Shift Gears within Information Security

Product Security Journey

Steven Carlson - 2023

Steven Carlson

Software Engineer who is passionate about clean secure code.

Helpdesk -> Software Engineer -> Security -> DevOps -> **Product Security**

https://about.me/rockrunner

The Talk

Are you in the correct room?

“This talk will be focused on discussing war stories from a product architect/engineer who lives within an information security department and is passionate about driving change. Attendees will get to experience a few different routes that have lead to success and others that might need to avoided. As an ever-evolving space, when reducing risk and deploy safe products to the market, we all have to find the correct gear to get us down the road.”

Keeping Security Top of Mind

Forethought vs Afterthought

Terms

Common Language

- AppSec is about securing the application.
- DevSec is about securing the developer and their actions.
- DevSecOps is about securing the application build chain.
- Product Security is about the end-to-end security of their organization’s software products.

Product Security

We, the Security Team

Recognize That Engineering Teams

- Want to do the right thing
- Are closer to the business context and will make trade-off decisions between security and other tasks
- Want information and advice so those trade-off decisions are more informed

Pledge to

- Lower the cost/effort side of any investment in developer security tools or practices
- Assist with preventative initiatives as we beg for your assistance reacting to security incidents

Understand that

- We are no longer gate keepers, but rather tool-smiths and advisors


Cybersecurity Sales Engineer
Security Engineer
Cybersecurity Software Engineer
Product Security Engineer
Cloud Security Engineer
DevSecOps
Cybersecurity Director
Application Security Engineer

War Stories

Overview

The plan

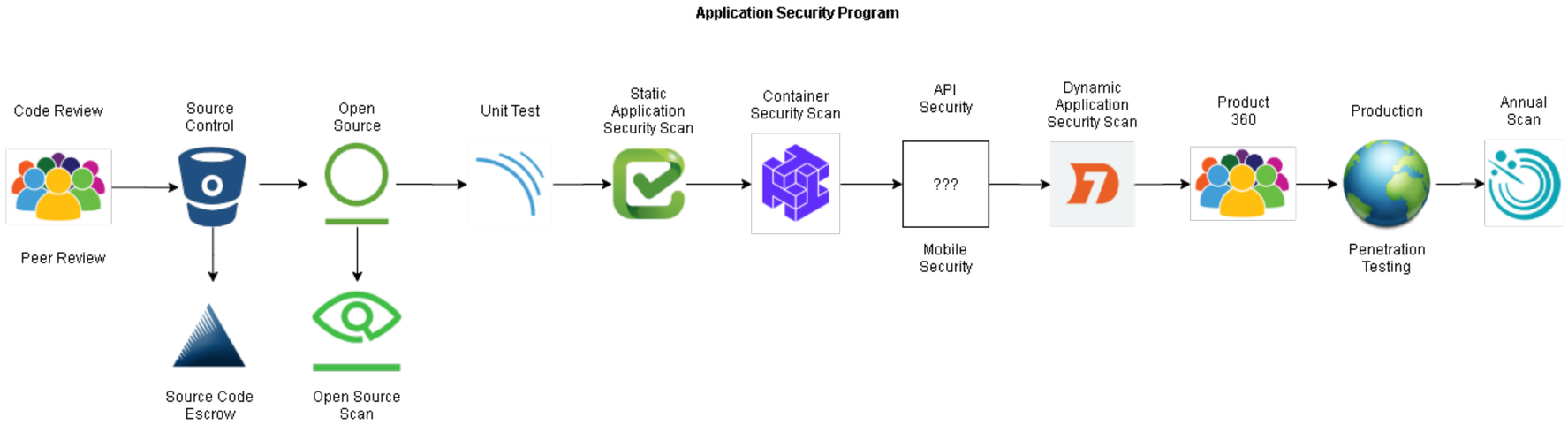
- Development Onboarding
- Empowering Champions
- Automation?
- Executive Buy-in
- Reporting Security Findings

