Mitigating Cyber Risks related to Open Source Software

SDG

[technology + passion] - risk

Points to be Discussed

- ❑ Statistics
- ❑ Advantages
- ❑ Open-Source Risks
- ❑ Best Practices

Some Statistics ….

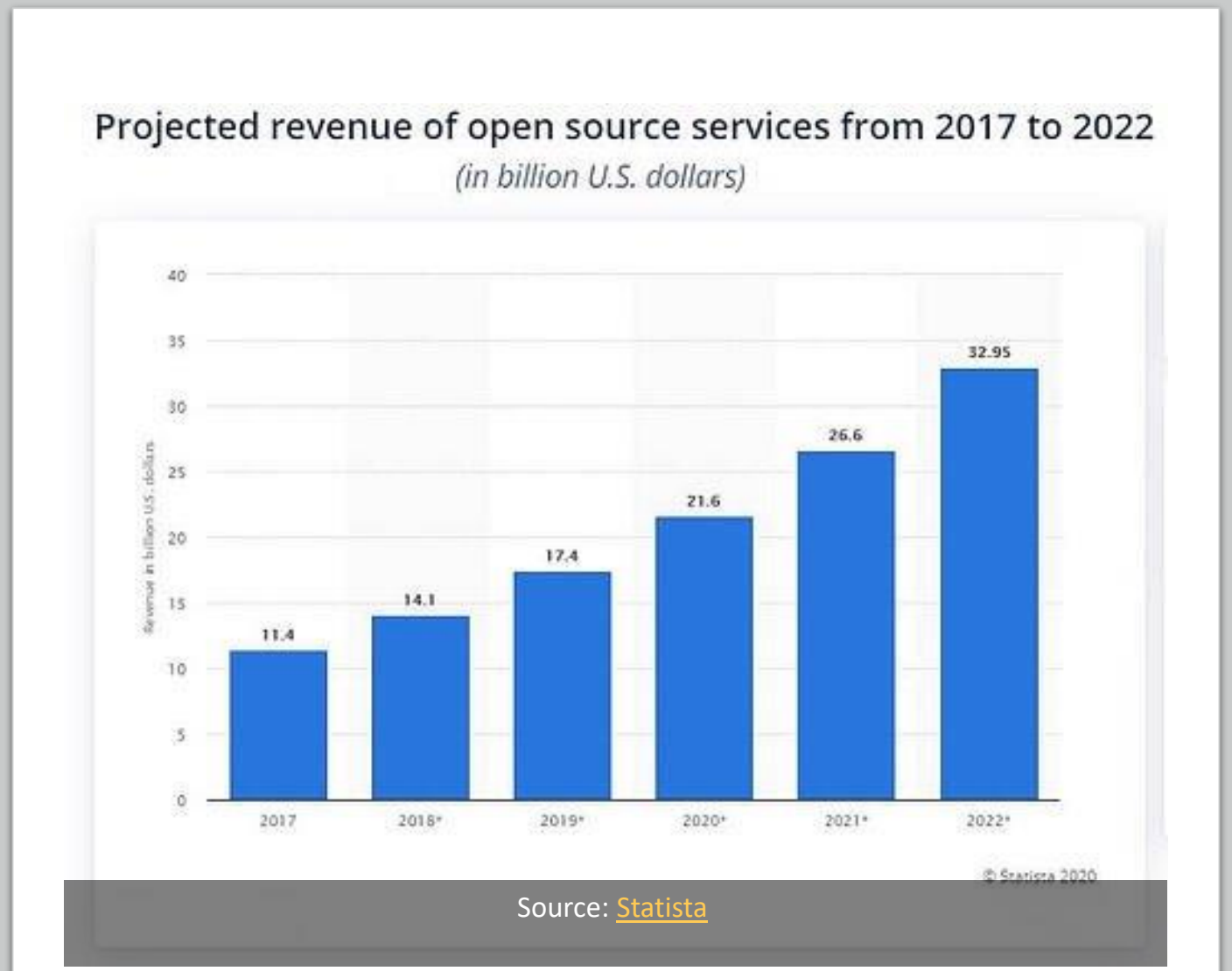
- As many as 93 percent of organizations use open source software and 78 percent run part or all of their operations on it, according to The Tenth Annual Future of Open Source Survey

- According to another survey, 97% of developers responded that they use open source components in their applications, with over 87% stating that they use it heavily.

