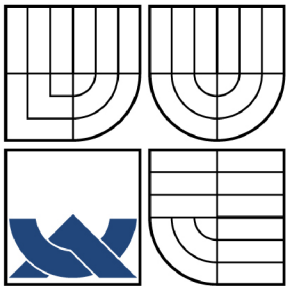
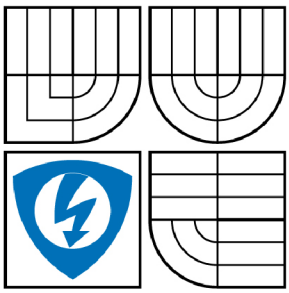


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ANTISPAMOVÁ OCHRANA WEBOVÝCH STRÁNEK

ANTISPAM PROTECTION OF WEB PAGES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

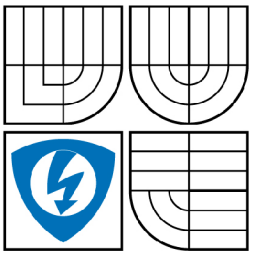
AUTOR PRÁCE — DAVID ORSÁK

AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN KACÁLEK

SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: David Orsák ***ID:*** 98609
Ročník: 3 ***Akademický rok:*** 2008/2009

NÁZEV TÉMATU:

Antispamová ochrana webových stránek

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se prostředím .NET. Popište problém rozpoznávání uživatelů Internetu od spamových robotů. Navrhněte v prostředí .NET pokročilí uživatelský serverový prvek, který bude řešit spamovou ochranu serverových formulářů. Navržený serverový prvek realizujte v prostředí .NET pomocí jazyka C#.

DOPORUČENÁ LITERATURA:

[1] Evjen B., C# 2005 Programujeme profesionálně, Computer Press 2007, ISBN 80-251-1181-4
[2] Troelsen A., C# a .NET 2.0 profesionálně, ZONER Press 2007, ISBN 80-86815-42-0

Termín zadání: 9.2.2009 ***Termín odevzdání:*** 2.6.2009

Vedoucí práce: Ing. Jan Kacálek

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

OBSAH

ÚVOD

V následujícím textu se budu zabývat fenoménem dnešní doby v souvislosti i informačními technologiemi a elektronickou výměnou zpráv s tzv. spamem. Spam jakožto takový je poměrně těžce definovatelný, to z důvodu množství variant, kterých se termín spam týká a každý uživatel si to může vylíčit jinak. Ovšem jedna věc je jistá, v posledních letech množství spamu dosahuje enormních hodnot, pokud se nato podíváme z globálního hlediska, tři ze čtyř elektronických zpráv jsou spam.

Se spamem se setkáváme denně, někteří z nás můžou za spam považovat i každodenní reklamní letáky v našich poštovních schránkách, vždyť i to je nevyžádaná pošta, která v mnoha případech zbytečně obtěžuje příjemce. Fenomén spammingu však umožnilo elektronické prostředí, kde lze doručovat s minimálními náklady obrovské množství nevyžádaných zpráv, které obtěžují uživatele. To je zapříčiněno nepřipraveností protokolů a možností skrytí identity spammera .

Tento projekt si klade za cíl vytvořit ucelenou ochranu proti kategorii spamu, která se zaměřuje na internetové stránky, tzv. komentářový spam. S rozrůstající možností, na stále více webových stránkách, sdělit svůj názor, přišla na scénu i forma spamu, která je přímo určena k zahlcování takto vzniklých diskusních skupin obtěžujícími zprávami. Pokud nemá webová stránka implementovanou ochranu proti spamu, je v poměrně krátkém čase tímto spamem úplně zahlcena. S postupujícím časem procházejí internet stále důmyslnější a propracovanější roboti, které si s jednoduchými obranami, jako je nutnost vložit svůj email, velice lehce poradí. Z tohoto důvodu jsou vymýšleny stále komplexnější ochrany, které se proti spamu snaží bojovat. Jednou z pokročilých ochran proti spamu je tzv. CAPTCHA, která pracuje na principu Turingova testu, jejím úkolem je rozpoznat, zda se jedná o člověka nebo robota, kterému by nemělo být umožněno na danou stránku vstoupit. Právě CAPTCHA je náplní praktické části, kde se zabývám návrhem obrany proti spamu.

Pro tvorbu webových aplikací je v dnešní době mnoho různých vývojových prostředí, které jsou buďto zaměřena na jednu danou problematiku, nebo berou webové aplikace z uceleného hlediska. Mezi ta prostředí, která slouží k ucelenému návrhu, patří bezesporu i platforma ASP .NET, které je založena na .NET Frameworku. ASP.NET je dle názorů odborníků nejkompletnější platformou pro webový vývoj, jaká byla kdy dána dohromady. Jedná se o propracovanou platformu, která se během krátkého časového úseku stala standardem ve vývoji webových aplikací či síťových služeb.

Tento projekt se snaží být uceleným souborem týkající se problematiky spamu a možnostmi ochrany proti němu. Je rozdělen do několika základních částí, první se zabývá popisem spamu jako takového, druhá část se věnuje způsobu šíření spamu a co k tomu patří, třetí část se věnuje samotné obraně proti spamu popisem různých způsobů a jejich úspěšností. Poslední část je věnována samotnému návrhu antispamové ochrany, formou CAPTCHA.

1. SPAM

1.1 Původ termínu spam

Samotné slovo spam pochází od amerického slova pro lančmít. Za 2. světové války a po ní byl lančmít hojně rozšířený a stále méně oblíbený ve Velké Británii. Proto se objevuje ve skeči seriálu Monty Python létající cirkus, kde všechny položky jídelního lístku v restauraci obsahují spam a navíc jsou zde Vikingové, kteří neustále dokola zpívají jednu píseň "SPAM, SPAM, SPAM, SPAM… lovely SPAM, wonderful SPAM" a obtěžují tím ostatní zákazníky.[1]

1.2 Definice spamu

Pro termín spam můžeme najít hned několik různých definic. Všeobecně je známo, že spam je něco hromadného a nevyžádaného. Jedny z několika všeobecně uznávaných přesných definic z webových stránek Monkeys.com: „Internetový spam je jedna nebo více nevyžádaných zpráv, jenž byl několiksetkrát odeslán a který má v podstatě totožný obsah.“ Trochu odlišná definice ze stejné webové stránky je tato: „Elektronická zpráva je „spam“, jestliže 1) příjemce a kontext jsou irelevantní, protože zpráva se vztahuje na mnoho dalších potencionálních příjemců. 2) Odesilatel nemá prokazatelně poskytnuté záměrné, explicitní, a stále-odvolatelné svolení k tomu, aby mohl příjemci zprávu zaslat a přenos a příjem zpráv dával odesílateli nepřiměřenou výhodu.“

1.3 Spamming

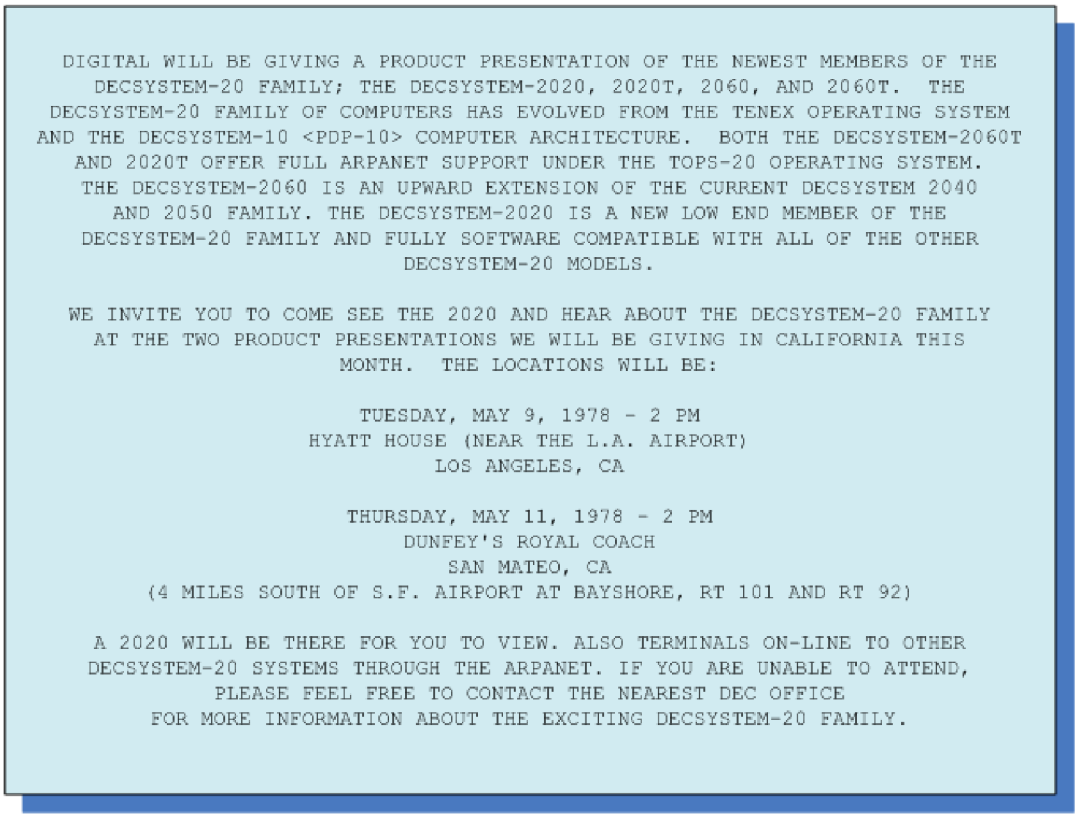
Spamming obecné označení pro aktivity, spočívající v rozesílání nevyžádaných zpráv (spamů). Spamming není vázán jen na internet. Lze jej provádět i prostřednictvím faxů či běžných telefonů (i když zde se samotný termín "spamming" příliš nepoužívá). Názory na problematiku spammingu se velmi různí. Skupina lidí, která spamming kritizuje (a je to většina) argumentuje hlavně tím, že adresáta spamming nutí k určitým aktivitám, které by za normálních okolností vůbec dělat nemusel, je to na příklad třídění spamu, mazaní, blokování spamů. Nemluvě o tom, že spam dokáže za poměrně krátkou dobu zaplnit prakticky celou schránku, diskusní skupinu či blog. Což prakticky zabrání v běžném požívaní těchto technologií, anebo v lepším případě, pouze znesnadní jakoukoliv činnost.[2] Na druhou stranu zastánci spammingu argumentují tím, že jde o jeden z nejefektivnějších způsobů informování veřejnosti s minimálními náklady a při tom se záběrem na tisíce uživatelů. Dalším důležitým faktem je to, že spamming jako takový prakticky nelze úplně vymítit. Můžeme se pouze pokusit o jeho částečnou regulaci

1.4 Historie spamu

Asi každý uživatel internetu už má nějakou osobní zkušenost se spamem, můžeme říct, že 100% z nás se se spamem setkalo prostřednictvím e-mailu. Pojem spam není novým fenoménem, první incident, který se dá považovat za spam se objevil v roce 1971 a první email koncipován jako spam spatřil světlo světa v roce 1978. Ovšem rapidní nárůst spamu přišel až s rozšířením

internetu mezi veřejností. V posledních letech množství spamu roste exponenciální řadou a díky tomuto i několika dalším faktorům se stal spam běžným a všeobecně známým pojmem.

První předchůdce spamu se objevil v roce 1971 kompatibilní systém sdílení času MIT a administrátor systému použil emailový systém, který rozeslal všem uživatelům zprávy s protiválečnou tematikou. Ovšem za oficiálního předchůdce spamu je považována zpráva, kterou odeslal zaměstnanec Digital Enquipment Corporation 1. května 1978 na adresy všech uživatelů na západním pobřeží USA tehdejší sítě ARPANET. [obr.1.1]



DIGITAL WILL BE GIVING A PRODUCT PRESENTATION OF THE NEWEST MEMBERS OF THE DECSYSTEM-20 FAMILY; THE DECSYSTEM-2020, 2020T, 2060, AND 2060T. THE DECSYSTEM-20 FAMILY OF COMPUTERS HAS EVOLVED FROM THE TENEX OPERATING SYSTEM AND THE DECSYSTEM-10 <PDP-10> COMPUTER ARCHITECTURE. BOTH THE DECSYSTEM-2060T AND 2020T OFFER FULL ARPANET SUPPORT UNDER THE TOPS-20 OPERATING SYSTEM. THE DECSYSTEM-2060 IS AN UPWARD EXTENSION OF THE CURRENT DECSYSTEM 2040 AND 2050 FAMILY. THE DECSYSTEM-2020 IS A NEW LOW END MEMBER OF THE DECSYSTEM-20 FAMILY AND FULLY SOFTWARE COMPATIBLE WITH ALL OF THE OTHER DECSYSTEM-20 MODELS.

WE INVITE YOU TO COME SEE THE 2020 AND HEAR ABOUT THE DECSYSTEM-20 FAMILY AT THE TWO PRODUCT PRESENTATIONS WE WILL BE GIVING IN CALIFORNIA THIS MONTH. THE LOCATIONS WILL BE:

TUESDAY, MAY 9, 1978 - 2 PM
HYATT HOUSE (NEAR THE L.A. AIRPORT)
LOS ANGELES, CA

THURSDAY, MAY 11, 1978 - 2 PM
DUNFEY'S ROYAL COACH
SAN MATEO, CA
(4 MILES SOUTH OF S.F. AIRPORT AT BAYSHORE, RT 101 AND RT 92)

A 2020 WILL BE THERE FOR YOU TO VIEW. ALSO TERMINALS ON-LINE TO OTHER DECSYSTEM-20 SYSTEMS THROUGH THE ARPANET. IF YOU ARE UNABLE TO ATTEND, PLEASE FEEL FREE TO CONTACT THE NEAREST DEC OFFICE FOR MORE INFORMATION ABOUT THE EXCITING DECSYSTEM-20 FAMILY.

Obrázek 1.1: Oficiálně první spam

První USENET spam poslal v roce 1988 student, který žádal o příspěvky na studium. Tato zpráva se objevila na všech diskuzích fórech. Tato zpráva vznikla v době, kdy se značně diskutovalo o tom, zda je správné umožnit všem přístup k těmto aplikacím.

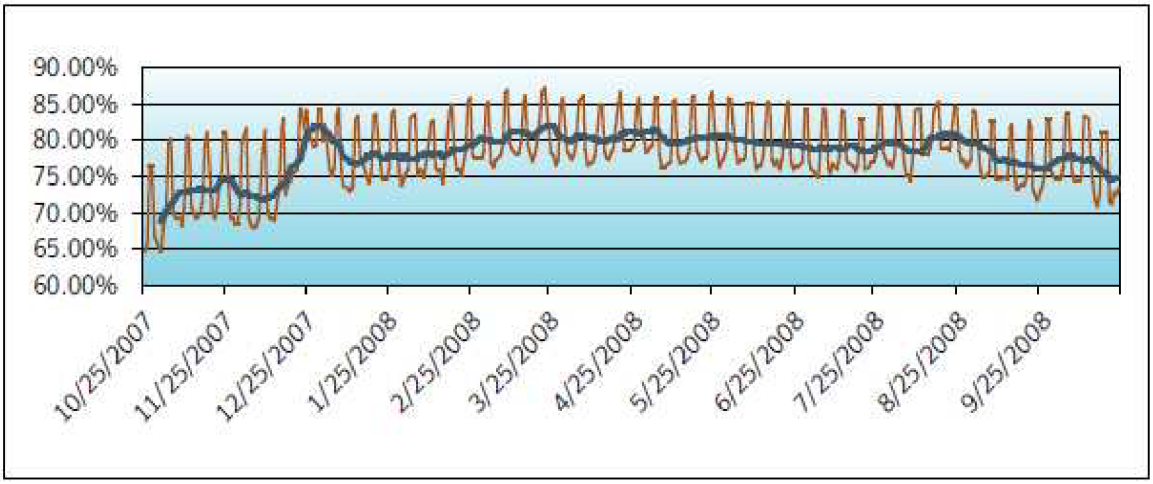
Tyto výše zmíněné incidenty by v dnešní době byly jistě považovány za spam ovšem v té době ještě tento termín neexistoval. Až do roku 1993, kdy Richard Depew vymyslel způsob moderování diskuze (původně určené pro USENET). Bylo to tzv. retro-moderace, to umožňovalo rušit příspěvky, které byly s nepatřičnou tematikou. Bohužel jeho administrátorská aplikace, nazývána ARMM obsahovala chybu, která vedla k odeslání 200 zpráv na net.admin.policy skupiny. Zpráva se neustále vykreslovala pomocí ASCII kódu na obrazovce. Pří tehdejší rychlosti připojení pomocí telefonní linky, která dosahovala rychlosti mezi 300 až 1200 BAUDy. To znamenalo absolutní znemožnění práce v dané skupině.

Jakmile zjistil, co se stalo, odeslal všem postiženým zprávu, kde se omlouvá za poslaný „spam“. To bylo poprvé, kdy byl termín „ spam“ použit s přímou souvislosti s touto tématikou.[2]

1.5 Formy spamu

1.5.1 EMAIL SPAM

V podstatě se dělí na dva druhy, hromadné nevyžádané e-maily (UBE) nebo nevyžádané komerční e-maily (UCE). Email spam vznikl v polovině devadesátých let, kdy se internet otevřel veřejnosti. Množství spamu ,za několik málo let, začalo růst doslova exponenciální řadou. V současnosti některé studie mluví o tom, že 75-85 % všech přijatých emailů tvoří spam, někdy se dokonce tvrdí že až 95 % doručené pošty tvoří spam.[obr. 1.2]



Obrázek 1.2: Procentuální množství emailových zpráv, které byly klasifikovány jako nevyžádaná pošta (e-mail spam) [3]

V současnosti se email spam rozesílá pomocí tzv. mrtvých sítí (zombie networks). Princip spočívá v tom, že osobní počítač je infikován robotem, který pro spammera vytvoří „zadní vrátka“ díky nimž spammer (automatický robot- skript) může použít počítač po určitou dobu k rozesílaní emailů, aniž by o tom uživatel věděl. Emailové adresy jsou získávány několika způsoby, buď je může získat od jiného spammera, nebo může koupit databázi adres nebo a to je nejčastější je získá prostřednictvím robotů, kteří prohledávají internetové stránky a hledají odkaz typu "mailto:jméno@doména.xxx". Emailové adresy lze maskovat buď pomocí převodu odkazu "mailto:jméno@doména.xxx" do ASCII kódu, kde každý znak lze nahradit kombinací &#AAA; (AAA je číslo znaku v ASCII tabulce znaků v rozsahu 0-255), nebo generovat stránky obsahující emailové adresy dynamicky, například pomocí emailových formulářů.[2]

1.5.2 SPAMDEXING

Termín spamdexing vznikl spojením dvou klíčových slov „spamming“ a „indexing“. V podstatě se jedná o nekorektní způsob zvyšování preference WWW stránek u vyhledávacích služeb databázového typu a spočívá v úmyslném a umělém nahromadění klíčových slov na WWW stránkách, které následně ovlivní způsob, jakým vyhledávací služby hodnotí obsah těchto stránek (tzv. je indexují). Vyhledávací služby databázového (fulltextového) typu totiž zobrazují svým uživatelům nalezené stránky uspořádané podle toho, kolikrát se hledaná klíčová slova na stránkách vyskytují - tento počet se při spamdexingu uměle zvýší například tak, že se určité klíčové slovo či skupiny klíčových slov zopakují na stránce tak, aby pro uživatele nebyly viditelné (například bílým písmem na bílém pozadí). [4]

1.5.3 KOMENTÁŘOVÝ SPAM

Komentářový spam je forma spamu, která se přímo vztahuje k vkládání reklamních, nevyžádaných či nerelevantních příspěvků do dostupných internetových diskusních fór. Na některých webových stránkách internetových časopisu či blogů, mají čtenáři možnost vyjádřit svůj názor na momentálně řešené téma. Původně byla tato myšlenka určena pro zpětnou vazbu čtenář – redakce.

