**GitHub Advanced Security**

**Date:** October 27, 2023 (Based on context)

**Presenter:** Sania Mirch (Head of Development, Solidify)

**Attendees:** Developers, DevSecOps Professionals

**1. Introduction & Context**

* **Focus:** This presentation focuses on GitHub Advanced Security, its features, configuration, and benefits, particularly for addressing security vulnerabilities and secret management in software development.
* **Shift-Left:** The core idea is to "shift left" security practices, integrating security considerations earlier into the development lifecycle rather than treating it as an afterthought. This involves embedding security checks and tools within the developer's workflow.
* **DevSecOps:** GitHub Advanced Security is presented as a key enabler of DevSecOps, aiming to bridge the gap between development and security teams by providing tools and insights within the developer's familiar environment.
* **Problem Statement:** Despite increased awareness and tools, the number of fundamental vulnerabilities is still increasing. Much of the code used is not custom-built but consists of dependencies and borrowed code (potentially 75-80% of a project) which significantly increases risk.
* *"...we see that we are not improving that much... More code, more errors of course, but because we are already working on it, it should be better."*
* **Secret Sprawl:** Secrets being committed directly to repositories (especially public ones) is a major issue. There has been a 50% increase in GitHub secrets being committed to public repositories from December 2018 to August 2021, and even more secrets are being missed.
* *"...from December 2018 to August 2021 we have a 50% increase in the GitHub Secrets committed to public repositories."*
* **Developer-Centric:** GitHub Advanced Security is designed to be developer-first and native to the environment they are already using. This aims to reduce friction and make security a part of their daily routines.
* *"...it's so close to me as a developers developer... I don't need to jump jump out to another tool and look there and then go back to my code, it's integrated..."*

**2. Key Features of GitHub Advanced Security**

* **Dependabot (Dependency Scanning):Dependency Graph:** Visual representation of all project dependencies, including versions and licenses.
* "*...basically representation and and the view where we can see all the dependencies that we have in our application..."*
* **Vulnerability Alerts:** Notifications about vulnerabilities found in dependencies.
* **Security Updates (Automated Pull Requests):** Dependabot can automatically create pull requests to update vulnerable packages to patched versions.
* **Version Updates (Housekeeping):** Can create pull requests to update packages to newer, non-vulnerable versions even if no vulnerability is found to keep code from going legacy.
* *"... a little bit of housekeeping in your code basically not not letting your code code go to Legacy too early"*
* **License Scanning:** Checks licenses of dependencies, providing feedback in pull requests to help maintain compliance.
* **Communication with Dependabot:** Developers can use commands in pull request comments to instruct Dependabot to re-create requests, ignore minor versions, etc.
* **Mean time to remediate:** Turning on Dependabot can reduce the average time to remediate a vulnerability from 180+ days to around 24 days
* **Secret Scanning:**
* **Secret Detection:** Detects secrets in code during commits and pushes, both for known provider secrets and custom patterns that can be added.
* **Push Protection:** Can block commits or pushes that contain secrets, forcing developers to address the issue.
* **Automatic Revocation (Public Repos):** GitHub partners with cloud providers to automatically revoke leaked secrets in public repositories.
* *"...for the public repositories there is also automatic reworking so the the GitHub has partnered up with uh 35 Plus Cloud providers..."*
* **Custom Patterns:** Allows you to specify additional secrets you want to be scanned for, up to 100 per repository, 500 per org.
* **Testing:** Custom patterns can be tested against sample code to avoid false positives before being enabled.
* **Code Scanning (CodeQL):Static Analysis:** Uses CodeQL (a semantic code analysis engine) to scan code for vulnerabilities, creating a database of your code and allowing queries on that data.
* *"...creates a database of your code of the execution tree and let you create queries language queries how to query those uh that code"*
* **Deep Semantic Analysis:** Understands the structure and semantics of code to find vulnerabilities more accurately.
* **Native Dev Experience:** Results are displayed within the GitHub interface alongside code.
* **Extensible:** Supports a growing library of community-provided code queries and custom queries can also be made.
* **Language Support:** Supports a wide range of languages (with a focus on Javascript and C# in the demonstration), and is extensible through SARIF for other tools.
* *"...in most of the cases it's quite easy it's a most of the languages is just almost out of the box experience but in some languages we need to configure that as well..."*
* **Automatic Code Fixes**: Code scanning provides documentation on how to fix code, and after a fix is implemented, the alert is automatically resolved.
* **Industry Standard Checks:** Code scanning out of the box has queries for OWASP top 10 and similar.
* **Security Policy & Advisories:Security Policy:** Allows for the creation of a security policy file in the repository, which defines support versions and how to report security issues.
* **Security Advisories:** Used to communicate discovered vulnerabilities within a project to others. Includes details such as affected ecosystem, package, and versions patched.
* **Integration with vulnerability database:** When a vulnerability is reported, it can be added to the broader public vulnerability database.

**3. Organization-Level Security View**

* **Centralized Overview:** Provides a single view of security risks across the entire organization.
* **Key Metrics:** Displays statistics on enabled security features, vulnerabilities, alerts, and open issues.
* **Filtering and Sorting:** Allows for filtering and sorting of alerts by severity, repository, and type.
* *"...a great way for us as a security people to actually get to know our organization and better..."*
* **Access Control:** Visibility is limited to relevant personnel, such as repository administrators or security champions.

**4. Key Takeaways**

* **Proactive Security:** GitHub Advanced Security facilitates a proactive approach to security.
* **Automation:** Leverages automation to address vulnerabilities and secret management.
* **Developer Integration:** Integrates with developer tools and workflows to reduce friction and increase adoption.
* **Improved Time to Remediation:** Significantly reduces the time it takes to find and fix vulnerabilities.
* **Reduced Risk:** Reduces the risk of breaches and data leaks.
* **Enterprise vs. Open Source:** Enterprise Licenses give access to organizational-level security views, custom secret scanning and more. Open source repos benefit from Dependabot and some base-level secret scanning.
* **Cost of Licenses:** Based on active user committers in the last 90 days.

**5. Questions Raised (and unresolved)**

* **Quality of Security Advisories:** How can Github ensure the quality of advisories, to prevent malicious advisories? Github notes there is a manual review process.
* **Purging Secrets:** How can you purge committed secrets that are not changeable, such as biometric data, from Github history? The presenter noted there are destructive tools but push protection is key.
* **Code Analysis Input Field Tracking:** Can the code analysis track vulnerabilities all the way from backend methods to front-end untrusted input fields? It can, but it depends on if they are within the same solution or not.
* **GitHub Actions Workflow Scanning:** Does the CodeQL code scanner also scan the GitHub Actions workflow definitions, or just source code? Further research is required here.
* **Integration with Copilot:** How GitHub Advanced Security works with Co-pilot. Co-pilot and the security suite are two separate tools.
* **Supported Package Managers:** What are the limitations to which dependencies Dependabot supports? Further research needed on which are supported and which aren't.

This briefing document aims to capture the essence of the presentation and provide a concise overview of the key points for anyone interested in understanding or utilizing GitHub Advanced Security.