Botnet Judo: Fighting Spam With Itself

Presented by:
Payal Singh

What is Botnet Judo?

Innovative spam filter

Source as a vantage point

“ Monitor new spam as it is created, and consequently infer the underlying template used to generate polymorphic e-mail messages by instantiating and monitoring botnet hosts in a controlled environment. “

What’s the innovation?

Increases the window of vulnerability by dynamic filtering

Template based approach

Virtually no false positives; very few false negatives

Template inference algorithm

Cheap and practical deployment

Previous spam filtering approaches

Content-based vs source-based

Bayesian filtering, IP blacklisting, sender reputation filtering, subject line blacklist, URL domain blacklist ...

Spam template

Anchor - constant text

Macros -

- Noise - random
- Dictionary - list

Micro-anchor - small length non-alphanumeric limiters for macros

Assumptions

Template-based spam

Limited number of templates

Initial messages based on single template - not intermixed

Interleaved messages

Group of messages belonging to more than one template

Judo doesn’t work well with such a set

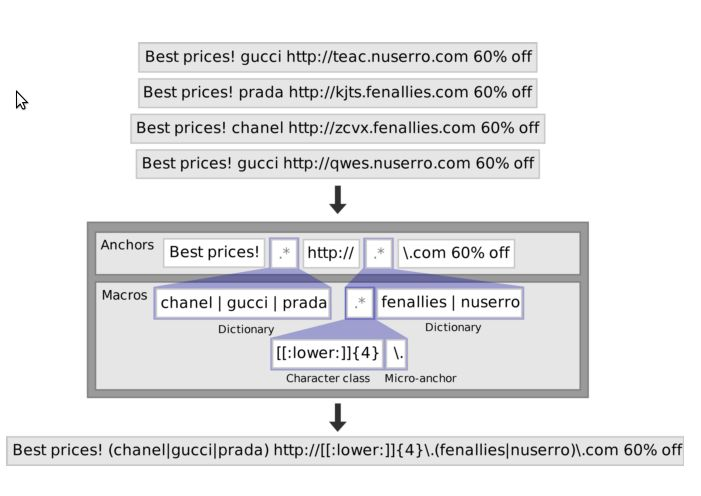
Produces amalgamated template - lacks precision

Signature Generator - Template inference algorithm

Regex signature by parsing messages

Maintain set of signature, updatable in real-time

Domain knowledge of email structure - message ID, subject, body, etc.



Anchor

LCS over q-gram substring, where q = minimum length of anchor substring

q = 6

Optimized by a two pass approach

- Find all substrings of length q common to all messages
- Treat each substring as a single character and find the LCS
- Reduces reading size of messages by converting them to substring sequences

Macros

Variant text between anchors

Dictionary Macros:

False positive - incomplete dictionary

Null hypothesis - probability of unfound element

= empirical probability of least frequently occurring element

(1−fn/(1+fn))^m (n distinct strings in m samples, fn is probability of lease frequently occurring element

Disjunction regex (a1 | a2 | … | an)

Micro-anchors

Small alphanumeric macro delimiters (<< q)

Identified with LCS on non-alphanumeric characters

Processing sequence - micro-anchors -> dictionary macros -> noise macros

Noise macros

Random characters generated from some character set (message IDs, etc.)

Same length strings - regex matches both character class and length

Variable length strings - only character class with + or * operators

Domain Knowledge

Improve algorithm’s performance by identifying message/email structure

Domain Knowledge: Header filtering

Exclude headers added by mail transfer agent to avoid generating templates involving spam collection environment information - incorrect anchors

Excluded headers example - To, From, IP address of mail server

Message must match all included headers to match the respective signature

Domain Knowledge: Special Tokens

Dates, IP addresses, etc.

“ If the output of a date macro, for example, were run through the template inference algorithm, it would infer that the year, month, day, and possibly hour are an- chors, and the minutes and seconds are macros. The re- sulting signature would, in effect, “expire” shortly after it was generated. “

Initially replaced with anchors, later converted to regex capturing all instances of the macro

Dynamic Signature Update

One signature per template

Signature set

Training buffer: contains messages that don’t match any signature

Training buffer is the input to template inference algorithm, outputs new signature

Parameter k - trade-off between signature selectivity and training time

- Too small - incomplete dictionary
- Too large - Longer generation time, general signatures

Second chance mechanism

Allowed updates after signature generation, building dictionary list

Faster

Anchor signature match: Added to training buffer and update signature

Performed incrementally

Pre-clustering

Large training buffer -> interleaved messages -> amalgamated signatures

Skeleton signatures - larger length of substrings (q ~ 14)

New skeleton signature - minimum number unclassified buffer messages

k = 10 (skeleton signature), k = 100 (full signature)

Unexplored

Execution Time

Linear time - increases with number of messages as input

Real-time updates are faster

Paper focused on accuracy rather than efficiency

Evaluation

Signature safety testing methodology - false positive rate

Effectiveness - false negatives

Corpora

Age bias

Evaluation types

Single template inference

Multiple template inference

Real-world development

Response time

Single template inference

Training sets generated from single templates

	False Negative Rate			
k	95%	99%	Max	Avg
1000	0%	0%	0%	0%
500	0%	0%	2.53%	<0.01%
100	0%	0%	0%	0%
50	0%	0%	19.15%	0.06%
10	45.45%	58.77%	81.03%	14.16%

(a) Self-propagation and pharmaceutical spam.