- According to Synopsys, 99% of commercial databases contain at least one open source component, and nearly 75% of these codebases contain open source security vulnerabilities.

Statistics…

- The overall market for open source is projected to stand at $21.6 billion in 2020 and then grow by over 30 percent by 2020, reaching close to $33 billion a year.



Source: Statista

Open Source – Advantages

- Generally free
- Continuously evolving and improving – fast fixes, integration
- Scalable and can be changed easily
- Less costly, low or no licensing fees
- Agile and customizable with real time improvements
- Easy to manage

Open Source Risks

Risk #1 – Software vulnerabilities & fixes

- Bugs are resolved quickly due to community support, however, information related to vulnerabilities is made public.
- This public information can be easily exploited by hackers.
- Eg. Equifax, Heartbleed etc.

Risk #2 – Security consideration in Open source software

- Open-source software comes with no claims or legal obligations for security and community support informing you how to implement it securely may be lacking. The developers responsible for creating software are often not security experts and may not understand how to implement best practices.

- Often open-source software includes or requires the use of third-party libraries, pulled in from package managers without inspection.

Risk #3 – Warranties & IP infringement issues

- Open-source software does not come with any warranties as to its security, support, or content. Although many projects are supported, they are done so by volunteers and the development of them can be dropped without notice.

- There are over 200 types of licenses that can be applied to open-source software, including Apache, GPL, and MIT. Many of these licenses are incompatible with each other, meaning that certain components cannot be used together since you have to comply with all terms when using open-source software. The more components you use, the more difficult it becomes to track and compare all of the license stipulations.
- What this means in practice is that if you use open-source software that is found to contain code with infringed rights, **you can be held responsible for infringement**.

Risk #4 – Untracked use of software

- Teams often have insufficient or non-existent review processes when it comes to which open-source components are being used. Multiple versions of the same component might be used by different teams or developers might be unaware of conflicting functionality or licensing.

- These issues can occur due to lack of knowledge of software or security functionality, lack of communication between teams or team members, or insufficient or absent tracking and documentation protocols.
- Unlike third-party proprietary software, which has built-in controls to prevent the use of multiple or incompatible versions, open-source components typically rely on the user to verify proper use.

Risk #5 – Operational challenges

- Primary concern from an operational standpoint is the failure to track open source components and update those components as new versions become available. These updates often address high-risk security vulnerabilities, and delays can cause a catastrophe, as was the case in the Equifax breach.

- Another issue is abandoned projects that perhaps begin with much active involvement from the open source community but eventually fade away as nobody updates them anymore. If such projects make their way into apps in the form of libraries or frameworks, your developers are responsible for fixing future vulnerabilities.

