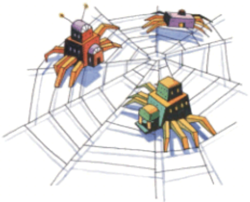
PUBCRAWL: Protecting Users and Businesses from CRAWLers

Grégoire Jacob[1,3], Engin Kirda[2],
Christopher Kruegel[1], Giovanni Vigna[1]

[1] University of California, Santa Barbara / [2] Northeastern University / [3] Télécom SudParis



* image from viralpatel.net

Fri Aug 10 2012

Introduction: are crawlers a threat?

What do web crawlers/spiders do?

- Browse the web in an automatic and systematic fashion
- Various types:
  - *web indexers* for search engines,
  - *link checkers* for site verification,
  - but also *scrapers* to harvest the content of sites

When does crawling become an abuse?

- Unauthorized large-scale crawls over web sites or social networks
- Use the collected data for competing products or services
  e.g., American Airlines-2003, Ryanair-2008, Facebook-2010
- Use the collected data for social engineering or targeted attacks
  e.g., StudiVZ-2009

Introduction: crawler prevention

How can we prevent crawlers from accessing a web site?

- Robot Exclusion Protocol:
  - ⊕ Access rules to limit crawlers to certain parts of a site
  - ⊖ Cooperation of the crawler is required
- User Authentication:
  - ⊕ Precise tracking of the user activity
  - ⊖ Forcing user to login might not comply with the business model
- CAPTCHAs and Traps:
  - ⊕ Tests/Traps that are easily solved/avoided by humans but remain hard for computers
  - ⊖ Potential usability issues and reduced user satisfaction
- **Crawlers need to be identified first to trigger prevention**

Introduction: crawler prevention

Why is the problem of detecting crawlers hard?

- IP-based identification of traffic sources
  - User-agent strings are unreliable
  - A same IP can host multiple users (proxy)
  - Authentication not necessarily available
- Passive detection is constraining
  - Request logs only contains basic information: *timing, HTTP header and URL information*
  - Request logs contains huge amounts of data to handle

Introduction: crawler prevention

How are crawlers currently detected?

- Learning techniques to extract crawlers' properties
- HTTP header artifacts:
  - ⊕ Betraying user-agent, missing referrer, ignored cookies
  - ⊖ Stealthier crawlers already handle these shortcomings
- Simple traffic statistics:
  - ⊕ Large request volume, short inter-arrival time, night traffic
  - ⊖ Large number of users behind a proxy show similar statistics
  - ⊖ These statistics become inadequate with distributed crawling
- **Need robust properties to distinguish:**
  - large proxies hosting a large number of users
  - stealthy crawlers mimicking browsers
  - distributed crawlers over multiple sources

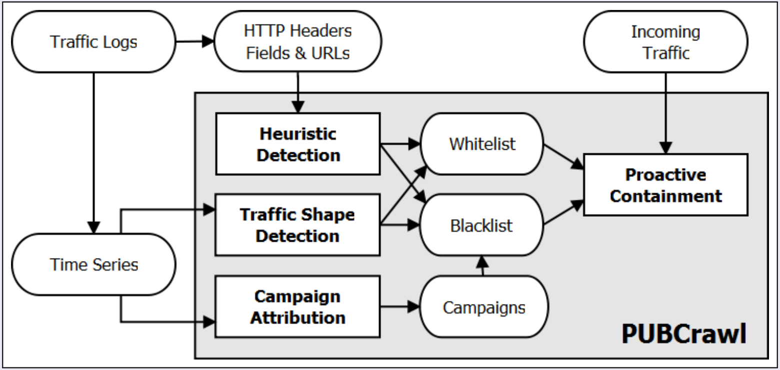
Introduction: containing crawlers

Our approach: PUBCRAWL

- Hypotheses on the traffic shape:
  - **User traffic:** *high versatility* on the short-term, *daily regularity*.
  - **Crawler traffic:** *high stability/versatility* on the long-term.
  - **Distributed crawler traffic:** *high synchronization* across sources.
- Traffic model based on time series
  - **Rational 1**: Independently, time-series can *model the traffic shape of a source* in a way that is *independent of the traffic volume*.
  - **Rational 2**: Combined, time-series offer valuable information about the *synchronization of sources*.
- Machine learning to extract distinctive characteristics of crawler traffic
- Clustering to identify synchronized traffic from crawling campaigns
- Automatic configuration of a containment strategy accordingly

