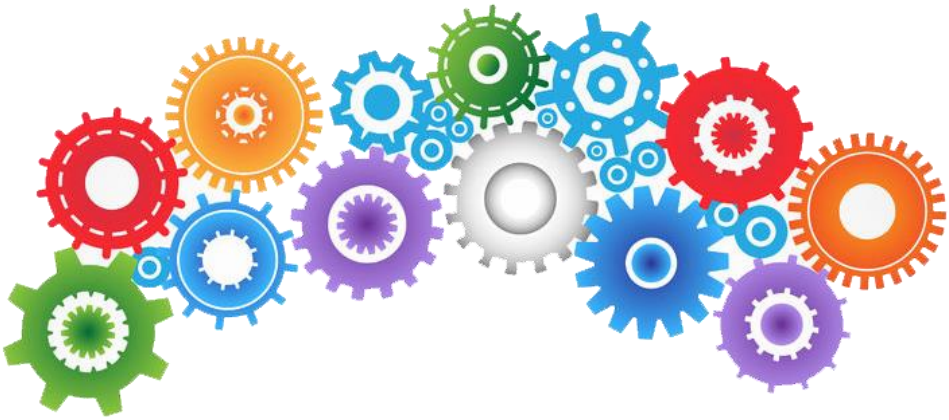
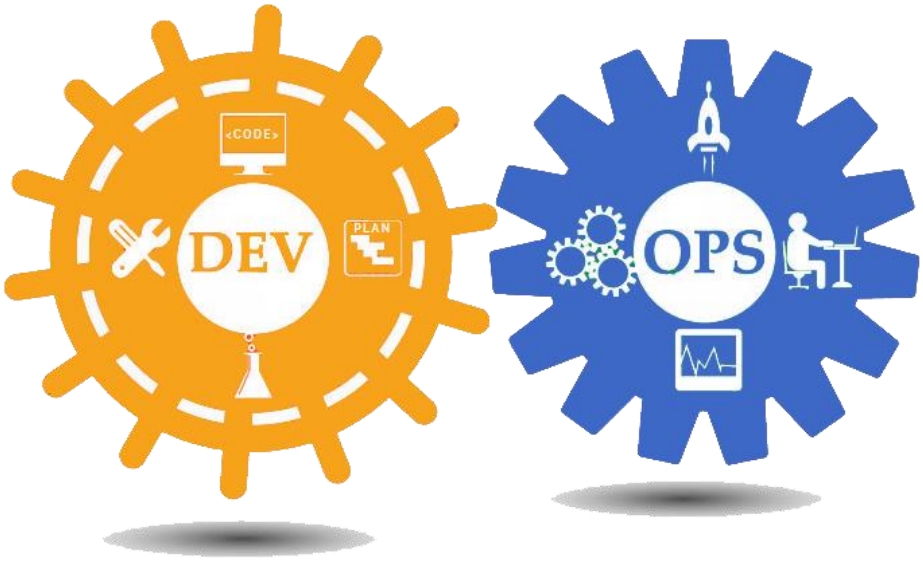
Use DevOps to Minimize Application Security Risks





Open Source Code Is Everywhere

Widespread open source usage is simply a fact in organizations of every size, across all verticals. This is not a passing trend – open source usage will continue to grow. Developers are going to keep incorporating open source components into their code: it's cost-effective, it helps companies accelerate the software development lifecycle (SLDC) and it frees up developers' time to give their organization's proprietary code the extra magic they need to stand out in today's marketplace.

A recent survey of Github found that virtually all (94%) of those who are employed use open source at least sometimes in their professional work (81% use it frequently), and 65% of those who contribute back do so as part of their work duties. According to the GitHub survey, most developers report that their employers accept or encourage use of open source applications (82%) and dependencies in their code base (84%), but some said their employers' policies on use of open source are unclear (applications: 13%, dependencies: 11%). 72% say that they always seek out open source options when evaluating new tools.

Reading this survey's findings, one cannot ignore the fact that open source software usage is growing day-after-day. But, unfortunately, many of these open source components come with liabilities in their license agreements, and one out of every 16 open source download requests is for a component with a known vulnerability.

