Detection of Web Defacements by means of Genetic Programming

Eric Medvet, Cyril Fillon and Alberto Bartoli
DEEI, University of Trieste, Italy
Web site defacement

- **Common** web attack: consists in replacing part or entire web page...
  - ...with **evident** disturbing content or political, social, religious messages
  - ...with **subtle** changes (links, forms, ...)
- More than **490,000** pages defaced in 2006, about 1500 pages/day
Detection: requirements

- Monitoring dynamic web pages **automatically**, that is:
- …without any assumption on expected content, appearance, behavior of the page
Detection: key idea

1. Observe the monitored page
2. Build a profile
3. Detect deviation from the profile

- This work approach: using Genetic Programming (GP) for points 2 and 3
Genetic Programming overview (I)

- The process of **solving** a problem by **searching** in a space of possible computer programs for the **fittest** individual computer program
Genetic Programming overview (II)

Random generation

Individuals

Fitness evaluation

Selection

Genetic operations

Replacement
GP overview: individual

Each individual is a parse tree representing a program or mathematical expression

```c
double doSomething(n, d) {
    if (d!=0)
        return n/d;
    else
        return 0;
}
```

\[
F(x,y) = (x+1)(x-10) + xy
\]
GP overview: main params

1. Functions and terminals
2. Fitness function
3. Stop criterion for iterative process

- Chosen basing on the problem knowledge and domain
Scenario

- We base on an **Anomaly-based defacement detection** system that we developed earlier *(IEEE Internet Computing Nov-Dec 2006)*
  - Monitors many remote web pages at regular intervals and raises an alert when a page deviates from its profile
  - Modular
Detector: how it works

- **Sensor**
  - Functional block
  - Quantifies page features

- **Aggregator**
  - Builds the profile
  - Detects deviations from profile

---

**S1**: # of links

Output: [23]

**S2**: relative frequencies of chars

Output: [0.03, 0.12, ..., 0.01]

Input: [23, 0.03, 0.12, ..., 0.01, ...]

Output: normal
Sensors

- In our prototype
  - 43 sensors
  - Grouped in 5 “categories”
  - Producing a numerical vector with 1466 elements
- “Many” page features from “many” points of view
Effectiveness indexes

- False positive rate (FPR)
- False negative rate (FNR)
GP in our tool (I)

- **Goal**: finding an optimal detection formula:
  - $f(v) = f(\{v_1, \ldots, v_{1466}\})$
  - $f(v) < 0 \implies$ negative reading
  - $f(v) \geq 0 \implies$ positive (anomalous) reading

- With low **FPR** and low **FNR**
GP in our tool (II)

- **Terminals:**
  - subsets of elements of vector $v$ output by the refiner, obtained with **feature selection**
  - constants $\{0, 0.1, 1\}$

- **Functions:** subsets of $\{+, -, *, /, \text{unary-}, \min, \max, \leq, \geq\}$

- **Fitness function:** $f = \text{FPR} + \text{FNR}$ on the learning set

- **Stop criterion:** $f=0$ or 100 iterations done
Feature selection (I)

- We aim at selecting best vector elements
- **Key idea**: basing on absolute correlation
  - $X_i$: random variable of $i$-th element $v_i$
  - $Y$: random variable of the desired output (0 or 1)
  - $c_i$: absolute correlation of $X_i$ with $Y$
  - $c_{i,j}$: absolute correlation of $X_i$ with $X_j$
Feature selection (II)

● Iterate on:
  1. Select the element $i$ with the highest $c_i$
  2. “Correct” the other elements: $c_{i,j} = c_{i,j} - c_i$

● In other words:
  1. Select the elements with the highest correlation with desired output but...
  2. … discard “duplicate” elements
Experiments: aggregators

● We compare the GP aggregator…
  ○ With 5 different function sets
  ○ With 5 different sizes for feature selection
● …to an anomaly-based aggregator that we developed earlier
  ○ Bases on domain knowledge
  ○ “Does not know” positive readings
● 25+1 aggregators
Experiments: data set

- **Negative readings**
  - 15 web pages observed for...
  - …1 month and downloaded...
  - …every 6 hours
  - totaling 125 reading for each page

- **Positive readings**
  - 75 readings extracted from a publicly available attacks (defacements) archive
Experiments: methodology

- For each page, for each aggregator
  1. We build a $S_{\text{learning}}$ with 50 negatives and 20 defacements
  2. We build a $S_{\text{testing}}$ with 75 negatives and 75 defacements
  3. We train the aggregator on $S_{\text{learning}}$
  4. We compute FPR and FNR on $S_{\text{testing}}$
Experiments: results

- GP performs better, but…
- … it is never really engaged
  - Only 1 generation
  - Trivial parse trees:
    \[ f(v) = 3v^{1233} - 15 \]
- Large attacker space!