Development Onboarding

K.I.S.S

Keep It Simple, Stupid

- People
- Process
- Technology
- Governance



Gitlab SDK

https://python-gitlab.readthedocs.io

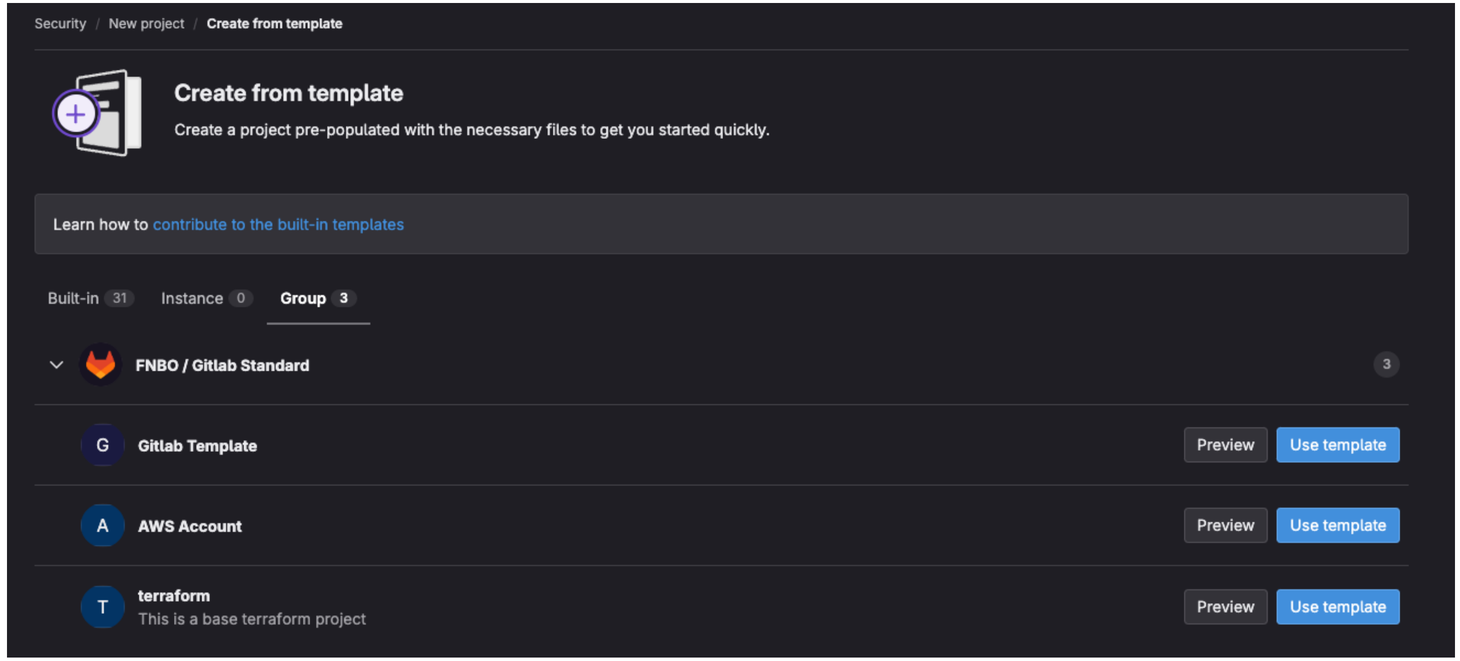
- People: Pure Dev
- Process: GitOps
- Technology:
  - GitLab
  - Docker
  - Python

```python
def main(args):
    sf.configure_logging()

    if str(args.gitToken) == "NONE":
        sf.logging.error("Missing environment variable GIT_TOKEN.. Exit")
        sys.exit(1)

    gl = gitlab.Gitlab('https://' + str(args.gitHost) + "/", private_token=args.gitToken)
    gl.auth()

    group = gl.groups.get(args.groupPath)
    sf.logging.info("GroupID: " + str(group.get_id()) + ", GroupName: " + str(group.attributes['name']))
```