Furthermore, looking at the heterogeneous sources of applications code, one can categorize today's applications into three main types, each with its own associated risk level:

1) The modern application – comprising of mostly open source code.
2) The legacy application – comprising a close to 0 amount of open source components.
3) The composite application – comprising of various types of source code components: proprietary, 3rd party and open source components.

Types of Applications

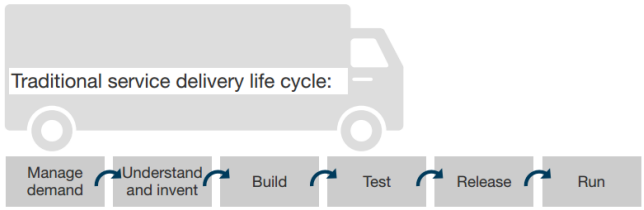
MODERN APP	LEGACY APP	COMPOSITE APP
90% Open Source Components	10% Open Source Components	70% Third Party
10% Proprietary Code	90% Proprietary Code	30% Proprietary Code
Modern languages e.g. Java, Ruby, Go, etc.	Mix of languages e.g. Java, C, C++, Fortran, Cobol	Mix of languages e.g. Java, C, C++, Fortran, Cobol
Mobile apps, web apps, IoT connected devices apps, APIs	Web apps, APIs	Mix of binaries
		Web apps, APIs, mobile apps, IoT connected devices apps, APIs

Source: Forrester, WhiteSource and Microsoft webinar, Shift Left your Application Security. Minimize Open Source Risks

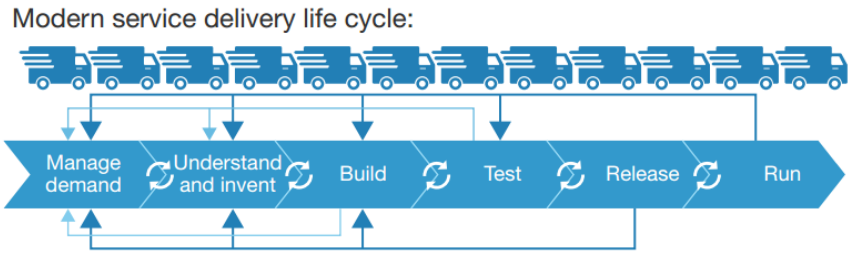
The Goal is Continuous Delivery and Continuous Security

The advantages of using open source components are obvious, and organizations are using more and more of them – either in the development of their products or in the internal tools that they use. As this practice continues to grow, it's important to pause for one moment and take a closer look at the processes and tools that we use when we work.

Old method: No coordinated effort, oftentimes too little too late in the life cycle



New method: Continuous Application Security Testing Security visibility across development life cycle to decrease discovery and remediation time



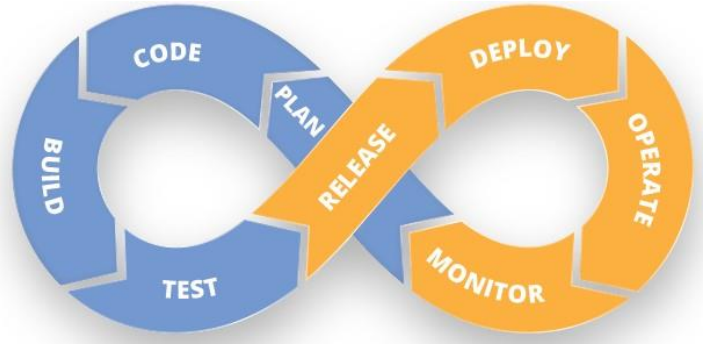
Source: Forrester, WhiteSource and Microsoft webinar, Shift Left your Application Security. Minimize Open Source Risks

Agility and modern service delivery are indeed a target and a reality already at many organizations. However, a recent MSDN blog post about open source management raises the following questions:

- Can we trust all the open source libraries that are in use?
- Do the open source libraries align with our organization's security standards?
- Do we accept all types of open source licenses?
- Can we track the OSS components that are added to our systems?

