Context-based Online Configuration Error Detection

*Ding Yuan*§, *Yinglian Xie*¶, *Rina Panigrahy*¶,
*Junfeng Yang*†, *Chad Verbowski*¶, *Arunvijay Kumar*¶

¶Microsoft Research, §UIUC and UCSD, †Columbia University,
Motivation

- Configuration errors are caused by erroneous settings in the software system.

---

**Misconfiguration Brings Down Entire .SE Domain in Sweden**

Oct 13, 2009 9:32 AM PDT | Comments: 1 | Views: 3,718

An incorrect configuration within Sweden's .SE zone caused temporary shutdown of all websites under the country code top-level domain. The configuration registry did not add a terminating “.” to DNS records...
Motivation

- Configuration errors are caused by erroneous settings in the software system
- Huge impact
- Configuration error is a major root cause of today’s system failures
  - 25% - 50% of system outages are caused by configuration error [Gray85, Jiang09, Kandula09]
  - This percentage is likely increasing
Existing Work

- Existing work focused on configuration error diagnosis
  - ConfAid [Attariyan10]
  - AutoBash [Su07]
  - Finding the Needle in the Haystack [Whitaker04]
  - PeerPressure [Wang04]
  - Self history constraint [Kiciman04]

Require manual error detection
Early Detection of Configuration Error

Why we need early detection?

- Prevent error propagation
- Hints for failure diagnosis
- Especially useful in monitoring servers

*Our goal*: Automatically Detect Configuration Errors
Early Detection of Configuration Errors

Why we need early detection?

- Prevent error propagation
- Hints for failure diagnosis
- Especially useful in monitoring servers

Windows Auto-Update disabled

- Stacked by malware

Security Alert
I am getting security alerts...

- It looks like you might be having a malware
- Seems my Windows Update was disabled long ago...
Challenge

- First thought: report any configuration change
  - $10^4$ writes/day per machine to Windows Registry
    - Majority are modifications to temporary Registry
Challenge

- First thought: report any configuration *change*
  - $10^4$ writes/day per machine to Windows Registry
    - Majority are modifications to temporary Registry
- Only monitor the changes to ‘important’ configuration?
  - Too complicated: 200K Registry entries on single machine [WangOSDI04]

Change user previledge
Our Observations

- Only those configurations that are *read* matter
  - Analyze read — configuration *access event*

![Diagram with read process and configuration data]

Auto-update process
Our Observations

- Only those configurations that are *read* matter
  - Analyze read — configuration *access event*
- Event sequences are repetitive and predictable
  - Externalize program’s control flow
  - Report deviation from repetitive sequence
Contributions

- **CODE**: online configuration error detection tool
  - Effective: detect configuration errors on-the-fly
  - Comprehensive: automatically monitor all the processes in OS (including kernel processes)
  - Reasonable false positive rate
  - Rich diagnostic information
  - Low overhead: < 1% CPU usage for 99% of time
Outline of the talk

- Motivations
- Background and Example
- Design and implementation
- Evaluation
- Related Work
- Limitations
- Conclusion
## Windows Registry

- **Centralized configuration storage**
  - Software, hardware and user settings
  - Key-Value pair
  - Standard interfaces for access Registry

### Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\Software\Policies\...\WinUpdate\AutoUpdate</code></td>
<td>True</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Windows Registry

- Centralized configuration storage
  - Software, hardware and user settings
  - Key-Value pair
- Standard interfaces for access Registry

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Software\Policies...WinUpdate\AutoUpdate</td>
<td>True</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Auto-Update Example

svchost.exe

28 events as the context

QueryValue

OpenKey

...WinUpdate\ ...

...WinUpdate \UpdateServer

http:// ...

...WinUpdate\AutoUpdate True

29th event

Periodically checks for Windows update.
Auto-Update Example – Error case

28 events in the context

Warning

svchost.exe

OpenKey

QueryValue

QueryValue

QueryValue

...WinUpdate\...

...WinUpdate\UpdateServer

http://...

...WinUpdate\AutoUpdate

Only when the modified Registry entry is read!

**Expected:** AutoUpdate = True

**Observed:** AutoUpdate = False

**Modified by:** explore.exe, at 2:03 PM, 4/6/2011
Design Overview

Event collection module

Extract frequent event sequences

Generate rules
  abc -> d
  abcd -> f

Learning

Rule: a b c -> d

Everytime ‘a b c’ occurs, ‘d’ will follow immediately

Analysis module
**Design Overview**

Event collection module

- **Epoch i**
  - Extract frequent event sequences
  - Generate rules: $abc \rightarrow d$
  - $abcd \rightarrow f$

- **Epoch i+1**
  - Match events against rules
  - Diagnose
    - Expected: $abc \rightarrow d$
    - Observed: $abc \rightarrow e$

Update

Analysis module

Learning

Detection

Rules

Time

Learning

Rules
Event Collection

- Monitor the configuration access events
  - Sequences faithful to the program’s control flow
  - Based on FDR [Verbowski08]
  - Negligible runtime & space overhead

```
<table>
<thead>
<tr>
<th>Thread 1</th>
<th>e₁, e₂, e₃, …</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 2</td>
<td>…, …</td>
</tr>
</tbody>
</table>
```

```
All processes
iexplore.exe arg1
  arg2
svnhost.exe
  …, …
```
Learn the frequent sequences

- Frequent Sequence Mining
  - **Efficiency**: streaming based method
- Sequitur algorithm [Manning97]
  - Streaming algorithm
  - Flexible pattern length

```
abcda b d abcdab cdf g f gh
```

- \( R_1 \): a b -- 5 times
- \( R_2 \): a b c d – 2 times
- \( R_3 \): a b c d a b – 2 times
Deriving Context -> Event rules

- Put every frequent sequence into a prefix tree

Sequence 1: a b c d
Sequence 2: f g h
Sequence 3: f k

Represents ‘ab -> c’

Each node is an event

Each edge might represent a rule

Only edges that are the only outgoing edge from the origin node are candidates to represent a rule
Deriving *Context* -> *Event* rules

- Not every candidate edge represents a rule

One Prefix Tree for all the processes launched by the same process name and argument.
Error Detection

- Report rule edge violation
  - Match incoming events against prefix tree

Report an error!