Gitlab Template

https://docs.gitlab.com/ee/user/admin_area/custom_project_templates.html

- People: GitLab Access
- Process: Merge Requests
- Technology:
  - GitLab



https://gitlab.com/nebraska-code-conference/gitlab-standard/gitlab-template

Empowering Champions

Shared Partnership

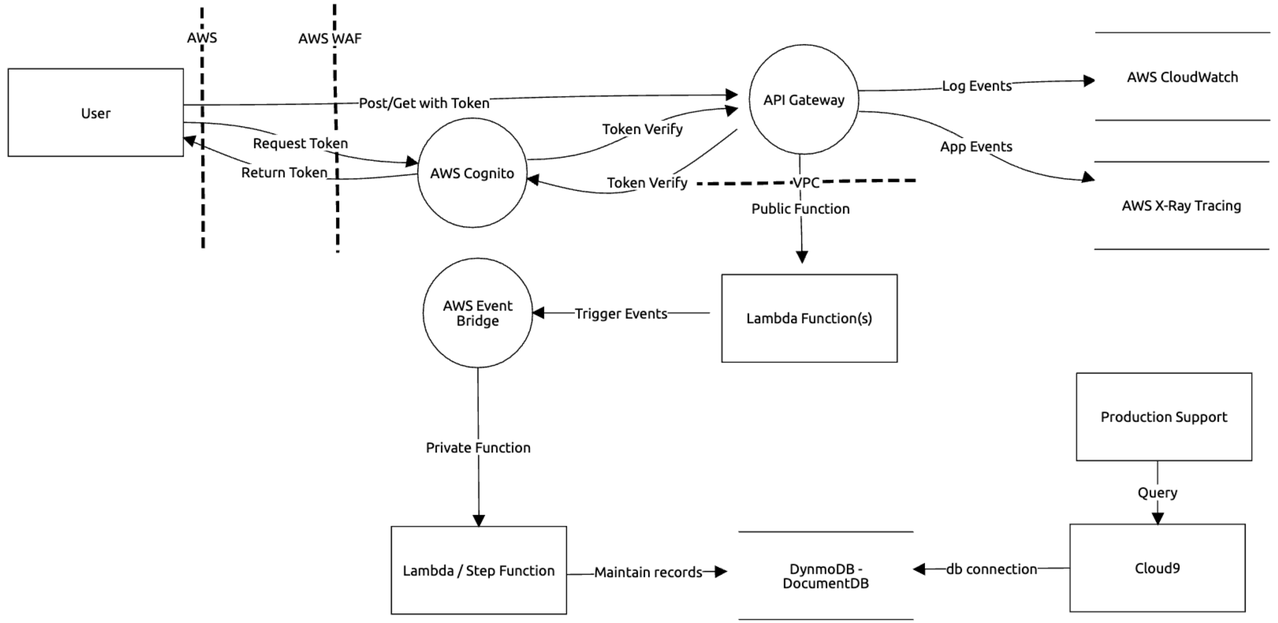
Not just knighting someone

- Create a clear picture
- Share ownership of the product's security score
- Establish a threat intel feed
- Recommend policy and standard modification

Threat Dragon

https://owasp.org/www-project-threat-dragon/

- People: Architect
- Process: Manual Review
- Technology:
  - Source Control
  - IDE
  - Threat Dragon