Use DevOps to Minimize Application Security Risks

DevOps approach applies agile and lean thinking principles to all stakeholders in an organization who develop, operate, or benefit from the business's software systems, including customers, suppliers and partners.



DevOps Cycle

By extending lean principles across the entire software supply chain, DevOps capabilities will improve productivity through accelerated customer feedback cycles, unified measurements and collaboration across an enterprise, and reduced overhead, duplication, and rework. Devops also offers competitive advantages to a business through three dynamic capabilities:

1) Decreasing total cost of production by tracing issues as soon as possible, during the development lifecycle (SDLC).
2) Speeding continuous innovation of ideas by enabling collaborative development and testing across the value chain.
3) Enabling continuous delivery of these innovations by automating software delivery processes and eliminating waste while still helping to meet regulatory concerns

Automate Your DevOps Cycle

DevOps is a rather new term emerging from the collision of two major related trends. The first was also called "agile system administration" or "agile operations"; it sprang from applying newer Agile and Lean approaches to operations work. The second is a much-expanded understanding of the value of collaboration between development and operations staff throughout all stages of the development lifecycle when creating and operating a service, and how important operations has become in our increasingly service-oriented world.

Going back to open source: if open source is everywhere, it must be managed like any other type of code. So many Application Life-Cycle Management (ALM) systems are handling proprietary and commercial code. So many discussions on methodologies of how to manage code and development processes. But not enough recognition, acceptance and methodology around open source management.

Applying the DevOps cycle to open source management requires the adaption of each of the steps specifically to open source characteristics. This requires a certain expertise - but it's certainly doable.

Plan a Secure and Cost-Effective Application

Agile development requires the product manager and the R&D manager to sync, sprint after sprint, on the user stories that will increase application security, regardless of the type of code that is being used, proprietary or open source.

Here is a list of example security focused stories that DevOps people should push to see as part of the Agile sprints:

- As a DevOps expert, I would like to automatically detect all open source components in the code, while running a build.
- As a DevOps expert, I would like to get real-time alerts on security risks, policy pitfalls, and software bugs.

- As a developer, I would like to discover the best open source components for my needs, while I search online.
- As a CISO, I would like to make sure that developers are downloading only authorized open source components.
- As a CISO, I would like to generate comprehensive, always up to date reports on my open source usage, in one click.

Code Only with Secure and Authorized Open Source Components

Developers want to code. They don't care much about costs, processes or policies. They want to use good, innovative and easy-to-deploy code. As a development manager or CISO, you are most likely worried about the code components that your developers are deploying.

Not only would you want to know what they are using, at best, you would like to control the open source components that developers are downloading, limiting them to secure, up-to-date and approved ones.

Such early detection of development issues, shifted left to the component-selection phase, will optimize your DevOps cycle, decrease cost and last but not least, ensure that the released product is the best possible one, security, quality and organizational-policy wise.

Source: Ponemon Institute Research

Shift Left Security Checkups to the Build Phase

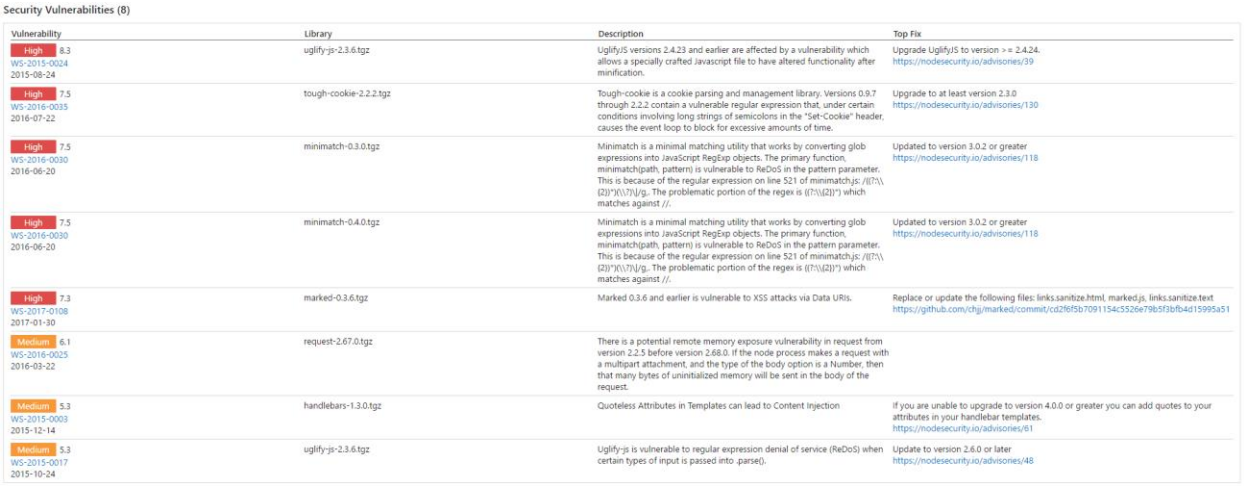
As a development manager, you want to make sure that your software goes through build cycles smoothly as possible. You are also most concerned about the percentage of propitiatory code bugs and open source vulnerabilities that can jeopardize the product's quality or in worst cases- invite cyber-attacks.

