Does IP matter with Open Source Software? – *Video Transcript*

Getting the software supply chain right

Click here for video

Introduction

Helllo, my name is Martin Callinan of Source Code Control. In this module, we're going to talk about the benefits of running a compliance program and introduce some of the industry best practices for managing open source software.

There are a number of benefits to running a compliance program for open source software. Firstly, from an organizational perspective, you can get clear data that gives you an indicator of the costs and benefits to using open source software in software developments. You'd also get increased knowledge of the breadth of solutions that you can leverage for your software developments and ultimately there will be a reduction in the risk or potential risk of infringement of open source licences. And finally, better collaboration with the open software community. This will help in recruiting and retaining software development talent in your organization.

Now there is more than one element to having a successful solution for managing open source software. Typically, it will involve people, processes and technology. Naturally, it is a technical problem, so organizations will probably look for a technical solution but without the people and the processes, ultimately it won't be a successful solution.

In 2017, the Linux foundation, which is the foundation behind the Linux kernel, introduced a project called OpenChain and the chain refers to the supply chain of software. A number of organizations globally, who are

heavily invested in open sourcing software developments came together to try to define the best practices for managing open source software through its supply chain. There are three key elements to OpenChain: First of all, there is a specification, which defines all of the processes required to manage open source software. Secondly, there is a conformance check. So any organization who adopts the specification can be assessed against that definition and be recognized as being conformant to the specification. And finally, there is a curriculum – so a training program to educate individuals within organizations of how to effectively manage open source software.

So the OpenChain specification has five main pillars. First of all is understanding your responsibilities to comply with open source software through the supply chain. Secondly is assigning responsibility for achieving that compliance – so who within an organization needs to be involved within that decision process. Thirdly, is reviewing and approving open source content – so in the previous sections we talked about attribution notices, licence notices and sharing source code – who has approval for those sorts of things within an organization's software development. Fourthly, there is delivering all the accurate and correct content – so, licence notices, offer of the source code, that we've already talked about. And lastly, there is understanding the community engagements, so developers contributing to open source projects; or indeed you turning your open source projects into a community project.

So, there are a number of best practices, which should be completed for an organization complying with open source software.

First of all, scrub for comments – so developers put comments in source code. If ultimately you are going to share your source code, you don't want any personal or derogatory comments in the source code. Include a read-me file, so this just defines what the application has been developed for. And then, talking more about the compliance with open source, a licence file, which is a consolidated list of all the third party components, how they are licensed and any attribution notices, as we talked about in previous modules and also, in each file, so for component, there should be a source code header. If your developers have modified any code, you need to clearly state that there has been modification, what the modifications are, and who did the modifications. Then, if you are distributing your software on a copyleft licence, like the GPL, there is the obligation to share the source code of your software. Now, to be strictly compliant, you need to ship the exact version that matches what the user is using, so you may need things like version control. And finally, if developers are contributing to an open source project, they will probably have to sign a Contributor License Agreement, so there needs to be some structure about how they do that and who is authorised to do that within your organization.

As we covered in a previous module, the Licence Notice will clearly state what the licence is for each component, and either a link to the licence, or the text of the licence.

Likewise, if there is any attribution – so if there is a named developer or organization behind the components of software, that attribution needs to be included in your software as well.

Recently, Google published their open source policy, so how they manage their software developers – and they've included boiler plates of how they licence their software. So, they have a standard licence header – their go-to licence is the Apache licence and there is a clear copyright statement, so for example, 'Copyright 2018, Google LLC.' So all their developers will use their boilerplates. So it's good best practice to be consistent in the way you copyright your software.

Now the challenge with all these attribution notices and licence notices, is that you get inconsistencies across individuals and organizations. To address this challenge, the Linux foundation has also incubated another project, called SPDX or Software Packet Data Exchange. It's just a standard way of formatting licence notices, attribution notices, that are both readable by humans but also by machines as well – and it drives the standardization and makes it easier to manage that supply chain of software. So as software goes downstream, you can just look for SPDX headers rather than different formats of licensing and this reduces the overhead for managing open source software.

With modification, if a developer takes an open source component and makes some changes, where you tweak some of the software to make it work within your application, there is an obligation to include a modification notice, on top of the attribution notice, copyright notice and licence notice.

