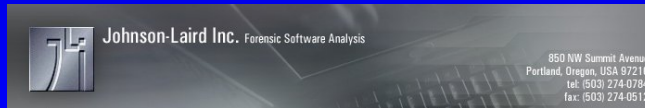




Software Pedigree Analysis: Trust But Verify

Presented by Susan Courtney, Barb Frederiksen & Marc
Visnick



Has something bad happened?

If so, what can I do about it?

Identifying The Risks

- Can you prove when your code was designed and developed?
- Do you identify/record all third-party code?
- Have you preserved a copy of each license?
- Do your developers understand restrictions placed on the use of third-party material?



What I'm Going to Talk About...

- Quick technology refresher
- Derivation and distribution
- Types of open source licenses
- Common development practices



My focus is on license function, not underlying ideology

Technology Refresher - Linking

- How to create an executable program?
 - Static Linking
 - Dynamic Linking



These are the traditional scenarios

Why Linking Matters...

- Have I created a derivative work?
 - Static Linking
 - Dynamic Linking



Complicated question for software. Have I created in essence one program with multiple parts, or multiple programs “aggregated” together?

Static linking

What if I include your material to which I’ve made modifications?

What if I include your material, but make no actual modifications to your material?

Dynamic linking

What if I have references to your material, but don’t call into your material until my program is run?

What if I never actually end up calling your material?

The Distribution Trigger

- License terms usually kick in on distribution
 - What about internal use?
 - Distribution of product to customers?
 - Web-hosted applications?



Nvidia shim scenario – end-user causes the ultimate loadable kernel module to be created... hence, there is no actual distribution by Nvidia. Should this matter?

Open Source License Types

- Two general categories of open source licenses:
 - Permissive Licenses
 - Reciprocal (“copyleft”) Licenses



Permissive Licenses

- MIT, BSD, Apache License, *etc.*
- Focus is usually on downstream attribution



Code under permissive licenses can generally be used in commercially-licensed applications

Frequently used in commercial applications

Fosters a knowledge commons, but likely a high rate of defection (“free riding”)

Perhaps offset by faster market penetration. Everyone’s using my software! I’m famous (if still poor)!

Reciprocal Licenses

- “Copyleft,” “Share-Alike,” “Viral”
- Examples: GPL, LGPL, Mozilla Public License
- Use my stuff if you’d like, but then you must “share-alike”



You are free to use my materials to create a work based on from my work, but you must redistribute (“share-alike”) the resulting work under same terms as you received the materials

Usually means making modified source code publicly available

Fosters a knowledge commons;
minimizes free riding

Examples of Reciprocal Licenses

- GNU General Public License (“GPL”)
 - Significant marketplace penetration
 - Strong reciprocal terms
 - A work “based on” GPL-licensed work must be redistributed under the terms of the GPL
- Affero GPL – deals with web apps



Terms = Basically, provide source code for the derived work

Various other housekeeping requirements (providing copy of GPL license text, etc.)

In 2008, ~88,000 out of ~129,000 SourceForge projects were licensed under the GPL

Sourceforge stats reflect in part dead or largely dormant projects

Affero GPL – if users can interact with your modified GPL program remotely through a computer network, you must make corresponding source available

Many companies make extensive use of copyleft material “behind the wall” to provide an end-user experience

No distribution = no obligation to release source

Affero GPL v3 tries to close loophole

“Remotely through a computer network”

Code must be made available

Common Development Practices

- Failure to document use of open source
- Code (license) laundering
- Failure to comply with license terms
- Use of “trial” versions of commercial software in production code



Failure to comply with license terms: “It’s OK – it’s only on the back end”

Not Just Open Source...

- Any Third Party Content
 - Source, Object, Media
- Risks include
 - Copyright infringement
 - License breach
 - Trade Secret theft
 - Patent infringement



You Have A Problem If:

- You don't know you are using
- You don't know what you are using
- You don't know the license terms
- You haven't identified where you are using it
or whether you are distributing



Classic Scenario

- Good intentions
- Team hits time, \$\$\$, or feature crunch
- Team “reaches out” for a solution
- No gatekeeper for downstream effects



Case Study

- TLA decides to compete with former “partner”
- Design team includes ex-employee of “partner”
- Developers hit a roadblock and ask ex-employee for possible solutions
- Ex-employee reaches out to old chums/documents
- Result: contaminated code, \$400M lawsuit



The Gate Keeper's Role

- Know what's there
- Know where its used
- Know the rules for reuse
- Know the license obligations
- Keep good records



Best Practices

- Develop and document corporate policy
- Educate developers
- Review policies
- Use a pedigree checklist
- Save development and QA artifacts
- Retain licenses and original source





Questions?

Susan Courtney
susan@jli.com

Barb Frederiksen
barb@jli.com

Marc Visnick
marc@jli.com

