The Diana Initiative

Floating the Goat

How to use DevSecOps to secure OWASP WebGoat

Chloe Potsklan, CISSP, M.Sc.

Agenda

whoami

High-Level Overview

Web App Sec, CI/CD, DevSecOps, Cloud Sec,
Open-Source, OWASP WebGoat, Leading the Change

DevSecOps Pipeline

Security Architecture and Design, Threat Modeling, DAST,
Logging, and Automation Server

Demo

Lessons Learned and Next Steps

Questions, Shoutouts, Resources, Appendix, TYSM

Chloe Potsklan, CISSP, M.Sc.

Cybersecurity Engineer, she/her

Chloe is a cyber security engineer with expertise in endpoint security, security architecture, and application security.

Deloitte.

savvycoders

FORDHAM UNIVERSITY

CISSP® | Certified Information Systems Security Professional

What is web application security?

What's DevSecOps?

What are CI/CD pipelines?

What’s cloud security?

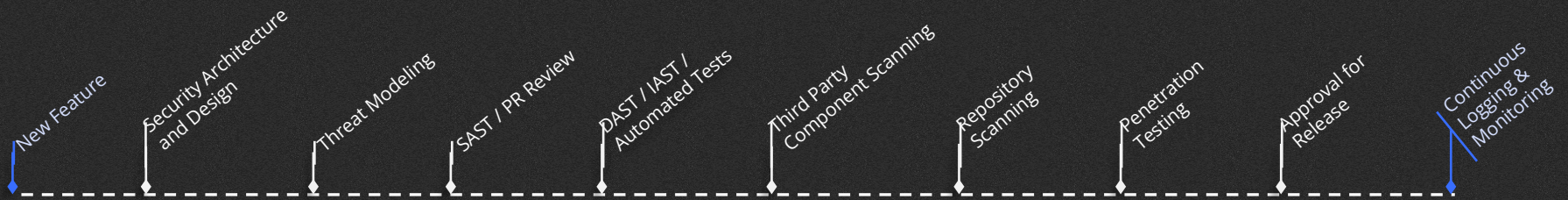
What are containers?

Open source?

OWASP WebGoat?

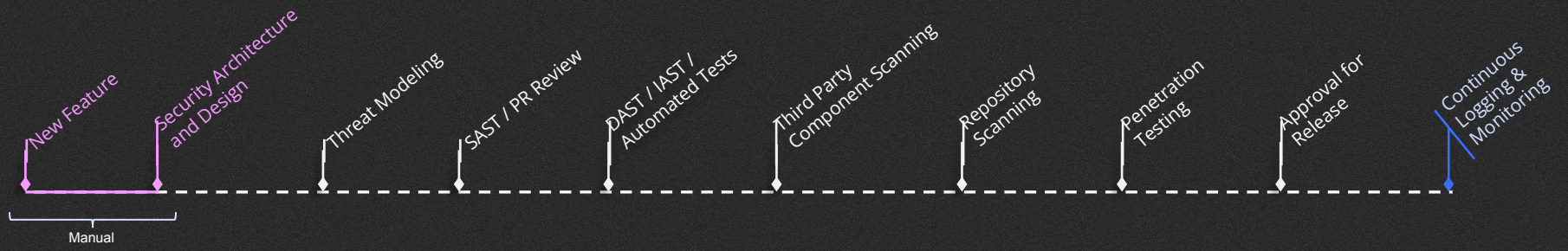
Leading the Change?