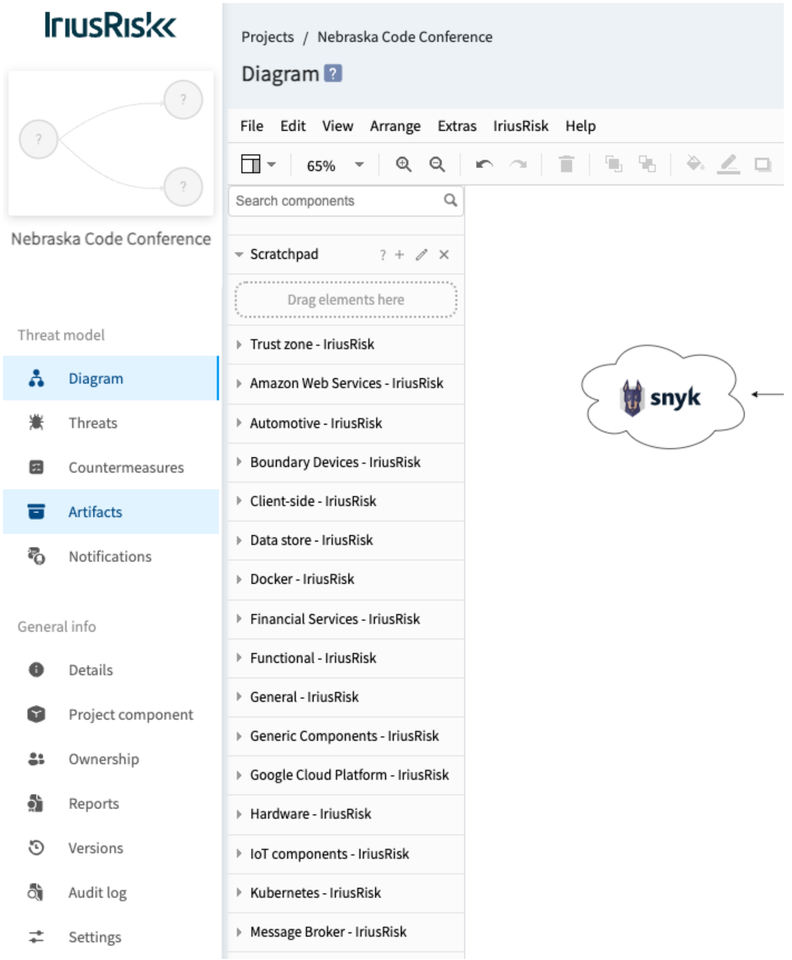


https://github.com/RockRunner007/security-programs/tree/master/threat-models

IriusRisk

https://community.iriusrisk.com/ui#!login

- People: Product Staff
- Process: Life Cycle
- Technology:
  - IDP
  - IriusRisk

