ErlGrey –

replizierte Mailfilterung in Erlang

Ignatios Souvatzis

Institut für Informatik
Universität Bonn
ignatios@cs.uni-bonn.de
The NetBSD Project
is@netbsd.org

GUUG–Frühjahrsfachgespräch 2007

universitätbonn

Gliederung

Das Problem

- Spam hat eine Häufigkeit erreicht, wo herausfischen von Nutzmails ohne elektronische Hilfe schwierig wird
- Das Problem ist konzeptionell schwierig: *Des einen Uhl ist des anderen Nachtigall.*
- Der Datenbestand solcher elektronischer Hilfsmittel ist leider wertvoll geworden.
- Viele elektronische Hilfsmittel werden mit der Zeit unwirksam.

Feste Sperrlisten für Einlieferer

- intern
  - nie vollständig
  - Spammer haben längst gelernt, ungesicherte Rechner anderer Leute als Relays zu benutzen
- von externen Dienstleistern (RBL, ORB, ...)
  - theoretisch eine gute Idee...
  - aber Gefahr von Sabotage

Feste Sperrlisten für Absender

- intern
  - nie vollständig
  - Spammer haben längst gelernt, Absender zu fälschen und sogar mit dem jeweiligen Empfänger zu korrelieren.
- von externen Dienstleistern
  - siehe oben
  - + Gefahr von Sabotage

Feste Sperrlisten für Betreff

- intern
  - nie vollständig
  - Gefahr von Falschpositiven
  - Spammer haben längst gelernt, den Betreff zu fälschen und zeitweise mit dem jeweiligen Empfänger zu korrelieren. (Inzwischen machen sie sich diese Mühe nicht mehr.)
- von externen Dienstleistern
  - siehe oben
  - + Gefahr von Sabotage

Benutzerdefinierbare feste Filter

z.B. procmail, deliver, ...

- umgehen das Nachtigallenproblem auf Benutzer- (statt auf System-)ebene , umgehen dadurch rechtliche Probleme
- mächtig
- langsam (oft muss ganze Email ausgewertet werden)

Filterung nach Inhalt

Eine Reihe von Spamfilterlösungen beinhaltet eine selbstlernende Komponente. Die Benutzerin braucht nur fehlklassifizierte Post umzuklassifizieren, und das Filter passt seine Musterdatenbank entsprechend an.

- umgehen das Nachtigallenproblem auf Benutzer- (statt auf System-)ebene , umgehen dadurch rechtliche Probleme
- mächtig
- langsam (oft muss ganze Email ausgewertet werden)
- ... und natürlich haben Spammer gelernt, damit umzugehen (seit ca. einem halben Jahr massiv!)
  - zufällig ausgewählte Lorem-Ipsum-Texte aus „dem Netz“ zur Täuschung des Filters
  - Nutzbotschaft in einer Bilddatei

Greylisting: warum?

Grundgedanke: ein großer Anteil des Spams wird von Spezialmaschinen eingeliefert, die

1. wechselnde IP-Adressen haben
2. keinen Mailspool benutzen (90% Erfolg von 1,000,000 Adressen sind immer noch 900,000)
3. und deswegen keine transienten Fehlercodes (4xx) auswerten
4. im Zuge von Pauschaltarifen für Privatanwender verlieren 1. und 2. an Bedeutung.

Greylisting: wie?

- unbekannte Absender- Empfänger- Einlieferer- Kombinationen über 4xx-Codes verzögern
- bei erneutem Verbindungsversuch erst annehmen, wenn Mindesthaltezeit (60s - 1800s) vergangen
- da zur Verbindungszeit, ist der Fehlercode billig (eine Textzeile / Mailkopf und -körper nicht gelesen)
- da zur Verbindungszeit, erreicht der Fehlercode den wirklichen Einlieferer und nicht den (pot. gefälschten) Absender
- theoretisch kein Verlust von Nutz-E-mail
- theoretisch nur Verzögerung von Nutz-Email von Unbekannten

Greylisting: Erfahrungen

1. vor ca. zwei Jahren 90% - 95% Wirksamkeit
2. Tendenz zu Greylisting-festen Werbemails nimmt zu
3. Greylisting muss auf allen MXen aktiv und idealerweise synchron sein, sonst kommt der Müll zur Hintertür 'rein.

