Security Controls in Shared Source Code Repositories

Best Practices for Protecting Source Code

Jonah Aney CSD-380 Module 11.2

12/13/2025

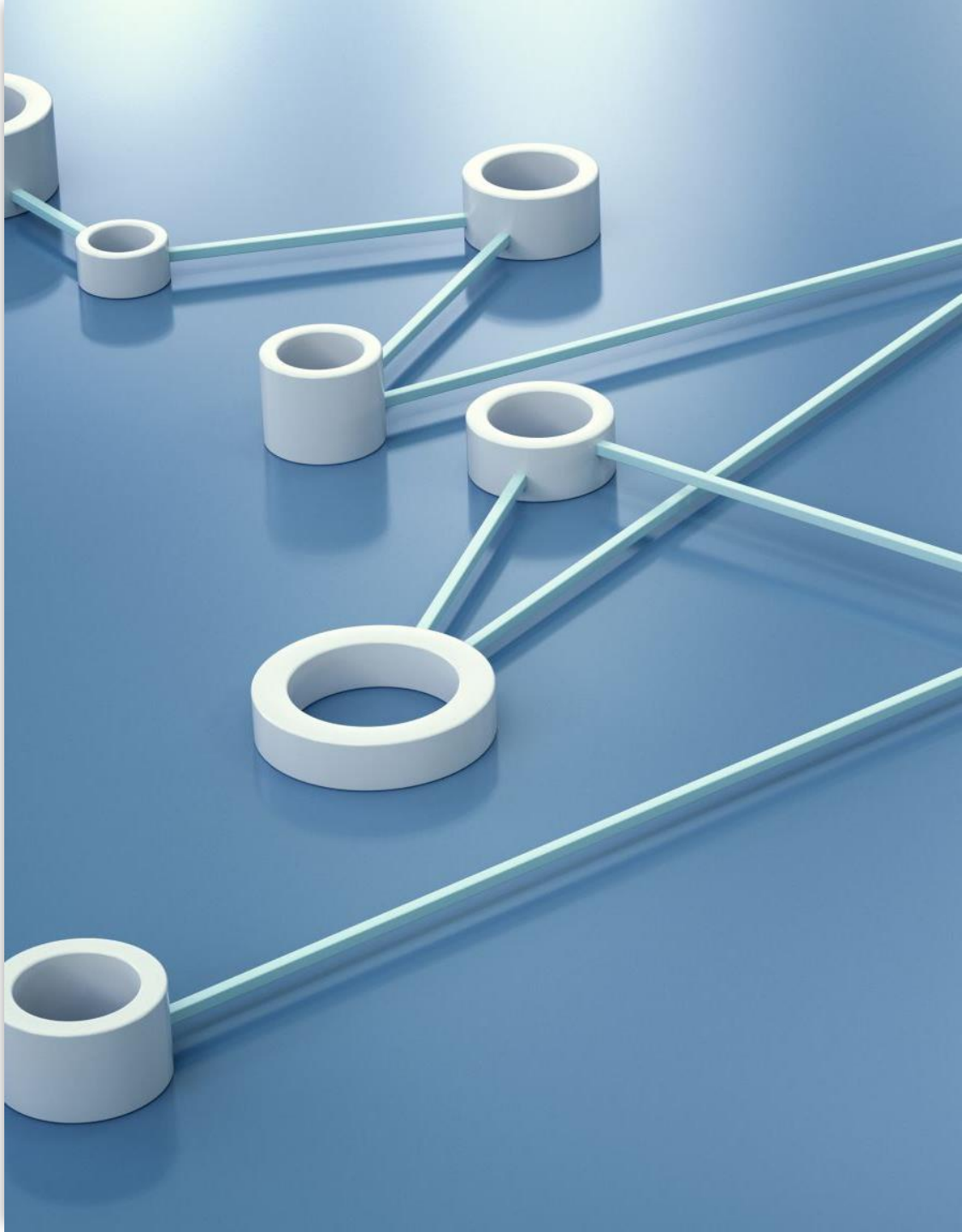
Purpose and Focus

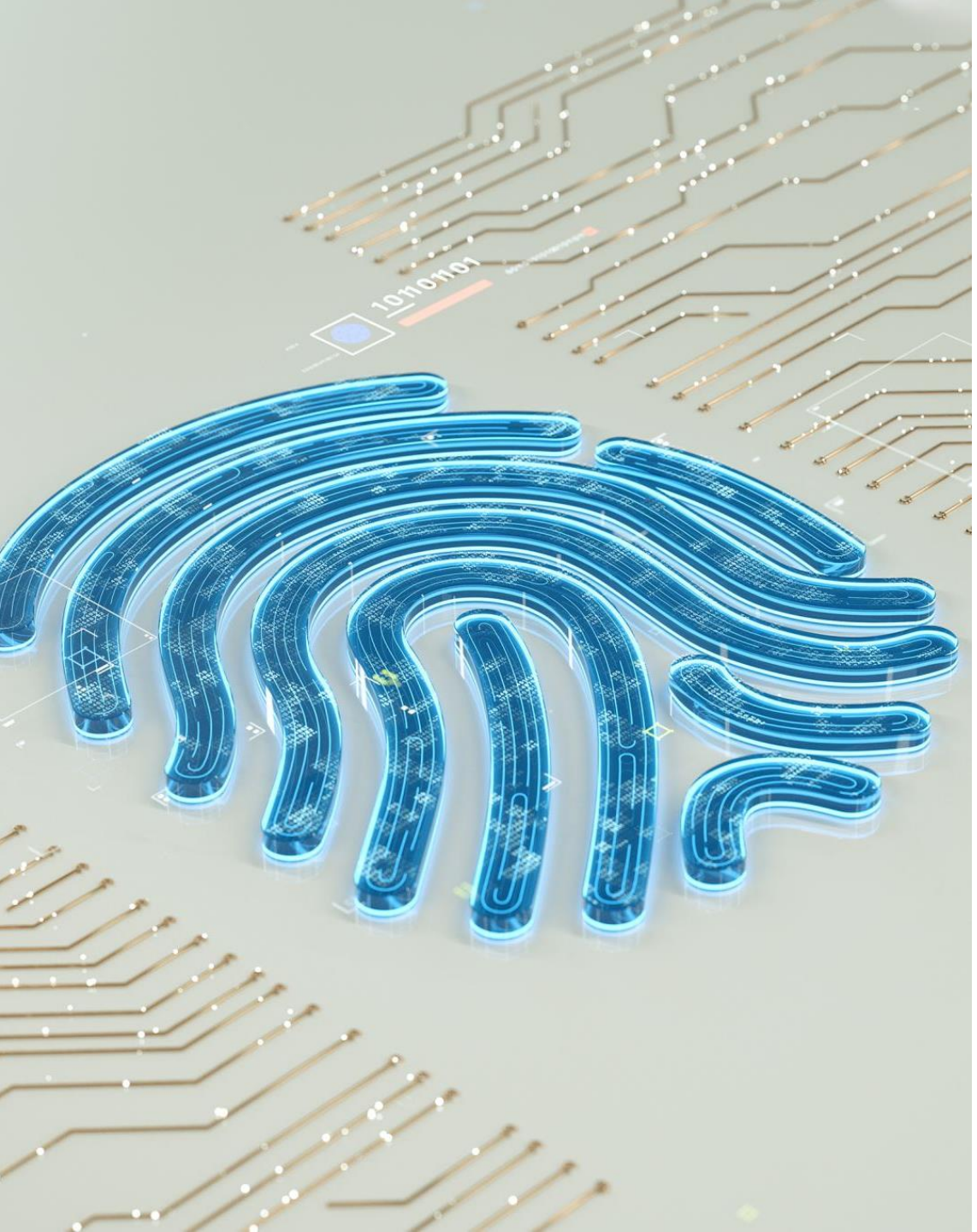
This presentation focuses on best practices for implementing security controls in shared source code repositories. As development teams increasingly collaborate using platforms like GitHub, securing source code is essential to protect intellectual property and prevent unauthorized access or attacks.

Why Repository Security Matters

Shared source code repositories often contain sensitive information such as proprietary code, configuration files, and credentials. Encryption Consulting explains that insecure repositories can expose organizations to data leaks and supply chain attacks. A compromise in one repository can affect many systems, making repository security critical.

- Shared repositories are common targets for attackers
- Source code may contain sensitive data or credentials
- Repository breaches can lead to supply chain attacks
- Attacks may impact multiple systems at once





Establish Strong Access Controls

Access control is a foundational security practice. Assembla recommends restricting access so users only have the permissions necessary for their role. GitHub also encourages using private repositories to prevent unintended exposure of source code. This reduces the overall attack surface.

- Limit repository access to authorized users only
- Use role-based access control (RBAC)
- Prefer private repositories when possible

Enable Authentication & Account Security

Strong authentication protects developer accounts from compromise. Encryption Consulting highlights MFA as a critical security control because it adds an extra layer of protection beyond passwords. Managing and rotating SSH keys and access tokens reduces the risk of stolen credentials being misused.

