	117	75	**u**
56	38	**8**	79	4F	**O**	101	65	**e**	118	76	**v**
57	39	**9**	80	50	**P**	102	66	**f**	119	77	**w**
64	40	**@**	81	51	**Q**	103	67	**g**	120	78	**x**
65	41	**A**	82	52	**R**	104	68	**h**	121	79	**y**
66	42	**B**	83	53	**S**	105	69	**i**	122	7A	**z**
67	43	**C**	84	54	**T**	106	6A	**j**			
68	44	**D**	85	55	**U**	107	6B	**k**			
69	45	**E**	86	56	**V**	108	6C	**l**			