Posudek na práci

Antispamming na principu výběru vyžádaných zpráv

Jan Nepožitek

Předmětem práce mělo být dle zadání prozkoumání možností řešení boje proti spammingu na základě výběru chtěných (vyžádaných) zpráv a základní implementace takového systému. Oproti většině běžně používaných systémů proti spamu filtrováním nevyžádaných zpráv tak měla být prozkoumána možnost opačného přístupu.

V úvodu práce autor konstatuje, že (cit.) „Prvním cílem této práce bylo stanovit, jak lze vlastně chápat zadání." Domnívám se ale, že autor zadání buď nepochopil, nebo se rozhodl jej zásadně změnit: z textu práce a z navrženého řešení totiž plyne, že autor téma ze zadání změnil na metodu dočasného odmítání potenciálně nevyžádaných zpráv (pokud ale bude zpráva poslána znovu, již nebude odmítnuta, ať je její obsah jakýkoli).

Provedená analýza stručně sepisuje známé možnosti boje proti spamům. Návrh systému pak volí kombinaci Bayesovských filtrů a Graylistingu, což jsou zcela běžné základní metody filtrování nevyžádaných zpráv, nikoli výběr chtěných zpráv. V analýze uvedený systém Challenge-response, který by mohl být použit právě pro výběr chtěných zpráv, je však zamítnut, aniž by bylo navrženo alespoň nějaké jeho možné využití.

Návrh autorova řešení neobsahuje žádné nové myšlenky. Spočívá v použití existujících komponent a jejich doplnění vlastním jednoduchým filtrem kontrolujícím, zda email propustit (pomocí převzatého modulu pro Bayesovské filtrování) nebo jej podrobit procesu Graylistingu, což je opět standardní řešení. Autor navíc tvrdí, že toto řešení je pro použitý mailserver (převzatý) transparentní. S tím ale naprosto nesouhlasím! Pokud autorův filtr nejprve převezme celou zprávu a pak ji teprve případně předá k dalšímu zpracování, pak tím vylučuje možnost, že přebírající mailserver již při zpracování hlavičky SMTP (před fází DATA) zjistí, že zprávu nepřijme např. na základě závadné emailové adresy či jiných znaků a ukončí komunikaci dříve se všemi důsledky, které z toho mohou plynout. Navíc, pokud by k tomu v autorově řešení došlo a mailserver chtěl komunikaci ukončit, pak to není bráno v potaz, protože zpráva je stejně předávána dále celá najednou, tedy včetně části, kterou už mailserver dále zpracovávat nechce a jejíž celé přijetí dříve může znamenat velmi netriviální objem dat. Nedomnívám se proto, že by řešení dobře mohlo v reálném prostředí splnit deklarované vlastnosti.

Podle mého názoru tedy návrh nedostatečně využívá možnosti tématu a netvoří dobré výsledné dílo.

Praktická část práce je tvořena dvěma částmi: serverem zpracovávajícím Graylisting a proxy, která filtruje nevyžádanou poštu mezi přijetím (cizím) mailserverem a jejím zpracováním jinou částí mailserveru.

První část (AspamDaemon) tvoří dva skripty v jazyce PHP, které jsou z programátorského hlediska provedeny naprosto katastrofálně jak návrhem, tak implementací, a několik tříd v jazyce Java pro obsluhu požadavků vzniklých použitím dříve zmíněných PHP skriptů (jedna ručně napsaná třída a několik vygenerovaných), což by ale bylo možné úspěšně vyřešit jednodušším způsobem – přímou akcí z PHP.

Filtrující proxy tvoří dva základní moduly (Aspam a AspamTh), jejichž úkolem je kontrola, zda přišlý mail smí být propuštěn dále nebo má projít procesem Graylistingu. Funkce obou částí je velmi jednoduchá, na úrovni velmi krátké zápočtové úlohy. Protože práce zřejmě není implementačního typu, nemuselo by to vadit, pokud by ale i takto malé dílko nebylo zatížena zásadními chybami (viz příloha).

Vytvořené řešení ani v návrhu ani v implementaci nepočítá s revokací rozhodnutí o zařazení emailových adres do whitelistu a blacklistu (je nutný případný ruční zásah do XML souboru se seznamy).

Instalace (popis viz str. 50 a následující) vyžaduje kromě ručního nastavování volacích parametrů dokonce i ruční vytváření spouštěcích skriptů a ruční opravování některých souborů. Také je potřeba ručně souborům nastavovat přístupová práva.

Implementační část práce tedy považuji za nedostačující.