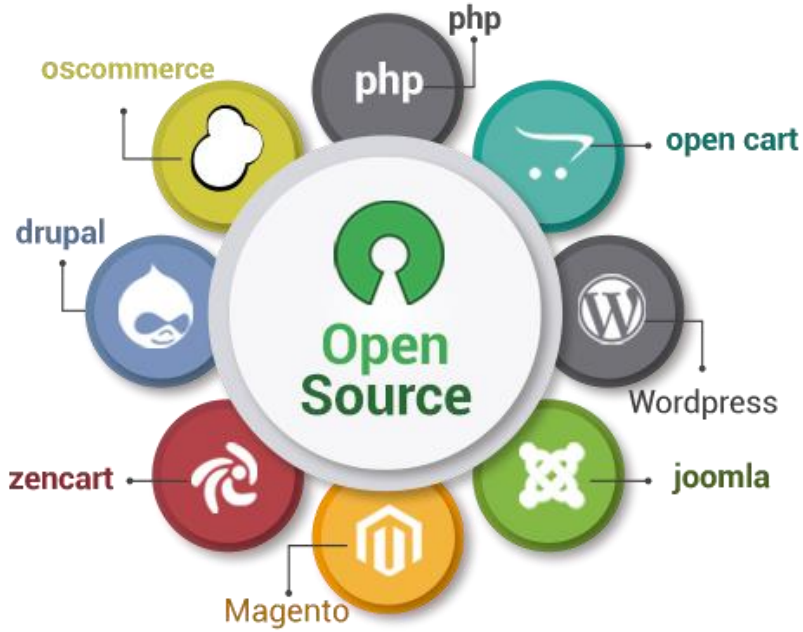
Best Practices

Way Forward…

sdge.com

Practice #1 – Assess the need

- Conduct a thorough Risk assessment of your environment to understand the inherent risks, need for the open source software, which software to be used and where it needs to be used.
- Use selection criteria to assess the software before use. The factors to be considered can be - Total Cost of Ownership, Technical support availability, scalability of solution, embedded security.
- Open Source Software (OSS) should be qualified after conducting relevant usability, stability and security tests. Only pre-approved and qualified OSS should be used and deployed within the organization.



How to choose – Questions to ask?

- **Does it do what you want?**
  - Does the software do what you want it to do? What are your requirements?

- **Is the software good for its role?**
  - Investigate prior uses of the software. Has anyone used it in the manner you want to use it?

- **Is the software actively used, developed and supported?**
  - The importance of an active community around the software shouldn't be underestimated. Check out the mechanisms for how support is provided: support forums, direct email support and issue trackers, and investigate whether they are actively used and have a good level of response to queries and issues.

- **Does the software have a future?**

- **How is the software provided?**
  - In most cases, good user and developer documentation is a must.
  - Are the prerequisites of the software well defined and straightforward to obtain and deploy, and do they fit your own requirements?

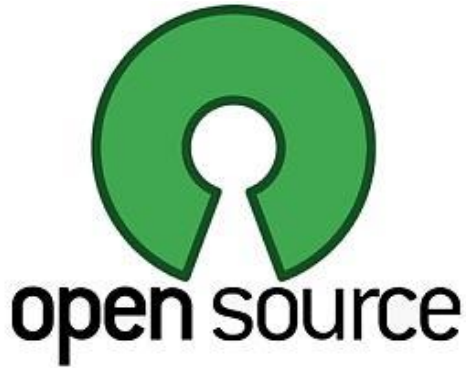
- **Choosing the right version**

Practice #2 - Governance

- Organizations should develop a comprehensive policy governing the usage of OSS.
- The policy should cover an acceptable usage of OSS and the acceptable risk appetite for OSS.
- Risk Assessment should on a minimum cover risks related to license requirements, operations (support for the software and stability of the software) and security (vulnerabilities and known exploits)

