Automatically Detecting Vulnerable Sites Before They Turn Malicious

Kyle Soska  
Carnegie Mellon University  
ECE / Cylab  
ksamka@cmu.edu

Nicolas Christin  
Carnegie Mellon University  
ECE / Cylab  
nicolasc@cmu.edu

Sponsors: NSF (CCF-0424422, CNS-1223762) and DHS S&T/CSD (N66001-13-C-0131)
Problem Setting

- Adversaries compromise websites
  - Economically Rational – monetize compromises
    - Neutral to victims
    - Maximize volume, efficiency, profits
  - Hacktivist – promote social, political, or religious agenda
    - Targeted attacks
    - Low volume
Economically Rational Adversary

- Decisions always maximize profit
- Probabilistic Polynomial Time
  - Cannot break standard crypto, session cookies, hashes, etc.
- Does not control significant portion of web
  - Cannot perform adversarial machine learning attacks by poisoning a random sample of the web
- Able to exploit vulnerable web software
Mode of Operation

Step 1: Find bug or vulnerability in popular web software or content management system (CMS)

Step 2: Enumerate sites containing vulnerability

Step 3: Exploit vulnerable sites

Step 4: Monetize and profit
Problem and Goal

- Existing approaches detect if a webpage is already malicious

- Is it possible to predict if a non-malicious website will become malicious in the future?
  - What would such a system look like?
  - What requirements are imposed on such a system?
  - What are the fundamental limitations?
System Design

Blacklists → Malicious Sites → Archive.org

Zone Files → Safe Sites

Archive.org → Template Filter → Feature Extraction → Stream Classifier

Alexa.com
System Properties

- **Efficiency**
  - Internet dataset

- **Interpretability**
  - Need to build intuition about why the site will become compromised

- **Robustness to Imbalanced Data**
  - Far more benign examples than malicious ones

- **Robustness To Mislabeled Data**
  - Blacklists may contain errors or be incomplete

- **Adaptive**
  - Internet is a concept drifting, requires active adaptation
Dataset

Blacklists -> Malicious Sites -> Archive.org

Zone Files -> Safe Sites

Template Filter -> Feature Extraction -> Alexa.com

Stream Classifier
## Dataset

<table>
<thead>
<tr>
<th>Type</th>
<th>Instances</th>
<th>Archived Instances</th>
<th>% Archived</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhishTank</td>
<td>91,555</td>
<td>34,922</td>
<td>38.1</td>
</tr>
<tr>
<td>Search Redirection*</td>
<td>16,173</td>
<td>14,425</td>
<td>89.2</td>
</tr>
<tr>
<td>.com Zone Files</td>
<td>336,671</td>
<td>336,671</td>
<td>N/A</td>
</tr>
</tbody>
</table>

- Zone Files: Feb 2010 – Sept 2013

* Leontiadis et al., 2014
Filtering

- Navigation
- User Content
- Social Media Links
Filtering – Based on [Yi et al., 2003]

- Compute entropy-like heuristic “Composite Importance” (CmpImp ∈ [0, 1]) for each element on a page

- Remove elements above a fixed threshold
Threshold = 0.99
Threshold = 0.1
Feature Extraction

- Blacklists
- Zone Files
- Malicious Sites
- Safe Sites
- Archive.org
- Template Filter
- Alexa.com
- Feature Extraction
- Stream Classifier
Feature Set

- Traffic Features
  - Site Rank
  - Links into site
  - Load Percentile
  - ...more

- Content Features
  - HTML Tags (type, content, attributes)
Dynamic Features

- Millions of unique HTML tags (including content)

- Solution: order tags by some statistic, select top $N$
  - ACC2 based on [Foreman, 2003]
  - Let $\mathcal{B}$, $\mathcal{M}$ denote the set of benign and malicious sites respectively, $\mathcal{w}$ the set of tags from a site, then ACC2 for a tag $x$ can be defined as:

$$s(x) = \left| \left| \{x \in \mathcal{w} : \mathcal{w} \in \mathcal{M} \} \right| \left| \mathcal{M} \right| - \left| \{x \in \mathcal{w} : \mathcal{w} \in \mathcal{B} \} \right| \left| \mathcal{B} \right| \right|$$
## Prominent Features After 90,000 Samples

<table>
<thead>
<tr>
<th>Feature</th>
<th>Statistic Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>meta{‘content’: ‘Wordpress 3.2.1’, ‘name’: ‘generator’}</td>
<td>0.0569</td>
</tr>
<tr>
<td>ul{‘class’: [‘xoxo’, ‘blogroll’]}</td>
<td>0.0446</td>
</tr>
<tr>
<td>You can start editing here.</td>
<td>0.0421</td>
</tr>
<tr>
<td>meta{‘content’: ‘Wordpress 3.3.1’, ‘name’: ‘generator’}</td>
<td>0.0268</td>
</tr>
<tr>
<td>/all in one seo pack</td>
<td>0.0252</td>
</tr>
<tr>
<td>span{‘class’: [‘breadcrumbs’, ‘pathway’]}</td>
<td>0.0226</td>
</tr>
<tr>
<td>If Comments are open, but there are no comments.</td>
<td>0.0222</td>
</tr>
<tr>
<td>div{‘id’: ‘content_disclaimer’}</td>
<td>0.0039</td>
</tr>
</tbody>
</table>
Varying Window Sizes

Feature: meta{‘content’: ‘Wordpress 3.2.1’, ‘name’: ‘generator’}
CMS Evolution

![Graph showing the evolution of CMS attacks across different versions of Wordpress. The graph plots ACC2 Value against Sample #, with distinct peaks indicating attack campaigns and a trend of low activity over time. The versions are labeled Wordpress 2.9.2, Wordpress 3.2.1, Wordpress 3.3.1, and Wordpress 3.5.1.](image)
Parking Page Feature

Similar value over 1 year later!

Feature: div{'id': 'content_disclaimer'}
Classification

- Blacklists
- Zone Files
- Malicious Sites
- Safe Sites
- Archive.org
- Alexa.com
- Template Filter
- Feature Extraction
- Stream Classifier
Classification

- Largely based on [Gao et al., 2007]
- Break input data stream into blocks
- Resample input blocks
- Train ensemble C4.5 decision tree classifiers using Hoeffding bounds [Domingos et al., 2000]
- Retrain periodically using new dynamic features
Classification Results

Good Operating Point
Limitations

- Only makes sense when page content and traffic statistics are risk factors of malice
  - Sites hacked via weak passwords or via social engineering attacks violate this
  - Sites that are maliciously hosted may violate this

- Requires some sites to become compromised in order to make predictions
Conclusions

- Predicting websites that become malicious in the future is possible!

- Acceptable performance can be achieved even on our modest dataset

Kyle Soska – ksoska@cmu.edu