As a development manager, you want that every time a new component is added to the build, your pre-integrated open source management tool will automatically calculate and assign a digital signature to that component, cross-reference it against an up-to-date database of known vulnerabilities to determine if it's an open source component and alert you on any known issues that are associated with that component.

As a DevOps expert and as a development manager, you want to know how healthy your build is and you want to address issues as soon as possible. Preferably the day after the build and certainly not in a periodic manner, when the product is already being used by thousands of users out there…

WhiteSource Platform: Vulnerabilities Dashboard

Remediating open source security risks should be an easy to handle task when the development manager can easily see the list of vulnerabilities, by priority, directing the responsible developer to handle these vulnerabilities one after the other and leveraging an easy-to-reach 'fix' link to the recommended remediation.



Security Vulnerabilities (8)

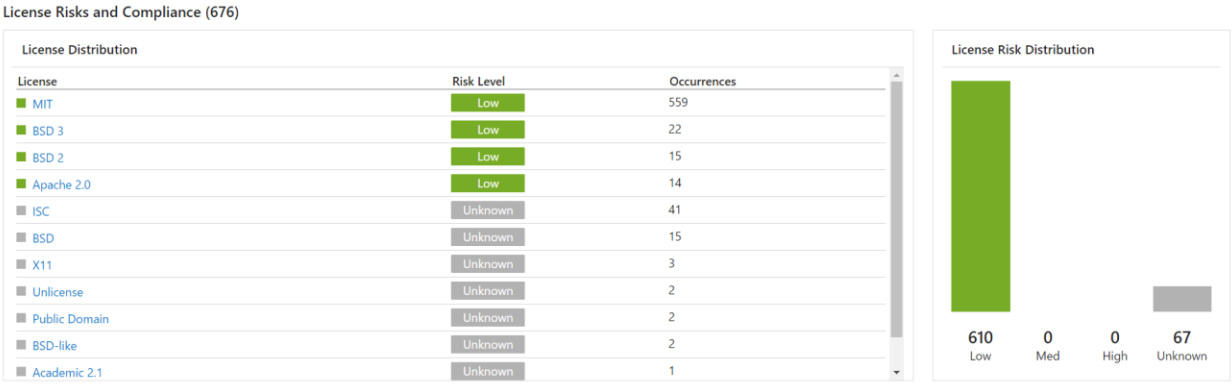
Vulnerability	Library	Description	Top Fix
High 8.3 WS-2015-0024 2015-08-24	uglify-js-2.3.6.tgz	UglifyJS versions 2.4.23 and earlier are affected by a vulnerability which allows a specially crafted Javascript file to have altered functionality after minification.	Upgrade UglifyJS to version >= 2.4.24. https://nodesecurity.io/advisories/39
High 7.5 WS-2016-0035 2016-07-22	tough-cookie-2.2.2.tgz	Tough-cookie is a cookie parsing and management library. Versions 0.9.7 through 2.2.2 contain a vulnerable regular expression that, under certain conditions involving long strings of semicolons in the "Set-Cookie" header, causes the event loop to block for excessive amounts of time.	Upgrade to at least version 2.3.0 https://nodesecurity.io/advisories/130
High 7.5 WS-2016-0030 2016-06-20	minimatch-0.3.0.tgz	Minimatch is a minimal matching utility that works by converting glob expressions into JavaScript RegExp objects. The primary function, minimatch(path, pattern) is vulnerable to ReDoS in the pattern parameter. This is because of the regular expression on line 521 of minimatch.js: /((?:\\{2})*)(\.)?\.)/g;. The problematic portion of the regex is ((?:\\{2})*) which matches against //.	Updated to version 3.0.2 or greater https://nodesecurity.io/advisories/118
High 7.5 WS-2016-0030 2016-06-20	minimatch-0.4.0.tgz	Minimatch is a minimal matching utility that works by converting glob expressions into JavaScript RegExp objects. The primary function, minimatch(path, pattern) is vulnerable to ReDoS in the pattern parameter. This is because of the regular expression on line 521 of minimatch.js: /((?:\\{2})*)(\.)?\.)/g;. The problematic portion of the regex is ((?:\\{2})*) which matches against //.	Updated to version 3.0.2 or greater https://nodesecurity.io/advisories/118