We covered on the previous module, talking about open source licensing, the offer of the source code. So, to be strict on compliance, you just need to offer the source code, you don't have to distribute the source code unless it is requested – but you are free to do that if you so wish – so somewhere, you need to communicate clearly how an individual or an organization can request the source code and how that is shipped to the end user.

Now with reference Contributor Licence Agreements, and there's two ways this can work for an organization – first of all, it's quite common for developers to contribute to open source projects. There's many high profile projects, such as the Linux kernel, Facebook has an open source program, where they have external people contributing. There's cloud solutions like Cloudera. Now there is a requirement for anyone contributing code to those projects, that they sign a Contributor Licence Agreement, either as an individual, or as an organization. So there are company contributor licence agreements. It's imperative that organizations have a way of managing this within the organization. So if a developer signs a contributor licence agreement, are they signing on behalf of the organization, or are they signing on behalf of themselves? And if they use a work email address, is there any risk associated with that for the organization?

The other side of the fence, is, maybe you want to turn software that you are developing into a community project. In that case, you may want to develop a contributor licence agreement of your own. There are a number of resources available which can guide you through that process. There's GitHub, which is a popular website for sharing code of all sorts and projects. They have a project called CLA hub, which guides you through the process of creating a contributor licence agreement – so it's basically a template. Another similar project is Project Harmony.

So now we've talked about the theory of managing open source software, I just want to talk through an example of how an organization has benefitted from putting in a program based on OpenChain. In the UK, there is the National Health Service, which is a public sector organization for health and NHS England has an organization called NHS Digital. A number of years ago, they started an open source program, called code for health, where they brought together communities of clinicians and communities of developers to work on applications that actually work and make a difference and that is overseen by what is called the Apperta foundation. The Apperta foundation is like the governance for the solutions that are made available for the NHS.

The goal of the NHS in creating an open source program is to create a library of software assets that are freely available to be used, modified, tweaked and re-distrubuted across the NHS, without any vendor locking into a particular supplier. By default, all the code is open source, and is openly shared – it can be leveraged outside the NHS and ultimately this will help manage clinical risk around the software being delivered.

So, as I said, there's this community of both software developers and implementation partners, and a community of clinicians developing applications. Now the NHS is a fragmented organization, there's hospitals and trusts across the UK – each one can be an individual customer. So, at the heart of it is product governance, which is overseen by the Apperta foundation, and this is where OpenChain and open source management comes into play.

So, as an example, there has recently been a case study published and it's all focused around one of the most successful open source applications – and it's one of the first applications that was delivered through Code4Health, it's called OpenEyes, which is an electronic medical records solution for eye care. Now, it's a community of projects and we talked about community projects earlier, but the actual delivery is across both the NHS and organizations around the world. So they wanted to control that supply chain of software, from the individual components developers used, merging into the final solution of OpenEyes, and finally being distributed and deployed in NHS organizations. So how do you trust that supply chain?

So Code4Health turned to OpenChain as a way of validating, they've got rigorous processes for managing open source risk. The journey for Code4Health started with education. There was a lack of knowledge across different disciplines involved in this process – so we talked about clinicians, we talked about software developers, but there wasn't a clear understanding of exactly what are the IP issues with open source, what

are the different types of open source licences and how do you, end to end, manage that from developing the software through to distribution of the software. So education was the starting point.

As processes were implemented, it was possible to assess how well those processes stack up against the OpenChain definition. So, at each stage, there was an assessment of the processes and do they conform with OpenChain. So eventually, there came to be a process which actually conformed with OpenChain and they were able to get recognized by the OpenChain team that they were conformant with OpenChain.

So what Code4Health has ended up with is a complete transparent process; transparent to the adopters of these solutions, that they are not engineering risk into the software and are not distributing at risk to the end users. With every release of code that's made available for deployment, there is a bill of materials which itemizes every open source component, how they are licensed and whether or not there are security vulnerabilities in those components. To define the rules for this, there is an open source policy which has been authored, which lists all of the licences out of the two thousand plus licences available that can or cannot be used within these health applications So, ultimately, the adopters of these solutions, the end users trusts, can have complete confidence that there is no risk being passed on to them in these open source solutions.

That concludes our series looking at open source software licensing. Look out for further content on the PatSnap Academy.