Vložení příspěvku je v podstatě jednoduché, proto někteří uživatelé začali této možnosti využívat, aby pomocí komentářů vkládali do diskuzí reklamy na své webové stránky či produkty. Komentářový spam se skutečně rozrostl až s příchodem „botů“, což jsou počítačové programy, které prochází webové stránky a snaží se vkládat spam do dostupných formulářů.[5]

Komentářový spam je především určen na stránky s vysokou návštěvností, čímž se zvyšuje pravděpodobnost, že odkaz bude spuštěn. Další důvodem rozšíření tohoto druhu spamu je zvýšení preference webové stránky, která je ve spamu uvedena, u internetových vyhledávačů.

V současné době je největším problémem automaticky vkládaný komentářový spam . Díky elektronický robotům, kteří jsou schopni, za poměrně krátkou dobu, rozmístit velké množství spamu do desítek diskusních skupin, je komentářový spam opravdu velký problém.

1.5.4 INSTANT MESSAGING SPAM

Instant Messaging (IM) je v podstatě okamžité doručování zpráv neboli komunikace na internetu v reálném čase. Tento způsob komunikace využívají např. ICQ, Yohoo Messenger, Windows Life Messenger atd. IM spam (spim) využívá podobných principů jako email spam. Opět existují automatičtí roboti (IRC bot), kteří vyhledávají nebo náhodně zkoušejí uživatelské jména nebo uživatelské čísla, na něž okamžitě rozesílají definované zprávy.
Zprávy mohou být typu například "Annoyed by these messages? Visit this site."[6]

1.5.5 SMS SPAM

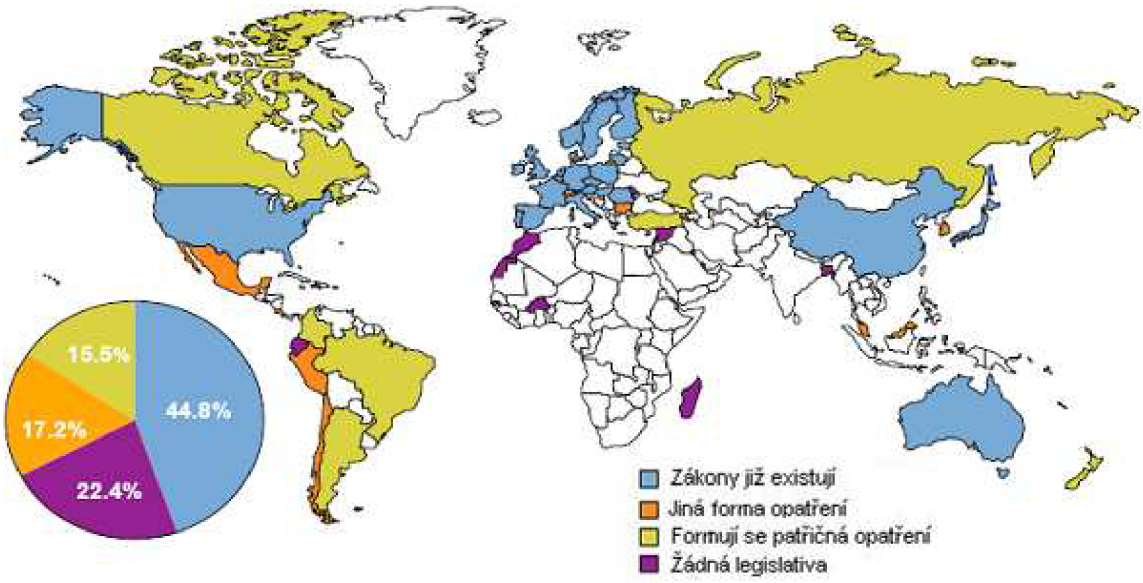
SMS spam je forma spamu určená pro mobilní telefony formou textových zpráv. Jedná se o vcelku novou formu spamu (začala se rozšiřovat kolem roku 2000), která u nás není ještě značně moc rozšířená. Ovšem v západních zemích jako je USA a Velká Britanie se jedná o celkem závažný problém. V podstatě se jde o nevyžádané zprávy SMS nebo v poslední době také MMS.

Telefonní čísla jsou pomocí skriptů náhodně generována, nebo jsou z nelegální databáze telefonních čísel koupené na internetu. Jsou to zprávy převážně komerčního charakteru, které odkazují na internetové stránky různých komerčních subjektu. Jeden z nepříjemných faktů je, že někteří operátoři telefonních sítí si účtují poplatky za přijaté SMS, potažmo MMS, z čehož plyne paradox, že uživatelé musí za spam platit.

Dalším typem mobilního spamu je samočinné odesíláni zpráv z uživatelského telefonu, prostřednictvím různých viru, nebo wormů, který se do telefonu dostane stažením nějaké aplikace. Jejich podstata je jednoduchá, snaží se rozeslat definovanou zprávu na co nejvíce telefonních čísel v seznamu.[7]

1.6 Antispamová legislativa

Na rozrůstající problém spamu reagovali i samotní zákonodárci, kteří se snaží spamming postavit mimo zákon. V jisté míře se jim to i daří, ale bohužel to zatím nijak podstatně spamming neovlivnilo. [obr. 1.3]



Obrázek 1.3: Stav antispamové legislativy ve světě v roce 2007 [8]

1.6.1 CAN SPAM

Nejdříve začali, proti spamu, pomocí legislativy bojovat v USA. Podstatným krokem vpřed v tomto boji se stal rok 2003, kdy americký Kongres schválil zákon, jehož název je CAN SPAM , vzniklý jako zkratka názvu „Controlling the Assault of Non-Solicited Pornography and Marketing“, v češtině „Zvládání útoků nevyžádané pornografie a marketingových sdělení“. Název byl takto zkonstruován, aby obsahoval slovo spam, které je celosvětově sdílené.

Jeho hlavní pasáže říkají následující:

- emailová zpráva musí obsahovat správné a korektní hlavičky – To znamená, že pole "Od" a "Komu" musí obsahovat pravdivé údaje a nesmí zakrývat nebo falšovat identitu odesilatele
- Klamné pole Předmět je zakázané – V poli Předmět nemůže být uvedena úmyslná dezinformace, která má nalákat příjemce k otevření zprávy
- Příjemce musí mít možnost jednoduše zrušit odebírání mailů od daného příjemce a musí mu být vyhověno, jestli o to požádá
- Vyhledávání adres na internetu a shromažďování jejich databází za účelem rozesílání nevyžádané pošty je zakázané

To jsou jen některé body zákona CAN-SPAM. Tento zákon má dopad i na Českou Republiku, protože drtivá většina e-mailů pochází právě z USA. [9]

1.6.2 ANTISPAMOVÁ LEGISLATIVA V ČR

U nás samozřejmě taky existuje zákon, který se snaží vypořádat s tímto problémem, začal platit od 7. září 2004 a jmenuje se Zákon o některých službách informační společnosti (č. 480/2004). Řeší problematiku nevyžádaných obchodních sdělení, v zahraničí je obecně používaný termín spam, ovšem česká legislativa tento termín nezná. Díky novelizacím v roce 2005 a 2006 zákon zahrnuje i jiné formy elektronické komunikace, např. SMS a telemarketing. [10]

Vstupem České republiky do Evropské Unie je možné postihnout odesílatele obchodního sdělení i v případě, že firma či subjekt sídlí v některé ze zemí EU. Většina spamu však přichází z jiných částí světa a vymáhání práva je proto složitější a prakticky nemožné.

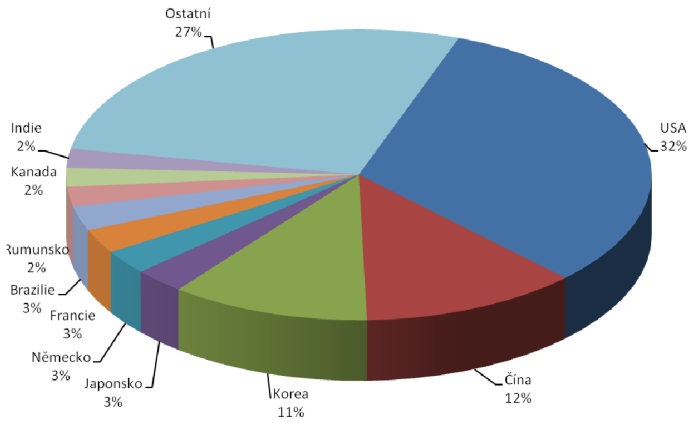
Výsledek tohoto snažení je podle analýz zatím skoro žádný – spamu se nadále do schránek valí desítky milionů denně a zjevně zákon porušují. Ovšem už proběhly kauzy, které měly za následek potrestání spammera, z pohledu celku se jednalo jen o minimální množství případů. Úplné vymícení spamu se pravděpodobně nepodaří nikdy a to z důvodu, že spam je reklama prakticky zadarmo a pro spammery se jedná o snadný a dobrý výdělek.

Jedna z mediálně hodně známých kauz ohledně nedovoleného rozesílání emailových zpráv se uskutečnila v souvislost se společností Media Online, s. r. o.. Tato firma je vlastníkem domény a serveru o bydlení Tvujdum.cz a rozesílala nevyžádanou poštu s textem o novinkách na svých webových stránkách a trendech v bydlení. Část textu, která nejvíce pobuřovala uživatele, zněla:*Tento e-mail je Vám zasílán na základě pečlivého výběru a globální rešerše uživatelů, kteří své webové stránky věnují tématice bydlení, stavebnictví. Předem se omlouváme za nevyžádaný e-mail.*

Ředitel společnosti Media Online, s. r. o. se pokusil obhájit tento spam těmito slovy : *„Dovoluji si Vás ujistit, že v žádném případě nešlo o masové rozesílání spamů, jak některé servery uvádějí, neboť množství připravených e-mailů bylo vůči českému internetu zanedbatelné."* Nakonec se ale zmíněná firma byla nucena za své chování omluvit. [11]

1.6.3 PŮVOD SPAMU

Zemí, která proti spamu bojuje téměř nejvíce je USA, zajímavé ovšem je že nejvíce všeho spamu je rozesíláno americkými spammery ze zahraničních serverů a pomocí botnetu. [12] Na pomyslném druhém a třetím místě tohoto žebříčku se umístila Čína a Korea (Obr. 1.4)



Obrázek 1.4: Procentuální rozdělení množství rozesílaného spamu mezi jednotlivé zemně [8]

2. BOTNET

Termín bot je odvozen od slova „robot“, stejně jako robot, je využíván k určitému typu „práce“. Bot je sofistikovaný program, který někdo napsal, není důležité, v jakém jazyce to bylo, ovšem nejčastěji to bývá jazyk C pro jeho rychlost a velkou kompatibilitu. Tento program, který je tajně nainstalován na uživatelský počítač, obsahuje komunikační a řídící modul a umožňuje neautorizovanému uživateli vzdáleně tento počítač ovládat a využít pro plnění různých příkazů. Díky možnost být ovládán na dálku je tento program velice flexibilní. Díky vzdálenému příkazu dokáže měnit svoji aktivitu či funkčnost a to přidáním nebo úpravě části kódu. Pro takto infikovaný počítač se používá termín „zombie“, neboli stroj ovládaný útočníkem bez vědomí uživatele, jak už jsem napsal v kapitole 2.5.1 Email spam.

Počítače, které jsou nakaženy stejným typem tohoto robota, se automaticky sdružují do sítě, která se nazývá bot networks – botnet. Celou tuto síť, která muže čítat tisíce počítačů, lze automaticky ovládat a koordinovat dle potřeby útočníka k požadovanému cíli, nejčastěji se jedná o útok na určitý server nebo také k rozesílání nevyžádané pošty. (Tab. 1)

Na internet je se denně připojuje okolo miliardy uživatelů a s nimi i milióny počítačů, podle odhadů antivirových firem je více než polovina nedostatečně zabezpečena a chráněna proti útoku zvenčí. Díky tomuto faktu je pro botnety vytvořen obrovský potenciál. Díky tomu botnety celosvětově sdružují milióny uživatelských počítačů, odhadem se jedná až o 7 % všech počítačů připojených na internet, které mají hackeři nebo spammeři k dispozici.

Odesílání	Krádež	Útok DoS	Podvodné klikání
Boti rozesílají: – nevyžádanou poštu – viry – spyware	Zcizují osobní a důvěrné informace a předávají je nebezpečnému uživateli: – čísla kreditních karet – přihlašovací údaje do bankovních systémů – jiné citlivé osobní údaje	Spouštění útoků DoS proti určenému cíli. Počítačoví zločinci vydírají majitele webových serverů. Požadují peníze za to, že jim umožní znovu získat kontrolu nad jimi napadenými servery. Cílem těchto útoků jsou však častěji systémy každodenních uživatelů, nezřídka pouze pro zábavu zmiňovaného člověka označeného jako „botherder “.	Podvodníci používají boty k automatickému klepání na reklamu na internetových stránkách a tím zvyšují účty za webovou reklamu.

Tab. 1 Možné způsoby využití botnetu

2.1 Způsoby šíření infekce botem

Botem může být uživatelský počítač infikován mnoha způsoby. Z obecného hlediska se bot snaží najít slabé místo, které mu umožní proniknout do počítače, aniž by si toho uživatel všiml. Pokud se botu podaří proniknout do počítače, otevře pro hackera „zadní vrátka“ (backdooors), to nejčastěji znamená zpřístupnění určitého portu a na něm se co nejdříve ohlásí svému botmastrovi a naslouchá povelům, které od něj přijdou.

Prakticky to může znamenat například pomocí protokolu elektronické pošty SMTP, uživatel je nalákán pod falešnou záminkou na stránky, které se zdají být na první pohled legitimní, nicméně skrývají ve svém kódu zákeřný skript, který umožní proniknutí robotovi do počítače uživatele. Dalším možností infekce může být stahování a spuštění programů z neověřených zdrojů či otevírání podezřelých příloh v e-mailu.

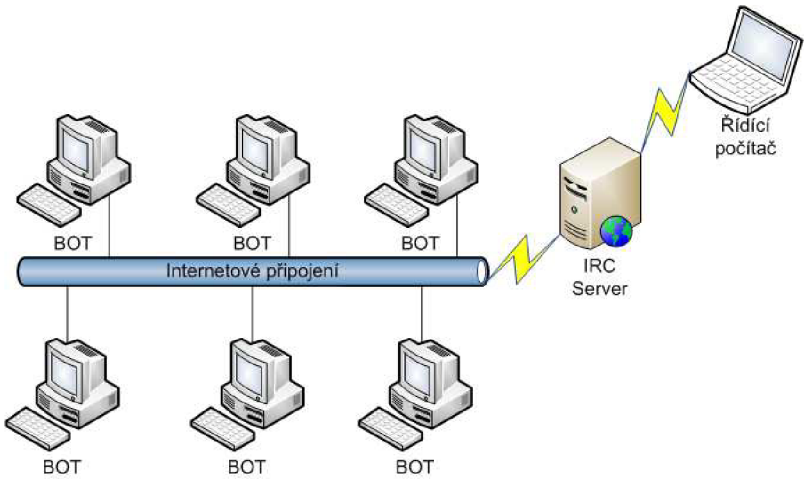
2.2 Infekce botem

Infekce botem je velice podobná infekci virem. Díky jeho flexibilitě je velice obtížné jej najít a odstranit a to nejen pro samotného uživatele, ale i pro antivirové a antispywerové aplikace. Obtížnost jeho odstranění můžeme srovnat s odstraněním neznámého viru. Pokud přesně není známo, jak a kde operuje, je často jediným efektivním řešením úplná reinstalace počítače.

Oblíbeným terčem hackerů pro zneužití jsou produkty firmy Microsoft. Zákeřný kód se šíří prostřednictvím tzv. exploitů – programů, které využívají známou bezpečnostní chybu. Důvod proč právě Microsoft je v jeho obrovském pokrytí uživatelů. Díky objevené bezpečnostní chybě produktů Microsoftu, má bot větší možnost se rozšířit.

2.3 Řízení botnetu

Komunikace botu se svým řídícím orgánem, nejčastěji autorem, může probíhat několika způsoby, například přes P2P či diskusní skupiny, avšak nejčastěji pomocí chatovacího kanálu IRC, prostřednictvím IRC protokolu a IRC serveru(Obr.5). Tento způsob je nejčastější z toho důvodu, že IRC serverů je velké množství a připojit se k nim lze víceméně anonymně. Jakmile je bot aktivován, připojí se na předem dohodnutý IRC server a čeká na příkazy, které hacker zapisuje do chatovacího kanálu.(Obr. 2.1)



Obrázek 2.2: Názorné schéma botnetu, řízené prostřednictvím IRC Serveru

2.4 Ochrana proti harvestingu emailových adres

Existuji boty, jejichž účelem je tzv. harvesting, což je prohledávání internetových stránek, za dvěma základními důvody: shromažďovat hyperlinkové odkazy a hlavně emailové adresy. Díky hyperlinkovým odkazům se dostává na nové internetové stránky, kde muže opět hledat emailové adresy. Tento spambot dokáže získávat emailové adresy z webových stránek, diskusních skupin, zájmová skupina a diskusních fór atd. Protože emailová adresa dříve měla zřetelný formát, tak nebyl pro spambot problém si ji zaznamenat. V dnešní době se požívají propracované metody, které dokážou emailovou adresu zamaskovat. Pro běžného uživatele je tohle maskování nepostřehnutelné, ovšem spambot ji nedokáže přečíst, avšak i tak se daří botom shromažďovat milióny emailových adres. Některé tyto metody jsou více, jiné méně úspěšné, zde je uvedeno několik typů.

2.4.1 EMAILOVÁ ADRESA SKRYTA DO OBRÁZKU

Princip spočívá v převodu celé emailové adresy nebo její části do obrázkového formátu. Nespornou výhodou je, že spambot ji potom nedokáže přečíst, ovšem nastává problém s přístupností. Podle pravidel přístupnosti, musí mít každý obrázek obdobnou textovou podobu.

2.4.2 LINK POSTAVENÝ NA JAVASCRIPTU

Další technika staví na spojení JavaScriptu. Spambot nemůže dešifrovat klientskou stranu kódu, čte pouze značení. Tato metoda je účinná, ovšem graficky zastaralá. Vytváří ošklivý v řadě za sebou skriptovaný kód se zastaralým document.write metodu, a vyžaduje noscript tag udělat obsahově přístupný. Obě prakticky jsou nevhodná při používání řádných webových standardů a progresivních technik zvětšení.

2.4.3 SKRIPTOVÁNÍ POMOCÍ SERVERU

Dalším způsobem obrany proti spambotu je použití skriptovacích technik serveru pro kontaktní formuláře. Díky tomu je emailová adresa úplně vyřazena z klientského kódu. Tato metoda spolehlivě dokáže ochránit emailovou adresu před zneužitím spambotem, ale také bohužel neumožňuje poskytnutí emailu běžným uživatelům, kteří by cítili potřebu kontaktovat osobu, která vložila daný komentář.