High 7.3 WS-2017-0108 2017-01-30	marked-0.3.6.tgz	Marked 0.3.6 and earlier is vulnerable to XSS attacks via Data URIs.	Replace or update the following files: links.sanitize.html, marked.js, links.sanitize.text https://github.com/chjj/marked/commit/cd2f6f5b7091154c5526e79b5f3bfb4d15995a51
Medium 6.1 WS-2016-0025 2016-03-22	request-2.67.0.tgz	There is a potential remote memory exposure vulnerability in request from version 2.2.5 before version 2.68.0. If the node process makes a request with a multipart attachment, and the type of the body option is a Number, then that many bytes of uninitialized memory will be sent in the body of the request.	
Medium 5.3 WS-2015-0003 2015-12-14	handlebars-1.3.0.tgz	Quoteless Attributes in Templates can lead to Content Injection	If you are unable to upgrade to version 4.0.0 or greater you can add quotes to your attributes in your handlebar templates. https://nodesecurity.io/advisories/61
Medium 5.3 WS-2015-0017 2015-10-24	uglify-js-2.3.6.tgz	Uglify-js is vulnerable to regular expression denial of service (ReDoS) when certain types of input is passed into .parse().	Update to version 2.6.0 or later https://nodesecurity.io/advisories/48

WhiteSource Platform: Security Vulnerabilities

Continuous Monitoring is the Heart of it All

When it comes to open source management, the responsible program manager would want to be on top of all known assets and all known vulnerabilities. That person needs an easy way to download a full and accurate open source Bill of Materials (BoM) report based on the last build.

The responsible development manager may also be asked by the CISO or by the legal parties at the company to address any license and compliance risks. An easy to produce 'License Distribution' type of report is therefore a critical type of requirement.



License Risks and Compliance (676)

License Distribution

License	Risk Level	Occurrences
MIT	Low	559
BSD 3	Low	22
BSD 2	Low	15
Apache 2.0	Low	14
ISC	Unknown	41
BSD	Unknown	15
X11	Unknown	3
Unlicense	Unknown	2
Public Domain	Unknown	2
BSD-like	Unknown	2
Academic 2.1	Unknown	1

License Risk Distribution

610 Low | 0 Med | 0 High | 67 Unknown

WhiteSource Platform: License and Compliance Risks

Bringing it all Together

Agile development is a continuous process. Sprint after sprint all people involved, from senior to junior, from CISO to product managers, from dev managers to test managers, from IT Ops to program managers; are all devoted to getting the best product out there, functionality, security and quality wise.

An optimized DevOps cycle is a key success factor in detecting issues and handling them as soon as possible, and as part of the development cycle and not after release or deployment.

When open source code is widely spread, an open source management product - continuous and pre-integrated into your DevOps cycle, is a critical enabler that DevOps teams should insist on implementing.