A System for High-Throughput Spam Analysis and Clustering

Riccardo Pelizzi

System Security Lab
Department of Computer Science
Stony Brook University

September 19, 2010
Introduction

- Spam is a short name for *unsolicited bulk email*.
  - Personal unsolicited emails: job offers, old friends, etc.
  - Solicited bulk emails: newsletters, phd@seclab, etc.
- It causes millions of dollars of damage every year, both directly (loss productivity, investment required to protect against Spam) and indirectly (increases revenue of malware writers).
- Spam became so widespread that created a market for antispam research and commercial products.
  - Bayesian filters, checksum-based filters, IP blacklists, URL blacklists, etc.
Motivation

- Problem: these aforementioned products mitigate the effects of Spam, but do not address its root causes.
- To this end, we need effective actions aimed to combat Spam on a global scale.
- Our goal: follow the flux of Spam upstream, to have a wider perspective on the problem
  - It is unpractical to manually visit every Spam URL to find Spam websites.
- Provide automated analysis tools to law enforcement: Messages → URLs → Spam pages → Clusters of pages
Overview

• SpamAnalyzer: a system for *High-Throughput Analysis* and *Clustering*
• Analysis
  • redirection analysis
  • screenshot generation
• Clustering
  • Image Shingling and Locality Sensitive Hashing (LSH)
• High-Throughput
  • No time to analyse every URL
  • Heuristics to analyse only a sample of incoming URLs.
Overview
Incoming Spam

- Incoming Spam is collected in a *Spamtrap*
  - A Spamtrap is a honeypot for Spam
  - Sidesteps the problem of classifying ham/spam: everything is spam!
  - Supplied by a Californian ISP to UCSB.
  - About 150k messages/day.
  - Representative?

- Every 2 hours, we fetch the URLs of the Spam messages from the last two hours.
Incoming Spam (2)

- It is not uncommon to find thousands of URLs such as:
  - http://vdf3g.chskr.cn/?fdsvkj=fsdv
  - http://ad56v.chskr.cn/?dsdkjr=askn
  - http://po1bk.chskr.cn/?oyinbd=exxc
- We do not care about every single URL, they are just used to discover potential landing pages.
- Heuristic: do not insert more than 1000 URLs for each registered domain.
  - Registered Domain: google.co.uk, yahoo.com
DNS Resolution

- URL domains are resolved using DNS queries.
- For efficiency, we detect *random domains*.
  - If the authoritative DNS for the domain is under the spammer’s control, it is possible to create an infinite number of subdomains for free.
  - They do not resolve to different websites! The DNS is configured to return the same IP for any subdomain.
  - Therefore, if we detect them, we can save thousands of DNS queries.
URL sampling

• The URLs are too many to analyze them all, we need a heuristic for sampling. Two contrasting goals:
  • We want to discover all landing pages.
  • We need to finish in 2 hours, so we want to analyze as few as possible.
• As the basic unit for the heuristic, we use registered domains.
  • Nr of URLs and Domains can vary considerably.
  • Registered domains cost money and they are therefore registered at a constant, reasonable rate.
URL Sampling (2)

- For each registered domain
  - Random Domain $\rightarrow$ Analyze only 1
  - Few Domains $\rightarrow$ Analyze them all
  - Too many Domains $\rightarrow$ group by URL minus querystring and use random sampling.
The sampled URLs are retrieved and analyzed for redirection. Two purposes:

- Obtain statistics about redirection and save nontrivial redirection attempts for later analysis.
- Minimize the number of candidates for the next phase, the most expensive of the whole system.

Some spammers actively oppose analysis: spoof user-agent and throttle traffic to hosts to avoid ban.

Support for multiple IPs
Redirection Analysis (2)

• 4 different ways to detect redirection, for performance and effectiveness.
• Increasingly expensive, to save time.
  • HTTP/Meta redirection
  • Javascript pattern matching
  • JSAnalyzer
  • Browser URL bar
JSAnalyzer

- JSAnalyzer is a virtual client that detects and logs redirection attempts.
- It uses Mozilla’s JS engine as a standalone component.
- This component does not provide DOM capabilities.
- Using python-spidermonkey, we inject python objects into the Javascript engine.
- The global namespace is a **Stub Object**.
  - `__getattr__` is redefined to return stub objects in case of unknown property or method.
  - Certain properties and methods are instrumented to log redirection, such as `document.location`.
- These objects contain instrumented version of specific DOM functionalities, such as `document.location`.
- This way, we don’t have to provide a complete DOM implementation.
JSAnalyzer Example

```javascript
foo();
document.write("<script>
bar = 3
</script>");
spam();
```

```
<script>
bar = 3
</script>
```

```
write()
```

```
execute_html
```

```
execute_html
```

```
execute_js
```

```
execute_html
```

```
execute_js
```

```
execute_js
```

```
execute_js
```
Screenshots

• The set of target pages is then opened with Firefox to get screenshots.

• We developed a browser plugin to turn Firefox into a batch process for downloading screenshots.

• Timeout heuristics because onLoad is not dependable.