System: PUBCRAWL overview

System architecture



System: heuristic detection

Detection of suspicious request content

- **Input:**
  - Suspicious values in the HTTP header fields
  - Suspicious visit patterns in the URLs
- **Heuristics:**
  - Original set of heuristics from the state of the art
  - Extended heuristics adapted to social networks
  - Decision by majority vote over heuristics

Source	Heuristic	Suspicious check
HTTP	High error rate	404 errors
HTTP	Suspicious referrer	none or directory query
HTTP	Unbalanced traffic	99% of traffic coming from a single-user agent
HTTP	Ignored cookies	new sequence number at every request
URL	No parameter use	no language choice depending on the profile
URL	Low page revisit	low overlap in the visited profiles
URL	Profile sequence	alphabetical order of visited profiles

System: traffic shape detection

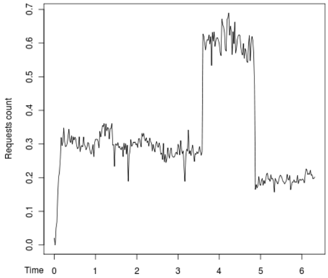
Detection of crawlers by traffic classification:

- **Input**: time-series with normalized amplitude and common time origin
- **Classification**:
  - Features: Auto-correlation and decomposition analyses
  - Classifiers: + Naive Bayes, Association Rules and SVM classifiers
    + Training over user and crawler traffic
    + Decision by majority vote over the classifiers output

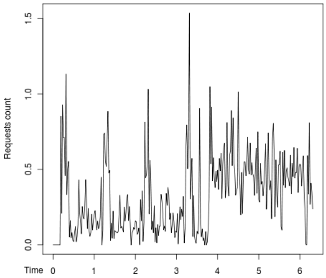
System: traffic shape detection

Time series generation

- Counting Process: volume of requests per time interval
- Fundamental differences in traffic shape between users and crawlers



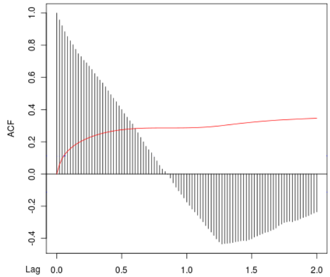
Crawlers: *very stable, sudden or no shift*



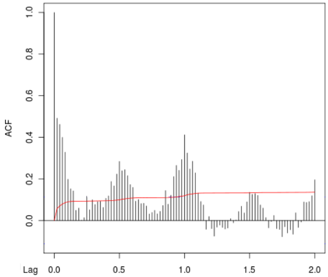
Users: *locally noisy, daily regularity, slow shifts*

System: traffic shape detection

Auto-correlation Analysis: Sample Auto-Correlation function (SAC)



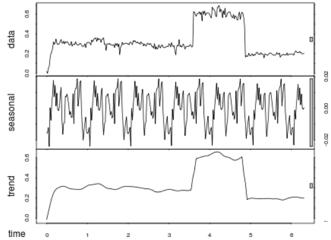
Crawlers: *linear decay, strong correlation at small lags, single oscillation, no local spike*



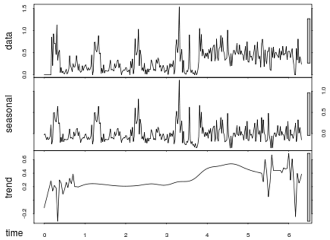
Users: *cut-off decay, multiple oscillation, local spikes at day lags*

System: traffic shape detection

Decomposition Analysis: Trend, Season and Noise components (STL)



Crawlers: *stable or square trend, predominant trend*



Users: *dispersed trend, predominant season*

System: campaign attribution

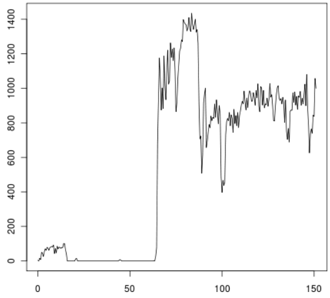
Detection of crawling campaigns by traffic clustering:

- **Input**: time-series with normalized amplitude and common time origin
- **Clustering**:
  - Similarity: Inverse of the squared Euclidean distance
  - Clustering: + Incremental clustering around series medoids
    + Cluster generation controlled by a minimal intra-cluster similarity
    + Candidate clusters selected by medoids of similar amplitude and deviation

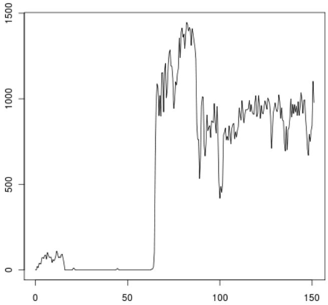
System: campaign attribution

Time series synchronization

- Distributed crawlers from a same campaign are highly synchronized



Malicious Crawler A



Malicious Crawler B

System: proactive containment

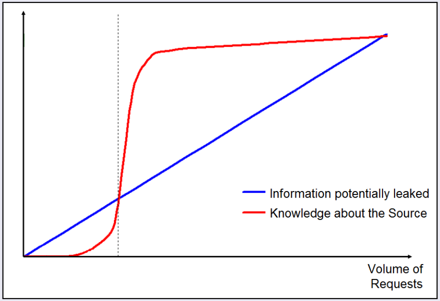
Strategy to trigger active responses:

- *Reduce the impact of active responses over users while limiting the quantity of information possibly leaked by crawlers*
- **Minimal volume sources**:
  - Traffic: user traffic with high probability
  - Minimal amount of information possibly leaked
- **Above average volume sources**:
  - Traffic: traffic coming from crawlers or users behind a proxy
  - Detected crawlers are **blacklisted**
  - Legitimate crawlers and stable proxies are **whitelisted**
- **Low to average volume sources:**
  - Traffic: mixed traffic between users and distributed crawlers
  - Time to collect sufficient traffic to apply classification
  - **Active responses** (*e.g.*, CAPTCHAs) to slow down potential leaks

System: proactive containment

Rational

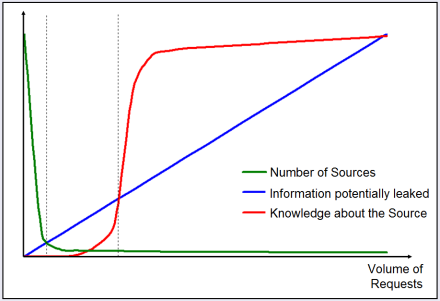
- Empirical trade-off between:
  - the acquired knowledge about this source and
  - the potential amount of information it may leak
  - the number of impacted sources



System: proactive containment

Rational

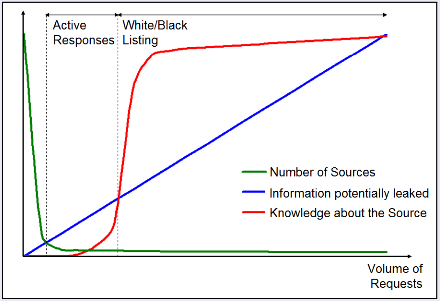
- Empirical trade-off between:
  - the acquired knowledge about this source and
  - the potential amount of information it may leak
  - the number of impacted sources



System: proactive containment

Rational

- Empirical trade-off between:
  - the acquired knowledge about this source and
  - the potential amount of information it may leak
  - the number of impacted sources