<table>
<thead>
<tr>
<th>Aggregator</th>
<th>FPR</th>
<th>FNR</th>
<th>( f )</th>
<th>( n_p )</th>
<th>( t_s )</th>
<th>( t_h )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomaly</td>
<td>1.42</td>
<td>0.09</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GP-10-( F_1 )</td>
<td>0.00</td>
<td>0.71</td>
<td>0.0</td>
<td>1.1</td>
<td>20.4</td>
<td>3.5</td>
</tr>
<tr>
<td>GP-10-( F_2 )</td>
<td>0.09</td>
<td>0.98</td>
<td>0.0</td>
<td>1.0</td>
<td>23.3</td>
<td>3.7</td>
</tr>
<tr>
<td>GP-10-( F_3 )</td>
<td>0.09</td>
<td>0.62</td>
<td>0.0</td>
<td>1.0</td>
<td>20.7</td>
<td>3.7</td>
</tr>
<tr>
<td>GP-10-( F_4 )</td>
<td>4.53</td>
<td>0.44</td>
<td>0.0</td>
<td>1.0</td>
<td>27.9</td>
<td>3.9</td>
</tr>
<tr>
<td>GP-10-( F_5 )</td>
<td>0.09</td>
<td>0.89</td>
<td>0.0</td>
<td>1.0</td>
<td>18.2</td>
<td>2.6</td>
</tr>
<tr>
<td>GP-20-( F_1 )</td>
<td>0.09</td>
<td>1.16</td>
<td>0.0</td>
<td>1.0</td>
<td>12.8</td>
<td>2.4</td>
</tr>
<tr>
<td>GP-20-( F_2 )</td>
<td>0.18</td>
<td>1.33</td>
<td>0.0</td>
<td>1.0</td>
<td>20.1</td>
<td>3.1</td>
</tr>
<tr>
<td>GP-20-( F_3 )</td>
<td>0.36</td>
<td>0.80</td>
<td>0.0</td>
<td>1.0</td>
<td>36.5</td>
<td>4.2</td>
</tr>
<tr>
<td>GP-20-( F_4 )</td>
<td>0.00</td>
<td>0.89</td>
<td>0.0</td>
<td>1.0</td>
<td>39.5</td>
<td>3.9</td>
</tr>
<tr>
<td>GP-20-( F_5 )</td>
<td>0.00</td>
<td>0.89</td>
<td>0.0</td>
<td>1.0</td>
<td>19.3</td>
<td>2.9</td>
</tr>
<tr>
<td>GP-50-( F_1 )</td>
<td>0.00</td>
<td>1.24</td>
<td>0.0</td>
<td>1.0</td>
<td>5.1</td>
<td>1.6</td>
</tr>
<tr>
<td>GP-50-( F_2 )</td>
<td>0.09</td>
<td>0.98</td>
<td>0.0</td>
<td>1.0</td>
<td>20.4</td>
<td>2.9</td>
</tr>
<tr>
<td>GP-50-( F_3 )</td>
<td>0.36</td>
<td>0.98</td>
<td>0.0</td>
<td>1.0</td>
<td>19.3</td>
<td>2.9</td>
</tr>
<tr>
<td>GP-50-( F_4 )</td>
<td>0.18</td>
<td>0.89</td>
<td>0.0</td>
<td>1.0</td>
<td>15.4</td>
<td>3.1</td>
</tr>
<tr>
<td>GP-50-( F_5 )</td>
<td>0.18</td>
<td>0.27</td>
<td>0.0</td>
<td>1.0</td>
<td>29.4</td>
<td>3.0</td>
</tr>
</tbody>
</table>
Experiments 2: improvement

- **Key idea**: using both defacements and genuine readings of other pages as positives
- **S\textsubscript{Learning}**: 50 negatives + 20 defacements + 14 other pages readings
- **S\textsubscript{Testing}**: 75 negatives + 75 defacements + 70 other pages readings
- More demanding test
Experiments 2: results

- GP still performs better...
- …and really works:
  - Many generations
  - Non trivial parse trees
- Feature selection is not necessary
Experiments: computation times

- Tuning procedure:
  - 100 s for GP aggregator (of which 5 sec in feature selection)
  - 10 ms for anomaly-based aggregator

- Single reading evaluation
  - 500 μs for GP
  - 100 μs for anomaly-based aggregator
- Questions?

- Thanks!