Securing the pipeline from end-to-end

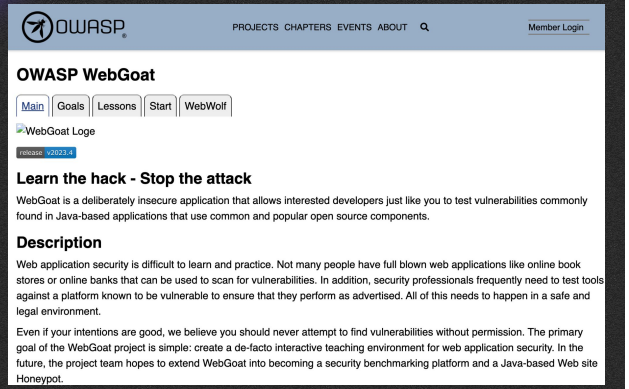


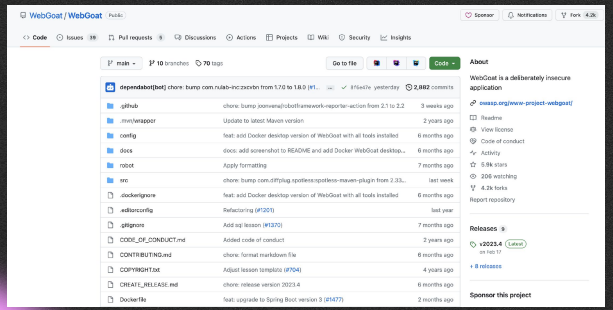
Continuous Integration

Securing the pipeline from end-to-end



Security Architecture & Design





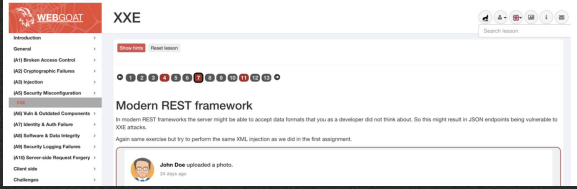
Introduction

WebGoat is a deliberately insecure web application maintained by OWASP designed to teach web application security lessons.

This program is a demonstration of common server-side application flaws. The exercises are intended to be used by people to learn about application security and penetration testing techniques.

WARNING 1: *While running this program your machine will be extremely vulnerable to attack. You should disconnect from the Internet while using this program.* WebGoat's default configuration binds to localhost to minimize the exposure.

WARNING 2: *This program is for educational purposes only. If you attempt these techniques without authorization, you are very likely to get caught. If you are caught engaging in unauthorized hacking, most companies will fire you. Claiming that you were doing security research will not work as that is the first thing that all hackers claim.*



1. Run using Docker

Already have a browser and ZAP and/or Burp installed on your machine in this case you can run the WebGoat image directly using Docker.

Every release is also published on DockerHub.

```
docker run -it -p 127.0.0.1:8080:8080 -p 127.0.0.1:9090:9090 webgoat/webgoat
```

If you want to reuse the container, give it a name:

