Cyber Resilience: Software Development Toolchain Code Repositories

By: Gerald Rigdon – 2023

IBM (*Cyber resilience defined*, n.d.) defines cyber resilience as "the ability to continue delivering intended outcomes despite experiencing challenging cyber events, such as cyberattacks, natural disasters or economic slumps." Further, they emphasize that resiliency is necessary to mitigate financial loss, gain customer trust, and increase competitive advantage. In terms of better understanding the topic, the IBM article describes resiliency based on stages of an Information Technology Infrastructure Lifecycle with concepts such as strategy, design, operation, and improvement.

In more relatable terms, NIST 800-160 (*Developing Cyber-Resilient Systems*, 2021), states cyber-resilient systems operate much like the human immune system which is exposed to numerous environmental hazards and responds with appropriate defense mechanisms and repair protocols to recover and maintain a healthy state. The key to resiliency is adaptation; recovering the minimal essential functions in order to survive. However, just like the body does not always recover to the previous state of health, likewise, cyber systems have limitations that must be understood in the context of risk management. Additionally, per NIST 800-160, identifying vulnerabilities is a way to learn and develop more resilient systems. For example, the software development toolchain is an area where compromise can result in long-term persistence of infection without detection. Although many things are considered part of the software development toolchain like compilers, linkers, interpreters, etc., our focus is on code repositories.

Repositories are a location for many software development assets such as source code. Given they are designed to help manage assets in a central location and facilitate collaboration among developers, they are a prime target for malware attacks (*Preventing Malware Injections: Best Practices for Secure Software Development*, 2021). Moreover, since code repositories are not all created the same, vulnerabilities arise when using a repository that has been compromised by a bad actor. One way to minimize such occurrences is to use repositories armed with enhanced cyber security mechanisms.

Git is a popular repository that is fully featured with security protections. However, when security features are optional, organizations must enforce their use. A case in point is signing commits to the repository. Signing is a cryptographically secure method of associating changes committed to the repository to verified users. Thus, if developers retrieve assets from a repository that have not been signed, they are increasing their likelihood exposure to tainted work from non-trusted sources. However, signing is not an all-encompassing security mechanism to be used in isolation; it must be used along with other sound methods such as good password management and two-factor authentication.

In fact, in 2019 (Kovacs, 2019) many Git repositories were deleted and held for ransom due to neglect of password fundamentals. In analyzing this case it was determined that the cyber attackers had obtained account credentials from Git configuration files where passwords were being stored in plaintext. Hence, Kathy Wang, the Director of Security for GitLab stated: “We strongly encourage the use of password management tools to store passwords in a more secure manner, and enabling two-factor authentication wherever possible, both of which would have prevented this issue.” Imagine the potential devastating financial impact of having all your source code deleted, stolen, and then having to pay to get it back. A resilient organization would need to recover from this setback while simultaneously working to detect, respond, and make improvements to protect against future similar attacks.

From (*Preventing Malware Injections: Best Practices for Secure Software Development*, 2021), the following practices have an impact on cyber resiliency when interacting with code repositories since they allow for earlier detection of compromise:

- Unit and Integration Testing – This testing should be done after every commit to a repository. This would include negative testing to ensure proper behavior for undesirable inputs and analyzing logs to make sure errors are not being ignored.
- Scanning – Ongoing execution of static analysis tools, antivirus scanners, etc. against assets retrieved from repositories.
- Fuzzing – Evaluating software behavior in the presence of deliberate invalid and random inputs.
- Code Review – The old-fashioned method of having a human set of eyes reading and reviewing the code both before commits and after retrievals from repositories.

However, vulnerabilities still exist even when fully using all repository security features. As discussed by (Przybylska, 2023), in a 2022 Dropbox breach, hackers stole 130 GitHub repositories. How? It started with a phishing attack using a fake email, login screen, and a malicious link. Once the attacker had obtained employee credentials, they stole the repositories along with thousands of names and email addresses. These types of recent social engineering incidents emphasize the need for continuous employee training and awareness. By learning from past mistakes and published NIST vulnerabilities like those associated with software development toolchains, companies can continue to make improvements with the goal of having more effective cyber-resilient systems that can detect earlier, recover faster, and evolve.

References:

Cyber resilience defined. (n.d.). ibm.com. https://www.ibm.com/topics/cyber-resilienceLinks to an external site.

Developing Cyber-Resilient Systems (2021, December). NIST Special Publication 800-160, Volume 2. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v2r1.pdfLinks to an external site.

Kovacs, E. (2019, May). *Hundreds of Git Repositories Held for Ransom*. securityweek.com. https://www.securityweek.com/hundreds-git-repositories-held-ransomLinks to an external site.

Preventing Malware Injections: Best Practices for Secure Software Development (2021, January). garantir.io. https://garantir.io/preventing-malware-injections/Links to an external site.

Przybylska, M. (2023, January). *Ultimate Review of the most infamous GitHub-related security incidents in 2022*. gitprotect.io. https://gitprotect.io/blog/ultimate-review-of-github-related-fackups-in-2022/Links to an external site.