If It Ain't Broke, Don't Fix It: Challenges and New Directions for Inferring the Impact of Software Patches

Jon Oberheide, Evan Cooke, Farnam Jahanian
University of Michigan

May 19th, 2009

HotOS XII
We can automatically infer the impact a patch will have on a software system.*

* DISCLAIMER: Prices and participation may vary. Additional charge for extra meat or cheese. Offer not valid in Switzerland.
**What is a Patch?**

A short set of commands to correct a bug in a computer program

**WolframAlpha**

<table>
<thead>
<tr>
<th>Definitions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun:</td>
</tr>
<tr>
<td>dressing</td>
</tr>
<tr>
<td>cloth covering</td>
</tr>
<tr>
<td>sewing</td>
</tr>
<tr>
<td>piece of material</td>
</tr>
<tr>
<td>connective</td>
</tr>
<tr>
<td>marking</td>
</tr>
<tr>
<td>program</td>
</tr>
<tr>
<td>piece of land</td>
</tr>
<tr>
<td>time</td>
</tr>
</tbody>
</table>
Different Interpretations

To a user:

![Image of two people with text about updates ready to install]

To a developer:

```diff
--- a/fs/cifs/connect.c
+++ b/fs/cifs/connect.c
@@ -3756,16 +3756,13 @@
   - kfrees(tcon->nativeFileSystem);
   tcon->nativeFileSystem =
+   kzalloc((length + 1), GFP_KERNEL);  
   - kzalloc((length + 2), GFP_KERNEL);
   if (tcon->nativeFileSystem)
```

To a machine:

![Assembly code diagram]

---

Jon Oberheide - HotOS XII - May, 2009
Our Target Audience

- System administrators
  - Intelligence: more than user, less than developer

- Tasked with patching complex systems
  - Software has bugs, software needs patching
  - Admin determines whether to apply patch

- Problems
  - Limited resources
  - Limited understanding of software system interactions
  - Lots of patches to assess!
Patch Frequency

- 3 Gentoo Linux machines
  - Workstation, data munger, web server
  - 923, 655, and 122 installed software packages
  - 1453 unique packages total

- During 2008 calendar year
  - 2402 new upstream versions
  - 260 working days, 8 hour workday
  - Over 1 update/hour for admin...non-stop!
PATCHES ONLY FIX THINGS, NOT BREAK THEM!

Most do.
But breakage can be very expensive.

MY TEST AND REGRESSION SUITES ARE FLAWLESS!

Not all are.
We aim to complement existing tools.

GET A BRAIN, MORAN!

Does not compute.
Food for Thought

On one extreme…

Would you expend resources to patch an area of code that is never exercised by your application?

On the other extreme…

Would you avoid patching an area of code that is a hot path due to the potential risk?

What about the non-extremes?
How do we balance the risk/benefit trade-off of a patch?
Our Approach

Simple intuition: Modifying more commonly used code will likely have a greater impact (whether positive or negative) on a software system.

- PatchAdvisor
  - Infer patch impact in an automated fashion
  - Combination of static analysis and dynamic tracing

- Our Goal
  - Provide administrator useful information about patch impact (or at least > 0 information!)
  - Enable informed decisions about patching
PatchAdvisor Overview

• Three stages of PatchAdvisor
  • Pre/post-patch CFG generation and diffing
  • Execution tracing and CFG overlay
  • Impact analysis and reporting
Impact Analysis

Three proposed functions to infer patch impact:

- **Naive Binary:**
  - Do we intersect at all with the modified code areas?

- **Trace Weighted:**
  - How often do we intersect with modified code areas?

- **Proximity Ranking:**
  - If we don't intersect, how close are we?
Impact Reporting

Open question: How can we most effectively convey actionable information to the administrator?

- Output of impact metrics?
- List of affected functions?
- List of impacted inputs?
- Risk index based on previous patches and/or failures of a package?
- SVN commits / mailing list messages that correspond to patched machine code?
- ...
Implementation

- **IDA Pro**
  - Disassembler, debugger
  - Intended for hostile code analysis
  - [http://www.hex-rays.com/idapro/](http://www.hex-rays.com/idapro/)

- **Pai Mei**
  - Extensible python framework for RCE

- **CFG generation, binary diffing, func/bb tracing, overlay**

- Not yet fully automated
Preliminary Evaluation

- Web stacks have many layers
  - HTTP server, scripting languages, dispatcher, ORM, backend DB, template engine, etc
- Psycopg2
  - Popular PostgreSQL Python bindings
  - Minor revision upgrade: 2.0.2 → 2.0.3
  - Innocent looking ChangeLog
  - Has a test suite!
- NULL deref when involving any FLOAT column types
Before Patch
Execution Trace
NULL check
After Patch
Future Directions

- Improved ranking heuristics
  - How do programs often fail in the real-world?
- Application-specific knowledge
  - Can we use application/domain-specific information to aid our inference (eg. web apps)?
- Patch splicing
  - Can we determine intra-patch dependencies and splice out high risk changes?
- Patch classification
  - Can we infer whether a patch fixes a semantic bug, performance issue, or security vulnerability?
Questions?

- Contact information
  - Jon Oberheide
  - University of Michigan
  - jonojono@umich.edu
  - http://www.eecs.umich.edu/fjgroup/