Don’t Tread on Me: Moderating Access to OSN Data with SpikeStrip

Christo Wilson, Alessandra Sala, Joseph Bonneau*, Robert Zablit, Ben Y. Zhao

University of California, Santa Barbara
*University of Cambridge
Problem: People Want Your Data

from: Birthday E-Cards <cardbot@evil-cards.com>
to: Christo Wilson <bowlinearl@gmail.com>
date: Monday, January 31, 2011 at 12:01 AM
subject: You got a Birthday E-Card!

You've received a Happy Birthday E-Card from Sandi Klopacic!
Visit this link to view your card: http://www.evil-cards.com/
Big Data = Big Problems

- 450 Million
- 150 Million
- 70 Million

- Crawlers are actively collecting OSN data
  - Pete Warden crawled 210 Million profiles
  - RapLeaf crawls and sells OSN data to marketers
  - 80legs offers crawling as a service (150K pages per $1)

- Yes, this includes researchers
- Many more emerging threats!
How Can the Crawlers Be Controlled?

- Users are unwilling/unable to defend themselves
  - Privacy paradox – paranoia vs. laziness
  - Privacy conscious people (us) are the minority
  - OSNs have notoriously difficult privacy settings

- OSNs must protect their users… but how?
  - Existing anti-crawler mechanisms are ineffective
    - Designed in the mid ’90s
    - Broadband + botnets = massive, distributed crawlers
  - More on this later

- New technology is necessary

---

Image © The New York Times

9 pages
50 settings
170 options
Introducing SpikeStrip

- Project goal: defend against malicious crawlers
  - Seamless to end-users and beneficial crawlers
  - Minimal impact on web server
  - Compatible with existing technology

- SpikeStrip uses novel “link-encryption” primitive
  - Used to track and rate-limit users

- Implement and evaluate SpikeStrip
  - Can impose arbitrary slowdowns on rogue crawlers
  - Only imposes 7% performance overhead on web server
Outline

• Overview

• Existing Defenses (and why they don’t work)

• Designing SpikeStrip

• Evaluation
Robots.txt

- File placed on web server that tells crawlers how to behave
- **Problem**: compliance is voluntary

```plaintext
robots.txt –
User-agent: *
Disallow: /cgi-bin/
Disallow: /tmp/
Disallow: /~joe/
```
HTTP Request Headers

- Filter requests based on HTTP Request Header information
- **Problem:** headers can be modified by clients

HTTP Request
GET /index.html HTTP/1.1
Host: www.slashdot.org
User-Agent: Mozilla/5.0 Firefox/2.0.0.9
IP Address Tracking

- Rate limit request on a per-IP basis
- Problem 1: NATs and Proxies
IP Address Tracking

- Rate limit request on a per-IP basis
- Problem 2: Botnets
Authenticated User Accounts

- OSNs require users to sign-up and log-in
- Ban user accounts that generate too much traffic

**Problem:** URLs are *session independent*
Outline

• Overview
• Existing Defenses (and why they don’t work)
• **Designing SpikeStrip**
• Evaluation
Link-Encryption

• State of the art in crawler defense isn’t enough
  ▫ URLs are still *session independent*
  ▫ Crawlers can switch accounts, share state between accounts
  ▫ Need a way to link URLs to clients

• Solution: *server-side link-encryption*
  ▫ Encrypt links using user’s session key and server-side secret key
  ▫ Uniquely binds all served URLs to the user

• Link-encryption enables reliable per-session tracking
  ▫ Rate-limit sessions to throttle crawlers
Link-Encryption Example

Session Key = XYZ789

Get /product.html

Session Key = ABC123

Secret Key = ********

HTTP 403 Forbidden Error

Product info and such.

</html>
Implications of Link Encryption

- Consider a BFS on an OSN website...