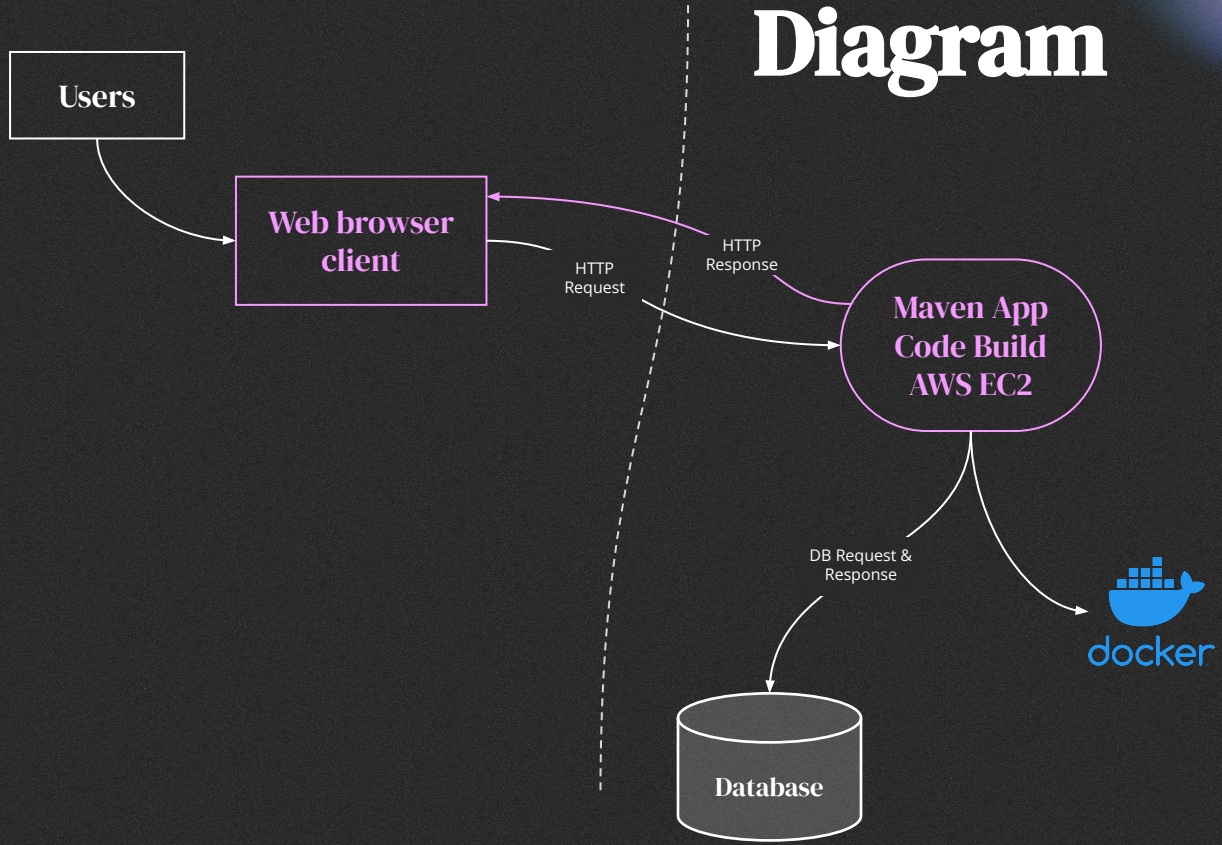
```
docker run --name webgoat -it -p 127.0.0.1:8080:8080 -p 127.0.0.1:9090:9090 webgoat/webgoat
```

As long as you don't remove the container you can use:

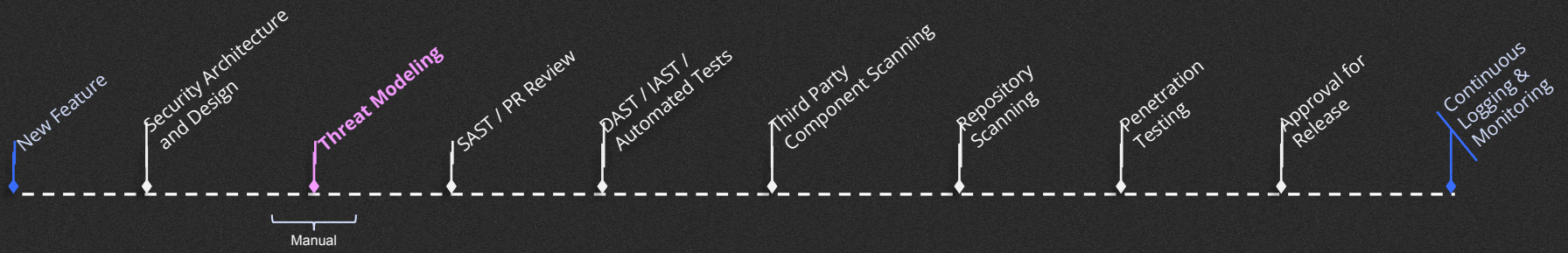
```
docker start webgoat
```

This way, you can start where you left off. If you remove the container, you need to use `docker run` again.