Practice #3 – Open source inventory & documentation

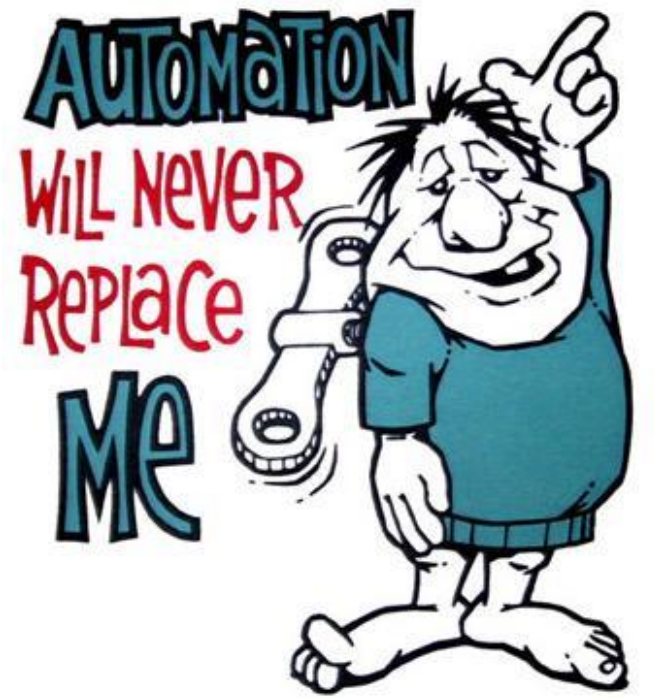
- Maintain a detailed inventory of OSS used within the organization detailing instances (number / quantity) of use, version etc.
- Where OSS is used as a component of your in-house application, a data call should be performed to determine what components are OSS and what versions are currently is use.
- The inventory should include libraries, frameworks, middleware and applications. Maintain a profile of each OSS to include the code's origin, where to get updates, and how often the community releases new versions.
- A comprehensive documentation of the components used should be maintained. This includes libraries, frameworks, middleware and applications.
- Maintain a repository of the source code of OSS deployed within the organization



Practice #4 – Installation of OSS

- Any OSS installed / deployed should adhere to the organization's system installation procedure. This should ensure that:
  - Only whitelisted OSS is deployed in the organization.
  - All deployments are vetted and approved through a formal system deployment / change management process.
  - All deployments are inventoried in the asset register.
  - Only authorized individuals such as the system administrators should install / deploy OSS.
  - Use OSS from reliable and trusted sites.
  - Wherever possible prefer source code to binaries.
  - Examples of trusted sites as recommended by Open Source Initiative include freshmeat.net, sourceforge.net, osdir.com, developer.berlios.de and bioinformatics.org f. Ensure that the OSS is tested and updated with the latest patches.

Practice #5 – Software security testing



- Integrate security within your build (Jenkins, Bamboo, TeamCity, etc.)
  - SAST
  - DAST
- Create a test framework to automate checks
- Constantly check code
- Perform a security assessment to identify and patch any known vulnerabilities in the OSS.
- For critical applications, it is recommended to do a combination of an automated static analysis (source code scanning) and dynamic analysis to find vulnerabilities in individual applications and define measures on how to fix them.