- REQUIRE MULTI-FACTOR AUTHENTICATION (MFA)

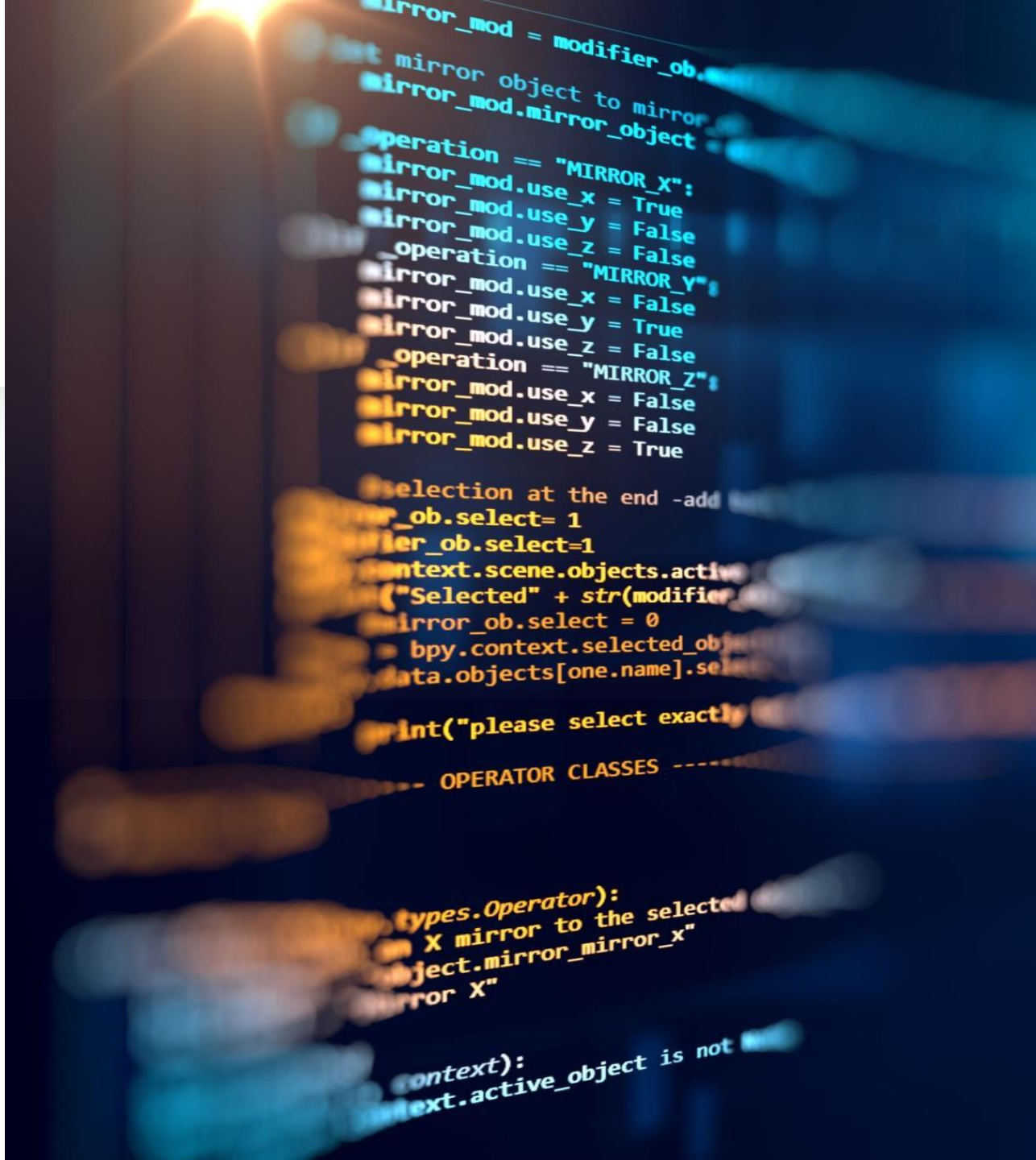
- SECURE AND ROTATE SSH KEYS AND ACCESS TOKENS

- REMOVE UNUSED OR OUTDATED CREDENTIALS

Code Scanning and Vulnerability Detection

GitHub recommends enabling automated code scanning tools to identify vulnerabilities in source code. Secret scanning and push protection help prevent accidental commits of sensitive information such as passwords or API keys. Assembla emphasizes that continuous scanning improves overall code security.

- Enable automated code scanning
- Detect vulnerabilities early in development
- Use secret scanning to prevent credential leaks
- Reduce human error in code reviews



Branch Protection and Review Policies

Branch protection rules help maintain code integrity. Protected branches prevent force pushes or direct commits to important branches. Requiring pull request reviews and automated checks ensures code is reviewed and tested before it is merged, reducing the risk of vulnerabilities entering production.