Schopnost dobře pracovat s literaturou autor neprokázal. V oboru existuje ohromné množství zdrojů (teoretické články a publikace i praktická řešení), autor však odkazuje pouze základní odkazy na dokumentaci jím použitých produktů třetích stran a jeden de facto standard (RFC 2821, základní pro SMTP). Navíc literatura je v seznamu na konci práce uvedena bez jakéhokoli rozumného pořadí (ani podle pořadí výskytu v textu, ani setříděná např. podle abecedy).

V textu práce chybí jakákoli zmínka o ověření, zda zvolené a implementované řešení v reálném použití splňuje požadavky – výběr pouze chtěných zpráv. Autor neuvádí, zda jeho dílo bylo kdy v provozu, byť testovacím.

Kapitola 5.3 „Závěr" je spíše novinářskou deklarací, než závěrem práce předkládané jako výsledek studia.

Vzhledem k uvedeným argumentům doporučuji, aby práce byla zásadně přepracována, a to ve všech částech.

Praha, 12.9.2007

RNDr. David Obdržálek

Příloha – poznámky k textu práce a k implementaci:

kap. 2.2, odstavec „Záměrné zpomalení SMTP komunikace":
Není pravda, že toto řešení vyžaduje nový protokol pro posílání emailu. SMTP komunikace se zcela běžně zpomaluje použitím standardních prostředků, které vyhovují tomuto protokolu.

kap. 2.2, odstavec „Signatury"
Součástí zprávy jsou i hlavičky, které je také třeba uvažovat a které na obsahu vlastních dat zprávy nijak nezáležejí.

kap. 2.2, odstavec „Stížnosti poskytovateli ISP, zákony proti spamu, ..."
Konstatování, že „tyto a další možnosti jsou mimo zaměření této práce, takže zde nebudou popsány" je pokus o obhájení, proč nebyly zpracovány. Dle mého názoru i tyto metody patří do boje proti spamu, takže v tomto přehledu neměly chybět.

kap. 3.2 „Použité součásti"
Autor pro své dílo v poznámce 14 stanovuje podmínku, aby instalovaná MTA komunikovala přes standardní vstup a výstup. Chybí alespoň naznačení, zda to je obvyklý postup, anebo zda to je specifický způsob řešení převzatého poštovního serveru qmail.

kap. 4.2.2. „Formát dat"
Pro každou emailovou adresu příjemce jsou data uložena v jednom souboru, kde jsou dohromady jak víceméně statická data – nastavení, tak dynamicky se měnící data – seznamy adres, což znesnadňuje správu.

Na přiloženém CD není žádný soubor readme či podobný, který by umožnil orientaci v adresářové struktuře.

Připomínky k vytvořenému dílu, část poskytující funkčnost Greylistingu:

Navržená architektura je zbytečně komplikovaná: žádost je přijata PHP skriptem, uložena do souboru. Na serveru běží démon, který ve stanovených intervalech kontroluje adresář se žádostmi, které případně zpracuje a pošle email původnímu příjemci (v případě žádosti o doručení mailu) anebo přidá záznam do seznamu povolených adres (v případě žádosti příjemce o povolení). Toto by celé mohlo být snadno řešeno již v PHP, kde jsou pro všechnu takovou funkčnost dostatečně silné prostředky, a to jak pro posílání emailu, tak pro práci se soubory.

Implementace:
skript request.php:
- pro tvorbu zaručeně unikátního jména souboru existuje přímo knihovní funkce, není potřeba ručně tvořit pomocí generování náhodných znaků ve funkci rand_chars (což z principu není zaručená metoda). Navíc zde použitá metoda by platformách nepoužívajících znakovou sadu ASCII nefungovala.
- nešikovné použití funkce echo ve funkci writeform donutilo autora prefixovak každé uvozovky v textu. Pro takové účely se v PHP používá konstrukce „heredoc". Také není nutné rozvíjet proměnné mimo řetězec, při použití řetězce uzavřeného v uvozovkách je rozvíjení automatické
- funkce check_email kontrolující správnost zadané emailové adresy nekontroluje adresu podle RFC
- test délky komentáře ve funkci check_text je sémanticky špatně uzávorkován, což vzhledem k typové volnosti jazyka PHP nezpůsobí syntaktickou chybu, ale nekontroluje to správně délku zadaného řetězce
- skript (a ostatně pak ani AspamDaemon) naprosto není odolný proti běžným technikám průniku typu code / tag injection – útočníkem zadaný text z html formuláře je přímo vložen do souboru anebo použit pro vytvoření opakovaného formuláře.

skript confirm.php:
- útočník může snadno přesunout libovolný soubor na libovolné místo zadáním vhodného jména souboru
- jedinou funkcí skriptu je přesun souboru zadaného jménem z jednoho adresáře do druhého. k přesunu souboru je v PHP standardní funkce rename (místo toho autor celý soubor přečte do proměnné, zapíše do druhého souboru a původní soubor smaže).