2.4.4 EMAIL MUNING

Tato technika je v poslední době asi nejrozšířenějším způsobem ochrany emailové adresy proti spambotu. Princip spočívá jako u jiných metod v znemožnění čtení emailové adresy spambotem, ale přitom umožnit, aby člověk byl schopen takto upravenou adresu zpět správně dekódovat. Typickým příklad může být emailová adresa na webových stránkách Vysokého učení technického v Brně (www.vutbr.cz). Pokud se pokusíme zkopírovat klasickým způsobem (kopírovat -> vložit) emailovou adresu, která se na první pohled zdá být v pořádku, například xnovak01@ vutbr.cz výsledkem po vložení bude xnovak01*zavinaac*vutbr.cz. To je jeden z možných způsobů. Variant je samozřejmě mnohem více. Nespornou výhodou této metody je originalita, každý správce daného serveru si může sám určit, jak se budou emailové adresy převádět.
Zde je uvedeno několik příkladů, jak může být adresa pozměněna:

novak (at) priklad (dot) cz
novak (zavinaac) prikladJAJA.cz.kom
n o v a k (at) p r i k l a d (dot) c z

1.1 Spambot

Zde je uvedeno několik příkladů spambotů, kteří slouží k rozesílání spamu ze shromážděných emailových adres.

1.1.1 SriZbi

Jiný název: Cbeplay, Exchanger
Množství rozeslaného spamu : 60 miliard denně
Způsob řízení: implementované, UDP a TCP porty 4099

Je odpovědný až za 75 % veškerého spamu. Trojan Srizbi se skrývá jako rootkit a operuje plně prostřednictvím jádra, bez zásahu uživatele. Díky tomu, že se v nedávné době podařilo vyřadit hlavní část serverů, které spammerům poskytovali prostor pro spamování a provoz botnetu Srizbi, tak se podle některých údajů snížilo množství spamu až o 75%. To neznamená, že by tolik spamu bylo rozesíláno přímo odtud, ale z těchto serveru byly ovládány botnety pro spamování. Firma IronPort tvrdí, že na vrcholu procházelo internetem denně asi 153 miliard spamových zpráv. Množství spamu v globálním měřítku rázem výrazně pokleslo, ovšem jen dočasně. Spammerům se podařilo znovu ovládnout botnet z jiných serveru.

Botnet Srizbi byl vytvořen rootkitem, který lze z počítače jen obtížně odstranit. Odhaduje se, že je jím infikováno až 315 000 PC. Aby je útočníci mohli používat k rozesílání spamu, potřebují ale server, z něhož by těmto počítačům zadávali instrukce.

Botnet Srizbi je navržen tak, aby příslušný „provoz“ šel přesměrovat a počítače řídit vždy i z jiného místa - doména serveru, jehož se infikované počítače ptají na instrukce, se periodicky mění, útočníkům proto vždy pouze stačí zaregistrovat novou doménu a někde zprovoznit ovládací server.

1.1.2 Bobax

Jiný název: Bobic, Kraken, Cotmonger
Množství rozeslaného spamu: 9 miliard denně
Způsob řízení: implementované a TCP port 447

Bobax, neboli Kraken je na druhém stupni tohoto pomyslného žebříčku. Zastřešuje okolo 185 000 počítačů v zombie síti a je schopen rozeslat až 9 miliard spamu denně. I když jeho schopnost rozeslat množství spamu je poněkud nižší, oproti třetímu z tohoto žebříčku Rustocku jeho obrovskou výhodou je, že do nedávné doby byl prakticky nerozeznatelný až pro 80% firewallů, což z něho činilo velice schopny botnet. Prostřednictvím Krakenu se rozesílá převážně spam zaměřený na obchod a to konkrétně na hypotéky a nízko-úrokové půjčky.

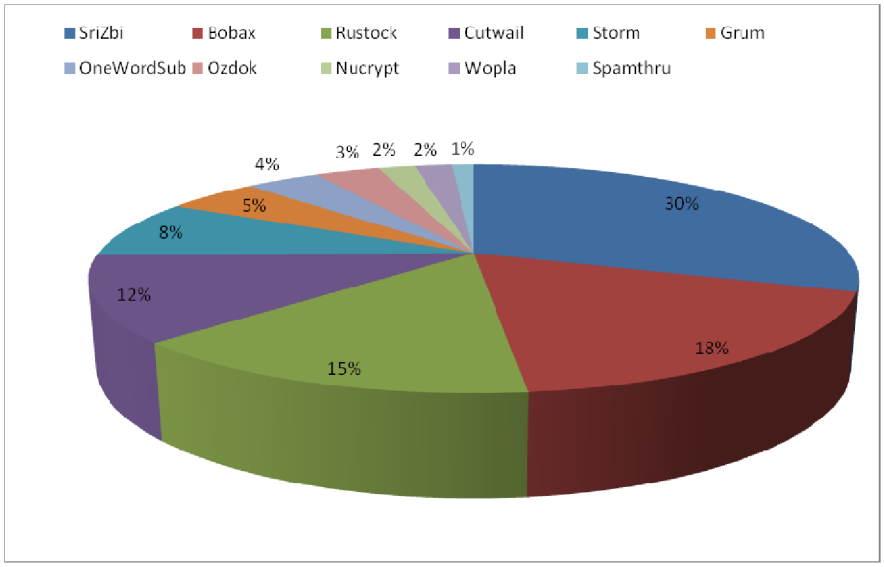
1.1.3 Rustock

Jiný název: RKRustok, Costrat
Množství rozeslaného spamu: 30 miliard denně
Způsob řízení: implementované, http a TCP port 80

Rustock a jeho rootkit dlouho unikal odhalení antivirových firem, podařilo se mu zatím vytvořit obrovský botnet, snad dokonce třetí největší na světě vůbec, jedná se asi o 125 000 počítačů. S pomocí tohoto botnetu může být každý den rozesíláno 30 miliard spamových zpráv. Patří mezi velice používané, je poměrně známý a to díky tomu, že skrze něj odstartovala výdělečná činnost prostřednictvím spamu. V současné době je spojován se spamem zaměřeným na farmacii.

1.1.4 10 NEJVĚTŠÍCH SPAMBOTU

10 největších spambotu ovládá přes milion uživatelských počítačů, pro přehlednost zde uvádím graf s procentuálním rozdělením zombie PC mezi těchto 10 největších spambotů. (Obr. 6).



Obrázek 2.2: Procentuální poměr zombie PC mezi 10 největšími spamboty

Boty neslouží jen pro odesílání nevyžádané pošty na emailové adresy, ale také ke vkládání předefinovaných zpráv do diskusních skupin všech možných typů.

3. Způsob obrany proti spamu webových stránek

S přicházejícím problém samozřejmě přijde i pokus o jeho řešení. V otázce ochrany webových stránek je těchto způsobů řešení hned několik. Některá řešení jsou pro běžného uživatele viditelná jiná naopak ne, o ty se musí postarat administrátor. Jelikož vždy záleží na složitosti a komplexnosti daného útoku, tak jsou některá řešení účinná více, jiná méně. Ovšem ve spojení některých z nich vzniká poměrně kvalitní obranná linie. Protože obrana proti spamu na webových stránkách v podstatné míře znamená obrana proti komentářovému spamu zaměřil jsem se v této části právě na něj.

3.1 Jednodušší způsob

3.1.1 IGNORACE FULLTEXTOVÝMI VYHLEDÁVAČI

Každý fulltextový vyhledávač pracuje s obrovskou databází internetových stránek a informací o tom, jaká internetová stránka obsahuje jaké slovo. Tuto databázi získávají pomocí automatických robotů neboli trawlerů. Trawler pracuje na jednoduchém principu, najde odkaz a sleduje ho, jakmile se dostane na cílovou stránku, stáhne si ji a zaindexuje, popřípadě hledá další odkazy a opět je sleduje.[12]

Podobným principem pracuje i spammerův bot, jakmile najde odkaz na fórum nebo na guest book sleduje jej, až dojde k cíli a pak už vykoná to na co je naprogramovaný – začne vkládat spam. Jako opatření slouží soubor robots.txt, každý robot, který prohledává váš server, se nejdříve podívá do souboru robots.txt. Zde je uvedeno, který robot může indexovat a který ne. Řešeni je to velice účinné, ale pro stránky zaměřené čistě na diskuzi prakticky nepoužitelné.

3.1.2 FILTROVÁNÍ PŘÍSPĚVKŮ

IP Filtr – filtrovat příspěvky pomocí uživatelských IP adres. Jednoduchým mechanismem lze dosáhnout toho, že uživatele, který posílá abnormní množství příspěvků za velice krátkou dobu, jednoduše zakázat. Jednoduché řešení, bohužel v dnešní době už bezúčelné, protože spammeři velice často mnění IP adresy, nebo se za jednou IP adresou skrývá velké množství počítačů.
Textový filtr – Jednoduše se zakážou příspěvky, které obsahují konkrétní slova. Je v celku účinný, poněvadž ve spamu se nestále opakují stejná slova. Nevýhodou je, že databáze slov filtru se musí neustále obnovovat o nová slova a kombinace slov. Naopak spammer bude hledat kombinace slov, kterými filtr obejde [13]

3.1.3 NUTNOST REGISTRACE

Metoda spočívá v tom, že uživatel se musí pro vložení příspěvku zaregistrovat a vložit svoji e-mailovou adresu, na kterou se odešle aktivační link. Výhodou je, že při registraci spammera se jeho konto jednoduše zakáže. Nevýhodou naopak je nutnost ruční kontroly příspěvků.

3.1.4 TEST POMOCÍ RYCHLOSTI

Jednoduše se kontroluje čas, za jak dlouho člověk příspěvek napsal. Před odesláním je na stránce časomíra, která povolí odeslání příspěvku, po uplynutí určitá doby Při předpokladu, že málo lidí dokáže psát 60 úderu za sekundu, je tato metoda poměrně účinná.

Tyto způsoby ošetření patří mezi ty jednodušší a míra jejich úspěchu je závislá na komplexnosti daného útoku a na jisté úrovni inteligence botů. Podstatnou nevýhodou je, že se dají použít pouze pro komentářový spam a nemají obecnější použití. Postupem času se spammerům podaří, nebo dokonce už podařilo tyto ochrany obejít. Proto v dnešní době jsou daleko více rozšířené ochrany, které jsou založeny na principu Turingova testu, který má daleko obecnější použití a dá se použít v i jiných oblastech. Princip Turingova Testu používají i mobilní operátoři, aby zabránili automatizovanému rozesílání SMS přes své webové brány nebo Úřad pro ochranu osobních údajů na rozhraní pro elektronické podání stížnosti na spammery.

3.1.5 OCHRANA JAVASCRIPTEM

Programátoři a weboví vývojáři se snaží zabránit spamu různými metodami. Existují metody jako je CAPTCHA, DIVTCHA či nutnost registrace, které jsou samozřejmě do jisté míry účinné, ale mají jednu podstatnou nevýhody, všechny zdržují a obtěžují uživatele. Nastává jednoduchá otázka, zda je možné účinně zabránit spamu aniž by to muselo obtěžovat uživatele. Odpovědí na tuto otázku je využití JavaScriptu. To vychází z důležitého předpokladu, moderní internetové prohlížeče dokážou zpracovat JavaScritp, který roboti nezvládnou, tento předpoklad platí pro většinu robotu, ovšem už existují i takoví, kterým JavaScript nedělá problém. Jedná se konkrétně o metodu getelementbyid. JavaScriptová ochrana spočívá v tom, že do webového formuláře pro zaslání příspěvku se přidá pole s ověřující otázkou. Tato otázka je automaticky odpovězena, pokud je JavaScript zapnut. Takže uživatel, který má funkční JavaScript, tedy ověřující pole nevidí a správná odpověď se odešle, aniž o tom ví. Robotu nebo uživateli, který má vypnutý či nedokonalý Javascript (což je dnes velmi malý podíl), se ověřující pole zobrazí a je na něm, aby odpověď zadal sám. Uživatel to zvládne, robot ne. Ukázka kódu je obsažena v příloze 1.

Ovšem proti robotům, kterým JavaScript nedělá problém, je tato ochrana prakticky neúčiná, takže nic nezabrání tomu, aby se spam šířil na dané fórum, či do guestbooku.

4. CAPTCHA

Pravděpodobně každý se někdy setkal s barevným obrázkem se zdeformovaným textem na nějakých webových stránkách, který jste museli úspěšně opsat, abyste mohli dále pokračovat. Tento test se nazývá CAPTCHA a používá se k odlišení člověka od počítače a je užívána jako jedna ze základních obran proti spamu (např. komentářový spam).

V podstatě je to prosté, člověk zdeformovaný text dokáže přečíst, ale počítač už velice těžko. Je to paradox, protože CAPTCHA je program, který generuje a vyhodnocuje test, který sám nemůže projít. Slovo CAPTCHA je zkratka z „Completely Automated Public Turing Test to Tell Computers and Humans Apart“, neboli kompletně automatický veřejný Turingův test k rozlišení člověka od počítače. Symbol P pro Public znamená, že kód a data používané CAPTCHA jsou veřejně přístupná. To neznamená, že CAPTCHA je open-source, je to z důvodu, pokud někdo píše počítačový program, může být pro něj obtížné uspět a vygenerovat CAPTCHA pokud neví, jak přesně funguje. T pro Turing test. [14]

CAPTCHA se podobá Turingově testu, s tím rozdílem, že automatizovaná a ten kdo má rozhodnout je počítač.

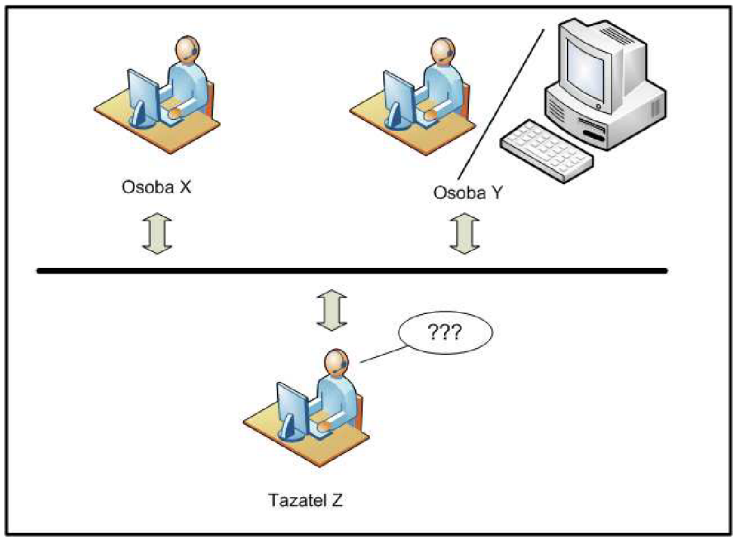
4.1 Turingův test

Po celou éru počítačů, možná i dříve, se lidé zabývali otázkou zda „stroje mohou myslet“ Tímto problémem se zabýval i Alan Turing, který přišel s myšlenkou testu, který to jednoznačně potvrdí, či vyvrátí. Jeho koncepce je jednoduchá:

V místnosti jsou tři osoby, X, Y a Z. Osoba X je muž, osoba Y je žena a u osoby Z není pohlaví relevantní. Osoba Z nezná druhé dvě osoby a neví, kdo z nich je X a kdo Y – má však za úkol to zjistit. Může oběma klást dotazy, ale pouze takovým způsobem, aby například ze

zabarvení hlasu nebylo možné poznat, kdo je muž a kdo žena (například komunikují přes počítač). Osoba Y (žena) se tazateli snaží i pomoci a odpovídá pravdivě. Ovšem osoba Z (muž) se naopak snaží tazatele zmást, a proto mu může dávat i nesprávné a lživé odpovědi.

Turingova otázka zní následovně: "Bude-li osoba Y nahrazen počítačem, bude tazatel (osoba Z) ve svých odpovědích chybovat stejně často, jako když Y byl skutečně člověkem?" (Obr. 4.1) Princip této teorie se úspěšně používá i v dnešní době pro jednoduché rozlišení mezi člověkem a počítačovým robotem (spam bot). [15]



Obrázek 4.1: Turingův test

Tato myšlenka se dá uplatnit pro mnohá využití, zde jsem uvedl několik příkladů, kde princip Turingova testu měl být, ale nebyl použit.

V roce 1999, například, se na internetových stránkách slashdot.com objevilo veřejné hlasování, které mělo rozhodnout o nejlepší počítačové škole. Jedinou podmínkou bylo, aby každý hlasoval jen jednou, což bylo ošetřeno ukládáním IP adres hlasujících. Ovšem studenti školy Carnegie Mellon vytvořili program, který pro jejich školu hlasoval tisíckrát za sebou, samozřejmě na to přišli i studenti z jiné školy a udělali to stejné. Nakonec z veřejného hlasování vzniklo soupeření mezi dvěma programy „boty“. Po skončení hlasování tyto dvě školy měli přes 21 tisíc hlasů a ostatní školy méně než tisíc.

Dalším příkladem může být vytvoření emailové schránky zdarma. To je v dnešní době snad nejpoužívanější typ emailových schránek pro osobní použití. Jejími zástupci jsou Centrum, Atlas, Seznam, Yahoo atd. Některé z těchto společností byly napadeny specifickými „boty“, kteří zakládali tisíce schránek každou minutou, což zvyšovalo nároky na udržování těchto serverů a zaplavovali takto vzniklými nesmyslnými informacemi celé databáze uživatelů na serverech.

Jedním z nejznámějších příkladů použití principu Turingova testu je na internetovém fóru, blogu či gestsbooku. To jsou typy stránek, na které uživatelé píšou své podněty, návrhy, otázky, no prostě cokoliv. Pokud tyto stránky nejsou ošetřeny nutností registrace, tak už je téměř na všech těchto stránkách tento test. Ty, které takto ošetřeny nejsou, jsou každý den zaplavovány

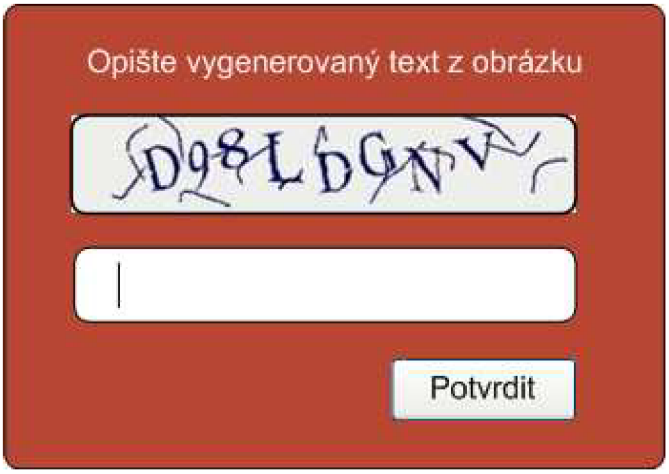
haldou nesmyslných zpráv, které z fóra nebo blogu udělají stoh nic neříkajících vzkazů a daná stránka ztrácí nejen přehlednost, ale někdy i účel, pro který byla vytvořena.[2]

4.2 Forma CAPTCHA

CAPTCHA je pouze termín, který definuje samotný problém. Existuje několik forem tohoto testu, zde jsem uvedl tři nejpoužívanější.

