Using Evolutionary Diversity to Identify Problematic Software Parameters

Errin W. Fulp, H. Donald Gage, and Matthew R. McNiece

WAKE FOREST UNIVERSITY
Department of Computer Science

BioSTAR Workshop, 2017
Software Configurations

- Behavior of a software system often depends on how it is configured
  - Occur on a variety of platforms, from mobile-devices to servers
- Configuration options allow the user to customize behavior
  - Allows different deployments and ability to tune performance
  - However this flexibility also has a cost...
Misconfigurations

- Misconfigurations are a continual source of problems$^a$
  
  Majority of failures in Yahoo Zookeeper service have been due to misconfigurations
  Misconfiguration caused Microsoft’s Azure unavailability for 2.5 hours
  Large number of Google production issues associated with misconfigurations
    .se domain unavailable for 1 hour due to DNS misconfiguration
  Facebook inaccessible for about 2 hours due to misconfiguration

- Many attacks can be attributed to poor software configurations
  - Most prevalent vulnerability for web-services

---

$^a$ “Proactive Detection of Inadequate Diagnostic Messages for Software Configuration Errors,” Zhang and Ernst, ACM ISSTA 2015
Security Checklists

• Security guidelines (e.g. DISA STIG and NIST checklists) are helpful
  – Settings are often too basic, may be out-of-date, or not applicable

• Consider DISA STIG recommendation changes (fixes?) over time

<table>
<thead>
<tr>
<th>Application</th>
<th>Additions</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache2.2 from 2011 until 2015</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>RHEL5.0 from 2012 until 2015</td>
<td>4</td>
<td>135</td>
</tr>
</tbody>
</table>

• Could there be alternative functional and secure settings?
Configuration(s) Search

• Consider all the possible configurations for a software system

• Possible to apply a search algorithm, but it is a difficult problem
Difficulty of the Problem

- Consider a single configuration with \( n \) parameters

<table>
<thead>
<tr>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeepAlive, allow requests over the same connection</td>
<td>0, 1</td>
</tr>
<tr>
<td>KeepAliveTimeout, wait time to wait for requests</td>
<td>0 - 1800</td>
</tr>
<tr>
<td>Indexes, automatic directory indexing</td>
<td>0, 1</td>
</tr>
<tr>
<td>LimitRequestBody, limit the message size</td>
<td>0 - 65535</td>
</tr>
<tr>
<td>LimitRequestFields, limit number of HTTP requests</td>
<td>1, 0</td>
</tr>
<tr>
<td>LimitRequestFieldSize, limit HTTP header field size</td>
<td>0 - 32767</td>
</tr>
<tr>
<td>max file handles /proc/sys/fs/file-max</td>
<td>0 - 10000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Many different setting combinations
- Parameters may be interdependent, forming *parameter chains*
- Finding a secure configuration is similar to the satisfiability problem
Evolutionary Algorithms

• Evolutionary Algorithms (EAs) are used as search heuristics
  – Better solutions are created from good solutions

• Typically given a cyber security problem, there isn’t one solution
  – Diversity and adaptability of EAs are helpful
Evolutionary Algorithm Components

- Chromosome – a model for a solution to the problem
- Fitness – value for ranking, we seek *better solutions*
- Processes – combining solutions to create new one
- Pool – set of chromosomes (*also called a population*)
Modeling Configurations as Chromosomes

- Configuration consists of multiple parameters
  - Individual configuration parameter settings are chromosome traits

<table>
<thead>
<tr>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeepAlive, allow requests over the same connection</td>
<td>0, 1</td>
</tr>
<tr>
<td>KeepAliveTimeout, wait time to wait for requests</td>
<td>0 - 1800</td>
</tr>
<tr>
<td>Indexes, automatic directory indexing</td>
<td>0, 1</td>
</tr>
<tr>
<td>LimitRequestBody, limit the message size</td>
<td>0 - 65535</td>
</tr>
<tr>
<td>LimitRequestFields, limit number of HTTP requests</td>
<td>1, 0</td>
</tr>
<tr>
<td>LimitRequestFieldSize, limit HTTP header field size</td>
<td>0 - 32767</td>
</tr>
</tbody>
</table>

\[ C = \{1, 800, 1, 32767, 0, 1\} \]