Floating the Goat Web Architecture Diagram

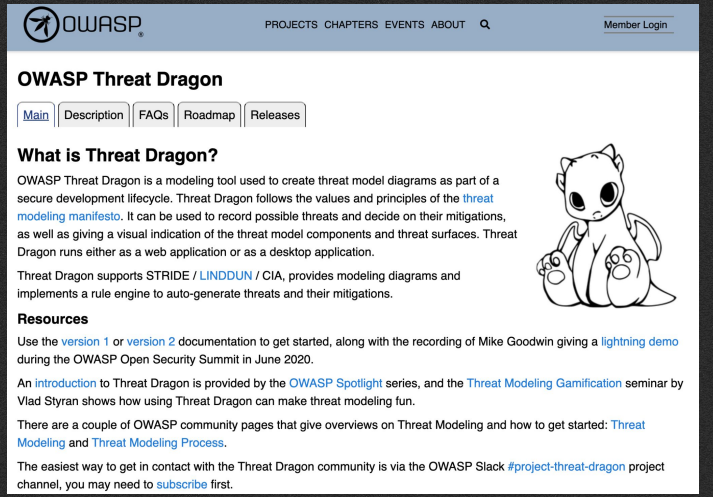


Securing the pipeline from end-to-end



Threat Modeling

OWASP Threat Dragon



OWASP Threat Dragon

Desktop installation

Home > Docs-2 > install-desktop

OWASP Threat Dragon

Threat Dragon comes in two variants, a desktop application and a web application.

Desktop application install instructions

Installable versions are available for download from the OWASP GitHub area:

- Windows (64 bit) installer
- MacOS installer
- Linux snap, AppImage, debian and rpm installers

Linux installer and AppImage

Packages for both Debian and Fedora Linux on AMD64 and X86-64bit platforms can be downloaded from the releases folder. Alternatively a platform independent snap installer can be downloaded, or use the AppImage provided.

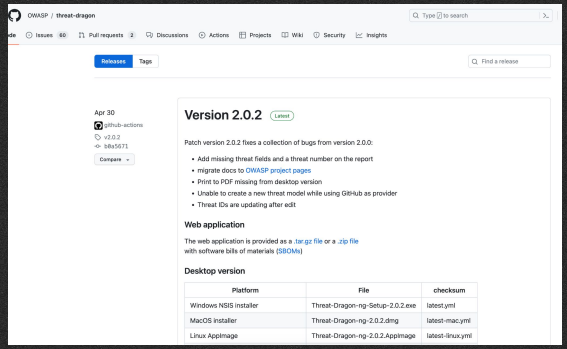
MacOS installer

Download the .dmg MacOS installer from the releases folder. Open the download and drag 'OWASP Threat Dragon' to the application directory. When the copy has finished then Threat Dragon can be run from launchpad or from Finder -> Applications.

Windows installer

Download the Windows .exe installer from the releases folder. Run the installer and invoke the application from the shortcut.

Table of Contents



OWASP Threat Dragon using STRIDE

Floating the Goat Threat Model

Owner: @cl_op
Reviewer: Users
Contributors:
Date Generated: Wed Aug 02 2023

OWASP Threat Dragon

Executive Summary

High level system description

Utilizing an open source project of an intentionally vulnerable web application, OWASP WebGoat on an AWS EC2 instance pulled from a docker image and then continue to set up a development environment and utilize different tools, build then test code, automate and monitor the pipeline, and finally shift our focus to continuous pipeline improvement

Summary

Total Threats	8
Total Mitigated	0
Not Mitigated	8
Open / High Priority	0
Open / Medium Priority	8
Open / Low Priority	0
Open / Unknown Priority	0