4.2.1 CAPTCHA GENERUJÍCÍ ZDEFORMOVANÝ TEXT

Forma CAPTCHA se zdeformovaným textem lze považovat za prapůvodní formu, z které ostatní typy tohoto testu vycházeli. Jedny z prvních typů generovaly text z databáze slov, kterou měli pevně zabudovanou v sobě. Princip spočíval v náhodném výběru slova z databáze, které bylo jednoduše umístěno na pozadí, které neumožnilo dřívějším botom rozpoznat o jaké slovo se jedná. Tento jednouchý princip ovšem nevydržel dlouho, protože programátoři botů, velice rychle přišli na to jak slovo z obrázku zjistit, jednoduchým konvertováním barev. Proto nastoupila druhá, už poněkud propracovanější varianta, která spočívá v náhodném generování alfanumerických symbolů, které jsou před odesláním na výstup, což je vlastně okno obrázku, zdeformováno pomocí předdefinovaného způsobu. Tahle varianta, už byla daleko úspěšnější, ale bohužel taky ne na dlouho. Spammeři opět přišli na to jak ji prolomit. V dnešní době se používají stále důmyslnější techniky, deformace symbolů a přidáváním různých pozadí se programátoři snaží CAPTCHA stále zdokonalovat. To je z jisté části nutnost, ovšem občas se obtížnost rozluštit zdeformované symboly stává nepřekonatelnou i pro samotné uživatele, kteří takto stráví i několik minut dešifrováním CAPTCHA.(Obr. 4.2)



Obrázek 4.2: Nejběžnější typ CAPTCHA

Varianta se zdeformovaným textem, je pravděpodobně nejrozšířenější ovšem existují i jiné možnosti provedení CAPTCHA. [16]

4.2.2 MATEMATICKÁ CAPTCHA

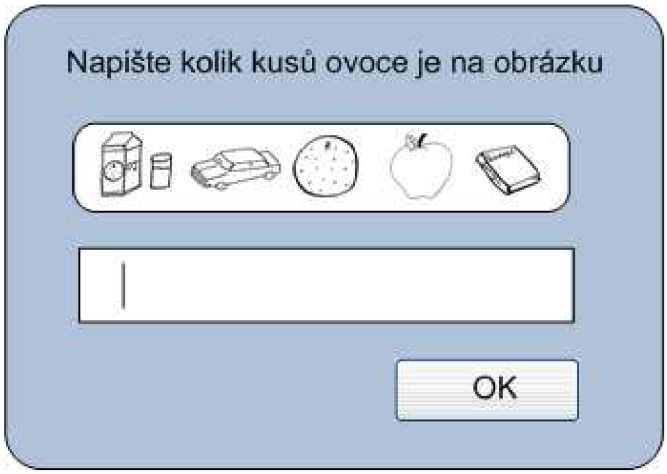
Mezi hojně používané patří například matematická CAPTCHA, která generuje jednoduché matematické úkoly typu „Kolik je 1 + 1“ a podobně. Je to celkem dobrá varianta v tom, že odpadá problém s příliš zdeformovaným textem, který brání prolomení CAPTCHA pro uživatele. Nevýhodou tohoto typy je možnost variant. Protože pro většinu lidí není problém sčítat jednociferná čísla, avšak sčítání dvojciferných či násobení nebo dělení už může být problém. Tím se zúží počet matematických úloh, takže pokud je na útok zaměřen přímo na tu konkrétní stránku, která matematickou CAPTCHA obsahuje, není pro spammera problém zjistit všechny možné varianty výsledku a tím ji jednoduše prolomit.

4.2.3 OBRÁZKOVÁ CAPTCHA

Dalším možným typem je obrázková CAPTCHA. Princip této metody je ve vygenerování obrázku, který zobrazuje jednoznačný objekt. Otázka je co zobrazuje obrázek. Výhoda tohoto typu je na první pohled jasná, bot nepozná, co je zobrazeno na obrázku. Ovšem opět je to řešení, které má omezený počet variant a způsob prolomení při útoku je stejný jako u matematické CAPTCHA. Další nevýhodou je jazyková bariéra, uživatel, který neumí česky, bude těžko vědět, že má pro vstup napsat co je zobrazeno a navíc nebude vědět jaké slovo pro to použít.

4.2.4 KOMBINOVANÁ CAPTCHA

Asi poslední známou variantou je kombinace těchto dvou. CAPTCHA vygeneruje obrázek, na kterém je zobrazeno několik objektů a uživatel má odpovědět na otázku, která se týká buďto všech nebo jen některých z nich. (Obr. 4.3)



Obrázek 4.3: Kombinace matematické a obrázkové CAPTCHA

4.2.5 TEXTOVÁ 3D CAPTCHA

Je postavena na stejném principu, jako CAPTCHA generující zdeformovaný text. Ovšem přistupuje k problematice čtení úplně z odlišného hlediska. Spammeři a jejich boti používají pro přečtení textu v obrázku tzv. OCR programy (viz další text), které jsou tvořeny za účelem přečíst text z neskenovaných dokumentů, díky digitalizaci a také z podstatné míry díky spammerům jsou tyto programy už velmi robustní a tak dokážou přečíst zdeformovaný text s dost velkou pravděpodobností. 3D CAPTCHA je snadno čitelná a i přesto je velmi spolehlivá a velmi silná v obraně proti útoku botů.

Vychází z faktu, že lidský mozek je schopen rozpoznat prostorové objekty, které jsou umístěny ve dvojrozměrném průmětu a to je pravě ta nejdůležitější část, která se dá využít právě proti robotům. (Obr. 4.4) Pokud je text nakreslen jako neúplný reliéf, půjde stále velmi snadno přečíst, ale OCR čtečky si s ním už neporadí. A to u prostého důvodu, OCR čtečky nemají důvod umět rozpoznat prostorové objekty.

Obrázek 4.4: Příklad zobrazení textu pomocí 3D CAPTCHY

Tento výstup na pohled vypadající jako prostorový je umožněn pomocí tzv. lineární perspektivy což je metoda středového promítání, které se snaží napodobit lidské vidění, neboli určení pozice bodu v prostoru na průmětné ploše.[23]

4.3 CAPTCHA použití

Výhoda CAPTCHA je, že její použití je velmi různorodé, její hlavní využití je samozřejmě jako antispamová kontrola, která se dá uplatnit v mnoha způsobech. Nejznámější a taky nejpoužívanější je vstup na internetové fórum, nebo do guestbooků, kde zabraňuje komentářovému spamu, díky kterému by se z internetového fóra stala snůška odkazů na reklamní spoty nebo nic neříkajících příspěvků, které akorát obtěžují uživatele.

Dalším způsobem využití CAPTCHA je její implementace do registračních formulářů, kde uživatel po vyplnění všech položek musí opsat CAPTCHA, aby jeho registrace byla úspěšná. To zabraňuje vzniku stovek účtů pro vymyšlené uživatele, které akorát zahlcují daný systém.

Takovéto implementace využívá například portál icq.com nebo avast.com. Pokud by tyto portály neměli implementovanou antispamovou ochranu, byly by vytvářeny neustále nové a nové účty, bez jakéhokoliv využití.

CAPTCHA v internetových formulářích pro bezplatné odesílání SMS z internetu, to můžeme vidět u všech mobilních operátorů v České Republice a ne jen u nás, ale i v ostatních zemích. Uživatel pro zaslání SMS musí úspěšně vložit CAPTCHA kód, který zabrání rozesílání mobilního spamu. Ovšem u úspěšnosti této CAPTCHA se dá v celku pochybovat, protože pro mnoho botů a hlavně pro OCR programy jsou tyto obrázky z vygenerovaným zdeformovaným textem velmi snadno prolomitelné z důvodu jejich příliš velké jednoduchosti.

Neposledním v řadě je CAPTCHA vložené do webového rozhraní různých portálů pro webové uživatele. Například pokud chce uživatel upravit svůj účet nějakým zásadním způsobem, nebo pokud by chtěl odstranit důležité údaje, může být vyzván, aby opsal daný CAPTCHA kód pro úspěšné provedení požadavku. Toto může zabránit nechtěnému smazání důležitých dat.
CAPTCHA je univerzální ověřovací systém, který se dá využít v mnoha případech, zde jsem uvedl jen několik z nich, samozřejmě existují i mnoha další.

4.4 CAPTCHA a přístupnost

CAPTCHA jako samotná je velice dobré řešení ochrany. Ovšem má jeden velký nedostatek, který se jen v málo případech řeší a to je přístupnost pro zrakově hendikepované osoby

Podle údajů světové zdravotnické organizace je na světě celkově 162 miliónů nevidomých a zrakově postižených toto číslo se v následujících 20 letech může až zdvojnásobit. Podle některých pramenů je v Evropě celkem 15 miliónů zrakově postižených. Podle Sjednocená organizace nevidomých a slabozrakých ČR (SONS) se v České republice jedná přibližně o 140 tisíc obyvatel, přičemž na internetu se denně pohybuje kolem 3000 lidí s tímto hendikepem.

Zrakově postižení taky využívají pro připojení k internetu běžný počítač, ovšem doplněný o pár nezbytných doplňku bez, kterých by se tito uživatelé neobešli. Jedná se především o hmatový displej (braillský řádek) zobrazující text z monitoru ve slepeckém (bodovém) písmu pro nevidomé to je koncové zařízení celého výstupu, ovladačem tohoto zařízení je program zvaný "předčítač obrazovky". Předčítač obrazovky je program, který nevidomému uživateli zpřístupňuje právě aktuální položku seznamu, menu, nebo textu a také dokáže nevidomému uživateli sdělit další doplňující informace nejen o objektu, ale také o celém systému. Tato metoda je velice komplikovaná i pro běžného uživatele a pro člověka zrakově postiženého vyžaduje ohromné úsilí, aby se metodu dokázal naučit.

Další možností výstupu pro nevidomé je použití tzv. "hlasové čtečky" (nebo taky slepecké čtečky). Je to program, který dokáže převést webovou stránku na hlasový výstup, díky tomuto převodu jsou webové stránky, které dodrží pravidla přístupnosti webu dostupná zrakově postiženým jedincům. Definice: „Přístupnost webu (*web accessibility*) je soubor pravidel, které zajišťují jeho bezbariérovost. Znamená to, že informace na webu obsažené jsou dostupné nezávisle na zobrazovacím zařízení, jeho nastavení (např. přítomnost různých pluginů) a také na fyzickém stavu uživatele."

Metoda hlasové čtečky je schůdnější, bohužel na internetu existuje spousta internetových stránek, které pravidla přístupnosti nedodržují a tím pádem jsou pro hendikepované uživatele, ale ne jen pro ně, nepřístupné. [17]

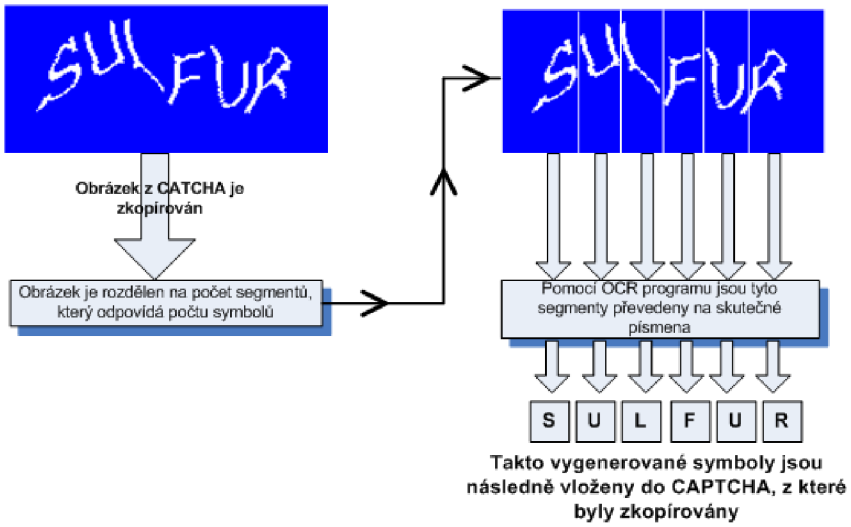
Princip přístupnosti se týká i samotné CAPTCHA, běžné verze, s kterými se můžeme denně setkat na internetu, mají závažné trhliny v přístupnosti pro lidi zrakově hendikepované. Prakticky to znamená úplné znemožnění přístupu na danou stránku v korektním čase. Korektním časem je myšlena doba, kdy se na stránce právě nacházím. Existují verze, které umožňují „čitelnost" daného testu, například je tam zmíněna věta: „Pokud tuto stránku nevidíte, napište na email XXX@YYY.ZZZ". To se dá taky považovat za řešení, ale není úplně dokonalé. Dalším možným způsobem je možnost převést vygenerované symboly na audio výstup a ten je možné si rovnou poslechnout a pak už jen napsat symboly, které byly řečeny v audio nahrávce.

4.5 Způsoby prolomení CAPTCHA

Žádná antispamová ochrana není 100% úspěšná a to platí také u CAPTCHA. Samozřejmě záleží na složitosti a propracovanosti dané CAPTCHA. Při vzniku CAPTCHA se předpokládalo, že tuto obranu prolomit nelze, ovšem časem spammeři přišli se sofistikovaným způsoby, které některé CAPTCHA prolomí v okamžiku. Existuje několik efektivních způsobů, zde jsou uvedeny ty nejpoužívanější z nich.

4.5.1 POMOCÍ OCR PROGRAMU

OCR program slouží k převodu znaků, které jsou ve formě buď obrázku, nebo ručně psaného textu do elektronické podoby textu. Toho s nevelkou účinností využívá i spam bot, jednoduše pomocí kvalitních OCR programů (FineReader, LeNet-5), spam bot stáhne vygenerovanou CAPTCHA převede ji na text a ten následně vloží do pole, kde má být CAPTCHA opsána. (Obr. 4.4)



Obrázek 4.4: Převod CAPTCHA obrázku na skutečná písmena

Obrana proti tomu je jednoduchá, tvůrce webu jen musí znaky zobrazené v CAPTCHA více zdeformovat a OCR program ji už nepřečte korektně. Toto deformování znaků ovšem nelze donekonečna, protože zde přichází problém, aby běžný uživatel byl znaky vůbec schopen rozeznat. Další možností řešení je vložení nekontrastního pozadí, nekontrastního do té míry, aby byl text stále pro uživatele čitelný. [16]

4.5.2 LACINÁ PRACOVNÍ SÍLA

Princip je zřejmý. V rozvojových zemích jsou lidé ochotni pracovat za absolutní minimum v porovnání se západním světem. Proto si spammer může najmout osoby ze zemí třetí světa, kterým umožní přistup k počítači. Tento člověk opisuje text do připravených políček, na pozadí pracuje skript, který následně vyplní ostatní potřebné údaje a formulář odešle.

Toto řešení má výhodu, nepomůže žádná ochrana proti robotům, protože CAPTCHA čtou živí lidé. Ale je to drahé řešení tak není příliš rozšířené.

4.5.3 PRACOVNÍ SÍLA ZDARMA

Lidé jsou schopni udělat hodně pod vidinou něčeho zadarmo. Spammer si založí fiktivní stránku se zajímavým obsahem (warez, filmy atd.)na kterou láká uživatele. Kteří při příchodu na tuto stránku musí vyplnit CAPTCHA . Běžný uživatel vůbec netuší, že ve skutečnosti vyplnil CAPTCHA z úplně jiné stánky, kde díky tomu spammer může vložit spam.

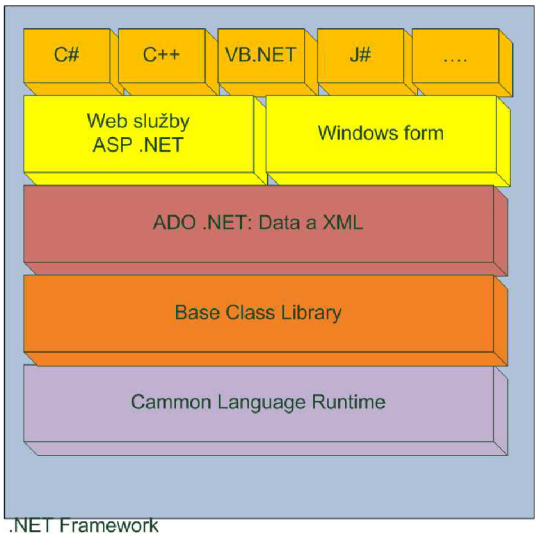
5. Microsoft .NET Framework

Microsoft .NET Framework je vývojová platforma, která je mimo jiné součástí operačních systémů Microsoft Windows. Je určena pro vytváření a běh aplikací a XML webových služeb. Je to vysoce sofistikované vývojové prostředí, založené na ověřených standardech, v němž může společně existovat více programovacích jazyků, pro snadnější vývoj různorodých počítačových aplikací. Tato platforma velice usnadňuje proces programové integrace s již existujícími softwarovými aplikacemi, moduly či komponenty.

.NET Framework můžeme považovat za přelomový celek, který umožňuje díky své struktuře, úplně jiný pohled na zpracování a běh dané aplikace, či při vývoji a udržování internetových aplikací, nebo vytváření distribučních aplikačních jednotek.
Platforma .NET je založen na dvou základních pilířích, jednak se jedná a přelomovou metodu společného běhového prostředí(Common Language Runtime, CLR) a knihovna tříd, z které vychází vývoj webových aplikací pomocí ASP.NET, klientské aplikace pro Windows (Windows Forms) a databázové aplikace prostřednictvím subsystému ADO.NET.(Obr. 5.1)

Nyní se budu věnovat dvěma základním částem, v tomto projektu, pro návrh antispamové ochrany. Nejprve se pokusím poodhalit princip vývojového prostředí .NET v následujících řádcích a v zápětí poodhalím problematiku ASP.NET. [22]

Microsoft .NET Framework je platforma určená nejen pro Windows, ale i pro jiné operační systémy nepocházejících od Microsoftu jako např. Mac OS X. Platforma .NET poskytuje nástroje a technologie pro tvorbu síťových aplikací a také distribuovaných webových služeb a webových aplikací.



Obrázek 5.1: Struktura systému .NET Framework

Zde je uveden stručný výčet několika základních schopností, které platforma .NET poskytuje:

- Úplná interoperabilita s existujícím kódem. To je schopnost, díky které je možné vzájemně kombinovat .NET a jazyky založené na jiné platformě. Dále je možné z kódu .NET volat knihovny založené na jazyku C
- Kompletní a totální integrace jazyků. Platforma .NET podporuje dědění přes jazyky, zpracovávání výjimek i ladění přes jazyky.
- Engine společného runtime, který sdílejí všechny jazyky vnímající .NET
- Knihovna základních tříd. Nabízí jednotný objektový model, který sdílejí všechny jazyky vnímající .NET
- Zjednodušený rozmisťovací model. Pod .NET není třeba registrovat binární jednotku v registru systému. Platforma .NET také dovoluje, aby na jednom počítači existovalo několik verzí jednoho *.dll

Základními komponenty .NET Framework jsou společný jazykový běhový modul (CLR, společný systém typů CTS, společná specifikace jazyků CLS a knihovna tříd rámce .NET FCL. Jazykový běhový modul CLR pracuje jako vrstva mezi operačním systémem a aplikací napsanou v .NET jazyce. CLR abstrahuje služby operačního systému a slouží jako vykonávací jádro pro řízené aplikace. Funkcí CLR je zkompilovat řízený kód, která se skládá z instrukcí zapsaných v pseudostrojovém kódu označovaném jako společný zprostředkovací jazyk CIL do nativního strojového kódu (x86). Takto vzniklá metoda jsou zkompilovaná většinou jen jednou a to při prvním spuštění. Pak se uloží do paměti a při dalším spuštění už nedochází ke zpoždění při kompilaci.