- Protect critical branches (e.g., main)
- Require pull request reviews before merging
- Enforce automated status checks

Security Policy and Reporting

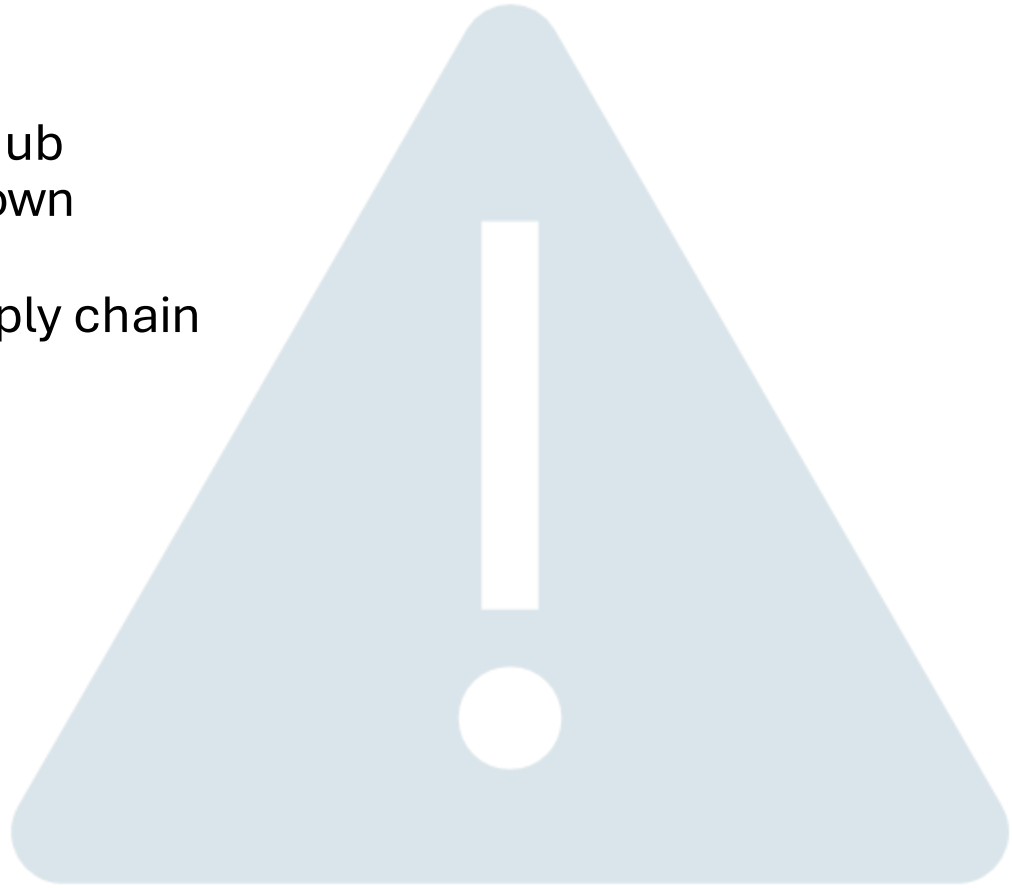
GitHub recommends adding a SECURITY.md file to repositories to clearly explain how security issues should be reported. This provides contributors with a safe and structured way to disclose vulnerabilities and helps organizations respond quickly and responsibly.

- Add a SECURITY.md file
- Define how vulnerabilities should be reported
- Encourage responsible disclosure
- Support and foster secure collaberation

Dependency & Threat Management

Modern applications rely heavily on third-party libraries. GitHub advises monitoring dependencies and enabling alerts for known vulnerabilities. Effective dependency management helps organizations identify risks early and reduce exposure to supply chain threats.

- Monitor third-party dependencies
- Enable vulnerability alerts
- Identify risks in external libraries
- Reduce software supply chain risks



Audit & Monitoring

Ongoing monitoring is necessary to maintain repository security over time. Reviewing audit logs helps detect unusual access patterns or unauthorized changes. Continuous monitoring allows teams to respond quickly before a security incident escalates.

- Review audit logs regularly

- Monitor access and activity changes

- Detect suspicious behavior early

Key Takeaways

In summary, securing shared source code repositories requires layered security controls. Layers such as access management, MFA, automated scanning, branch protection, security policies, and monitoring all work together to protect source code while supporting secure collaboration at the same time.

LIMIT ACCESS AND ENFORCE MFA

USE AUTOMATED SCANNING TOOLS

PROTECT BRANCHES AND REQUIRE REVIEWS

MONITOR DEPENDENCIES AND ACTIVITY

References

- Assembla. (n.d.). *Source code security: Best practices to protect your repositories*. https://get.assembla.com/blog/source-code-security/
- Encryption Consulting. (n.d.). *Are code repositories safe for your source code?* https://www.encryptionconsulting.com/are-code-repositories-safe-for-your-source-code/
- GitHub Docs. (n.d.). *Quickstart for securing your repository*. https://docs.github.com/en/code-security/getting-started/quickstart-for-securing-your-repository