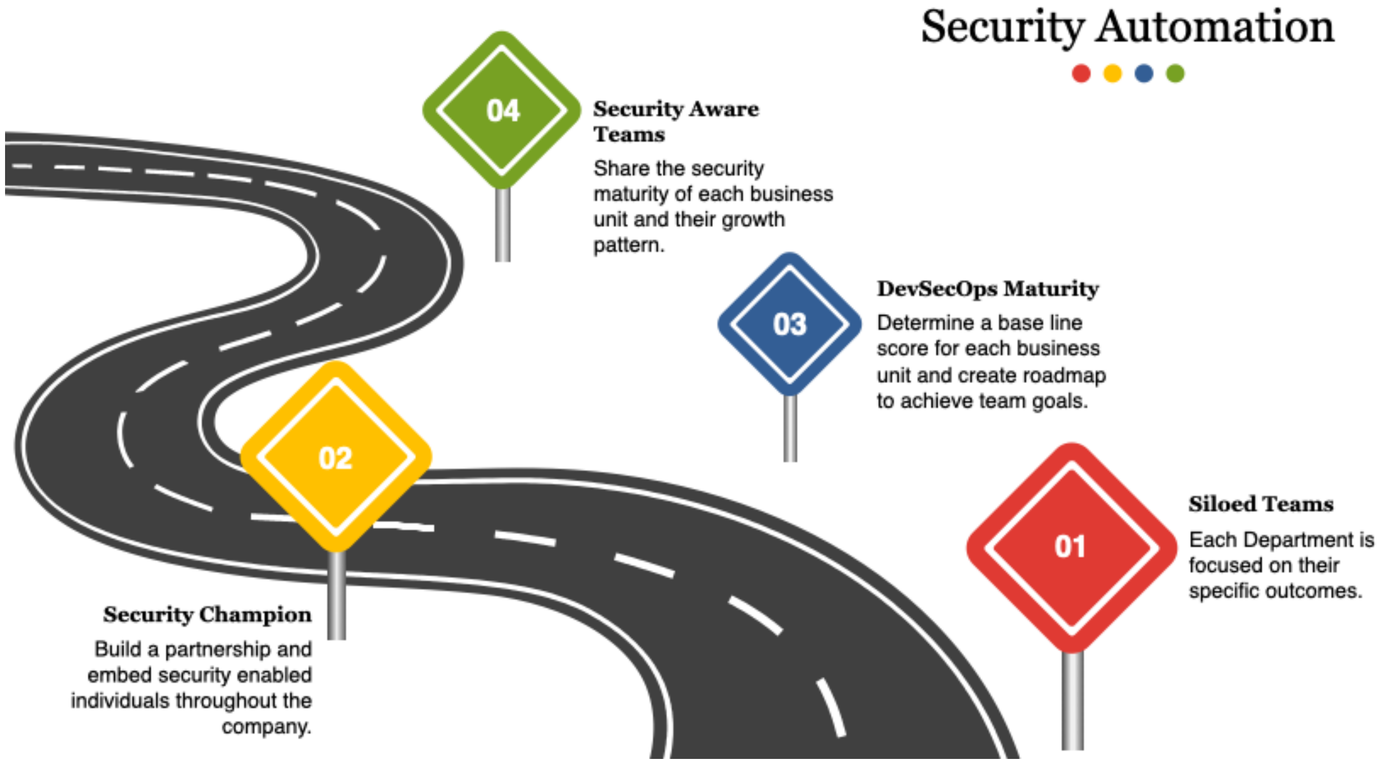


Automation?

Context is Key

Build the case with Product team(s)

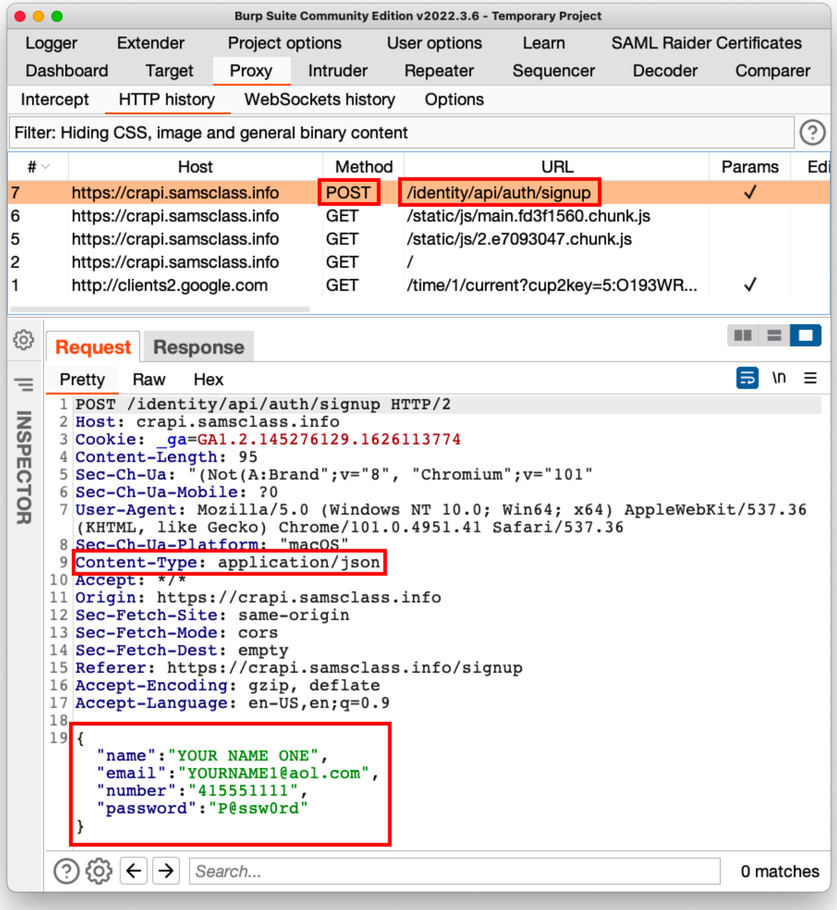
- Training and Education
- Build Relationships
- Regular Feedback Loops



