Predicting Vulnerable Components: Software Metrics vs Text Mining

James Walden$^1$  Jeff Stuckman$^2$  Riccardo Scandariato$^3$

$^1$Northern Kentucky University, USA  
$^2$University of Maryland  
$^3$KU Leuven

International Symposium on Software Reliability Engineering (ISSRE 2014)
Topics

Web Security
Prediction
Data Set
Methodology
Results
Conclusions
Extras
Millions of websites hit by Drupal hack attack

31 October 2014 Last updated at 11:30 GMT

Millions of sites are managed and updated using Drupal

Up to 12 million websites may have been compromised by attackers who took advantage of a bug in the widely used Drupal software.

Related Stories
Our Applications
Our research questions were to determine whether

1. \( \text{IsVulnerableFile} \equiv f(\text{SWmetrics}) \)

2. \( \text{IsVulnerableFile} \equiv f(\text{Textminingfeatures}) \)

and if such models could be constructed, did one model type perform differently than the other

\( R_{SM} \equiv R_{TM} \)
Software Metrics

1. *Lines of code (LOC)*
2. *Lines of code (non-HTML)*
3. *Number of functions*
4. *Cyclomatic complexity*
5. *Maximum nesting complexity*
6. *Halstead’s volume*
7. *Total external calls*
8. *Fan-in*
9. *Fan-out*
10. *Internal functions or methods called*
11. *External functions or methods called*
12. *External calls to functions or methods*
T_BREAK: 3,
T_CASE: 1,
T_ELSEIF: 1,
T_IF: 2,
T_ISSET: 1,
T_OPEN_TAG: 5,
T_PRINT: 3,
T_REQUIRE_ONCE: 2,
T_STRING: 5,
T_SWITCH: 1,
T_VARIABLE($return): 1
1. Fewer vulnerabilities than defects reported.
2. Fewer projects track vulnerabilities than defects.
3. The “many eyes” effect works differently:
   - Defects cause problems for users, who report them.
   - Vulnerabilities triggered by odd input sequences rarely entered by users.
   - Vulnerabilities offer opportunities to threats, who therefore keep them secret.
4. Different skills required to find vulnerabilities.
5. Developers limit vulnerability information to reduce opportunities for exploitation.
6. No repository of standard vulnerability prediction data sets for model evaluation.
### Application Data

<table>
<thead>
<tr>
<th>Application</th>
<th>Vulnerabilities</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drupal</td>
<td>97</td>
<td>30 (all 6.x)</td>
</tr>
<tr>
<td>Moodle</td>
<td>51</td>
<td>71 (all)</td>
</tr>
<tr>
<td>phpMyAdmin</td>
<td>75</td>
<td>95 (since 2.2.0)</td>
</tr>
</tbody>
</table>

For each application, the data set includes:

- Source code for each included version.
- Source code metrics for each PHP file for each version.
- Mapping of vulnerabilities to files in each version.
- Vulnerability identifiers from vendor or CVE.
Vulnerability lifetime.

1. Vulnerability introducing commit.
2. Inclusion of vulnerable commit in a released version.
3. Persistence of vulnerability in code, potentially moving to different files through renames or refactorings.
4. Vulnerability fixing commit.
5. Inclusion of fixing commit in a released version.
Vulnerability Location Process

1. Determine main release branch of application.
2. Identify commit which fixed the vulnerability on main branch.
3. Construct indicator string by examining code containing vulnerability.
4. Locate first version where indicator string appears.
5. Trace code changes from introduction to fix to validate vulnerability presence.
Building the Software Metrics Model

1. **PHP Source Code**
2. **Manual Location**
3. **Vulnerabilities per file**
4. **>0**
5. **PHP file classification**
6. **Machine learning**
7. **PHP file metrics**
8. **Metrics**
9. **Prediction model**
Building the Text Mining Model

1. **PHP Source Code**
   - **Tokenizer**
   - **Bag of words**

2. **Manual Location**
   - **Vulnerabilities per file**
   - **>0**
   - **PHP file classification**

3. **Machine learning**
   - **Prediction model**
### Performance Indicators

<table>
<thead>
<tr>
<th>Recall</th>
<th>Inspection Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R = \frac{TP}{TP+FN} )</td>
<td>( I = \frac{TP+FP}{TP+TN+FP+FN} )</td>
</tr>
</tbody>
</table>

- \( R \): (% of vulnerable files found)
- \( I \): (% of files to inspect to be possible to find TP)
## Undersampling

<table>
<thead>
<tr>
<th></th>
<th>Vulnerable files</th>
<th>Total files</th>
<th>P-rate (%)</th>
<th>Text features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drupal</td>
<td>62</td>
<td>202</td>
<td>30.68</td>
<td>3886</td>
</tr>
<tr>
<td>PHPMyAdmin</td>
<td>27</td>
<td>322</td>
<td>8.39</td>
<td>5232</td>
</tr>
<tr>
<td>Moodle</td>
<td>24</td>
<td>2942</td>
<td>0.82</td>
<td>18306</td>
</tr>
</tbody>
</table>

Positive rates (percentage of vulnerable files) are low above, so we undersample to create a balanced training set by

- Keeping all positives.
- Randomly selecting a number of negatives equal to the number of positives.

