Security Controls in Shared Source Code Repositories

Joe Huffer

12/15/2025

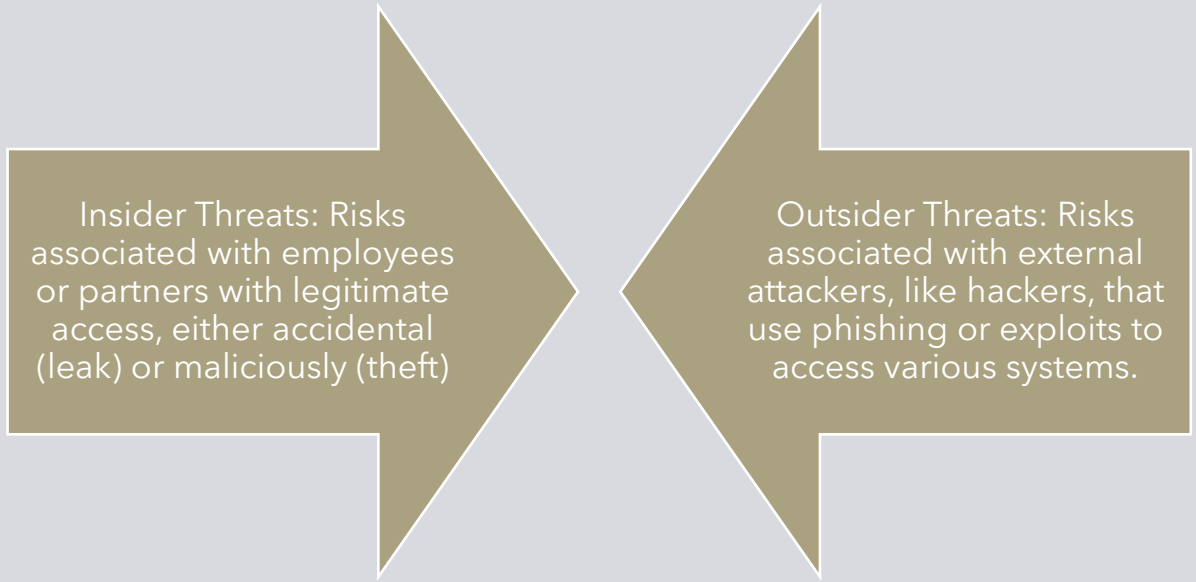
CSD380

Professor Sampson

Introduction

- Defining ‘Secure Source Code Collaboration’: Shared Source Code Repositories are essentially the locations and platforms that we as developers can store code that will then be available for collaboration. Prefixing the term ‘Secure’ gives the context that we are performing this collaboration, securely!
- The importance of this concept can be captured by:
  - Using Secure Code Collaboration to protect Intellectual Property.
  - Secure and preserve a company (or group of developers) reputation.
  - If unsecure, source code is vulnerable to hackers, which exposes source code to exploitation (or fiscal/workload ramifications).
- The real-world consequences of lack of sufficient security have impacts large companies such as: Adobe, Nissan, and Microsoft – and there are countless others that have suffered from source code being leaked.

Security Risks in Sharing Repositories



- Potential Consequences can include:
  - Loss of resource or financial impact to companies.
  - Vulnerabilities that could lead to application or systems being compromised.
  - Regulatory fines due to exposure of PII data (resulting in lawsuit or other situations)

Best Practices in Repository Access Control

Role-Based Access:

Defined as access that is limited to only essential personnel (Developers, Admin, Other Technical Support)

Authentication Measures:

Methods such as 2FA, Two-Factor Authentication, can be implemented to protect access to code repositories.

Auditing

This includes the process of regularly reviewing access permissions to remove inactive users and potentially updating roles.