Postman + Burp

https://www.postman.com/api-platform/api-testing/

- People: Security Team
- Process: Internal PEN test
- Technology:
  - Postman Enterprise
  - Burp Suite Professional

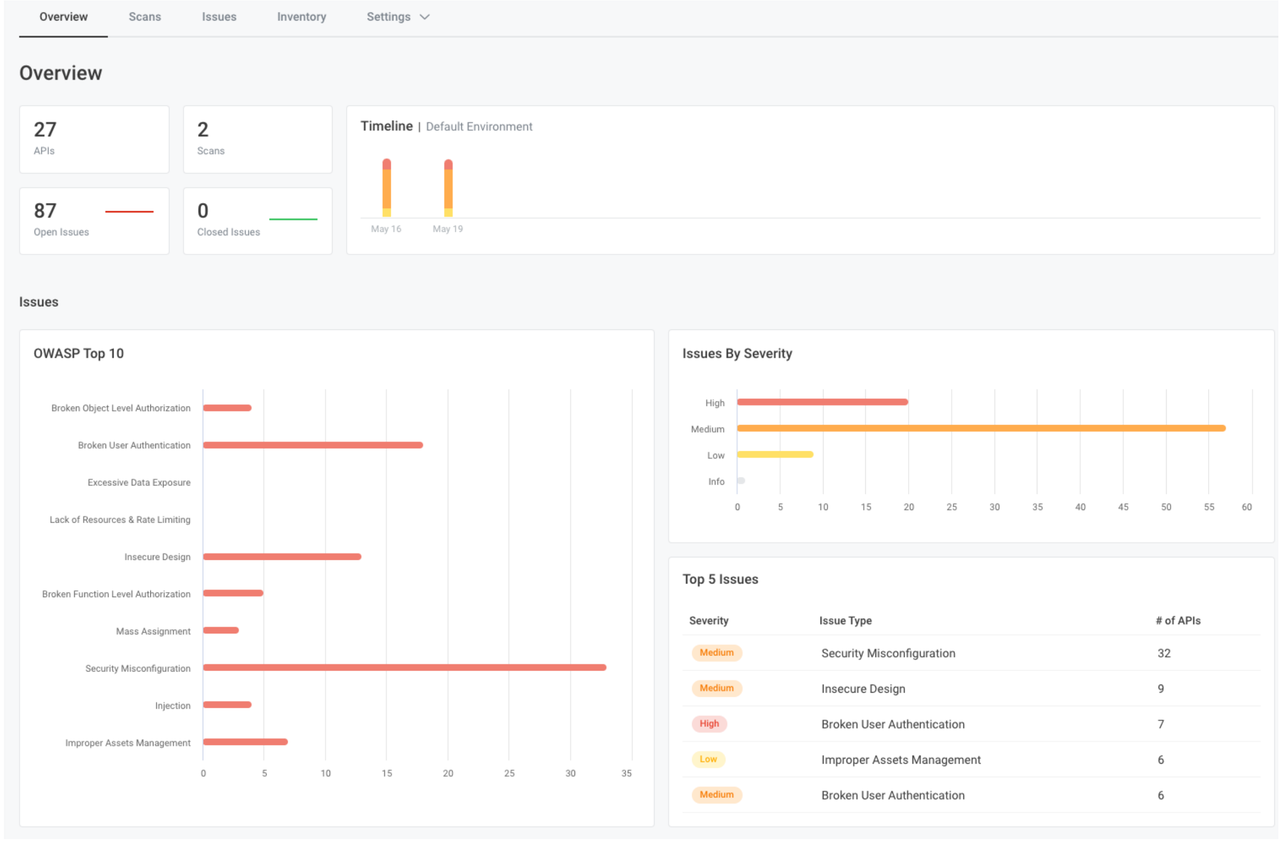


https://github.com/RockRunner007/crAPI

Noname Security

https://nonamesecurity.com/

- People: Dev and Security Team
- Process: SDLC + Review
- Technology:
  - Noname Platform
  - OpenAPI Specification