Toto je do značné míry také nevýhoda, protože při složitějších aplikacích jako jsou hry a podobné, je kompilace velice náročná na systémové prostředky počítače. U méně náročných aplikaci to ovšem nevadí. [18]

V prostředí CLR existuje věc, usnadňující práci s operační pamětí – Garbage Collector. Jedná se o sadu složitých algoritmů pro uvolňování nepotřebných programových objektů z paměti.

Společný systém typů CTS plně popisuje všechny možné datové typy a programovací konstrukce, které runtime podporuje, specifikuje, jak mohou tyto entity spolu vzájemně komunikovat.Společná specifikace jazyků CLS definuje podmnožinu společných typů a programovacích konstrukcí, na nichž se dohodnou všechny programovací jazyky .NET.
Knihovna tříd rámce .NET FCL poskytuje objektově orientované rozhraní API, do něhož řízené aplikace zapisují. FCL obsahuje obrovskou sbírku opakovaně použitelných tříd, rozhraní a druhy hodnot schopné rychleji a lépe umožnit vývoj procesu.

Výsledkem použití CLS a CTS je rovnocennost programovacích jazyků. Jinými slovy, pro vývoj .NET aplikací je možné použít jeden z několika programovacích jazyků vyšší úrovně. Pří uvedení této platformy autoři některých programovacích jazyku příliš nevsázeli na velký rozkvět platformy .NET a čekali, že se zařadí mezi ostatní vývojová prostředí s vlastním jádrem. Ovšem postupem času přišli na to, že se spletli. Brzy si tito pesimisté začali uvědomovat jaký je ve skutečnosti skrytý potenciál v .NET Framework a pochopili, že pro bude velice účelné ke skupině jazyků .NET přidat i svoji variantu. Tím se množství podporovaných jazyku rozrostlo na několik desítek.

Může se jednat například o:

- C#, nový jazyk vyvinutý pro .NET
- Visual Basic .NET
- J#, což je jazyk se syntaxí rozšířeného jazyka Java
- managed C++, kde slovíčko managed označuje možnost psát řízený kód pro .NET
- JScript
- A další desítky jazyku COBOL, Pascal, A# atd.

Technologií, které jsou určeny pro tvorbu a návrh webových aplikací či prostředí je v dnešní době vcelku velké množství. Ovšem dle mého názoru žádná z nich neumožňuje tak komplexní přístup k danému tématu jako právě platforma .NET. Tato platforma umožňuje nejen vytvářet a spravovat webové aplikace a služby, ale je to ucelený soubor pro návrh téměř jakékoliv aplikace. Právě pro její komplexnost a relativní jednoduchost je ideální prostředím pro vytvoření aplikace, za účelem sofistikované ochrany proti spamu webových stránek.

6. Jazyk C#

Programovací jazyky Visual Basic a Visual C++ stály odjakživa na rozdílných stranách spektra vývojářů. Visual Basic kladl důraz především na vysokou produktivitu práce a nabízel programátorům co možná nejlehčí a nejpřátelštější vývoj aplikací, a to i za cenu poněkud

omezeného přístupu k systémovým zdrojům. Na druhé straně, Visual C++ šel takřka na hranice možností, protože dovoloval programátorům převzít kontrolu nad systémem plně do svých rukou, i když někdy bylo nutné obětovat notnou dávkou pracovní produktivity.

Vzhledem k tomu, že je .NET radikálním odklonem od předchozích technologií, vyvinula společnost Microsoft specificky pro tuto platformu nový programovací jazyk C#. Je to programovací jazyk, který používá podobnou syntaxi jako Java, ale identický s ní není. C# se nedá považovat za pouhý odvar Javy, oba tyto jazyky jsou založeny na syntetických konstrukcí C++. Podobně jako je Java v mnohých ohledech vyčištěnější verzí C++, je C# v mnohých ohledech vyčištěnější verzí Javy. Je také třeba říci, že mnohé syntaktické konstrukce jsou vymodelované podle různých aspektů Visual Basic a C++. Vzhledem k faktu, že je C# hybrid hned z několika četných jazyků, je výsledkem produkt, který je synteticky tak čistý jako Java, ale stejně jednoduchý jako Visual Basic.

Základní charakteristiky jazyka C# jsou:

- C# je čistě objektově orientovaný
- Obsahuje nativní kontrolu komponentového programování
- Podobně jako Java obsahuje pouze jednoduchou dědičnost s možností násobné implementace rozhraní.
- Vedle členských dat a metod přidává vlastnosti a události.
- Správa paměti je plně automatická. O kurentní uvolňování zdrojů aplikace se stará garbage collector
- Podporuje zpracování chyb pomocí výjimek.
- Zajišťuje typovou bezpečnost a podporuje řízení verzí.
- Zajišťuje zpětnou kompatibilitu se stávajícím kódem, jak na binární, tak na zdrojové úrovni

Jednou z nejdůležitějších věcí, kterou je potřeba si uvědomit v souvislosti s jazykem C# dodávaným jako součást platformy .NET společnosti Microsoft, je to, že umí produkovat jen kód, který se může vykonávat jen uvnitř runtime .NET. [22]

7. ASP.NET

ASP.NET je nejkompletnější platformou pro webový vývoj, jaká byla kdy dána dohromady. Vychází ze svého předka ASP, který byl pouhou sadou nástrojů pro vkládání dynamického obsahu do webových stránek. Platforma ASP.NET je naproti tomu plnohodnotná platforma pro vývoj plnohodnotných a velice rychlých webových aplikací.

ASP.NET je technologie nové generace od společnosti Microsoft pro vytváření webových aplikaci na straně serveru, je založena na Microsoft .NET Framework. Za počátek vzniku této technologie můžeme považovat rok 1996 a technologii ASP od firmy Microsoft. ASP je technologie nezávislá na programovacím jazyce, která umožňuje vykonávat kód na straně serveru

a následné odeslání výsledku uživateli. To znamená, že webová stránka s příponou .asp obsahuje kód, vykonávaný na IIS serveru a prohlížeči odešle pouze výsledek v jazyce HTML. Programovací jazyky, které se u technologie ASP používají nejvíce: VBScript a JScript. ASP postupem času přestalo vyhovovat a nestačilo pro vývoj moderních webových stránek. Proto se v roce 2000 ukončil jeho vývoj a nastoupila nová technologie ASP .NET, která má v současné době verzi 3.5
ASP.NET se v porovnání od dřívějších vývojových platforem odlišuje v těchto věcech:

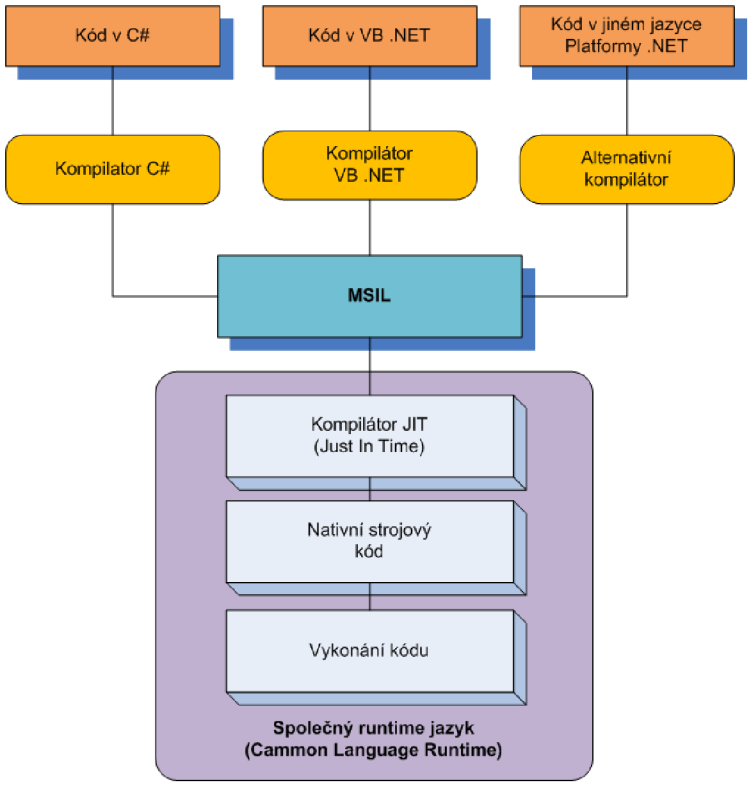
- ASP .NET obsahuje úplný objektově orientovaný programovací model, který obsahuje architekturu řízenou událostmi, založenou na ovládacích prvcích, což podporuje zapouzdřování kódu a jeho opětovného využívání
- ASP .NET dává možnost psát v kterémkoliv z podporovaných jazyků .NET (například C#, Visual Basic, J# a další, které mají kompilátory od jiných výrobců)
- ASP .NET slouží také jako platforma pro budování webových služeb, což jsou opětovně využitelné jednotky kódu, které mohou volat jiné aplikace přes platformy na počítači.
- ASP .NET vyniká vysokým výkonem, to je dáno tím, že komponenty se kompilují na požádání,a tudíž se neinterpretují pokaždé, když se použijí

To je pouze několik ze základních rysů, které jen částečně naznačují hlavní rozdíl v technologii ASP.NET a jiných technologiích

ASP.NET je integrováno s .NET Framework což je obrovská výhoda. Platforma .NET Framework je rozčleněn do kolekce funkčních částí, zahrnujících více jak 7000 druhů (třídy, struktury, rozhraní a další klíčové funkce). V podstatě to znamená, že v ASP.NET používá funkční části .NET Frameworku úplně stejně jako jakémkoliv jiném druhu aplikace .NET, neboli pro webový vývoj jsou k dispozici stejné nástroje, jako pro bohatě vybavené klientské aplikace.

Další z mnoha výhod, ASP.NET se neinterpretuje, ale kompiluje. Interpretace spočívá v tom, že na serveru vykonávaná aplikace se musí nejprve složitě převést do strojového kódu řádek po řádku, to se děje pokaždé kdy je aplikace vyvolána. Kompilace na rozdíl od interpretace je poněkud rychlejší a méně náročná na výkon. Kompilace ASP.NET je rozdělena na dvě etapy. Kód napsaný v jazyce .NET se nejprve převede do přechodného jazyka MSIL (Microsoft Intermediate Language) to se může nastat automaticky, když se stránka poprvé požaduje nebo se může vykonat předem. Druhá etapa kompilace nastává těsně před tím, než se stránka skutečně vykoná. Kód MSIL se zkompiluje do nativního strojového kódu (např. x86). Tato etapa se nazývá kompilace just-in-time (JIT). (Obr. 6.1)

Výhodou je, že kód MSIL se vytvoří jen jednou, což znamená, že se nemusí kompilovat, vždy když přijde požadavek na webovou stránku či webovou službu. Kompilace se znovu odehraje jen tehdy, když dojde k modifikaci původního kódu.



Obrázek 7.1: Kompilace webových stránek ASP.NET

ASP.NET je objektově orientované. Nejenže kód má plný přístup ke všem objektům v .NET Frameworku, může také těžit z veškerých konvencí objektově orientovaného prostředí (OOP). ASP. NET může například vytvářet opětovně využitelné třídy, standardizovat kód s rozhraním, nebo začlenit nějakou užitečnou funkcionalitu do zkompilované komponenty určené k šíření. Dalším problémem,s kterým si ASP.NET velice dobře poradila je podpora různých zařízení a prohlížečů. Přestože ASP.NET stránky dokážou získat informace o prohlížeči klienta a o jeho vlastnostech a díky tomu následně dynamicky upravit stránky, aby odpovídaly vlastnostem daného klienta, existuje i druhá možnost. ASP.NET se snaží o využití webových ovládacích prvků.Ty realizují svůj kód HTML přizpůsobivě - berou v úvahu schopnosti klienta. Příkladem toho mohou být prvky pro ověřování vstupních dat (tzv. validátory) , které pro rozšíření svých schopností používají JavaScript a DHTML, za předpokladu, že je klient podporuje. To umožňuje těmto validačním prvkům dynamicky zobrazovat chybové zprávy, aniž by klient musel odeslat stránku zpět na server k dalšímu zpracování. [19]

Uvádí se, že díky ASP.NET můžeme množství kódů potřebné pro běh dané webové aplikace zredukovat až o 70 %. Čehož se sice v praxi nikdy úplně nedocílí, ale díky množství definovaných funkcí u díky schopnosti implementace nových funkcionalit s využitím velmi malých úprav kódu je kód o poznání kratší, přehlednější a více flexibilní.

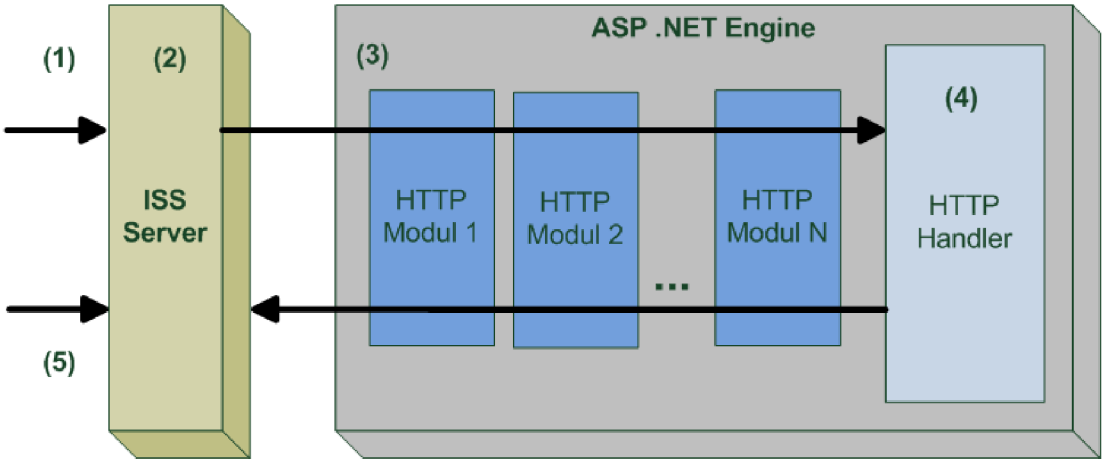
Ani Microsoft neodhadl, s jakým nadšením bude tato technologie přijata. ASP.NET se rychle stalo standardem pro vývoj webových aplikací s technologiemi firmy Microsoft.

7.1 Zpracování stránky prostřednictvím ASP .NET

Vždy když dorazí žádost na webový server pro ASP .NET webovou stránku, je tato žádost Web serveru odejmuta k ASP .NET Engine. V tomto enginu pak žádost prochází několika body, které zahrnují ověření přístupových práv, aktivování uživatelského stavu a podobnými, o to se starají http moduly. Po projití všemi požadovanými moduly se žádost dostane k http handleru, který ověří požadavky a na základě nich odešle odpověď zpět na Web server a ten odešle požadovanou stránku klientovi, který o ni žádal. (Obr. 6.2)

Http moduly slouží k vyčlenění kódu sdíleného více WWW aplikacemi, které jsou napsány v ASP.NET. S ASP.NET jsou dodávány moduly, které se v aplikacích starají například o cachování dat nebo o autentizaci uživatele.

Http handler v ASP.NET je autonomní funkční modul pro zpracovaní požadavků, jež jsou cíleny na soubory se specifickou příponou či jménem. [20]



Obrázek 7.2: Zpracování stránky prostřednictvím ASP .NET [20]

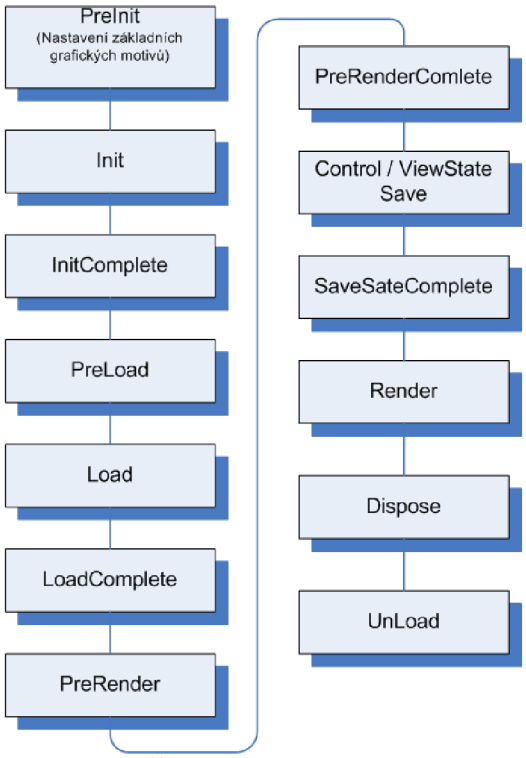
1 – Příchozí požadavek na Web server
2 – Žádost je ze serveru odejmuta k ASP .NET Engine
3 – Žádost ověřována skrz http moduly
4 – Žádost je zpracována http handlerem a odeslána zpět na Web server
5 – Výsledek je odeslán zpět na stranu klienta

Toto je jen velice obecný popis jakými kroky prochází ASP .NET stránka. Jelikož tento projekt se zabývá vytvoření webové aplikace, tak je níže uvedeno, jakým způsobem dochází ke zpracování samotného kódu dané aplikace.

Platforma ASP .NET jak už jsem uvedl výše je přelomová v tom směru, že narozdíl od běžné ASP, či jiných technik je založena na Objektově Orientovaném programovaní.Cyklus webové stránky je zahájen žádostí od klienta na její zobrazení. Server na základě požadavku nahraje

klientovi danou stránku a po jejím zpracování je tato stránka opět odstraněna. Od nahrání až do odstranění stránky se dle jejího vytvořeného objektového modelu zpracují všechny informace, které jsou stránce předány. Ve výsledku je pro klienta vygenerován HTML kód.

Zpracování klasické HTML stránky probíhá ve zpracování celého kódu od první řádky stránky a veškeré zpracování je ukončeno posledním řádkem stránky. Takto fungovalo i klasické ASP, ovšem v ASP .NET aplikacích to funguje zcela jinak. Jedná se o OOP, takže klasické, jednoúčelové zpracovaní od prvního řádku kódu po poslední řádek jde zcela stranou. ASP .NET nabízí mnohem více. (Obr. 6.3)



Obrázek 7.3: Životní cyklus ASP .NET stránky

Zde je uvedeno jakými nejdůležitějšími fázemi ASP .NET stránka prochází.

- **`Init`** – Inicializace

Inicializace jak už sám název napovídá, slouží k inicializaci objektů uvnitř stránky a také nastavení pro zpracování požadavku. Tyto požadavky vyvolá událost *`Init`*, která vychází z metody *`OnInit()`*. Takto zinicializované objekty a nastavení jsou používány po celou dobu zpracování stránky.

- **`LoadViewState`** – Nahrání dat ve `ViewState`

ASP .NET stránka si umí uchovávat stavové informace všech prvků v ní obsažených a to právě pomocí *`ViewState`* a také informace o stránce samotné, která je také prvkem. Tyto informace jsou předdefinované v metodě *`LoadViewState()`*.

- **`LoadPostData`** – Zpracování vstupních dat

V této metodě dochází ke zpracování popřípadě aktualizaci, čí změně dat. Zpracování dat je automatické, ovšem aktualizace a následná změna dat je možná jen tehdy pokud je bráno v potaz rozhraní `IPostBackDataHandler`.

- **`Load`** – Načtení a zpracování stránky

V metodě `OnLoad()` události `Load` jsou zpracovány všechny požadavky. Všechny objekty na dané ASP .NET stránce jsou uspořádány do stromu (myšleno pouze obrazně) a jsou inicializovány na základě dat předaných od klienta. Od tohoto okamžiku je možné jednotlivým prvkům upravovat vlastnosti, či s nimi jinak manipulovat.