Floating the Goat Threat Model

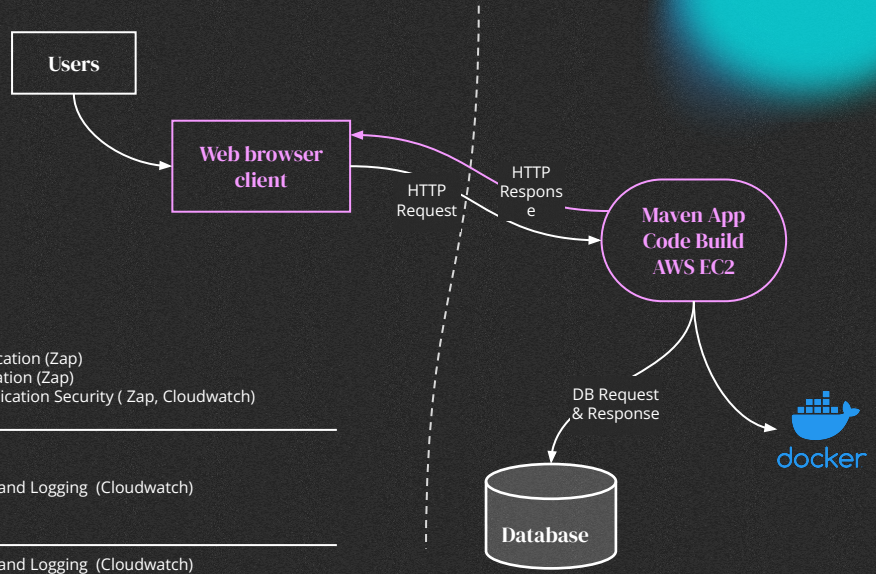
Localhost
browser (Actor)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
3	Spoofing	Spoofing	Medium	Open		Involves illegally accessing and then using another user's authentication information, such as username and password	Authentication: Who are you? Authentication is the process where an entity proves the identity of another entity, typically through credentials, such as a user name and password Authorization: What can you do? Authorization is how your application provides access controls for resources and operations Communication Security: Who are you talking to? Communication Security ensures all communication done is as secure as possible

maven app
code build
aws ec2 (Process)

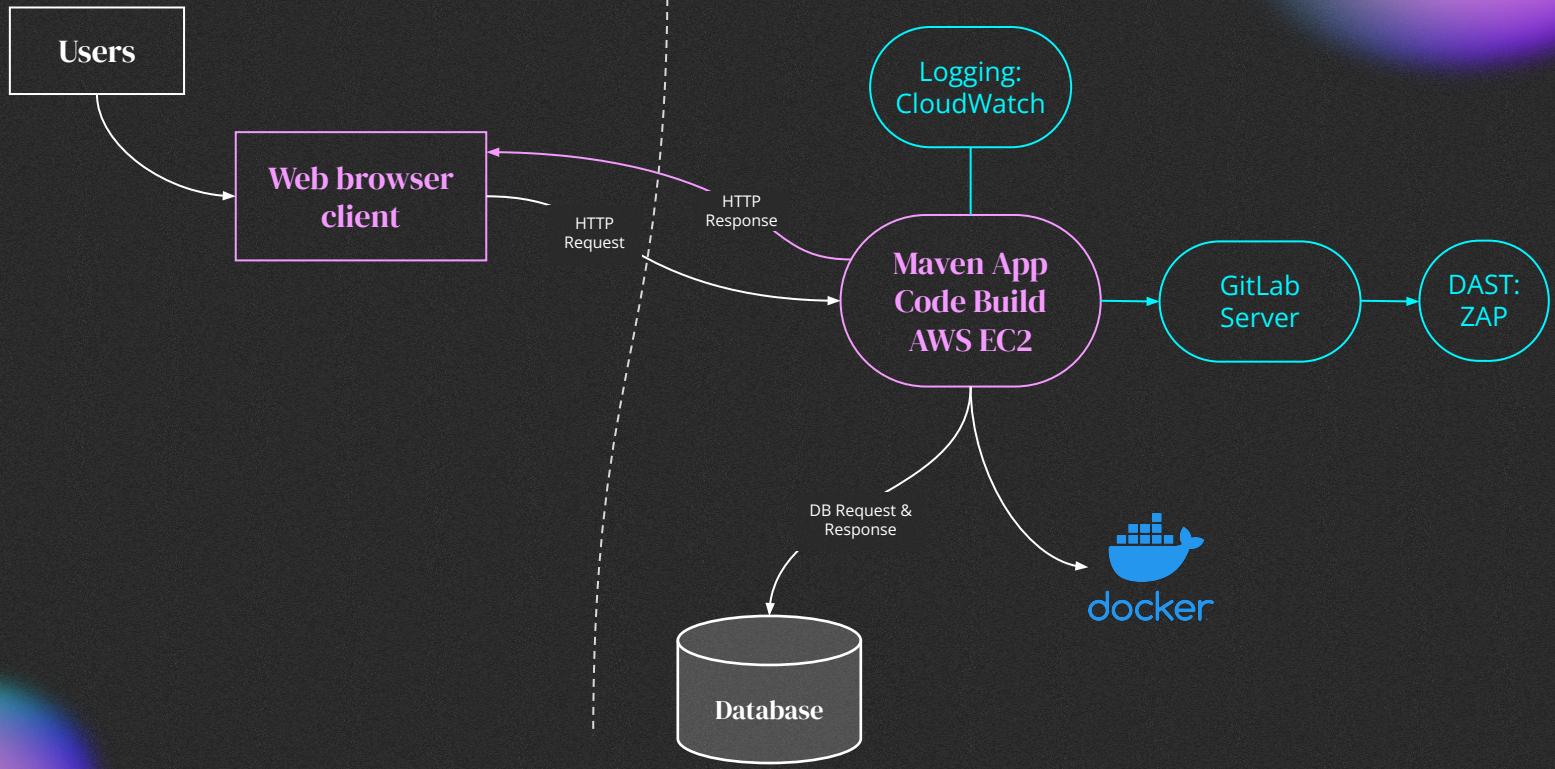
Number	Title	Type	Priority	Status	Score	Description	Mitigations
5	Repudiation	Repudiation	Medium	Open		Associated with users who deny performing an action without other parties having any way to prove otherwise—for example, a user performs an illegal operation in a system that lacks the ability to trace the prohibited operations. Non-Repudiation refers to the ability of a system to counter repudiation threats. For example, a user who purchases an item might have to sign for the item upon receipt. The vendor can then use the signed receipt as evidence that the user did receive the package	Auditing and Logging: Who did what and when? Auditing and logging refer to how your application records security-related events
8	Spoofing	Spoofing	Medium	Open		Provide a description for this threat	Provide mitigation or prevention for this threat

