Spam Control on the Web

Paul M. Winkler
PyGotham II
June 2012

Me: Paul Winkler
slinkp@gmail.com
programmer at OpenPlans
http://openplans.org

The Problem

- Comment / post spam evolves
- There will never be a silver bullet
- Old projects in maintenance mode get increasingly hard to keep spam-free

"An Internet service cannot be considered truly successful until it has attracted spammers."

- Rafe's law

http://rc3.org/2006/10/20/rafes-law/

The Story of Btwlzyq

Tracking a resourceful spammer on http://communityalmanac.org.

Existing Solutions

A brief taxonomy / survey

See also appendix B - prior art.

Existing Solutions: Form Modifiers

- Captcha (obtrusive)
- Other problem solving (obtrusive)
- Honeypot Fields (unobtrusive)

By design, only useful against bots.

Content Filtering Services

Remote APIs:

- Bayesian filters
- IP blacklist

Good: Unobtrusive. Large training set.

Bad: Network overhead. Reliabilty.

Content filters: Local

- Bayesian filters and IP blacklists, but also:
- IP throttling
- Other metadata filtering: link counting, admin users, …

Good: Unobtrusive. Low I/O overhead. Reliable.

Bad: Requires training.

Summary

- Lots of solutions
- None are sufficient alone
- Complement each other
- *n* solutions == *n* APIs
- *n* APIs integrated into *m* web apps == *aaargh*

Trac SpamFilter plugin - a flexible approach

- "all of the above" approach
  - 14 filters and 3 captchas
- extensible, easy to code new filters
- highly configurable
  - select filters
  - assign karma scores to filters (positive = good)
  - set minimum karma needed for posting

Trac SpamFilter plugin (cont'd)

- Records possible spam in database for moderation
- Moderation UI: rough but useful
- Moderation includes training
- Lots of tests

About the Trac plugin

http://www.cmlenz.net/archives/2006/11/managing-trac-spam

SpamAssassin has a similar multi-filter strategy, but is designed for use with email, not web:

http://wiki.apache.org/spamassassin/BlogSpamAssassin

Filters - remote

stopforumspam.py

akismet.py

blogspam.py

defensio.py

extlinks.py

httpbl.py

ip_blacklist.py

linksleeve.py

typepad.py

Filters - local

regex.py

bayes.py

extlinks.py

ip_blacklist.py

ip_regex.py

ip_throttle.py

session.py

Captcha

recaptcha.py

image.py (uses PIL)

expression.py ("what is three plus twelve") … looks unfinished

It only works with Trac.

Hamage Control!

- Goal: Decoupling SpamFilterPlugin from Trac
- Prototype
- http://github.com/slinkp/hamage

Hamage Control: modes of operation

- Python library API
- WSGI middleware
- Hybrid
- Native integration with every framework
  - Nooo.

Python API: Filters

```
class MyFilter(object):
    def test(self, req, author, ip):
        return (score, 'reason')
```

Positive score = ham, negative = spam.

Python API: FilterSystem

```
>>> from hamage.filter import FilterSystem
>>> config = {
...   'options': {'min_karma': 1},
...   'filters': ['hamage_extlinks']}
>>> config['options']['backend_factory'] =
'django_orm'
>>> filtersys = FilterSystem(config)
```

Python API: FilterSystem

```
>>> filtersys.strategies
[<hamage.filters.extlinks.ExternalLinksFilterStrategy object at ...>]
>>> filtersys.backend_factory
<class'hamage.backends.django_hamage.models.DjangoBackendFactory'>
```

Python API: FilterSystem

```
>>> from hamage.filter import Request
>>> req = Request.blank('/foo',
...remote_addr='10.20.30.40')
>>> filtersys.test(req, author='fred',
...      changes=[('Old content', 'New content')])
Traceback (most recent call last):
...
hamage.filter.RejectContent: Submission rejected as
potential spam
```

Python API: FilterSystem

```
>>> filtersys.min_karma = 0
>>> filtersys.test(req,
...     author='fred',
...     changes=[('Old content', 'New
content')])
(0, [])
```

Python API: FilterSystem

```
>>> lotsa_links = 'http://somewhere.org ' * 100
>>> filtersys.test(req, author='fred',
...      changes=[(None, lotsa_links)])
Traceback (most recent call last):
...
hamage.filter.RejectContent: Submission rejected as
potential spam  (Maximum number of external links per
post exceeded)
```

Python API: Registering filters

```
# Put entry points in your setup.py
setup(name='hamagecontrol',
   entry_points={
    'hamage_filters': [
      'hamage_extlinks =
hamage.filters.extlinks:ExternalLinksFilterStrategy',],
   'hamage_backends': [
      'django_orm
=hamage.backends.django_hamage.models:DjangoBackendFact
ory',],
...
```

WSGI Middleware

Request: Client POST → hamage (filtering) → application

Response: application response → hamage (form field and error message injection) → client

Demo

- Django running via wsgiref server, behind WSGI middleware

Wish List: RESTful web service?

- Use with any language
- Scale independently
- Would love to do this
- … later

Performance & Scaling

- Run cheapest filters first; allow them to short-circuit.
- Parallelize slow filters (eg. network IO)
  - How?
- Asynchronous operation

More about async

- Use case: speed
- Use case: moderation
  - Integration just got tougher
  - Feedback just got really hard
    - Don't bother?

More on logging

- Remember Btwlzyq?
- Consistency matters

Parting shot

buy my cheap replica rolexes

No page rank for you buddy

Appendix A. Links

- Hamage Control:https://github.com/slinkp/hamage
- Django integration demo code: https://github.com/slinkp/pygotham_hamage_demo
- These slides: https://github.com/slinkp/pygotham_hamage_demo/blob/master/pygotham2_hamage_slides.odp?raw=true
- Trac plugin: http://trac.edgewall.org/wiki/SpamFilter
- About WSGI: http://lucumr.pocoo.org/2007/5/21/getting-started-with-wsgi/
- About entry points: http://stackoverflow.com/a/9615473/137635

Appendix B. Prior Art

Python packages related to spam. Too many for one slide, see https://gist.github.com/2896944#file_prior_art.txt