- Queued URLs are bound to session

- Prevents session-switching
Rate Tracking and Limiting

• Reliable tracking enables rate limiting
  ▫ Very tight limits – no need to pad for NATs
  ▫ Enforcement – drop requests, ban accounts, etc

• Challenge: Scaling to high volume OSN sites

• Solution: Counting Bloom Filters
  ▫ Often used in high-throughput network security contexts
  ▫ SpikeStrip uses d-left CBF – fastest and most space efficient CBF variant
SpikeStrip Overview

- Link-encryption creates per-session “views” of the OSN
  - URLs are unique within each view
  - Binds URLs and clients
  - Enables reliable client tracking

- Prevents bad behavior
  - Crawlers can’t switch sessions
  - Distributed crawlers can’t share URLs
  - Enables strong rate-limiting

- Doesn’t hinder normal users and useful crawlers
  - Whitelist safe URLs using regex
  - Whitelist IPs/domains of good crawlers

Summary

New!
Outline

• Overview
• Existing Defenses (and why they don’t work)
• Designing SpikeStrip
• Evaluation
SpikeStrip Evaluation

Questions:

- How much server overhead does SpikeStrip cause?
  - Implemented SpikeStrip as an Apache 2 module
  - 256-bit AES encryption, d-left CBF

- How effective is SpikeStrip at throttling crawlers?
  - Created mock OSN called Fakebook
    - Based on data from London
    - 3.5 Million pages
    - Typical LAMP setup (Linux, Apache, MySQL, Python)
    - 10 load-balanced web servers, 1 DB
SpikeStrip Micro-Benchmarks

On average, SpikeStrip reduces Apache performance by 7%
SpikeStrip vs. Crawlers

- Rate limit = 1000 requests per hour
- 0.25 Requests Per Second (RPS)

- Successfully throttles crawlers
- Speed penalties can be made arbitrarily harsh, depending rate-limiter parameters
Conclusion

- OSN users are defenseless against malicious crawlers
- It’s up to OSNs to secure users’ data
- SpikeStrip uses novel link-encryption technique
  - Overcomes traditional user tracking challenges
    - Disambiguates users behind NATs/proxies
    - Renders botnets ineffectual
  - Minimal inconvenience for end-users
- SpikeStrip works in practice
  - Imposes minimal overhead on server
  - Successfully throttles crawlers
  - Works with existing Apache setups
SpikeStrip for Apache 2.x is Open Source!

Source code and benchmark tests available at
Why are OSNs so popular?

Because **everyone** uses them!
(Corollary: they have **lots** of data)

- **Sharing and Socializing**
- **Convenience**
- **Games and Interaction**
Existing Defenses Against Crawlers

- Passive Defenses
  - Robots.txt
  - HTTP Request Header Filtering

- Active Defenses
  - Relies on identifying, tracking, and rate limiting clients
  - Usually done by IP address

- Authentication Based Defenses
  - Control access by authenticating users
  - Use CAPTCHAs to control account creation
  - Ban users who break the rules
Link Opacity

• Link-encryption makes links opaque


facebook.com/secure/AnvTR64Iz  ➔  facebook.com/christowilson

• Not useful for security – metadata allows disambiguation

<a href="http://facebook.com/secure/AnvTR64Iz">Christo Wilson’s Profile</a>
Link Sharing

• Link-encryption makes it hard to share links
• Lots of web tech already does this
  ▫ Shopping carts
  ▫ AJAX
• Important pages to people ≠ important pages to crawlers
  ▫ Crawlers need friends lists and search results
  ▫ People want pictures
• Solution: permalinks

en.wikipedia.org/wiki/Facebook vs.
en.wikipedia.org/w/index.php?title=Facebook&oldid=366580719
End-to-End Latency

![Graph showing time per request (S) vs. number of requesting threads for Apache with SpikeStrip and Apache. The graph indicates a delay of 90ms.]
Does SpikeStrip Ruin OSN Research?

**NO!**

- SpikeStrip enables OSNs to set up controlled access channels for researchers
  - i.e. *.ucsb.edu can crawl at rate X for Y days
- This arrangement benefits both parties
  - Researchers can crawl in a secure way
    - No need deal with account bans, etc
  - OSNs can control who has access and their bandwidth allocation