Threat modeling the proposed architecture

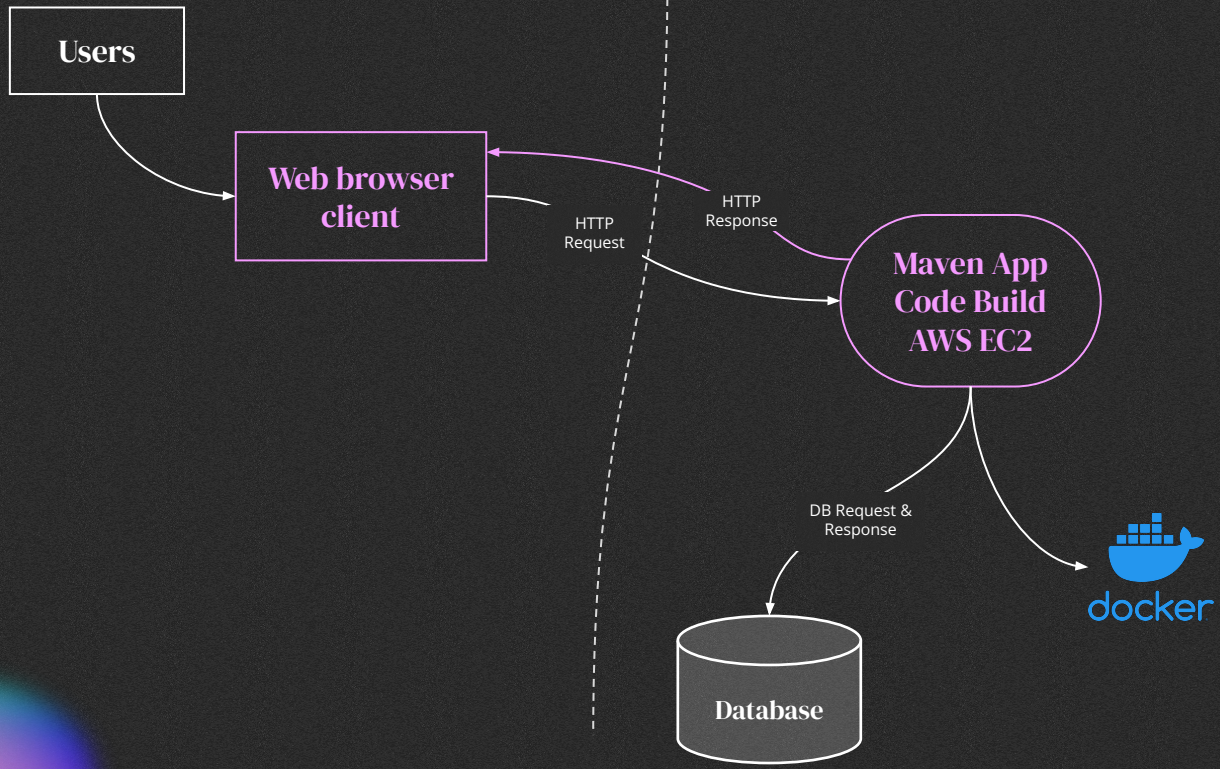


Web browser client, Maven App Code Build AWS ec2	Spoofing	Involves illegally accessing and then using another user's authentication information, such as username and password	Authentication (Zap) Authorization (Zap) Communication Security (Zap, Cloudwatch)
Maven App Code Build AWS ec2	Repudiation	Associated with users who deny performing an action without other parties having any way to prove otherwise—for example, a user performs an illegal operation in a system that lacks the ability to trace the prohibited operations.	Auditing and Logging (Cloudwatch)
HTTP Response	Information Disclosure	Involves the exposure of information to individuals who are not supposed to have access to it—for example, the ability of users to read a file that they were not granted access to, or the ability of an intruder to read data in transit between two computers	Auditing and Logging (Cloudwatch) Sensitive Data: (Zap) Communication Security (Zap, Cloudwatch) Cryptography (Zap) Authentication (Zap) Authorization (Zap) Exception Mgt (CloudWatch, Zap) Session Mgt CloudWatch, Zap) Input validation (Zap)
Maven App Code Build AWS ec2	Tampering	Involves the malicious modification of data. Examples include unauthorized changes made to persistent data, such as that held in a database, and the alteration of data as it flows between two computers over an open network, such as the Internet	Auditing and Logging (Cloudwatch) Sensitive Data: (Zap) Authentication (Zap) Authorization (Zap)