.. a b c e ..

Represents ‘abc -> d’

A few heuristics to suppress false positives
What is the expected event

Help to recover from the error

.. a b c e ..
**Diagnostic Information**

- What is the expected event
  - Help to recover from the error
- The context of the violation
  - Understand the error

... a b c e ...
Diagnostic Information

- What is the expected event
  - Help to recover from the error
- The context of the violation
- Which process modified the Registry that caused the error? And when?
  - Write buffer
- Examine the side effect of rolling back the Registry to its old data
  - All the other rules involving the new Registry data
Evaluation methodology

- False negative rate
  - Real configuration errors
  - Error injection

- False positive rate
  - Deployed on 10 actively using desktops and a server cluster with 8 servers running

- Performance
How many real world errors do we catch?

<table>
<thead>
<tr>
<th>Error Description</th>
<th>machines reproduced</th>
<th># of cases detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>explorer-double-click</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>ie-advanceoptions</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>ie-search</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ie-smbrandbitmap</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ie-brandbitmap</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ie-title</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>explorer-policy</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>explorer-shortcut</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>ie-password</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>ie-workoffline</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>outlook-emptytrash</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>42</strong></td>
<td><strong>41</strong></td>
</tr>
</tbody>
</table>

Missing only 1 out of 42
Exhaustive Registry Corruption

- Exhaustively corrupted every Registry Key frequently accessed by Internet Explorer
  - Among 387 successfully corrupted Keys, CODE detected 374 (97%) of them
- CODE can effectively detect most of the Registry related configuration errors
False Positive Rate

- Deployed on 10 actively used desktop machines, 8 production servers
  - Over 30 days
  - Includes 78 software updates

<table>
<thead>
<tr>
<th>Warnings/day</th>
<th>Average</th>
<th>Max</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>0.06</td>
<td>0.27</td>
<td>0</td>
</tr>
<tr>
<td>Desktop</td>
<td>0.26</td>
<td>0.96</td>
<td>0</td>
</tr>
</tbody>
</table>
Performance

- In all machines, CPU overhead is negligible
  - 1% over 99% of time
  - 10% - 25% peak usage
Performance

- In all machines, CPU overhead is negligible
- Memory Usage between 500MB – 900MB
- We can use one CODE process to monitor multiple servers with similar configuration setting

![Graph showing Memory Usage (MB) vs. Number of servers monitored. The graph indicates a 7% increase in memory usage as the number of servers monitored increases.]
Related work

- Configuration error diagnosis
  - Key value pair based approaches [Wang04, Kiciman04]
  - Virtual Machine based [Whitaker04]
  - ConfAid [Attariyan10]
  - AutoBash [Su07]
- Sequence Analysis [Hofmeyr98, Wagner01]
  - Used in security
  - Different design
- Bug detection tools using symbolic execution
  - KLEE [OSDI08]
Limitations

- Cannot detect errors during installation
- Windows only
  - Key challenge on other systems: intercepting configuration accesses
- Still non-zero false positive rate
  - Limitation in truly differentiate user’s rare intentional changes from errors
Conclusion

- CODE: Automatic online configuration error detection tool
  - Simple observation: key configuration access events form highly repetitive sequence
  - Effective and Efficient
Thanks
## Top five causes for False Positives

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Association</td>
<td>The default program used to open different file types is changed.</td>
<td>24.1%</td>
</tr>
<tr>
<td>MRU List</td>
<td>Changes to most recently accessed files tracked by applications (e.g., explorer and IE)</td>
<td>12.7%</td>
</tr>
<tr>
<td>IE Cache</td>
<td>The meta-data for the IE Cache entities is changed.</td>
<td>3.8%</td>
</tr>
<tr>
<td>Session</td>
<td>The statistics for a user login session is updated</td>
<td>3.8%</td>
</tr>
<tr>
<td>Environment Variable</td>
<td>Environment Variable Changes</td>
<td>2.5%</td>
</tr>
</tbody>
</table>

Intentional configuration change that occurs infrequently
Impact of Software Updates

- During the month-long deployment on 10 desktops, only 5 warnings were due to software Updates (out of total 78)
  - 2 environment variable updates, one display icon update, one DLL update, one daylight saving time
- There was one most intrusive update
  - Office update from SP2 to SP3
  - 200 patches, modified 20,000 keys
  - Only 10 keys overlapped with CODE’s rule, causing only 1 warning
Comparison with state-based approach

<table>
<thead>
<tr>
<th>Num/day/machine</th>
<th>CODE</th>
<th>State-based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Max</td>
</tr>
<tr>
<td>Server</td>
<td>0.06</td>
<td>0.27</td>
</tr>
<tr>
<td>Desktop</td>
<td>0.26</td>
<td>0.96</td>
</tr>
</tbody>
</table>