- Some parameters settings will affect security, however which are security related is possibly unknown (*more on this later*)

- Therefore a chromosome will be a configuration

- Chromosomes will be be ranked, need a measure of *fitness*
Chromosome (Configuration) Fitness

- Fitness is based assessing the attacks encountered

- We have investigated three approaches
  1. Common Vulnerability Scoring System (CVSS) 6 part vector

<table>
<thead>
<tr>
<th>CVE</th>
<th>CVSS Vector</th>
<th>CVSS Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default ssh password</td>
<td>AV:N/AC:L/Au:N / C:P/I:P/A:P</td>
<td>7.2 (high)</td>
</tr>
</tbody>
</table>

- Attack signatures, count the **types** of attacks
- Simplest fitness indicates if the system was hacked (**binary fitness**)
Evolutionary Algorithm Processes

- Selection identifies parents for new chromosomes (configurations)
- Crossover combines selected chromosomes to create new chromosomes
- Randomly change traits (configuration parameter settings)
Evolving Configurations

Continually iterates and updates configurations based on environment.
Resolving Misconfigurations

Linux Apache configuration consisting of 200 parameters, 12 of which are misconfigured. Two attacks occur (red and blue), each targeting a unique set of 6 misconfigurations [SafeConfig’15].

- EA finds good configurations (settings for all parameters)
  - Perhaps only a small set of parameters are misconfigured
- Prefer to identify and manage only problem parameters
  - Remaining parameters can be set based on performance
  - Identification is also beneficial to administrators
Comparing Configurations

EAs can find multiple diverse solutions (multi-stability).

- Additional information can be gained from comparing configurations

- Can identify vulnerabilities using cross-generational information
  - Settings that consistently appear with good/bad configurations
Comparing Configuration Settings

- **Compare secure configurations, vulnerable parameters always have secure settings, other parameters can change**

- Maintain counters per each setting
  - Increment/decrement if part of a good/bad configuration
  - Good settings have positive counts, bad settings are negative

Create a counter for each setting of each parameter
Initialize setting-counters to zero

Given an assessed configuration
foreach setting used in the configuration
  if the configuration is secure then
    Increment the associated setting-counter by 1
  else
    Decrement the associated setting-counter by 1

- More diverse configurations provides faster identification
Diversity

EA can find diverse secure configurations, assuming multiples solutions exist.
Experimental Results

• Interested in the effect of different size configurations
  – Configurations with different number of parameters
  – Configurations with different number of settings per parameter

• All configurations had 6 independent misconfigurations
  – Measured the number of generations (EA iterations) required to resolve and identify all 6 problematic parameters
  – 50 independent experiments performed per configuration and average performance recorded
Configuration Size

- 6 of settings per parameter, different number of parameters

- Resolved vulnerabilities in 10 generations, identified in 30 generations
  - Configuration size did not affect resolution or identification

- Additional configurations needed to identify the errors
  - EA provides diverse solutions over multiple generations
Number of Parameter Settings

- 100 parameter configurations, with different number of settings

Number of Generations

Resolution with Different Settings

Number of Generations

Identification with Different Settings

- Number of settings did effect the of generations required
  - More settings required more generations to resolve and identify

- Additional configurations needed to identify the errors
Conclusions and Future Work

- Software configuration management remains a difficult problem
  - Configurations are increasing in size and complexity

- Evolutionary algorithms can identify correct configurations
  - However all parameters are configured, not just problem parameters

- Possible to compare configurations to find common settings
  - If configurations are diverse, then they share parameters of interest
  - Simulation results indicate the approach can resolve and identify

- Several areas for future work
  - More experiments with complex configurations (chains)
  - Use machine learning approaches for parameter identification
Turn back, you’ve gone too far