- **`RaisePostDataChanged`** – Došlo ke změně dat

Zpracování metody `RaisePostDataChangedEvent()`, která se inicializuje prostřednictvím `RaisePostDataChanged`, vychází faktu, zda na dané stránce došlo ke změně u konkrétních prvků oproti předcházejícímu stavu. Například vyplnění, `TextBoxu`, dojde ke zpracování metody `RaisePostDataChangedEvent()`.

- `RaisePostBackEvents` – Vyvolání události

Vyvolání události `RaisePostBackEvents` je zapříčiněno chováním klienta a vyvolává příslušné události, jež byly zaregistrovány v metodě `OnInit()` pro příslušný prvek. Tuto událost mohou vyvolat pouze prvky, které mají implementován interface `IPostBackEventHandler`.

- **`PreRender`** – Předrenderování stránky

V této metodě se naskýtá poslední možnost změnit nějakým způsobem vlastností prvků, ještě před tím, než je jejich stav uchován ve `ViewState`. Událost je vyvolána pouze u těch prvků, které budou na stránce zobrazeny.

- **`SaveViewState`** – Uložení `ViewState` (stavu stránky a prvků)

K uložení stavu prvků na stránce dojde při zpracování metody `SaveViewState()`. Stav prvků, které jsou uloženy prostřednictvím této metody, je převeden na textový řetězec a následně zakódován pomocí *BASE64*. To je z toho důvody, aby nikdo neoprávněný nemohl tyto informace využít. `ViewState` se hojně využívá například u nekupních košíků internetových obchodů

- **`Render`** – Vygenerování stránky

V metodě `Render()` dochází k vykreslování všech prvků na stránce, které se mají zobrazit klientovi. Vykreslování spočívá ve vygenerování kódu pro dané rozhraní na klientově straně. Může se jednat o HTML, nebo například XML.

Pokud předefinujeme metodu `Render()`, potom ovlivníme konečný výstup, jenž naše stránka (prvek) poskytne klientovi. Změny vlastností u prvků stránky, které bychom provedli v této metodě, se projeví na klientovi, avšak nebudou již promítnuty do zpětného zpracování.

- **`Dispose`** – Odstranění objektů

V této fázi dochází k odstraňování objektů, které se při zpracování stránky použily.

- **`UnLoad`** – Odstranění stránky ze zpracování

A na konec je vyvolána událost `UnLoad()` a tato odstraní všechny prvky stránky ze zpracování. Po ukončení zpracování této metody je již stránka odeslána klientovi a zobrazena v prohlížeči.

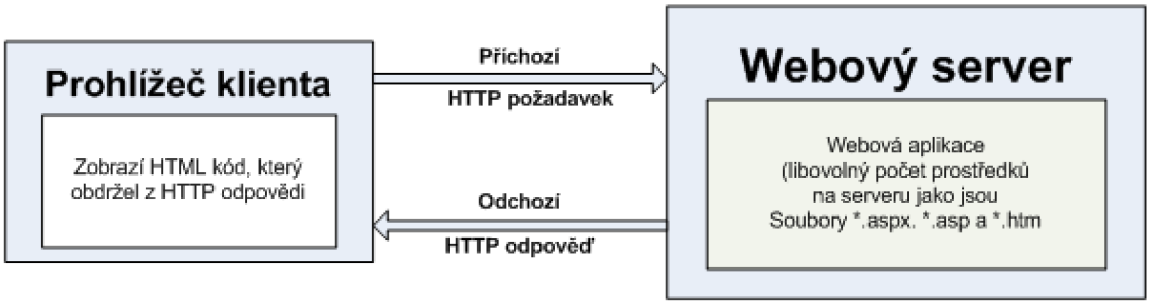
Další velice důležitou metodou je metoda `CreateChildControls()`, která provádí zpracování prvků uvnitř stránky a prvků uvnitř prvků (kontejnerů). Tato metoda zajišťuje upořádání všech prvků. Jelikož k volání této metody dochází několikrát v průběhu zpracovávání aspx stránky není možné ji přesně zařadit.

Poprvé je volána při nahrávání stránky (`Load`), následně během navazování prvků na data, nebo při vykreslování (`Render`).[19]

7.2 WEBOVÉ APLIKACE A WEBOVÉ OVLÁDACÍ PRVKY

Webové aplikace jsou hodně odlišným druhem, pokud je porovnáme s klasickými desktopovými aplikacemi. Prvním zřejmým rozdílem je samotný princip, běžná webová aplikace vždy zahrnuje alespoň dva stroje propojené v síti (pro vývoj samozřejmě stačí pouze jeden, který má roli klienta i serveru). Vzhledem k tomu musí konkrétní stroj spolupracovat s konkrétním přenosovým protokolem, aby bylo možné odesílat a přijímat data, jedná se o HTTP protokol.

HTTP protokol je založený na textu a je vybudován pomocí standardního schématu požadavek/odpověď. Když stroj spustí nějaký webový prohlížeč (Mozilla Firefox, Internet Expolerer atd.), vznikne http požadavek na přístup ke konkrétnímu prostředku (např. soubor *.aspx nebo *.htm) na vzdáleném stroji serveru. (Obr. 7.4).



Obrázek 7.4: Cyklus požadavku a odpovědi HTTP

Jakmile příslušný webový server obdrží příchozí webový http požadavek, specifikovaný prostředek může obsahovat nějakou logiku, která extrahuje veškeré vstupní hodnoty dodané

uživatelem (například hodnota zadaná do textového pole), aby mohla naformátovat řádnou HTTP odpověď. Tyto HTTP požadavky mohou být dynamicky generovány pomocí různých technologií (ASP .NET, CGI, ASP, Java atd.) , tyto požadavky jsou následně převedeny do HTML kódu, který je následně zobrazen u klienta.

Dalším podstatně odlišným aspektem webového vývoje od desktopového programování je skutečnost, že HTTP je v podstatě bezestavový přenosový protokol. Což v podstatě znamená, že jakmile server odešle odpověď klientovi, okamžitě zapomene všechno o předchozí komunikaci. Ovšem díky webovým vývojovým technologiím je možné podniknout různá opatření, aby se tomuto „zapomínání“ zabránilo.

Tento problém lze jednoduše ukázat například na internetových obchodech, kde jsou uloženy položky v tzv. nákupním košíku. Samotné ASP .NET poskytuje řadu způsobů, jimiž se dá zpracovat stav – mnohé z nich jsou samozřejmou součástí každé webové platformy (proměnné relace, cookies, aplikační proměnné), některé jsou ovšem velmi odlišné a zcela nové (například zobrazení ovládacího prvku či cachováné do paměti)

Rozdílu mezi webovými aplikacemi a těmi desktopovými je celá řada, ovšem pro základní představu zde uvedená jsou dostatečné. Stejně jako u desktopových aplikací existují i v těch webových ovládací prvky, které programátorovy podstatně usnadňují práci. Existuje jich celá řada od těch klasických, které mají své ekvivalenty u desktopových aplikací, až po prvky které slouží čistě jen pro potřeby webu. [19],[21]

7.2.1 Ovládací prvky

Samotný návrh CAPTCHA budu provádět pomocí *Custom Controls*, které jsou pro tento účel velice vhodné. Jejich nespornou výhodou je jejich rychlost a jednoduchá schopnost implementace do různých webových aplikací a prostředí. Můžeme je považovat za univerzální prvky, které jsou přímo určeny pro několikanásobné použití. To je pro tento účel nutné, protože cílem toho projektu je vytvořit komplexní ochranu proti spamu webových stránek, k čemuž je CAPTCHA přímo určena. Z toho důvodu je nutné se s těmito prvky více seznámit, což splňuje následující text.

S ovládacími prvky se můžeme setkat prakticky ve všech odvětvích vývoje a návrhu různých programů či aplikací. Využívají se ke zkrácení doby vývoje a dodávají aplikaci vzhled a řízení, jež je konzistentní s ostatními aplikacemi budovanými pro danou platformu. Obecně mezi ovládací prvky patří například panely nástrojů nebo stavové řádky, jež jsou charakteristické téměř pro každou aplikaci, zejména ve Windows.[19]

7.2.1.1 UŽIVATELSKÝ OVLÁDACÍ PRVEK (User Control)

Je vlastní ovládací prvek tvořený kódem HTML a serverovým skriptem. Jedná se v podstatě o jednoduchý prostředek, který běží jen na dané webové stránce, a proto nevyžaduje vkládání kód na straně serveru. *User control* jsou vhodné pro jednoúčelové funkce, která jsou využity jen v jedné aplikaci, naopak nejsou vhodné pro standardizování opakujícího se odkazu na všech stránkách webu. Například pokud se jedná o vybudování a opětovné použití záhlaví, zápatí čí

navigačních pomůcek. Ovšem pro svou jednoduchost a intuitivní tvorbou patří mezi jedny z nejpoužívanějších komponent ASP .NET.

Uživatelské ovládací prvky mají tří základní využití:

- Rozčlenění složitých uživatelských ovládacích prvků UI a jejich převedení do opakovaně použitelných komponent
- Vytváření dynamických stránek personalizovaných pro jednotlivé uživatele
- Umožnění práce s mezipamětí na podstránkové úrovni – to znamená ukládání statického obsahu stránek, přičemž ostatní stránky lze zpracovávat dynamicky

Typickým příkladem je možnost poskytnout uživatelům na několika stránkách jednotný způsob pro zadávání informací, z nichž se skládají kontaktní údaje. Dá se to vyřešit tak, že pro kontaktní údaje se vytvoří uživatelský ovládací prvek, který bude obsahovat několik textových polí a několik validátorů. Díky tomu, že se jedná o uživatelský ovládací prvek, může se s ním pracovat jako s jediným objektem a je možno jej vkládat na jakoukoliv webovou stránku.[21]

7.2.1.2 SERVEROVÝ OVLÁDACÍ PRVEK (Custom controls)

Na rozdíl od uživatelských ovládacích prvků serverové ovládací prvky, jak už název napovídá, vyžadují obousměrnou komunikaci mezi klientem a serverem. Platforma ASP .NET podporuje dva výrazně odlišné typy serverových ovládacích prvků: Ovládací prvky HTML a webové ovládací prvky. Ovládací prvky HTML se přímo vztahují k syntaxi HTML, což umožňuje vytvářet webové stránky ASP .NET v editorech určených pro HTML a rychlou integraci těchto prvků při běžném vývoji pomocí HTML.

Na rozdíl od toho, webové ovládací prvky umožňují vytvořit bohaté serverové ovládací prvky, schopné se realizovat z desítek odlišných prvků HTML, ale přitom stále poskytovat jednoduché rozhraní založené na objektech. Díky tomuto modelu je pak možné pracovat s mnohem vyšší úrovní abstrakce a s větší mírou funkcionality. Nesporou výhodou těchto prvků je jejich možnost rychle a pohodlné implementace do webových aplikací, bez toho aniž by vývojář této aplikace musel vytvářet, či jinak upravovat zdrojový kód. Díky této nesporné výhodě je i samotný návrh CAPTCHY realizován pomocí *Custom Control*. [21]

Rozdělení ovládacích prvků

- Jednoduché ovládací prvky
- Tlačítkové ovládací prvky, vytvářejí různé druhy tlačítek ve webových formulářích
- Seznamové ovládací prvky, zobrazují posté seznamy položek
- Datové ovládací prvky, používají datové vazby k zobrazení informací získaných z databází a jiných zdrojů

- Ovládací prvek `Calendar`, přidává do webových formulářů interaktivní kalendář
- Ověřovací ovládací prvky, které ověřují zadání uživatele před a po předání formuláře serveru

Jednouché ovládací prvky
Jsou to prvky, které se mapují na standardní prvky HTML, některé vracejí i klientské skripty, ale pouze za speciálních podmínek. Jejich nespornou výhodou je, že se velmi snadno se používají. Patří zde například `TextBox`, `Label`, `HyperLink`, `Image`, `Table` atd.[19]

Tlačítkové ovládací prvky
Tyto ovládací prvky zahrnují tři typy tlačítkových prvků: `Button`, `LinkButton` a `ImageButton`. Funkčně dělají naprosto totéž, odesílají formulář, v kterém jsou obsaženy na server. Rozdíl spočívá v jejich vzhledu. `Button` vypadá jako klasické tlačítko, `LinkButton` jako hypertextová odkaz a `ImageButton` se zobrazí prostřednictvím definovaného obrázku. Téměř každá webový formulář obsahuje jedno nebo několik tlačítek, aby mohl uživatel odeslat formulář na server.[21]

Seznamové ovládací prvky
Obsahují čtyři členy ovládacích prvků: `ListBox`, `DropDownList`, `CheckBoxList` a `RadioButtonList`. Všechny mají společnou jednu vlastnost, jsou určeny k tomu, aby uživateli nabídly seznam definovaných položek. Ovládací prvky `ListBox` a `DropDownList` zobrazují textové položky, které uživatel může vybírat. Obě se odesílají zpět prohlížeči jako značky <select> HTML. `CheckBoxList` a `RadioButtonList` zobrazují pole zaškrtávacích políček a přepínačů [21]

Datové ovládací prvky
Jedná se o tři ovládací prvky, jejichž hlavním úkolem je vázat se k datovým zdrojům a zobrazovat výsledky jako HTML. Těmito prvky jsou `Repeater`, `DataList` a `DataGrid`. `Repeater` se používá k zobrazení položek získaných z nějakého datového zdroje. Prvek `DataList` také slouží k zobrazení položek z datového zdroje, ovšem podporuje vícesloupcové formátování, výběr položek a úpravu položek Prvek `DataGrid` zobrazuje tabulková data ve vysoce upravitelných tabulkách HTML a podporuje stránkování, řízení, úpravu a další funkce. Tyto tři prvky patří mezi nejsilnější a nejpoužívanější ovládací prvky v ASP .NET.[22]

Ovládací prvek `Calendar`
Tento prvek je významný z jednoho důvodu, vytvoří vlastního kalendáře jako tabulky je velice jednoduché, ovšem napsat kód k tomu, aby zachycovala nějaké konkrétní datum vybrané uživatelem, je velmi zdlouhavé a nyní už zbytečné. Ovládací prvek `Calendar` automaticky zobrazuje aktuální měsíc, není-li uvedeno jinak, automaticky spouští určitou událost po výběru data a samočinně zobrazuje předcházející i následují měsíc po stisku některého z jejich navigačních kláves [19]

Ověřovací ovládací prvky

Validační ovládací prvky – validátory patří mezi velice užitečné části ASP .NET potažmo jazyka C#. Jejich účelem je ověření platnosti zadaných dat na straně uživatele ještě před odesláním stránky na webový server, tak aby byla splněna určitá omezení. Na straně klienta se zajistí, aby vstupní data formuláře byla vyplněna podle požadavků, které jsou předem definované. Existuji v podstatě dva typy kontroly kontrola na straně serveru a na straně klienta.

Kontrola na straně serveru probíhá vždy, zajišťuje, aby data nebyla serveru podvrhnuta, tak že která nebyla zkontrolována klientským skriptem. Kontrola na straně klienta je výhodná v tom, že data se nemusí neustále posílat tam a zpět, takže neprobíhá zbytečná komunikace klient server a tím se tak nezatěžuje server.

Konstrukce validačních ovládacích prvků je velmi silná a zároveň i velmi flexibilní, jejich použití je velmi prosté. Nastavením vlastnosti `ControlToValidate` na nějaký z ovládacích prvků na formuláři se zajistí jeho kontrola schvalovacím prvkem dle výběru. Existuje několik typů těchto prvků:

`RequiredFieldValidator` – testuje zda daný prvek, ke kterému je připojen, obsahuje nějakou hodnotu – musí být vyplněn. To je podmínkou pro úspěšné odeslání formuláře. Je možné u něj nastavit tyto vlastnosti

- **`ErrorMessage`** – text, která se zobrazí při nesplnění podmínek
- **`ControlToValidate`** – získá, nebo nastaví ovládací prvek, jehož platnost se ověřuje

`CompareValidator` – porovnává hodnotu ovládacího prvku s nějakou jinou hodnotou v jiném ovládacím prvku nebo s fixní konstantní hodnotou. Nastavitelné vlastnosti

- **`ErrorMessage`** – text, která se zobrazí při nesplnění podmínek
- **`ControlToCompare`** – určuje ovládací prvek, proti kterému se bude porovnávat.
- **`ControlToValidate`** – získá, nebo nastaví ovládací prvek, jehož platnost se ověřuje buď proti `ControlToCompare` nebo konstantně.
- **`Operator`** – určuje, jaká operace se bude provádět při porovnávání (rovnost, nerovnost, větší, menší…)
- **`Type`** – datový typ, který bude zadáván do ovládacích prvků (`String`, `Integer`…).
- **`ValueToCompare`** – konstanta, proti které se může porovnávat.

`RangeValidator` – kontroluje, jestli je daná hodnota z předefinovaného rozsahu. Nastavitelné vlastnosti

- **`ErrorMessage`** – text, která se zobrazí při nesplnění podmínek
- **`ControlToCompare`** – určuje ovládací prvek, který se bude kontrolovat.
- **`MaximumValue`** – maximální hodnota ovládacího prvku.
- **`MinimumValue`** – minimální hodnota ovládacího prvku.
- **`Type`** – datový typ, který bude zadáván do ovládacího prvku (String, Integer…)

`RegularExpressionValidator` – zkontroluje, jestli hodnota sdruženého vstupního ovládacího prvku vyhovuje vzoru specifikovaného regulárního výrazu. Nastavitelné vlastnosti

- **`ValidationExpression`** – obsahuje regulární výraz sloužící ke kontrole.

`CustomValidator` – umožňuje vybudovat vlastní funkci pro ověřování platnosti, která provede validaci daného ovládacího prvku. Dává programátorovi největší svobodu při kontrole údajů v ovládacím prvku k němu připojeném

- **`ClientValidationFunction`** – ověřovací funkce provádějící se na straně klienta
- **`ServerValidate`** – vyvolává se při ověřování prvků na straně serveru.

`ValidationSummary` – Zobrazí souhrn všech chyb nalezených při validaci stránky jako seznam. Tyto chyby se dají zobrazovat několika způsoby, je možno nastavit, jestli se zobrazí jako prostý seznam, nebo jako odstavce textu a zda se budou zobrazovat v dokumentu, jako „vyskakující“ okno zprávy, nebo obojí dohromady.

Všechny tyto prvky až na `ValidationSummary` obsahují ještě důležitou vlastnost Display. Tato vlastnost může být:

- **`Static`** – čeká se s kontrolou až na odeslání formuláře serveru, který provede kontrolu a vrátí výsledek.
- **`Dynamic`** – provádí kontrolu okamžitě po vyplnění ovládacího prvku pomocí klientských skriptů. Tyto skripty se však jako všechno ostatní v ASP.NET generují v závislosti na schopnostech prohlížeče. Pokud je prohlížeč podporovat nebude, přistoupí .NET framework ke kontrole na straně serveru.

Serverové prvky se dají využít k rozličnému použití, od návrhu jednoduchých speciálních tlačítek až po vytvoření plně hodnotné webové aplikace, která je díky vlastnostem prvku schopna integrovaní prakticky do všech internetových systémů postavených na bázi platformy .NET [22]

Tato integrace se provádí prostřednictvím DLL knihoven, do kterých jsou tyto prvky po vytvoření zkompilovány. Tuto knihovnu lze jednoduše přidat do již vytvořené aplikace, patřičně ji zaregistrovat, a pokud je potřeba provést další nezbytné nastavení. Nastavení a registrace se může částečně lišit pro každý prvek, ovšem to záleží na použití a účelu prvku. Jelikož návrh CAPTCHA je proveden prostřednictvím *Custom Controls*, bude dále v textu popsáno, jakým způsobem se tyto konkrétní prvky registrují.