We used Weka’s SpreadSubSample filter to undersample.
Hypotheses

Our null hypotheses are that the recall and inspection metrics will not differ between software metrics (SM) and text mining (TM).

\[ H_0^R : \mu\{R_{SM}\} = \mu\{R_{TM}\} \]
\[ H_0^I : \mu\{I_{SM}\} = \mu\{I_{TM}\} \]

We use a Wilcoxon rank-sum non-parameter test for independent samples with a significance level of 0.05 for determining if a difference exists.
Testing the Models

1. **PHP Source Code**
   - **Tokenizer**
   - **Bag of Words**
   - **Prediction Model**
   - **Predicted Classification**

2. **Manual Location**
   - **Vulnerabilities in each PHP file**
   - **>0**
   - **Observed Classification**

3. **Performance Indicators**

---

**Flowchart Description**:
- PHP Source Code is tokenized.
- Tokens are converted into a Bag of Words.
- The Bag of Words is used in the Prediction Model to predict vulnerabilities.
- Observed classification is compared with the predicted classification.

---

**Keywords**:
- Web Security
- Prediction
- Data Set
- Methodology
- Results
- Conclusions
- Extras
## Cross-validation results

<table>
<thead>
<tr>
<th></th>
<th>Indicator</th>
<th>SW metrics (%)</th>
<th>Text mining (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Drupal</strong></td>
<td>Recall</td>
<td>µ 76.9</td>
<td>σ 2.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>σ 3.3</td>
</tr>
<tr>
<td></td>
<td>Inspection</td>
<td>µ 45.5</td>
<td>σ 2.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>σ 2.3</td>
</tr>
<tr>
<td><strong>PHPMyAdmin</strong></td>
<td>Recall</td>
<td>µ 66.3</td>
<td>σ 12.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>σ 9.8</td>
</tr>
<tr>
<td></td>
<td>Inspection</td>
<td>µ 42.0</td>
<td>σ 2.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>σ 4.6</td>
</tr>
<tr>
<td><strong>Moodle</strong></td>
<td>Recall</td>
<td>µ 70.4</td>
<td>σ 10.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>σ 6.1</td>
</tr>
<tr>
<td></td>
<td>Inspection</td>
<td>µ 32.1</td>
<td>σ 4.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>σ 3.7</td>
</tr>
</tbody>
</table>
Conclusions


2. Effective vulnerability prediction models can be developed for web applications.

3. Text mining outperformed software metrics with
   - Significantly better recall.
   - Almost same inspection rate (cost).

4. Models built on one application could not effectively predict vulnerable files in other applications.
1. Build prediction models that predict vulnerable components in future versions.
2. Build prediction models based on vulnerability categories.
3. Include data for Drupal 5.x and possibly Drupal 7.x.
Extra Slides
## Vulnerabilities by Class

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Drupal</th>
<th>Moodle</th>
<th>PHPMyAdmin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Injection</td>
<td>2</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>CSRF</td>
<td>8</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>XSS</td>
<td>32</td>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>Path Disclosure</td>
<td>0</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>Authorization issues</td>
<td>39</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>Other</td>
<td>16</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
### Additional performance metrics

<table>
<thead>
<tr>
<th>Indicator</th>
<th>SW metrics (%)</th>
<th>Text mining (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drupal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>52.0</td>
<td>57.1</td>
</tr>
<tr>
<td>FP rate</td>
<td>31.6</td>
<td>26.9</td>
</tr>
<tr>
<td>Accuracy</td>
<td>71.0</td>
<td>75.4</td>
</tr>
<tr>
<td>PHPMyAdmin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>13.2</td>
<td>14.3</td>
</tr>
<tr>
<td>FP rate</td>
<td>39.8</td>
<td>40.6</td>
</tr>
<tr>
<td>Accuracy</td>
<td>60.7</td>
<td>60.6</td>
</tr>
<tr>
<td>Moodle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>1.8</td>
<td>2.3</td>
</tr>
<tr>
<td>FP rate</td>
<td>31.7</td>
<td>28.3</td>
</tr>
<tr>
<td>Accuracy</td>
<td>68.3</td>
<td>71.8</td>
</tr>
</tbody>
</table>