Application Crash Vulnerabilities
Thanumalayan Sankaranarayana Pillai, Vijay Chidambaram, Ramnathana Alagappan, Samer Al-Kiswany, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau

File-System Crash Consistency
- Ensures logical consistency of internal metadata
- Important for FS developers and researchers
- Much research, multiple techniques
  - FSC, Soft Updates, Journaling, COW …

Application-Level Crash Consistency
- Applications don't use internal metadata of FS
- What happens to user data during a system crash?
- Maintain application-defined consistency on user data structures: Application-Level Consistency

State of the art
For effective application-level consistency, applications depend on specific details of file-system implementation
- Bad situation
  - Many file systems in use
  - New file systems constantly invented
- Application-level consistency is important
  - Modern applications store many data structures
  - Google Chrome initialization: 500+ files
    - Data structures like page cache, history
    - Cache should have only complete entries
  - Photo application: Thumbnails match pictures

Example: Atomic File Rewrite
- User updates a file
- User wants update to be atomic
  - File should be fully in original state or updated state

Wrong Protocol
fd = creat("temp")
write(fd)
rename("temp", "grub.conf")

- Omitting fsync() might result in a zero-filled file
  - Because FS can re-order write() and rename()
- Wrong protocol is commonly used
- Works under most common file systems
  - Ext4, btrfs etc. explicitly ensure correctness
  - Observation: FS implementation affects applications

Discovered Vulnerabilities

Total Unique Vulnerabilities: 65

Current FS Impact

Patterns

Variation of File System Implementation Details
Persistence Property of a File System (True / False):
Does a system call sequence only result in a given, desirable set of post-crash states?

Safe Rename
Atomic file rewrite is ensured even when omitting fsync() in the wrong protocol

System Call Sequence
fd = creat("temp")
write(fd)
rename("temp", "grub.conf")

Post-Crash states
print "Hello"
kern vmlinuz
in id intrd.img

(or)
grub.conf (Original)
grub.conf (Garbage)

Safe Append
When appending a file, appended part will never contain garbage

System Call Sequence
append(LogA, "1.00 Msg")

Ordered Append
Append calls to files get persisted in ordered

System Call Sequence
append(LogA, "1.00 Msg")
append(LogB, "2.00 FAULT")

Ordered dir-ops
Directory operations (create, unlink, rename …) get persisted in ordered

Safe new file
After fsync() on a new file, another fsync() on the parent directory is not needed

<table>
<thead>
<tr>
<th>File Systems</th>
<th>Safe rename</th>
<th>Ordered append</th>
<th>Ordered dir-ops</th>
<th>Safe append</th>
<th>Safe new file</th>
</tr>
</thead>
<tbody>
<tr>
<td>ext3 – ordered</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ext3 – writeback</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ext4 – ordered</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ext4 – original</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Btrfs</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>