Na závěr jsou zde uvedeny hlavní rozdíly mezi *User Control* a *Custom Control*

- Návrhová podpora:
  - *User Control*: dobrá
  - *Custom Control*: minimální

- Složitost (náročnost implementace):
  - *User Control*: malá
  - *Custom Control*: vysoká
- Nasazení komponenty
  - *User Control* : omezené, jednoduché a jednoúčelové aplikace
  - *Custom Control* : globální, více obecné, složitější systémy

- *User Control* se vytváří stejně jako ASPX stránka – je možné využít stejný designér ve Visual Studiu .NET, jako pro běžný návrh internetové aplikace. Naproti tomu, *Custom Control* musí mít HTML výstup programovaný ručně, což zahrnuje více času stráveného psaním a laděním kódu.
- *User Control*, pro použití, musí být nakopírován v každé jednotlivé aplikaci, která jek využívá Pro použití *Custom Control* stačí jedna dll knihovna umístněna v Global Assembly Cache - zásobníku knihoven pro všechny aplikace, které ji chtějí využívat.
- *Custom Control* se kompiluje jako dll knihovna, zatímco *User Control* je obsažen v ASCX souboru, v případě codebehind má navíc ještě další dll knihovnu pro kód dané ASCX stránky. Oba tyto soubory je nutné mít v aplikaci pro provoz komponenty. Z toho plyne větší náročnost na údržbu – v případná změny v kódu se musí tato změna vložit do každé stránky, kde je *User Control* použita. U *Custom Control* se změny automaticky reflektují okamžitě po přeložení knihovny ve všech aplikacích.

Z toho plyne, že každý typ, jak *User Control*, tak *Custom control*, má několik nesporných výhod a zaleží už hlavně na typu použití, pokud se uvažuje nad využitím jen pro jednu aplikaci a komponenta nevyžaduje uživatelský přívětivější rozhraní je vhodná *User Control*, pro rychlost návrhu, v opačném případě, pokud uvažujeme o několika násobném využití, je jednoznačná volba *Custom Control.* [21]

Implementace prvků do webové stránky

Při předpokladu, že ovládací prvek je již vytvořen, je nutné pro jeho správnou funkci jej zaregistrovat pro danou stránku. Pro registraci musíme vložit do zdrojového, například `Home.aspx`, direktivu `@Registrer` s odpovídajícími parametry. V ukázce kódu níže, je zobrazena registrace obou prvku, jak *Custom Control* tak *User Control.*

```
<%@ Register Assembly="Captcha1" Namespace="Captcha1" TagPrefix="cc1" %>
<%@ Register TagPrefix="user" TagName="NavigLisXml" Src="NavigLisXml.ascx" %>
```

Na první pohled je zřejmý rozdíl mezi těmito dvěma prvky, *User Control* se přímo odkazuje na zdrojový soubor `NavigListXml.ascx`, naproti tomu *Custom Control* má zahrnutu direktivu `Assembly`, která označuje jeden nebo více souborů seskupených do logické jednotky, která se skrývá v knihovně DLL. Společné pro oba prvky je `TagPrefix`, která deklaruje název, pod kterým se odkazujeme na daný prvek v kódu stránky, který je zobrazen na další straně.

Další parametry, už závisí na samotném zdrojovém kódu stránky a nebo jsou odvozeny z registračního pole, které ovšem může obsahovat jen souhrnné informace o celém prvku a ne pří

jednotlivé funkční parametry. Tyto funkční parametry jsou zobrazeny u *User Control*, jedná se například o `BarvaPopredi`, `Font-Name` a jim podobné, díky těmto parametrům lze upravovat jednotlivá nastavení prvků.

```
<user:NavigLisXml ID="NavigLisXml1" ZdrojXml="Odkazy.xml" BarvaPopredi="white"
                  Font-Name="verdana" Font-Size="10pt" Font-Bold="true"
                  BarvaPrechoduMysi="black" RunAt="server" />
<cc1:CaptchaImage1 ID="CaptchaImage1_1" runat="server" />
```

Prostředí ASP .NET není programovací jazyk, je to pouze platforma, která nám umožňuje komplexní návrh webového prostředí či webové aplikace. Umožňuje návrh od fundamentálních záležitostí až po úplná více úrovňová prostředí. Její inovační schopností je možnost implementace kódu, který byl napsán v kompatibilním jazyce, což v současné době splňuje velké množství jazyků přizpůsobených pro platformu .NET. Jazyk C# byl pro tuto platformu přímo navržen proto je ideálním východiskem pro účely tohoto návrhu. [19]

8. Návrh CAPTCHA

Jak už jsem uvedl výše CAPTCHA slouží jako funkční ochrana nejen proti spamu, ale jako plnohodnotný test, který má úkol rozlišit jednu věc. Zda se jedná o běžného uživatele a tudíž člověka nebo se jedná a skript, která má jeho chovaní pouze napodobit a tím se dostat na pro něj nepřístupná místa. Díky možnosti implementace CAPTCHA do rozličných webových služeb a aplikací je proto tento projekt ideálním řešením. Účelem tohoto návrhu je poskytnout několik možných variant CAPTCHA jako webových aplikací, kde bude mít možnost provozovatel konkrétní služby uzpůsobit tuto antispamovou ochranu dle vlastních požadavků a pro konkrétní účely.

CAPTCHA je tvořena třemi různými variantami své nejznámější a nejpoužívanější podoby. Provozovatel má možnost definovat různé parametry pro jednotlivé typy CAPTCHA. Zde je uveden jen stručný výčet typů, bližší informace k samotným funkcím a možnostem nastavení je uveden dále. První z těchto zmiňovaných tří je ta nejvíce rozšířená forma, jedná se o CAPTCHA ve které se generuje náhodně určitý počet symbolů s různě definovaným pozadím. Dalším z této trojice je matematická CAPTCHA, která bude umožňovat nastavit obtížnost a množství generovaných matematických úkolů. A posledním ze zmiňovaných bude obrázková forma CAPTCHA, která bude doplněna o několik drobných vylepšení oproti výše zmíněné v kapitole 4. Častým problémem tohoto typu antispamové ochrany je její nepřístupnost pro lidi se zrakovým hendikepem, tomuto problému se chci také při svém návrhu věnovat a to vytvořením audio CAPTCHA, které by měla usnadnit a částečně zjednodušit problém, se kterým se tito lidé musejí denně potýkat.

8.1 CAPTCHA generující zdeformovaná text

CATPCHA generující náhodně zdeformovaný text je nejvíce používanou formou, proto její vlastnosti mají největší variabilitu. Její základní vlastností je zobrazit náhodně generovaný řetězec zdeformovaných symbolů, na různém pozadí, které pro úspěšné prolomení je nutno správně opsat. Tato varianta také obsahuje audio výstup, kde lze v případě potřeby přehrát vygenerovaný text.

Vlastnosti samotné aplikace:

1. **Text pro uživatele [TextProUzivatele]** umožňuje nastavit zprávu, která bude uživateli zobrazena a bude sloužit jako nápověda
2. **Zobrazení tlačítka pro přehrání audia** [**showPlayButton]** pokud by provozovatel chtěl nemuselo by být umožněno přehrát audio, což ovšem není vhodmé, protože by tak captcha nebyla přístupná pro zrakově hendikepované uživatele

Vlastnosti generované řetězce je možno měnit několika způsoby:

1. **Počet symbolů [PocetZnaku]** v řetězci, jedná se o určité množství, z praktického hlediska je vhodné volit z rozsahu 4–9 symbolů, pokud by byl nastaven nižší počet, nebyl by problém díky malému počtu variant CAPTCHA prolomit a při vyšším počtu nežli 9 by se jednalo o neadekvátní množství, které by uživatele spíše odrazovalo
2. **Typ symbolu [TypZnaku]** v řetězci, je možno nastavit, zda budou zobrazeny jen numerické symboly nebo náhodně generovaná písmena
3. **Velikost písma [FontSize]** jak budou zobrazené zdeformované znaky velké
4. **Barva písma [FontColor]** při nastavování barvy je možno volit z kombinace dvou barev, text bude ve výsledku zobrazen specifickým způsobem
5. **Pokřivení textu [FontWarp]** zde je možno nastavit, jak bude výstupní text zdeformovaný
6. **Font písma [ImageFont]** zde je možno zvolit typ písma, nebo po zadání hodnoty „-1“ je font automaticky náhodně generován

Vlastnosti pozadí lze také nastavit dle potřeby:

1. Natavení rozměrů výsledného obrázku je možno provádět pomocí voleb **[SirkaObrazku**] a **[VyskaObrazku]** ovšem hodnoty zde uvedené platí určitá omezení, výška nesmí být menší než je 60 a výška nesmí být menší nežli 160. Pokud by tyto hodnoty byly nižší, text by nebyl příliš čitelný a všechny znaky by se nemuseli vejít do vygenerovaného obrázku
2. **Barva pozadí [BackColor**] opět se jedná o kombinaci dvou barev, z kterých bude vykresleno pozadí
3. **Způsob vykreslení pozadí [BackgroundHatchStyle]** zde je možno volit z několika možností jak bude následná CAPTCHA vykreslena
4. Deformace textu **[WarpInImage**] jednoduše lze povolit, či naopak zakázat
5. Šum obrázku **[Solnicka**] zde je možno nastavit míru šumu, který je obsažen ve výsledném obrázku

Několik možných variant jak by obrázek CAPTCHA mohl při různých nastaveních vypadat (Obr. 8.1)

Obrázek 8.1: Možné varianty zobrazení

8.2 CAPTCHA generující matematickou úlohu

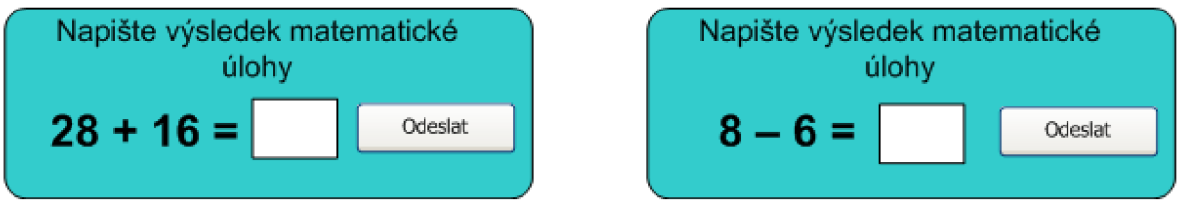
Matematická CAPTCHA je jednou z méně používaných, z důvodu její omezené možnosti vytvářet nové varianty. Proto jsem se rozhodl ji implementovat více užitečných funkcí. Její nejběžnější variantou je sčítáni dvou jednociferných čísel. Jelikož dnešní roboti, kteří napadající právě CAPTCHA zatím nedokážou rozeznat, že se jedná o matematický výraz a určit jeho výsledek, má tento typ vysokou pravděpodobnost úspěchu v obraně proti botom. Matematická CAPTCHA bude možno generovat automaticky, nebo dle nastavení provozovatele. (obr. 8.2)

Vlastnosti samotné aplikace:

1. **Text pro uživatele [TextProUzivatele]** umožňuje nastavit zprávu, která bude uživateli zobrazena a bude sloužit jako nápověda

Vlastnosti, která je možno upravit:

1. **Počet cifer**, je možnost volit pouze mezi dvěma možnostmi a to jednociferná čísla, anebo dvouciferná
2. **Typ operace**, zde bude na výběr ze dvou matematických operací + a –, provozovatel může přímo zvolit jednu z nich a nebo se bude generovat náhodně
3. **Vlastni hodnota [VlCislo],** provozovatel může zvolit, zda budou hodnoty proměnných náhodně generovány, nebo může vložit vlastní hodnotu. Náhodné generování se provede po vložení „0“ do položky **[VlCislo]**



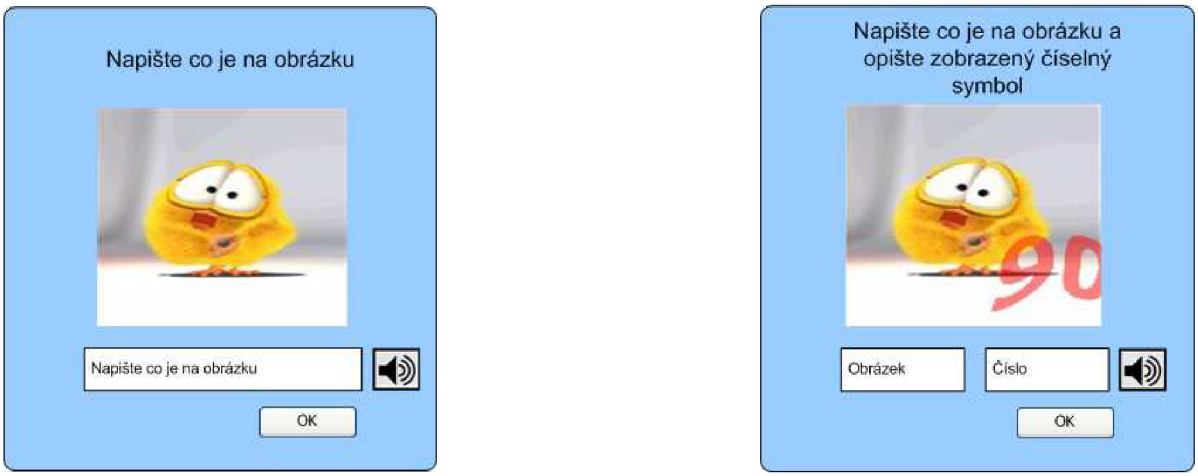
Obrázek 8.2: Vizualizace matematické CAPTCHA

8.3 CAPTCHA generující obrázek

Tento typ CAPTCHA jsem se rozhodl částečně upravit. Běžně rozšířená verze této formy vygeneruje obrázek z omezeného počtu, který je do CAPTCHA implementován předem. Z čehož plyne nevýhoda už na první pohled a to je opět malá možnost kombinací. Běžně bývá v databázi okolo 30 obrázků, takže šance, že bot se trefí do správné možnosti, je poměrně vysoká. V tomto případě bude také generován obrázek z databáze, ale do obrázku mohou být vloženy další symboly a to konkrétně náhodně vygenerované dvojčíslí. Uživatel musí napsat co je zobrazeno na obrázku a pokud bude vložen číselných symbol, tak i ten. (Obr. 8.2). Pro ošetření přístupnosti je do aplikace implementována i audio CAPTCHA, která přehraje náhodně vygenerovaný řetězec pěti čísel.

Vlastnosti aplikace:

1. Natavení rozměrů výsledného obrázku je možno provádět pomocí voleb **[Sirka**] a **[Vyska]** ovšem hodnoty zde uvedené platí určitá omezení, velikost výstupního obrázku, lze nastavit libovolně, ale provozovatel musí dbát na to aby výsledné zobrazení příliš nedeformovalo obrázek.
2. **Text pro uživatele [TextProUzivatele]** umožňuje nastavit zprávu, která bude uživateli zobrazena a bude sloužit jako nápověda
3. **Zobrazení tlačítka pro přehrání audia** [**showPlayButton]** pokud by provozovatel chtěl nemuselo by být umožněno přehrát audio, což ovšem není vhodmé, protože by tak captcha nebyla přístupná pro zrakově hendikepované uživatele
4. **Zobrazení čísla** ve formě vodoznaku **[NumberInImage],** lze jednoduše povolit, nebo naopak zakázat



Obrázek 8.2: Rozdíl mezi obrázkovou CAPTCHA s vloženým vodoznakem a bez něj

8.4 Audio CAPTCHA

Audio CAPTCHA, není v podstatě další typ, ale jedná se jen o odlišný způsob kontroly, je spojena s CAPTCHOU generující zdeformovaný text a CAPTCHOU generující obrázek a to z důvodu přístupnosti. Po stisknutí příslušného tlačítka se přehraje zvuková stopa, obsahující určité symboly, které po úspěšném opsání do kontrolního pole znamenají „prolomení" CAPTCHA. Návrh těchto kontrolních prvků byl proveden v jazyce C#, kde je podpora přehrávání audia implementována v podobě knihovny, která podporuje přehrávání pouze s anglickým přízvukem, což znamená, že česká slova a diakritika by byla příliš zkomolená. Z tohoto důvodu Audio CAPTCHA přehrává generované symboly pouze v angličtině.

Knihovna, která byla pro přehrávání textu použita, se jmenuje Microsoft Speech Object Library a je již obsažena v komponentech samotného .NET Framework SDK. Tato komponenta obsahuje velké množství funkcí pro zpracovávání zvuku, obsahuje funkce pro rozpoznávání řeči a je schopna zpracovat jako audio výstup vygenerované symboly, proto byla pro tento projekt ideálním řešením.

Vygenerovaný řetězec symbolů je nejprve rozložen na jednotlivé symboly, které ke kterým jsou následně přiřazena názvosloví (například pro 1 je to „one"). To transformovaný text je odeslán do metody, která je schopna díky enginu Text To Speech, převést text na audio nahrávku a ta je následně přehrána po stisku tlačítka.

9. Implementace do webové aplikace

Jak již bylo zmíněno několikrát v předchozím textu, samotný návrh je proveden pomocí *Custom control*, každá ze tří variant je v podstatě samostatný webový prvek, který má plnit kontrolní funkci, v převážné většině případů, zaměřenou na antispamovou kontrolu. Programová část každého z těchto prvků se skládá z několika části, které je potřeba pro úspěšnou implementaci a bezproblémovou funkčnost správně zaregistrovat. Pro CAPTCHU generující zdeformovaný text a pro CAPTCHU generující obrázek, platí ještě další speciální potřeba registrace v konfiguračním souboru stránky, ale to bude uvedeno až později.

Každý samostatně vytvořený serverový prvek, musí být do webové aplikace přidán jako funkční knihovna DLL, to lze provést následujícím způsobem.

Přidání *Custom control* do *ToolBoxu*

Po zkompilování *Custom control*, je vytvořena DLL knihovna, kterou lze jednoduchým způsobem přidat do *ToolBoxu* v Microsoft Visual Studiu, či do Visual Web Developer.

1. V ***ToolBox*** menu se klikne pravým tlačítkem a vybere ***Choose Items***
2. Zobrazí se menu ***Choose Toolbox Items***, kde můžeme prvky vybírat a přidávat dle libosti, ovšem náš vytvořený prvek se musí vyhledat a to prostřednictvím tlačítka ***Browse***
3. Následně už jen prvek vybereme, klikneme na ***OK*** a prvek se automaticky přidá
4. Po úspěšném přidání *Custom Control*, se objeví ikona s příslušným názvem v ***ToolBoxu***

Tento způsob je velmi snadný, ovšem platí jen za předpokladu, že využíváme služeb vývojářských programů od společnosti Microsoft. Ovšem platforma .NET umožňuje vytváření aplikací prakticky v jakémkoliv textovém editoru, kde se už tento způsob využít nedá. V případě, že použijeme textový editor, je implementace sejná jako je uvedeno v kapitole 7, jen musí být knihovna příslušného prvku nahrána v adresáři BIN dané webové aplikace.

Registrace v konfiguračním souboru stránky

Jedním z prvotních cílů .NET Framework byla podpora instalací pomocí příkazu XCOPY – to znamená instalace aplikací jejích zkopírováním do nějakého adresáře na pevném disku a možností odinstalovat pouhým smazáním jejich souborů a adresářů. To znamená, že řízené aplikace se ukládají do textových souborů XML. Konfigurační soubor, neboli Web.Config, je soubor XML, v němž si aplikace ASP .NET ukládají konfigurační údaje.