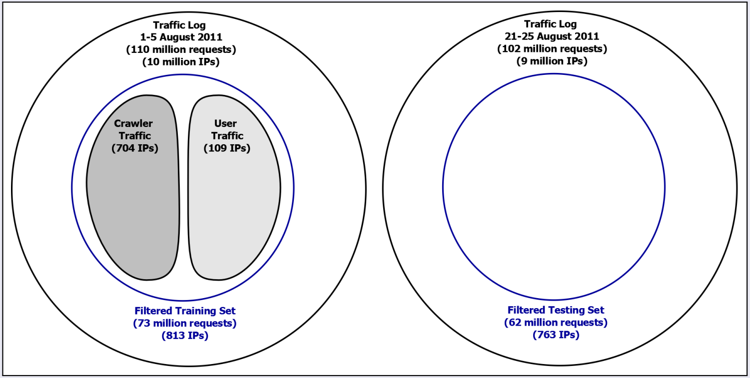
Evaluation: dataset presentation

Social network traffic logs

- Anonymized traffic from a large social network:
  public profiles accesses (no authentication required)
- Request logs:
  time, encrypted source IP, encrypted subnet, user-agent, target URL, server response, referrer, cookie
- Filters:
  - obvious crawlers (more than 500,000 requests per day)
  - insufficient traffic (less than 1,000 requests per day)

Evaluation: dataset presentation

Training and testing sets



Evaluation: training the system

System configuration

- Configuring heuristic thresholds:

Accuracy	Former features	New features	Combined features
	47.84%	82.50%	**77.81%**

- Training traffic shape classifiers:

Accuracy:	Bayes	Rules	SVM	**Vote**
Cross validation	98.39%	96.36%	98.55%	**98.99%**
Two third split	97.45%	96.19%	95.11%	**96.90%**

- Tuning the intra-cluster similarity threshold:

Precision	Recall	Accuracy
99.03%	85.54%	**94.35%**

- Tuning the proactive containment strategy:

Minimal Vol.	Sufficient Vol.	Impacted IPs	Impacted Requests
100 requ.	1000 requ.	0.1%	3.2%

Evaluation: testing the system

System evaluation

- Heuristics detection:

Accuracy	Former features	New features	Combined features
Accuracy	38.84%	86.34%	**74.19%**

- Traffic shape classification:

Accuracy	Bayes	Rules	SVM	**Vote**
Global	93.05%	87.55%	94.36%	**94.89%**
Legitimate crawlers	92.54%	87.10%	97.18%	**93.95%**
Unauthorized crawlers	88.89%	96.27%	100.00%	**100.00%**
Masquerading crawlers	98.27%	86.71%	98.84%	**98.84%**
Crawlers (TP/FN)	93.66%	87.68%	97.79%	**95.58%**
Users (TN/FP)	82.50%	85.00%	32.50%	**82.50%**

Evaluation: testing the system

System evaluation

- Campaign attribution:

Precision	Recall	Accuracy
92.84%	80.63%	**91.89%**

Agent	#Clust.	#ClassC	#IP	Req/day
Legitimate crawlers				
Bingbot	5	11	211	6 million
Googlebot + Feedfetcher	9	11	113	4 million
Yahooslurp	4	9	71	500 thousand
Baiduspider	1	1	23	50 thousand
Voilabot	3	3	20	19 thousand
Facebookexternalhit/1.1	1	1	8	14 thousand
Crawlers with suspicious agent strings				
" "	2	16	22	330 thousand
Python-urllib/1.17	2	51	54	140 thousand
Mozilla(compatible;ICS)	1	10	10	70 thousand
EventMachine HTTP Client	1	3	3	3 thousand
Gogospider	1	2	3	2 thousand
Masquerading crawlers				
Gecko/200805906 FireFox	1	10	73	350 thousand
Gecko/20100101 FireFox	9	12	25	60 thousand
MSIE6 NT5.2 TencentTraveler	1	1	30	7 thousand
Mozilla(compatible; Mac OS X)	1	1	4	8 thousand
googlebot(crawl@google.com)	1	1	4	1 thousand

Evaluation: system evasion

Evasion techniques and remediation

Evasion	Potential remediation
Browser header mimicry	traffic-shape detection
Traffic shape randomization	traffic-shape detection
Traffic shape engineering	*model of user traffic needed*
Distributed crawling	campaign attribution
Traffic de-synchronization	*robust similarity measure*

Conclusion: PUBCRAWL

Contributions

- Solution to the detection and prevention of crawlers
- Traffic model relying on time-series:
  - More robust than traditional HTTP field values
  - More robust than simpler statistics over the traffic volume and speed
- Detection based on learning techniques over traffic shape features
- Identification of crawling campaigns by clustering of distributed traffic
- Optimization of the containment strategy according to detection
- Large scale deployment in production at a well-known social network