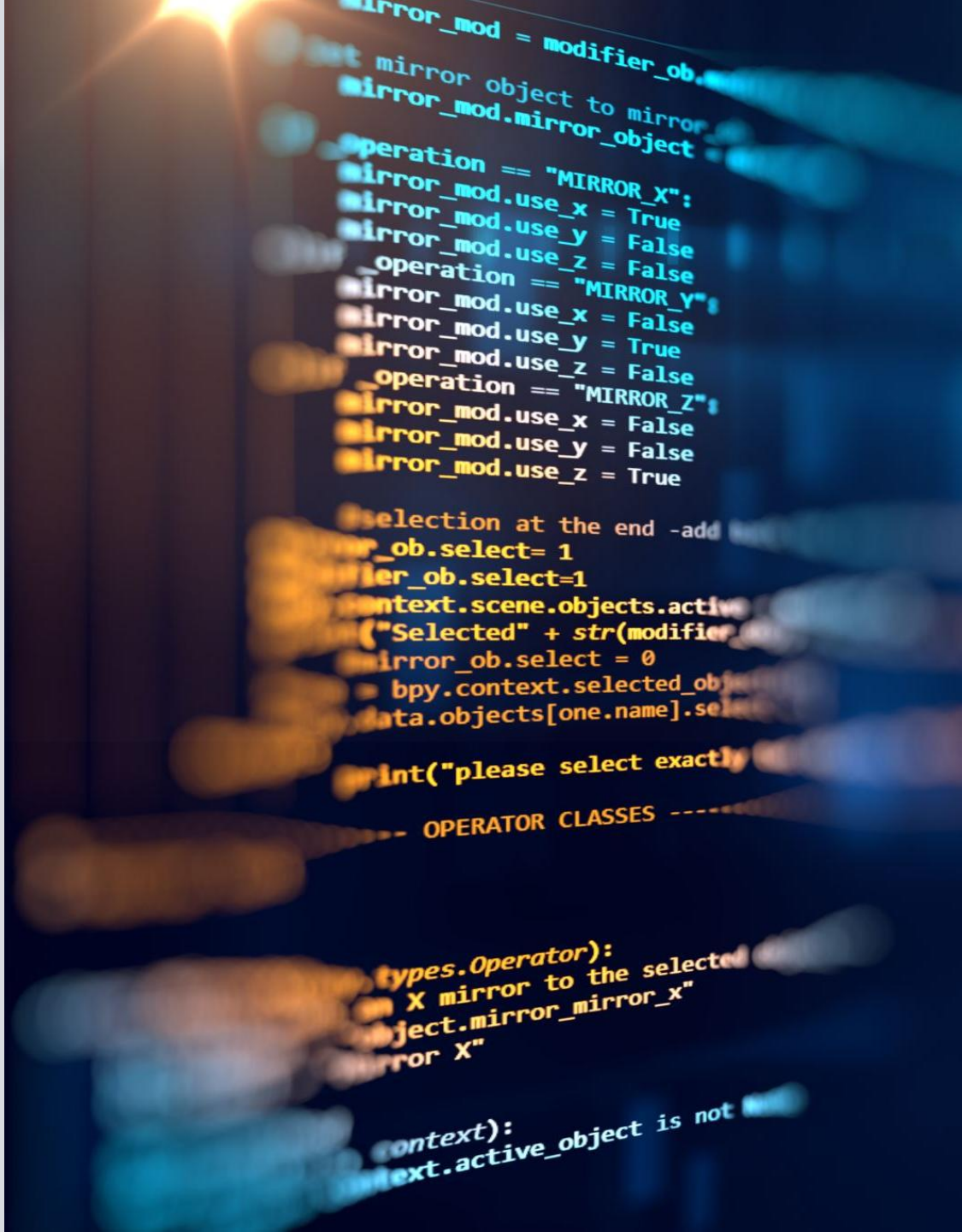
Other Tips Specific to Github!

You can Manage Visibility Settings to restrict access to the public.

You can also carve members into teams, define roles, and this helps streamline permissions in the repository interface menus.

Secure Code Scanning

- Static Application Security Testing (SAST):
  - Involves scanning source code during development to detect vulnerabilities.
  - Examples online presented this as SQL Injections, using Cross Site Scripting, and “buffer overflows” (Berecki, 2022)
- Dynamic Application Security Testing (DAST)
  - Used for to assist in identifying vulnerabilities during runtime.
  - You would see examples of this to test server configurations for stability.
- Github-Specific Code Scanning
  - Software integrations like CodeQL can automatically analyze code and identify potential security vulnerabilities.
  - Advanced tooltips can be used to establish custom rules for those scannings, also



Encryption and Network Security

Encryption:

- Used to encrypt sensitive data ‘in-transit’ (ex: data during uploading), and ‘at rest’ (ex: files stored somewhere) using protocols such as AES-256 (Advanced Encryption Standard – 256 Bit key Length).

Data Monitoring:

- The use of activity tracking to gather real-time metrics for measuring unauthorized access or attempts.

Endpoint Security:

- DLP, or Data Loss Prevention, tools can be deployed to:

- Block Unauthorized USB access
- Encrypt data on portable storage devices

- Antivirus/antimalware tools can also be implemented to prevent malicious code from invading at-risk devices or workspaces.

Network Security

- Last but certainly not least, this is implemented to establish firewalls that restrict unauthorized access, use VPNs to create secure connections between employees and repository servers, and can be regularly monitored by an individual or team to address evolving threats.

Security Policies, and Legal Protection

Creating a SECURITY.md File:

- This provides contributors with clear instructions on reporting vulnerabilities.
- You can specify here which versions and timelines you have regarding security and patching.

Provide Training to Developers:

- Workshops, peer-review/mentorship with senior developers, and good code-documentation are in my experience essential if you want developers to become trained and knowledgeable about security in coding.

Protect Important IPs:

- You'll want to protect important servers, that's a given (if not all technical assets within your company, as well as patents, algorithms, methods, personal best practices, and other code-related data.

Update and Enforce Policy Updates:

- We've all seen policies updated. These polices are updated to reviewed often to establish and inform about an organizations security policies.

Conclusion

- To wrap up, we've covered the following topics:
  - Combining elements of access control, scanning tools, encryption + endpoint + network security elements, as well as policies and legal protection.
  - And that in doing all of this, we can help ensure that a repository and other online/e – assets are protected via a multi-layered approach!

Works Cited

- Berecki, Balázs. "Best Practices for Source Code Security." *Endpoint Protector Blog*, 8 Apr. 2022. https://blog.endpointprotector.com/best-practices-for-source-code-security/

- "About CodeQL." *CodeQL Documentation*, GitHub. https://codeql.github.com/docs/codeql-overview/about-codeql/

- "Quickstart for Securing Your Repository." *GitHub Docs*. https://docs.github.com/en/repositories/securing-your-repository

- "GitHub Repositories Threat Model." *GitHub Well-Architected*, https://wellarchitected.github.com/library/application-security/scenarios-and-recommendations/threat-model/

- "Exploiting Repos: 6 Ways Threat Actors Abuse GitHub and Other DevOps Platforms." *SentinelOne*, https://www.sentinelone.com/blog/exploiting-repos-6-ways-threat-actors-abuse-github-other-devops-platforms/

- "What You Need to Know About Code Repository Threats." *Cyberint*, https://cyberint.com/blog/threat-intelligence/what-you-need-to-know-about-code-repository-threats/