Thesen zur Mailfilterung

- Maßnahmen müssen auf allen MXen synchron laufen, um wirksam zu sein
- dynamische aufgebaute Datenbestände sind zwar keine Nutzdaten, repräsentieren dennoch unmittelbar vermiedene verlorene Arbeitszeit, unmittelbar gesparte Virenscannerzyklen etc. und sollten daher gesichert werden
- neue Strategien sollten ohne Dienstunterbrechung einzuführen sein, da alle Maßnahmen nur begrenzte Lebensdauer haben

Postfix-Policy-Filter

- smtpd baut eine TCP/IP- oder `AF_LOCAL`-Verbindung zum Policy-Server auf.
- Protokoll besteht aus Textzeilen der Form Schlüssel=Wert
- Anfrage wird mit einer Leerzeile beendet
- Antwort ebenfalls Schlüssel=Wert (1 Zeile)
- Antwort wird mit einer Leerzeile beendet
- Protokoll- und Adressbehandlung bleibt beim smtpd
- alle ohne Inhaltsanalyse möglichen Informationen verfügbar
- alle smtpd-Aktionen möglich, in der Regel:
  - dunno (weitere Regeln entscheiden)
  - defer_if_permit (falls weitere Regeln erlauben würden, verzögern

Spracheigenschaften [Armstrong 2003]

- deklarative Sprache
- Modulkonzept
- Codeersetzung im Betrieb möglich
- leichtgewichtige parallele Prozesse
- Prozesskommunikation über Interpreter- und Hostmaschinengrenzen hinweg.
- teilweise Compiler auf Maschinencode
- Open Telecom Platform (OTP) enthält viele Bausteine für komplexe Applikationen, z.B. generische Server, eine verteilte Datenbank,
- Webserver YAWS

Die Datenbank mnesia

- in Erlang integriert: keine Datenumwandlung nötig
  - Strings werden komprimiert gespeichert.
- soft realtime
- Transaktionen für beliebigen Erlang-Code
- ortsunabhängiger Zugriff
- Knoten können `disc_only` , `disc` (gecacht), oder `ram` -basiert sein.

Skizze

- policy-Server-modul und Datenbank/Entscheider-Modul getrennt, dadurch potentiell neue Zugriffsmethoden möglich (Milter? Eigener smtpd?)
- Disk-Knoten der verteilten Datenbank sichern gegen Datenverlust
- MXe enthalten Disk- oder RAM-Knoten
- policy-Server in Erlang redet mit lokaler Datenbankkopie
- policy-Server wird von lokalem Postfix per TCP abgefragt

Code

. . . siehe Editorfenster, demnächst Homepage

. . . teils in Proceedings

Demonstration

. . . wenn das Netz funktioniert

Aufwand

Prototyp:

- Datenbank mit einer Tabelle mit einem Schlüssel
- Server selbstgebaut, statt Behaviour-Modul gen_server
- Zeilen selbst zusammengesucht, statt zeilenweises Einlesen
- 3–4 Manntage
- 198 Codezeilen
- Übergang zwischen Codeversionen großenteils unterbrechungsfrei

Zusammenfassung

- Mailfilterung sollte im Betrieb an neue Gegebenheiten angepasst werden können.
- Mailfilterung sollte auf allen Mail eXchangern synchron stattfinden.
- Erlang und OTP erlauben, ein solches System mit minimalem Zusatzaufwand zu implementieren und ohne Betriebsunterbrechung zu aktualisieren.

Ausblick

- automatisches Aufräumen (bisher nicht implementiert)
- Großbetrieb: RAM-Bedarf?
  evtl. `disc_only` – Tabellen
- Großbetrieb: Geschwindigkeit?
  evtl. lokale Tabellen als erste Stufe, nur bestätigte Kontakte in die permanente Tabelle

Weiterführende Literatur

J. Armstrong
Concurrent Programming in Erlang.
GUUG FFG (Proceedings), 2003.

Håkan Mattsson, Hans Nilsson and Claes Wikström,
Mnesia - A Distributed Robust DBMS for Telecommunications Applications,
International Workshop on Practical Aspects of Declarative Languages (PADL'99), LNCS 1551, pp. 152-163. San Antonio, USA, January 18-19, 1999.

URLs

- PostFix
  http://www.postfix.org/
- (Open Source Erlang/OTP)
  http://www.erlang.org/
- HowTO für Erlang mit SSL
  http://www.erlang.org/doc/doc-5.5.2/lib/ssl-3.0.12/doc/html/ssl_distribution.html
- Hier taucht demnächst Material über ErlGrey auf:
  http://theory.cs.uni-bonn.de/%7Eignatios/