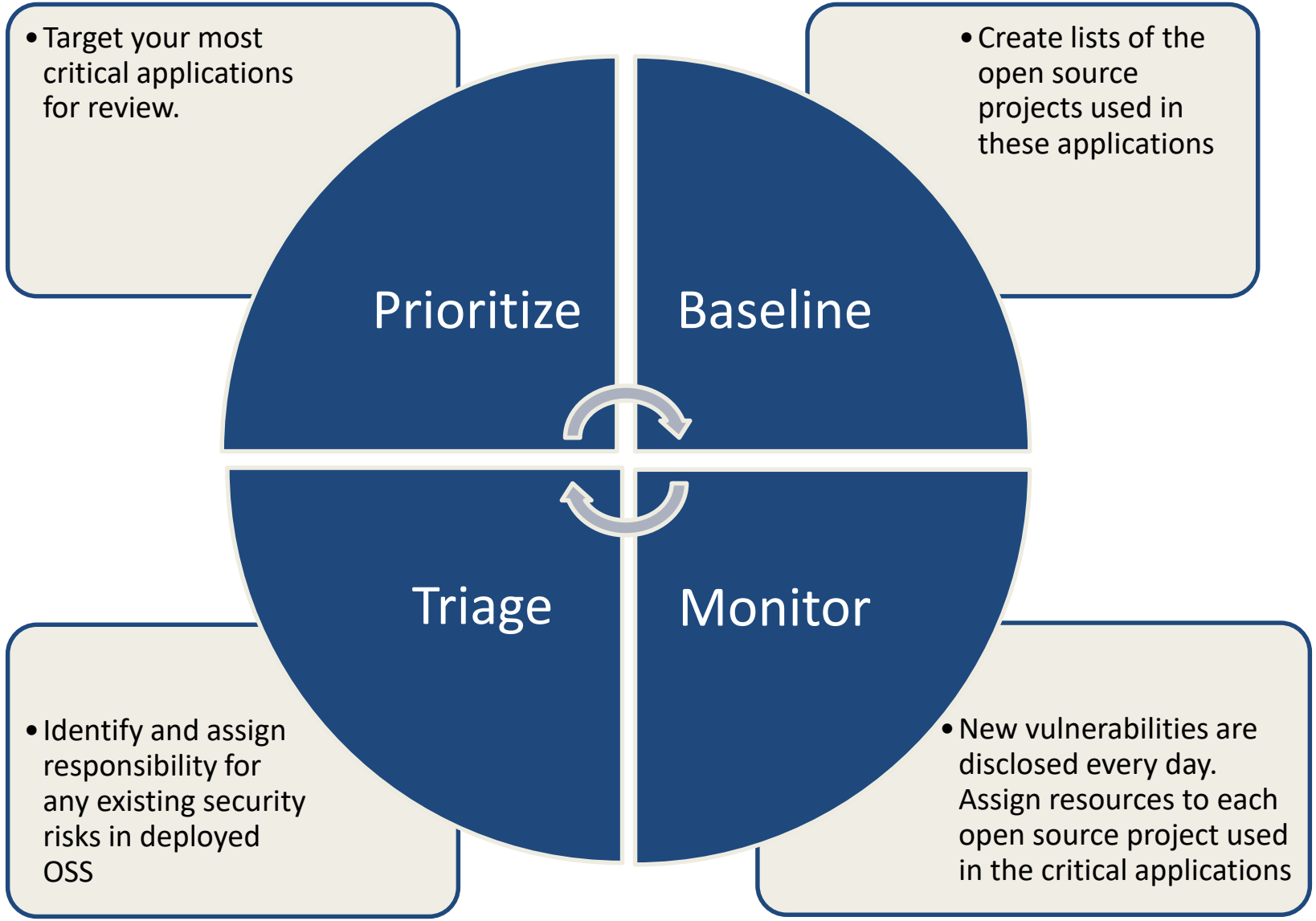
Practice #6 – Application hardening & Patch management

- As with any other software, the OSS should be configured in a secure manner.
- The organization's Patch Management process should monitor and update patches released for the OSS.
  - Check the community associated with your open source code.
  - When a new vulnerability is identified, the organization should explore possible mitigation strategies that can be implemented until a patch is available.
  - Once a patch is released, test the patch for stability and applicability within your test environment prior deploying on your production systems.
  - Have a time bound approach to patch all vulnerable OSS in place.

Practice #7 – Training of employees

- Enterprises should ensure that their developers have a general understanding of cybersecurity, as well as the latest trends and updates. Your developers should be able to identify common security issues that arise in open source code, if not fix them.
- Similarly, the security team should be involved in the development process from the early stages. Rather than making security an after-thought, it should be a priority from the very beginning of a project.

Basic Principles of Open Source Management



Open Source has its own pros and cons but with the emerging threat landscape and agile environment, it is the need of the hour.

Although, whether to choose from open source or proprietary software is entirely based on an organization's business needs – Security, Scalability and Stability of the tool/software should be analyzed with due diligence.

Open source is an excellent model that can be found in many of today's projects. However, to ensure secure open source code, you need to acknowledge the security risks that come with open source software. You have to make sure that each of your open source components is delivering value to the project and are secure.

sdgc.com
23

Thank You

Contact Details –

Meetali Sharma
meetalisharma81@gmail.com
www.meetalisharma.com
https://www.linkedin.com/in/meetali-sharma/