SCRM Position Paper for NIST May 26th 2021

Industry Best Practices for Securing the Software Development life-cycle

Authors: David Wray, CTO Public Sector Alliances, david.l.wray@mircrofocusgov.com, Shlomi Ben-Hur Software Engineer (Security), shlomi.ben-hur@microfocus.com, Mark Conway, Director Product Group Strategy & Ops, Mark.Conway@microfocus.com

Topic Area - *Initial list of secure software development lifecycle standards, best practices, and other guidelines acceptable for the development of software for purchase by the federal government.*

Background

As one of the world's largest IT Management software producers, with over 300 products in our portfolio and 40,000 customers, Micro Focus has over forty years of experience with continuous improvement of software development lifecycle standards, best practices and reducing risks within our **product to market (P2M)** process. Our software is in use in every Federal CFO Act agencies and many of our products are used to manage IT infrastructure, networks and mission critical capability. We are a provider of DevOps, Cloud, IT Operations Management and Cyber Security offerings.

We are very interested in participating with this outreach request on helping achieve the goals outlined within the recent Presidential Executive Order (EO) and would like to present our current approach and methodology for reducing security risks within our P2M Process. Our hope is that we partner with NIST and collaborate to improve software supply chain risks capability as well as help propose viable solutions to current issues that require government sponsorship/support.

Our P2M Process

Several years ago, we aligned with Agile and DevOps practices and have established a common **digital software factory** that supports all our product development. Today this digital software factory process is supported by **enterprise services that significantly reduces our supply chain risks** by significantly reducing and in some cases **eliminating threat vectors within our P2M process**. For example, continuously strive to eliminate tools, work processes and manual tasks that local development teams performed during their unique software build process with common enterprise services. These common enterprise services are key to our ability to significantly reduce many of the threat vectors we have identified within our P2M process. Mainly, we limit a local development team's ability to perform local tasks and have policies that mandates that our shared enterprise services that greatly reduce potential attack vectors for adversaries. When we started this effort several years ago, we had hundreds of software build products in use by our development teams, thousands of processes and very limited enterprise visibility into our P2M life-cycle. Today, we have rationalized our software build tool chain down to approximately a dozen core

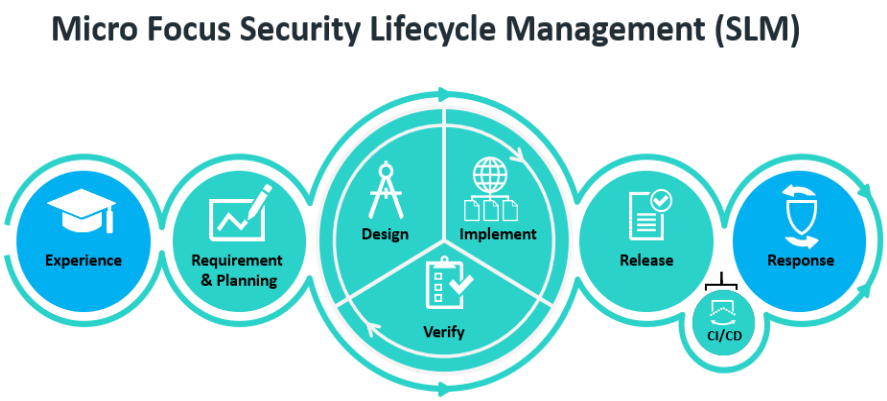

Figure 1 - High Level Components of our Software Development life cycle threat vectors

products. We produce over 2000 commercial product releases a year, currently scan between 10 and 20 million lines of code daily and have automated the major components within our P2M life-cycle to reduce errors, risk & improve quality. We believe that scale of our P2M capability is unprecedented and may be the **world's largest and most comprehensive model and could be used as a benchmark for other organizations to be measured**.

We have over a dozen enterprise services established, each with service and business owners that we continuously monitor, innovate and improve upon to further reduce supply chain risks within our P2M process and we continue to develop new services and common tools. For example, an **insider threat capability**, a **software build kill chain**, a common automation server, a **CVE Tracker Service**, Software Compliance and Dependency discovery**, Machine Readable SBOM creation,** etc. As one of the world's largest software producers of IT Management tools we plan to productize many of these capabilities as offerings to complement our existing portfolio of offerings to help reduce software supply chain risks.

We have a list of recommended security controls that we would like to discuss that are organized by threat vectors for each of the high-level components in Figure 1. This list contains hundreds of security controls that we believe are a good start for NIST to consider as guidelines.

Build Server – Security Controls

#	Threat Vector	Security Control
1	Infrastructure related attacks	• OS Hardening; patches; NW segmentation; MW scanning; AV; host-based IDS, etc. • IAM via SSO • Least privileges – who can access, what can they do on the OS level
2	Uncontrolled/Approved build tools may lead to system compromise	• Define list of approved build tools – TBD • Define requirements for an approved build tools – TBD
3	Insecure configuration of build tools may lead to compromise	• Verify the security of the build tools (i.e., patches, secure configuration, logging, auto update) • Define proper authorization, authentication, least privileges concept • Access via VPN only / use 2FA (?) • Enable access only via SSO/SAML
4	Insecure interfaces may lead to system compromise	• Define secure interface to/from build tools from/to relevant components
5	Build from insecure 3rd party repo or source code repo may lead to system compromise	• Allow access to pull 3rd party libs only from approved 3rd party repos. • Allow access to pull source code only from approved source code repos. • Alert in case of pull from non-approved repo – TBD
6	Insecure jobs configurations may lead to system compromise	• Pipeline jobs protection → allow only DevOps team to manage relevant pipeline jobs, define AAA, ACL's etc. • Pipeline jobs protection → allow only signed commits to be integrated into pipeline • Pipeline jobs protection → allow 3rd party only from vetted repositories (TBD – define 'vetted') • Pipeline jobs protection → allow build scripts/configurations to run only by least privileges users. • Developer jobs → limit ability of developers to interact with automation software
7	Malicious code/libs may lead to system compromise	• Static code analysis; SCA; image scanning (Clair/Trivy/Anchor), other?; JAR/WAR/DLL • Scan for libs which are not used – remove/alert(?) when found
8	Inability to maintain artifacts' integrity may lead to system compromise	• Artifacts will be uploaded to an approved "Products Repo"

*Figure 2 - **Example** Security Controls & Threats for one high level component for the software build process.*

We also have concerns that we would like to communicate that we believe must be addressed. We are hoping that NIST will accept our offer to present to the working group so we can provide the complete list of security controls we are tracking/implementing as well as all the potential concerns & issues. We welcome questions and input and look forward to helping government achieve improved software supply chain security.

Partial List of Potential Concerns & Issues

1. We believe NIST should limit transitive dependency requirements within the creation of an SBOM to one level. Tracking 3rd party SBOM's and their dependencies is extremely difficult.
2. We no longer publish our SBOM on our website due to supply chain security concerns. We would welcome a machine readable SBOM format. Industry needs NIST to adopt a standard.
3. The current Free CVE/NVD solutions sponsored by government requires better alignment with SWIDs to support automation. We currently wrote our own CVE Tracker, and can provide assistance.
4. We believe further refinement of how/who must report SBOM within SaaS offerings is needed.
5. Open source tools that support software builds for IoC, Testing, security compliance, code coverage, etc. must be classified as "Critical"
6. We have concerns that open source vendors will not be able to meet new guidelines.