• Reads the URL bar as a final redirection detection.
Clustering

• Distance: Jaccard Index on Image shingles
  • The image is cut into equal squares (we take the CRC of the square to compress the information) called *shingle*.
  • Fails against shifting and noise, but resilient against partial loads and banners.
  • Jaccard Index: \[ \frac{a \cap b}{a \cup b} \]

• We could use a hierarchical clustering algorithm, but the dataset is huge and the space is 300-dimensional.
• Can we do better than \( O(N^2) \)?
  • Nearest Neighbour: No.
  • Approximate Nearest Neighbour: Yes, in the average case.
Clustering(2)

- The idea behind LSH is to hash the set of pages $P$ in such a way that similar pages have a much higher collision probability than dissimilar pages.
- The key for LSH is to provide a family $H$ of functions $h : S \rightarrow V$, with $S = \mathcal{P}(V)$ satisfying the following conditions for each pair of pages $p_1, p_2$:
  - if $J(p_1, p_2) \geq T$, then $P([h(p_1) = h(p_2)]) \geq C_1$
  - if $J(p_1, p_2) \leq cT$, then $P([h(p_1) = h(p_2)]) \leq C_2$

where
- $T$ is the similarity threshold
- $C_1$ and $C_2$ are the probabilities that two similar items collide and two different items collide respectively. Ideally, $C_1 \gg C_2$.
- $c < 1$ sets the size of the transition interval $[cT, T]$. 
• Generally, our family $H$ is not parametrized. How can we tailor it to an appropriate threshold and accuracy?
We can concatenate multiple function from $H$. If we define:
\[ lsh(p) = h_1(p), h_2(p), \cdots, h_k(p) \]
($k$ is known as sketch size), then we have:
\[ P([lsh(p_1) = lsh(p_2)]) = P([h(p_1) = h(p_2)])^k. \]

If we perform $l$ iterations, define $S$ as the set of pairs which were similar in at least one iteration, and define
\[ P([lsh(p_1) = lsh(p_2)]) = v \]
we get:
\[ P((p_1, p_2) \in S) = 1 - (1 - v^k)^l = g_{k,l}(v) \]

Tradeoff
Clustering (5)

Collision Probability vs Jaccard Similarity

- $k=5, l=7$
- $k=13, l=100$
Clustering (6)

- Time to discover the mysterious function!
- For the Jaccard Similarity, a suitable family $H$ is
  
  $$h_i(p) = \min(\pi_i(p))$$

  where $\pi_i$ is a permutation of $V$, chosen uniformly at random.
- We use modular algebra instead of true random permutations
  
  $$\pi_i(x) = c_{i1}x + c_{i2} \pmod{P}$$

  where $P$ is a prime number bigger than $V$, and $(c_{i1}, c_{i2})$
Clustering (7)

• Follow these steps \( l \) times to build the candidate set
  • Choose \( k \) functions from \( H \) randomly
  • Calculate \( lsh(p) = h_i(p), \ldots, h_k(p) \) for each \( p \in P \). Put it in \( L \).
  • Sort \( L \) according to \( lsh(p) \)
  • Scan through \( L \): if a group of pages with identical \( lsh(p) \) is encountered, add all pairs to \( S \)
• Finally, use \( J(p_i, p_j) \) to filter the set for dissimilar pages.
Clustering (last!!!)

- Use a single linkage algorithm to form clusters from pairs.
- Complexity is $O(n^2)$ for the worst case, because linkage and filtering depend on the number of pairs.
- In practice, it is much faster.
- Evaluate “suspiciousness”
Incoming Spam

- Domains to Registered domains ratio is very unstable → random domains.
- It is trivial to generate a huge number of unique URLs or domains. That is not the case for registered domains (100/day)
Redirection

- SpamAnalyzer recognizes 9 types of redirection.
- About 75% of pages use some kind of redirection
  - Consistent with Spampscatter’s results

<table>
<thead>
<tr>
<th>Type</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>63.51%</td>
</tr>
<tr>
<td>Refresh</td>
<td>0.01%</td>
</tr>
<tr>
<td>JavaScript (Static)</td>
<td>1.16%</td>
</tr>
<tr>
<td>JavaScript (Dynamic)</td>
<td>2.69%</td>
</tr>
<tr>
<td>Meta-HTTP</td>
<td>4.93%</td>
</tr>
<tr>
<td>Frame</td>
<td>0.41%</td>
</tr>
<tr>
<td>No redirection</td>
<td>25.36%</td>
</tr>
<tr>
<td>Suspicious</td>
<td>1.90%</td>
</tr>
<tr>
<td>Complex</td>
<td>0%</td>
</tr>
<tr>
<td>Total Analyses</td>
<td>8733</td>
</tr>
</tbody>
</table>
Redirection (2)

Open Text File
Hosts

- Geolocation: most hosts are in China and Eastern Europe.
- 57% of registered domains are .cn.
- HTTP server fingerprints reveals that Nginx’s market share is larger for scam hosts.
Clustering (again!)
Clustering (∞)
Clustering (∞)

> Highest

> Highest
Anti-Analysis

- Random Domains
- JavaScript obfuscation
- Fake HTTP error codes
- Malformed URLs
- IP ban