Tato potřeba registrace v Web.Config platí jen pro CAPTCHU generující zdeformovaný text a pro CAPTCHU generující obrázek. Jelikož tyto dvě varianty mají v sobě integrovanou funkci, pro práci s obrázky a se zvukem, prostřednictvím HTTP Handleru je nutno tyto položky zaregistrovat.

HTTP Handler se používá na zpracování individuálního požadavku od klienta (v tomto případě internetového prohlížeče) . Handler umožňuje ASP .NET Frameworku zpracovat požadavky na konkrétní URL nebo na skupinu URL adres přesně určeným způsobem. Což v podstatě znamená, že pro načítání obrázků a audia není potřeba, aby server zbytečně načítal celé stránky, ale pouze a jen ty části, kterých se to týká.

Registrace těchto Handlerů je zobrazena níže nejdříve pro CAPTCHU generující zdeformovaný text a následně pro CAPTCHU generující obrázek. Tento způsob registrace se musí přesně dodržet, jinak by vzniklo riziko, že stránka nebude správně fungovat.

Každý HTTP handler musí obsahovat tři atributy. Atribut *verb* určuje, jaký typ požadavku bude handler zpracovávat. V tomto případě *GET*. Atribut *path* může obsahovat relativní URL cestu, která bude zpracovaná tímto handlerem, anebo jeho masku (například "*.jpeg"). Posledním atributem je *type*, určuje "úplnou cestu" k třídě, která představuje handler. Skládá se z plného názvu. Konfigurace pak vypadá následovně.

Web.Config CAPTCHA generující zdeformovaný text

```
<httpHandlers>
      <add verb="GET" path="generatedImage.axd"
        type="Captcha1.CaptchaHandler" />

      <add verb="GET" path="generatedAudio.axd"
          type="Captcha1.AudioHandler"/>
</httpHandlers>
```

Web.Config CAPTCHA generující obrázek.

```
<httpHandlers>
      <add verb="GET" path="generatedImageControl.axd"
      type="ImageControl.ImageControlHandler" />

      <add verb="GET" path="generatedAudioImageControl.axd"
          type="ImageControl.AudioHandler"/>
</httpHandlers>
```

```
<appSettings>
    <add key="PathToImg" value="Cesta k obrázkům" />
</appSettings>
```

CAPTCHA generující obrázek obsahuje navíc ještě určení cesty k adresáři s obrázky, které se budou náhodně generovat. Tato je zaregistrovaná v oddílu `appSetings`, kde se definují datové položky specifické pro aplikaci.

Po provedení těchto nezbytných náležitostí by měly být kontrolní prvky úspěšně implementovány ve webové aplikaci. Ovšem to ještě nezaručuje jejich plnou funkci, i když se prvky budou zobrazovat i náhodně generovat svůj obsah, stále není implementována funkce, která nám zjistí, zda proběhla kontrola v pořádku.

V zdrojovém kódu serverových ovládacích prvků CAPTCHA je implementováno rozhraní `IValidator`, které zajišťuje samotnou kontrolu vstupních dat na straně uživatele a porovnává ji předem definovaným způsobem. V tomto případě se jedná o kontrolu, která zajišťuje úspěšné zvládnutí CAPTCHA. V každém ze tří případů vytvořených prvků je validace prováděna částečně odlišným způsobem, ale to není podstatné, důležitá je posloupnost metod, jakými `IValidator` prochází. U každého ze tří typů se jedná o stejný postup.

```
IValidator Members

protected bool IsVisible()
public void Validate()
protected virtual bool EvaluateIsValid()
public bool IsValid()
```

1. V metodě `IsVisible()` se zjišťuje, zda je prvek skrytý, návratová hodnota je False, či nikoliv a návratová hodnota je True
2. Metoda `Validate()` porovnává hodnotu z metody `IsVisible()`, pokud je tato hodnota False, tak metoda `Validate()` odeěle hodnotu true do metody IsValid(), v opačném případě, kontrola pokračuje prostřednictvím metody `Validate()`
3. V metodě `EvaluateIsValid()` probíhá samotná kontrola CAPTCHA vůči vstupním datům od uživatele, pokud jsou definované podmínky splněny (například pokud symboly od uživatele odpovídají vygenerovaným symbolům) je návratová hodnota True, která je odeslána do metody `IsValid()`
4. V poslední časti validátoru, v metodě `IsValid()` se porovnávají vstupní hodnoty, které přišli od předchozích činitelů, pokud je vše v pořádku je odeslána hodnota True, která je zpracována v samotném kódu stránky viz dále

V každé ze tří vytvořených aplikací, je integrována metoda `IsValid`, která zjišťuje, zde je kontrolní podmínka splněna. Pokud ano je navrácena hodnota `True`, pokud ne je vrácená hodnota `False`. Příklad použití je uveden na další straně, jedná se jen o ukázku, použití může být různé.

```
protected void Button1_Click(object sender, EventArgs e)
    {
       if (!this.IsValid)
        {
            //zde muze byt funkce, ktera se vykona v pripade hodnoty False
            return;
        }

         //funkce, ktera se vykona pukud je captcha podminka uspesne splnena

    }
```

Tento popis by měl být dostačující k tomu, aby každý uživatel byl schopen si do vlastní internetové stránky vložit a případně nastavit některou, ze tří variant CAPTCHA.

10. Testování

Nedílnou součástí všech projektů by mělo být testování. Ať už je to testování hmotnosti při výrobě automobilu nebo funkčnost mobilního zařízení, musí probíhat kontinuálně s vývojem daného produktu.

Co se týče testování webových aplikací, probíhají zde především testy funkčnosti, validity, korektnosti a správné funkce databáze.

10.1 Funkčnost

Tento test má odhalit případné neošetření výjimek nebo chybu ve zdrojovém kódu. Test probíhal kontinuálně a bylo zjišťováno, zda se prvky zobrazují tak, jak mají, popřípadě se zobrazí adekvátní chybová hláška. Chybová hláška zobrazuje obecné údaje, tak aby uživatel nemohl odhalit slabé místo aplikace.

10.2 Validita

Testování validity zdrojového kódu probíhalo pomocí programu Visual Studiu 2005, při každé kompilaci stránky. Výsledek kontroly ukázal, že ve zdrojovém kódu nejsou žádné chyby.

ZÁVĚR

Tento projekt se zabývá v dnešní době velmi rozšířeným fenoménem nevyžádaných zpráv spamem, v různých jeho formách a různých prostředích. V úvodu práce, tedy v teoretické části, jsem se věnoval problematice spamu. Rozebral jsem jeho historii, možné formy, v kterých se nejběžněji vyskytuje a jak na něj pohlížejí státní orgány z pohledu legislativy. Tato část je ve skutečnosti natolik obsáhlá, že není v mezích tohoto projektu se ji plně věnovat, ovšem pro tyto účely jsou uvedené informace dostačující.

Ve druhé části teorie jsem se zabýval problémem, který je se spamem úzce spojen – spambot. Samotné boty, patří mezí jedny z největších problémů internetové komunikace, kterým musí denně čelit miliony lidí. Spambot je nejpoužívanější metodou k rozesílání spamu, o čemž svědčí kapitola, která zobrazuje několik příkladů těchto robotů ve virtuální podobě. Tyto dvě kapitoly slouží pouze k tomu, abychom si udělali představu, s jak velkým problém se potýká dnešní doba
informačních technologií.

Samotnou náplní projektu je těmto problémům se umět bránit a úspěšně vzdorovat proti neustálím útokům ze strany spammerů. Pro tento účel již bylo vymyšleno mnoho metod, které jsou více jiné méně účinné, právě v třetí části se touto problematikou zabývám. Jsou zde vypsány ty nejběžnější a nejpoužívanější metody jednoduššího rázu a také forma obrany, která je stěžejní pro tento projekt, CAPTCHA, což bylo náplní i praktické části.

Návrh antispamových ochran byl realizován v prostředí ASP .NET prostřednictvím jazyka C#, tato kombinace byla ideální, protože jazyk C# byl přímo vytvořen pro platformu .NET Framework. Tato technologie je v dnešní době již standardem, díky kterému je možné vytvářet internetové aplikace na vysoké úrovni. Z tohoto důvodu serverové prvky CAPTCHA, mají velkou možnost uplatnění na internetových stránkách založených na této technologii.

V praktické časti, jak už jsem zmínil, jsem navrhl tři plnohodnotné serverové prvky různých forem CAPTCHA. Nejpoužívanější v dnešní době, je klasická CAPTCHA generující zdeformovaný text, která má proto i největší variabilitu nastavení. Další dvě formy, CAPTCHA generující matematickou úlohu a CAPTCHA generující obrázek nejsou používány příliš často což je do značné míry i výhoda, protože tento fakt zvyšuje pravděpodobnost jejich úspěch při obraně proti spamu. Všechny tři prvky jsou plně schopny okamžité implementace do internetových stránek, aniž by tvůrce webu musel zasahovat do zdrojového kódu prvku. Což jsem předvedl na ukázce internetových stránek, které jsou přiloženy na CD.

Domnívám se, že cílů práce definovaných v úvodu a zadání bylo úspěšně dosaženo.

REFERENCE

[1] *Spam (electronic) : Origin of the term "spam"* [online]. [2005] [cit. 2008-11-05]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Spam_(electronic)>.

[2] KANTOLA, Raimo . *Peer to Peer and SPAM in the Internet*. [s.l.], 2004. 161 s. Helsinki University of Technology. Seminární práce. Dostupný z WWW: <http://portal.acm.org/>.

[3] A Monthly Report – November 2008. *The State of Spam* [online]. 2008 [cit. 2008-12-10]. Dostupný z WWW: <http://www.symantec.com/business/theme.jsp?themeid=state_spam>.

[4] PETERKA, Jiří. *Archiv článků a přednášek Jiřího Peterky: Spamdexing* [online]. 1. 1. 1998 [cit.2008-11-05]. Dostupný z WWW: <http://www.earchiv.cz/a98/ a801k180.php3>.

[5] *Forum spam* [online]. [2005] [cit. 2008-11-10]. Dostupný z WWW: <http:// en.wikipedia.org/wiki/Forum_spam>.

[6] *Messaging spam* [online]. [2006] [cit. 2008-11-10]. Dostupný z WWW: <http:// en.wikipedia.org/ wiki/Messaging_spam>.

[7] GOMEZ HIDALGO, JM, BRINGAS, GC, FRANCISCO CARRERO GARCÍA, EPS. *Proceedings of the 2006 ACM symposium on Document engineering*. New York, USA : ACM, 2006. Dostupný z WWW: <http://portal.acm.org/>. ISBN:1-59593-. Content Based SMS Spam Filtering, s. 107-114.

[8] INTERNATIONAL TELECOMMUNICATION UNION, (ITU). *WORLD INFORMATION SOCIETY REPORT 2007*. [s.l.], 1.5.2007. 175 s. Report. ISBN 92-61-11.

[9] *The CAN-SPAM Act: Requirements for Commercial Emailers* [online]. 1. 4. 2004 [cit. 2008-11-12]. Dostupný z WWW: <http://www.ftc.gov/bcp/edu/pubs/business/ecommerce/ bus61.shtm>.

[10] *Úřad pro ochranu osobních údajů* [online]. [2000] [cit. 2008-12-18]. Dostupný z WWW: <http://www.uoou.cz/>.

[11] MLEJNEK, Miroslav. Spam – historie a zákon. *Internet* [online]. 24. 10. 2007 [cit. 2008-11-12]. Dostupný z WWW: <http://www.swmag.cz/153/spam-historie-a-zakon/>.

[12] Zakázání přístupu vyhledávačům. *Jak psát web* [online]. 2004 [cit. 2008-11-18]. Dostupný z WWW: <http://www.jakpsatweb.cz/robots-txt.html>.

[13] *Interval.cz - wedesign a e-komerce denně* [online]. [2001] [cit. 2008-11-12]. Sada článků týkajících se spamu. Dostupný z WWW: <http://interval.cz/>. ISSN 1212-865.

[14] von AHN, Luis, BLUM, Manuel, LANGFORD , John. *Communications of the ACM*. [s.l.] : [s.n.], 2004. Dostupný z WWW: <http://portal.acm.org/>. ISSN:0001-078. Telling humans and computers apart automatically, s. 56-60.

[15] *Paradoxy umělé inteligence: Turingův test 50 le poté* [online].29.03.2005 [cit.2008-11-14] Dostupný z WWW: <http://www.transhumanismus.cz/ library.php?source=turing50>.

[16] LUIS, Von Ahn, HOPPER, Nicholas J. , LANGFORD, John. *Lecture Notes in Computer Science : Advances in Cryptology*. 2003th edition. [s.l.] : [s.n.], 2003. Dostupný z WWW: <http://www.springerlink.com/>. ISBN 978-3-540-140. CAPTCHA: Using Hard AI Problems for Security , s. 294-311.

[17] *E - bariéry osob se zdravotním postižením* [online]. 2002-2008 [cit. 2008-11-16]. Dostupný z WWW: <http://www.sons.cz/docs/e-bariery/>.

[18] TROELSEN, Andrew .*C# a .NET 2.0 profesionálně*.[s.l.] Zoner Press, 2006.1200 s. ISBN 80-86815-42-0

[19] MACDONALD, Matthew , SZPUSZTA, Mario . *ASP.NET 2.0 a C# : tvorba dynamických stránek profesionálně* . [s.l.], Zoner Press , 2006. 1376 s. ISBN 8086815382

[20] *Understanding ASP.NET View State* [online]. 2004 [cit. 2008-11-15]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/ms972976.aspx>.

[21] PROSISE, Jeff . *Programování v Microsoft .NET : Webové aplikace v C#, ASP.NET a .NET*. [s.l.], Computer Press, 2003. 736 s. ISBN 80-7226-879-1.

[22] ROBINSON, Simon , et al. *C# Programujeme profesionálně*. [s.l.], Computer Press, 2003. 1160 s. ISBN 80-251-0085-5.

[23] HOZÍK, Martin . *Textová 3D CAPTCHA* [online]. 2008 , 7. června 2008 [cit. 2009-05-20]. Dostupný z WWW: <http://doublethink.cleverweb.cz/22-textova-3d-captcha>.

Výpis použitých zkratek

ASP - Active Server Pages
ASP .NET - Active Server Pages.NET
ADO .NET - ActiveX Data Objects.NET - představuje soubor tříd pro přístup k datům v technologii *.NET*
API - Application Programming Interface - rozhraní pro programování aplikací
ARPANET - Advanced Research Projects Agency Network - předchůdce Internetu
ASCII - American Standard Code for Information Interchange - kódovou tabulku která definuje znaky anglické abecedy, a jiné znaky používané v informatice.
CAN-SPAM - Controlling the Assault of Non-Solicited Pornography and Marketing Act – americký zákon pro kontrolu spamu
CAPTCHA - Completely Automated Public Turing Test To Tell Computers and Humans Apart - plně automatický veřejný Turingův test k odlišení počítačů a lidí
CTS - Common Type System - společný typový *systém*
CLS - Common Language specification - sada základních pravidel, která by měl splňovat každý jazyk vyhovující specifikaci CLI
CIL - Common Intermediate Language - společný mezijazyk pro zpracování nižších programovacích jazyků
-
COBOL - COmmon Business Oriented Language – programovací jazyk pro obchodní a později i databázové aplikace.
CGI - Common Gateway Interface - protokol pro propojení externích aplikací s webovým serverem
CLR - Common Language Runtime – slouží jako vykonávací jádro pro řízeneé aplikace
DoS - Denial Of Service - technika útoku na internetové služby nebo stránky
DLL - Dynamic Link Library - spustitelný modul obsahuje procedury, funkce,data nebo zdroje
DHTML - Dynamic HTML
DIVTCHA - DIV Turing Test to tell Computers and Humans Apart
FCL - Framework Class Library – knihovna tříd rámce .NET
HTTP - Hypertext Transfer Protocol - protokol určený k přenosu WWW stránek
HTML - HyperText Markup Language – jeden z jazyků pro vytváření stránek v prostředí WWW
IM - Instant Messaging – internetová komunikace pomocí textových zpráv
ICQ - I Seek You - software pro instant messaging využívající protokolu OSCAR
IRC - Internet Relay Chat - jedna z prvních možností komunikace v reálném čase po internetu
IP - Internet Protocol - protokol používaný na síťové vrstvě OSI modelu
IIS server - Internet Information Services -
JIT - Just In Time - speciální metoda překladu pro urychlení běhu interpretovaných programů.
MMS - Multimedia Messaging Service – multimediální zpráva

MSIL - Microsoft Intermediate Language - jazyk, který je nezávislý na procesoru
Mac OS X - operační systém pro počítače Macintosh
OCR - Optical Character Recognition - **optické rozpoznávání znaků**
OOP - Object Oriented Programming – objektově orientované programování
PC - Personal Computer
P2P - Peer To Peer – architekura sitě klient-klient
SMS - Short Message Service – krátká textová zpráva
SMTP - Simple Mail Transfer Protocol - je internetový protokol určený pro přenos zpráv elektronické pošty
SONS - Sjednocená Organizace Nevidomých a Slabozrakých
TCP - Transmission Control Protocol – jeden ze základních internetových protokolů, představuje transportní vrstvu.
USENET - spřežka "user" a "network"
UCE - unsolicited commercial e-mail - hromadné nevyžádané komerční e-maily
UBE - Unsolicited Bulk E-mail - hromadné nevyžádané e-maily
UDP - User Datagram Protocol - zpřístupňuje nespolehlivé a nespojované,ale rychlé a efektivní přenosové služby síťové vrstvy
UI - User Interface – uživatelské rozhraní
URL - Uniform Resource Locator - unifikovaná metoda jednoznačné lokalizace zdroje na Internetu
WWW - World Wide Web - „celosvětová pavučina“
XML - Extensible Markup Language - rozšiřitelný značkovací jazyk

SEZNAM OBRÁZKŮ

Přílohy

1. Ochrana proti spamu JavaSciptem

ukázka kódu:

```
<form action="zpracuj_komentar.php" method="post">
<!-- zde jsou normální pole formuláře -->
<p id="spamprotirobotum">
Ochrana proti spamu. Kolik je dvakrát tři?
<input type="text" name="robot" value="" id="protirobotum">
</p>
<script>
/* tento skript vloží číslici do příslušného políčka automaticky */
document.getElementById("protirobotum").value="6";
document.getElementById("spamprotirobotum").style.display = "none";
</script>
<p><input type="submit" value="Odeslat"></p>
</form>
```

Serverový skript, který zpracovává data z formuláře, pak jenom zkontroluje, zda mu byla odeslána správná hodnota:

```
if ($_POST["robot"]*1==6)
// pridej prispevek
```

2. Obsah přiloženého CD

Obsah.txt	– obsah CD
Antispam ochrana.pdf	– Vlastní text BP
Captcha	– adresář obsahující aplikaci včetně zdrojového kódu CAPTCHA generující zdeformovaný text
ImageControl	– adresář obsahující aplikaci včetně zdrojového kódu CAPTCHA generující obrázek
MatchaControl	– adresář obsahující aplikaci včetně zdrojového kódu CAPTCHA generující matematickou úlohu
WebSite	– webové stránky obsahující příklad použití pro jednotlivé varianty
Obrázky	– adresář obsahující obrázky, které slouží jako podklad pro ImageControl
CustomControl	– adresář obsahující jednotlivé serverové prvky zkompilované do formátu *.dll