		False Negative Rate			
k	s	95%	99%	Max	Avg
1000	99.8%	0%	0.22%	100%	0.21%
500	81.8%	100%	100%	100%	18.21%
100	55.0%	100%	100%	100%	45.04%
50	42.9%	100%	100%	100%	57.25%
10	40.9%	100%	100%	100%	62.13%

(b) Stock spam.

Table 2. False negative rates for spam generated from Storm templates as a function of the training buffer size k. Rows report statistics over templates. The stock spam table also shows the number of templates s for which a signature was generated (for self-propagation and pharmaceutical templates, a signature was generated for every template); in cases where a signature was not generated, every instance in the test set was counted as a false negative. At $k = 1000$, the false positive rate for all signatures was zero.

Multiple template inference

Training buffer includes spam from multiple templates

Delay - Hold up messages for a short time to wait and see if it matches any updated signatures/filters

False negative rate decreases with increase in delay

False negative rate increases with increase in training buffer size

Observed miscalculation in template generation due to false dictionary complete assumption in one of the corpora

Multiple template inference evaluation

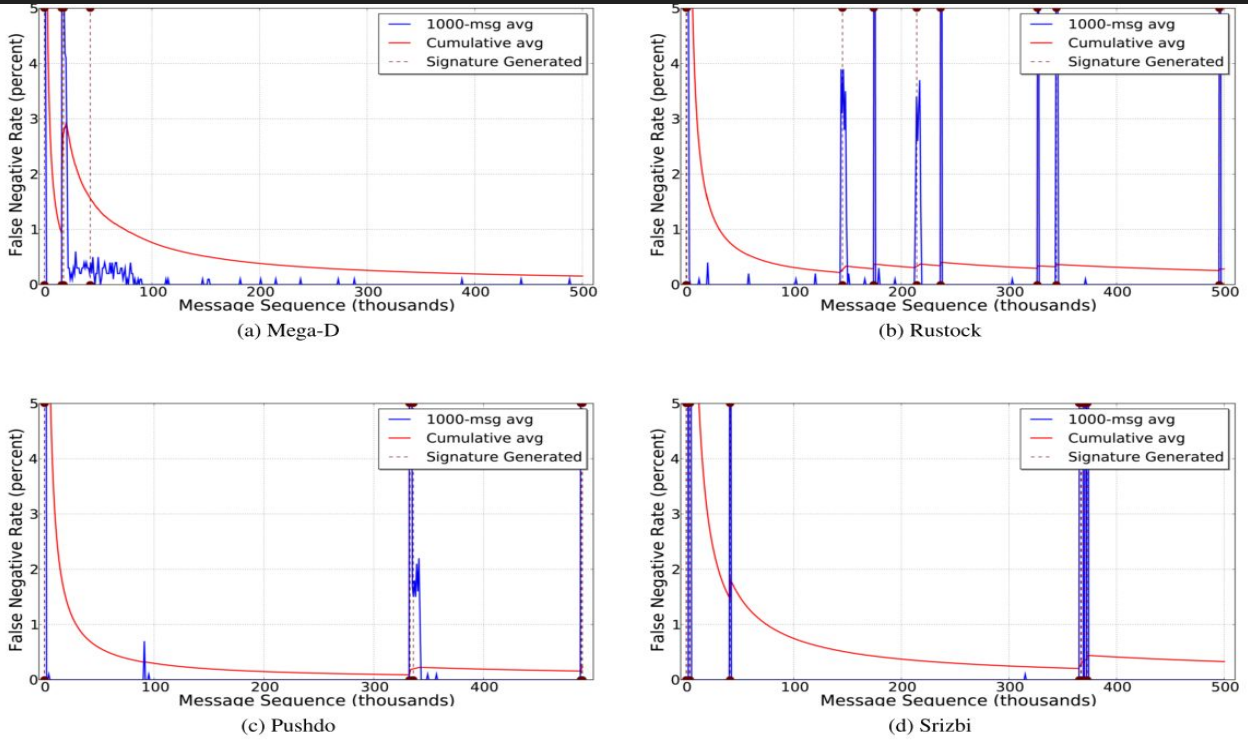


Figure 5. Classification effectiveness on Mega-D, Rustock, Pushdo, and Srizbi spam generated by a single bot, as a function of the testing message sequence. Experiment parameters: $k = 100$, $d = 0$

Real-world deployment

Multiple bot instances, one for training and the other for testing

~1% false negative rate - higher because asynchronous training and test buffers

Relatively higher false negative rate - unsynchronized template switching

Near-perfect false positive rate for k >= 500 in most bots

- Whole email
- Signature generated by processing spam source
- Safe signatures - at least one anchor and macro

Response Time

Average of 2 - 10 seconds, based on message length and training buffer size

Training set generation

- Spam rate of botnets
- Interleaving

Pros

Safe - no false positives

- Content + header
- unsafe signatures are rejected

Reliable - very low false negatives

Cheap - Easily integrable and deployable

- Black box
- Regex

Cons

Execution time: unoptimized

Unsafe signatures - incomplete dictionaries, false anchors, multiple signatures per template

Spammer reaction - Defeat Judo

VM Detection

Proof of genuine spam node - sending email to external IPs

Advanced spam generation languages - multi-pass directives

Template distribution - multiple templates across each bot making it harder for Judo to generate reliable signature based on individual bot messages

Questions?