Project Proposal - Github Repo Integration Security Scanner

1. Introduction

A typical company has 100+ SaaS services in use on top of their cloud workloads and on-prem systems. Many of these systems and services are connected and share company data from one system to another. Some of the shared data is sensitive and companies should know where the data is flowing and how the integration is protected.

While not all companies identify as software companies, many of them at least have some IT integration infrastructure code in Github or Gitlab. This code may be built by in-house engineers, but even more commonly it's built by contractors who come in to build a point-to-point integration and then leave.

Finding out which integrations a company has is currently a daunting task. Even more challenging is to know if the integrations have been built securely or not.

2. Project goals

Goal is to build a scanner which, when granted access to a Github repository, can produce a list of likely integrations in the repo. Here an integration is referring to network connections to third party services: Either the source code connects to a 3rd party (including webhooks) or it uses a 3rd party sdk that connects out. Typically both kinds of integrations require some kind of an API key or token to be provided.

The deliverable itself should be code or a container that can live in a backend environment and can be tasked by other backend services to scan particular repositories and return a list of findings and related metadata.

3. Technologies

The scanner itself can be built with any modern backend technology, e.g. Python.

The team can select the best technology for detecting integrations, but some simpler or more complex form of an AI is probably a good idea to include. You can find some ideas for finding integrations in the appendix at the end of this document.

4. Requirements for the students

The team will require backend development and design skills. Previous experience with software integrations and Git is a plus. This topic also allows the team to research approaches to analysis and hence some creativity is a big plus.

The difficulty of the topic is from moderate to demanding depending how far the automated analysis is taken.

5. Legal Issues

Intellectual Property Rights (IPR):

- A. All IPRs to all Results will be transferred to the Client.

Confidentiality:

- A. The client will share some confidential information with the students.

Any other legal issues, e.g. if the default contract template does not cover something that needs to be agreed

6. Client

Valo Security is a startup established in 2023. The service hasn't launched yet at the start of the course. Valo team consists of cyber security industry experts with backgrounds in companies such as F-Secure, Oura, Spotify, Google, Snap, Salesforce, ServiceNow, and Nokia.

Representative of the client is Valo Security CTO Mika Ståhlberg. The representative has 25 years of experience in both security and software development, has been using Scrum since 2004 and has been the client representative on this class before – and has also taken the course in 1999-2000.

Valo team is prepared to help the project team as much as needed, but as a new startup a working room or similar facilities are not available. But kickoffs with pizza and such will certainly be provided.

Client representative:

Valo Security
Product Owner
Mika Ståhlberg
e-mail: mika@valosecurity.com
+358 50 518 9050

Appendix: Some ideas for features to use in detecting integrations

Search for Keywords Search the codebase for common keywords or terms associated with SaaS integrations, such as "API key," "access token," "endpoint," "webhook," "OAuth," and the names of specific SaaS providers (e.g., "Stripe," "Twilio," "Salesforce"). This can help identify potential integration points. Attention should be paid to the performance of the string search algorithm chosen – e.g. Aho-Corasick is a lot better than basic search.
Configuration Files Look for configuration files that might contain credentials or configuration details for external services. These files could include API keys, tokens, URLs, and other relevant information.
Dependency Management Examine the project's dependency management files (e.g., package.json, requirements.txt). Dependencies often indicate the use of external libraries or SDKs that interact with SaaS services.
Network Requests Scan the code for network requests, HTTP requests (GET, POST, etc.), and relevant libraries like Fetch, Requests, or Axios. These could indicate interactions with external APIs.
Webhooks and Callbacks Check if there are any implementations of webhook endpoints or callback functions. These could be related to SaaS services sending notifications or data to the application.
Integration Documentation If the project has documentation, search for any mentions of external integrations. Developers often document the integration points and how they're used.
Environment Variables Look for references to environment variables, as these might contain sensitive information like API keys. Review any scripts or configuration that sets up these environment variables.
Third-Party Libraries Check the third-party libraries and frameworks used in the project. Some libraries might be specific to integrating with certain SaaS services.
Code Scanning Logic Consider using code scanning libraries that can automatically identify certain connections. These kinds of libraries are typically used for vulnerability analysis (static code analysis), e.g. E.g. https://github.com/returntocorp/semgrep
Testing and Logging If available, review the testing code and logging statements. They might reveal interactions with external services during test scenarios or runtime.
Version Control History Look at the commit history to identify any changes related to integrations or connections with external services. Comments, commit messages, and diffs can provide insights.