Guidelines to Enhance Software Supply Chain Security - Onapsis

Onapsis Position Paper - v1.0 - May 26, 2021

For more than 10 years, Onapsis, the Boston-based leader in cybersecurity for mission-critical applications has helped organizations secure their most critical applications by researching vulnerabilities and threats and providing products and technology to assess, comply and monitor to protect against and prevent cybersecurity risks affecting mission-critical applications. The leading research performed by the Onapsis Research Labs led to reporting over 800 zero-day vulnerabilities and multiple alerts generated by CISA that helped organizations address these critical threats:

- 2021-04-06 - CA: Malicious Cyber Activity Targeting Critical SAP Applications
- 2020-07-13 - Alert AA20-195A: Critical Vulnerability in SAP NetWeaver AS Java (RECON)
- 2019-05-02 - Alert AA19-122A: New Exploits for Unsecure SAP Systems (10KBLAZE)
- 2018-07-25 - CA: Malicious Cyber Activity Targeting ERP Applications
- 2016-05-11 - Alert TA16-132A: Exploitation of SAP Business Applications

Based on the experience working with governmental organizations including civilian agencies, companies in the critical infrastructure sector and some of the largest corporations (top 20% of the Fortune 100) in the US and around the world, Onapsis is uniquely positioned to provide guidance on the following 3 areas, with a strong focus on cyber risk in critical software or that are mission-critical for companies:

Area 1- Criteria for designating "Critical Software"

Onapsis is proposing that software matching any of the following functional criteria to be considered for designating "Critical Software":

Functional Criteria	Description
Level of privilege or access required to function	If the software manages data that is classified or sensitive but unclassified[1] (personal information, confidential business information, trade secrets, information related to the protection of critical infrastructure assets, etc) and requires segregating its access appropriately according to different roles and profiles in a business, then it is designated as "Critical Software"
Integration	If the software has [outgoing] interfaces with pre-established authentication to at least one other critical software, then it is designated as "Critical Software"
Dependencies	If the software (any of the following): ● (Process Dependencies): enables critical/essential operations of the federal government and its agencies to achieve their missions. ● (Data Dependency): stores and processes data that is classified or sensitive but unclassified then the software is designated as "Critical Software"
Direct access to networking and computing resources	If the software has network access to multiple other critical software assets, this allows lateral movement that may lead into access to "Critical Software" through multiple jumps, hence it is designated as "Critical Software".
Performance of a	A Mission Critical system[2] is essential to the survival of a business or

[1] 12 FAM 540 SENSITIVE BUT UNCLASSIFIED INFORMATION (SBU) by US Department of State
[2] Mission Critical - Glossary by NIST or Mission critical by Wikipedia

function critical to trust	organization. If the software is supporting a Mission Critical system including one that is essential for business operation or to an organization, then the software is designated as “Critical Software”.
Potential for harm if compromised	If the software ● is involved (stores or processes) in a breach of personal Information (according to HIPAA/GLBA/Privacy Act/COPPA/FERPA/FCRA), or ● is harmed and it disrupts essential operations on critical infrastructure[3], or ● or its data (stores or processes) is unauthorized altered or deleted and it disrupts normal operations on the federal government or its agencies or critical infrastructure Then the software is designated as “Critical Software”

Area 3 - Guidelines outlining security measures that shall be applied to the federal government’s use of critical software.

While current security standards are comprehensive, the challenge resides on their interpretations for different technologies and applications. **There is a knowledge gap between the application-agnostic standards and the real-life implementations of “Critical Software”**. Specialized experts in different agencies may interpret those standards differently, leading to different measures for the same applications. It does not contribute to best practices nor to a cost-effective solution. The deep level of detail and specificity provided by the National Checklist Program’s (NCP) publicly available checklists (benchmarks) is essential to a proper standardized implementation of security measures in “Critical Software”.
The National Institute of Standards and Technology (NIST) should launch a public-private initiative that involves private-sector specialized experts to extend the different checklists (benchmarks) to the most common applications considered “Critical Software” by contributing with their expertise. These checklists (benchmarks) should be used as the minimum security baseline to determine compliance to best practices.
Onapsis, as a recognized leader on the security of mission-critical applications (such as SAP, Oracle and business critical SaaS applications), offers itself to contribute to the building of these checklists (benchmarks).

Area 4 - Initial minimum requirements for testing software source code

Based on Onapsis' experience securing mission-critical SAP Applications, it is Onapsis' recommendation that at a minimum for any software that is designated as “Critical Software”:
Depending on the technologies used for developing the “Critical Software”, different SSDLC implementations may be required, for example, any custom developments on top of critical software such as SAP should be developed following a specific SSDLC which involves the following best practices:

- All Developers must use SAST[4]
    - Perform code scanning early and as often as possible, when developers create code.
    - Involve your developers (a code scanner helps in developing better code)
    - Ensure a 4-eye principle
    - Implement an exception approval workflow for false-positives
    - Continuously monitor production for SSDLC bypasses
- There must be a SAST control with a 4eye principle in place whenever code used as or operated on top of critical software that is shipped from dev to qa.
- There must be a continuous control in place that checks all production code for critical software (depending on the technology that may be achieved by scanning the code in production or scanning the code that is released towards production. In the latter case evaluation methods should be in place that detect any deviations from actual deployed code in production vs expected code according to a change management system). This control builds the foundation for establishing vulnerability management in code.

[3] Critical Infrastructure Sectors - Glossary by NIST or Critical Infrastructure Sectors by CISA
[4] Definition of SAST - IT Glossary by Gartner