Executive Buy-in

Shift Left and Extend Right

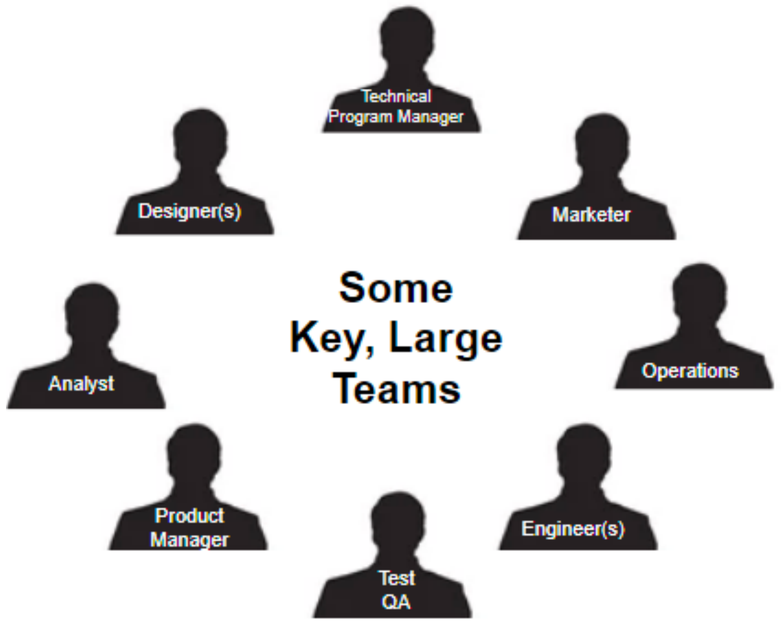
Oldie but a goodie

- Embedding a security engineer within a product team
- Accept work that is achievable and generally understood
- Establish a position within leadership and engineering

Enablement Team

Team Structure and their actions

- People: Product Leadership
- Process: Workshopping
- Technology:
  - Text editor of choice



Product Security Program

Risk vs Threat Organization

- People: Security Leadership
- Process: Workshopping
- Technology:
  - Text editor of choice

Product Security Program

Threat-led defensive programming is an approach that prioritizes the identification and mitigation of potential threats throughout the software development lifecycle. By integrating security measures from the beginning and ensuring their sustainability, this approach enhances the overall resilience and robustness of software systems. Here's an explanation of each element:

- Proactive: Proactive defensive programming involves anticipating and addressing potential threats before they can be exploited. This includes conducting thorough threat modeling and risk assessments during the design phase to identify potential vulnerabilities and attack vectors. By proactively identifying security weaknesses, developers can implement appropriate security controls and countermeasures to prevent or mitigate potential threats.
- Defensible: Defensible programming focuses on building software systems that are resistant to attacks and can effectively defend against them. This involves implementing secure coding practices, adhering to security standards and guidelines, and leveraging defensive programming techniques. Developers should validate and sanitize input, implement access controls and proper error handling, and apply encryption and authentication mechanisms. By building a defensible software system, developers can minimize the likelihood and impact of successful attacks.
- Sustainable: Sustainable defensive programming emphasizes the long-term maintenance and management of secure software systems. This includes regular security assessments, vulnerability scanning, and penetration testing to identify and address emerging threats. It also involves staying up to date with security patches, updates, and best practices. Developers should establish processes and resources to ensure that security measures remain effective and relevant throughout the software's lifecycle, even as new threats emerge.
- Integrated: Integrated defensive programming involves the seamless integration of security practices into the software development process. Security should not be an afterthought but rather an integral part of every stage, from requirements gathering to deployment. By integrating security controls, code reviews, and testing into the development workflow, developers can identify and address vulnerabilities early on. Collaboration between developers, security teams, and other stakeholders is crucial for a holistic and integrated approach to security.

By adopting a threat-led defensive programming approach that is proactive, defensible, sustainable, and integrated, developers can build software systems that are resilient against attacks and provide robust protection for sensitive data and critical functions. This approach helps ensure that security is not an add-on but a fundamental aspect of the software development process, ultimately enhancing the overall security posture of the system.

https://github.com/RockRunner007/security-programs/blob/master/product-security.md

Reporting Security Findings

Proof is in the Pudding

Find out the truth before spending money

- Should we partner with this vendor?
- Should we use this vendor’s software?
- How often should we require 3rd party PEN tests?
- How will this software effect our network?

N+1 Tools

Tools for everyone!!!

- People: Security
- Process: Manual Review
- Technology:
  - ASM
  - Security Headers
  - SSL Tester
  - N+1

GitLab Automation

Sbom is like the car fax for your 3rd party software

- People: Dev + Security
- Process: CI
- Technology:
  - CI Tool
  - Cyclonedx CLI
  - Snyk or SBOM Vendor

```yaml
artifacts:
  stage: artifacts
  image: node:latest
  allow_failure: true
  script:
    - apt-get update && apt-get install -y jq python3 python3-pip python3-venv
    - pip3 install cyclonedx-bom
    - cd ${CI_ENVIRONMENT_NAME}
    - python3 -m venv . venv
    - source bin/activate
    - pip3 install -r requirements.txt
    - echo "generate sbom"
    - cyclonedx-py -r --format json -o sbom.json
  after_script:
    - ls
    - ls ${CI_ENVIRONMENT_NAME}
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
  artifacts:
    when: always
    untracked: false
    expire_in: 30 days
    paths:
      - "${CI_ENVIRONMENT_NAME}/sbom.json"
```

https://github.com/RockRunner007/gitlab-templates/blob/main/sbom/sbom.gitlab-ci.yml

Manual

Manual

Linux Command: man man

- Build security, as more than bolt it on.
- Rely on empowered product teams, more than security specialists.
- Implement features securely, more than security features.
- Rely on continuously learning, more than end-of-phase gates.
- Adopt a few key practices deeply and universally, more than a comprehensive set poorly and sporadically.
- Build on culture change, more than policy enforcement.

Resources

Books, Website, and more

Books & Publications

- Application Security Program Handbook by Derek Fisher
- Designing Secure Software by Loren Kohnfelder
- Clean Code by Robert Martin
- Software Transparency by Chris Hugh and Tony Turner
- Threats by Adam Shostack

Online

- SecurityChampionSuccessGuide.org
- attack.mitre.org
- nist.gov/itl/csd/secure-systems-and-applications
- hockeyinjune.medium.com/product-security-14127b5838ba
- santikris2003.medium.com/product-security-dev-sec-tips-2fdb1698a3b3
- https://media.defense.gov/2023/Jun/28/2003249466/-1/-1/0/CSI_DEFENDING_CI_CD_ENVIRONMENTS.PDF

Glossary

Words and things

- Application Security - the process of developing, adding, and testing security features within applications to prevent security vulnerabilities against threats.
- Infrastructure Security - the security provided to protect infrastructure, especially critical infrastructure such as cloud or datacenter resources.
- Security Operations - a centralized unit that deals with security issues on an organizational and technical level.
- Software Development Life Cycle (SDLC) - a conceptual framework describing all activities in a software development project from planning to maintenance. This process is associated with several models, each including a variety of tasks and activities.

Steven Carlson

Software Engineer who is passionate about clean secure code.

Helpdesk -> Software Engineer -> Security -> DevOps -> **Product Security**

https://about.me/rockrunner

Feedback

Please remember to fill out the evaluation forms