Mitigations coming from the threat model

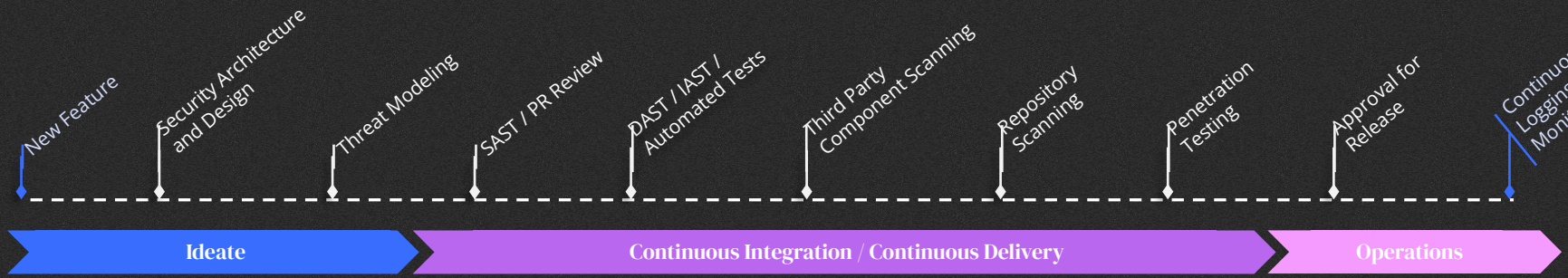


Mitigations coming from the threat model



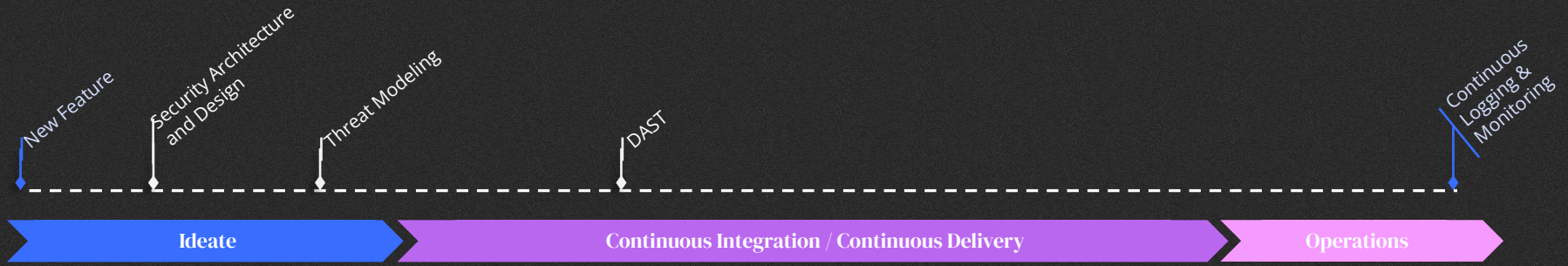
Securing the pipeline from end-to-end

Ideate to CI/CD to Operations



Securing the pipeline from end-to-end

Ideate to CI/CD to Operations



Identified Challenges & Risks

- Security flaws in the design and architecture of the application
- Slow, inefficient, and insecure deployment
- Security flaws in source code
- Lack of visibility that can make malicious behavior more difficult to detect

Processes to Mitigate Risks

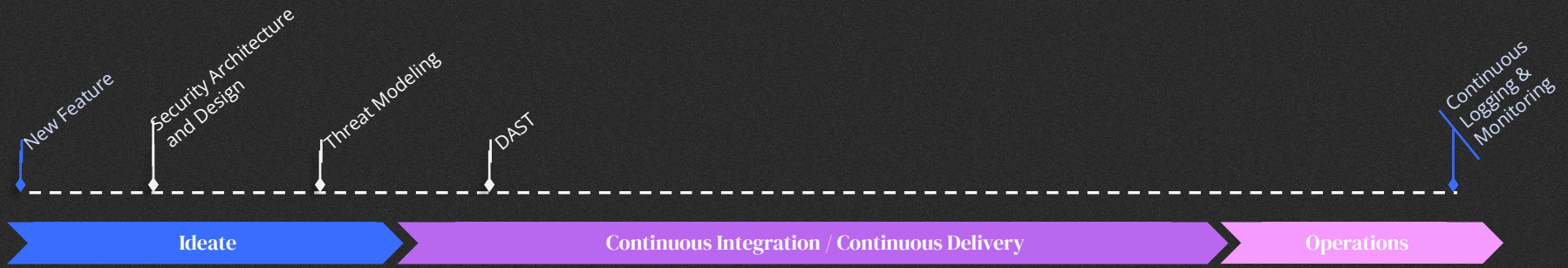
- Security Architecture & Design
- Threat Modeling
- DAST
- Continuous Logging & Monitoring

Wins 🎉

- ✔ Secure architecture and design process
- ✔ Automated Pipeline
- ✔ Automated Security Testing
- ✔ Continuous Logging & Monitoring

Securing the pipeline for this demo

Design to CI/CD to Operations



Demo

Let’s set up the development environment

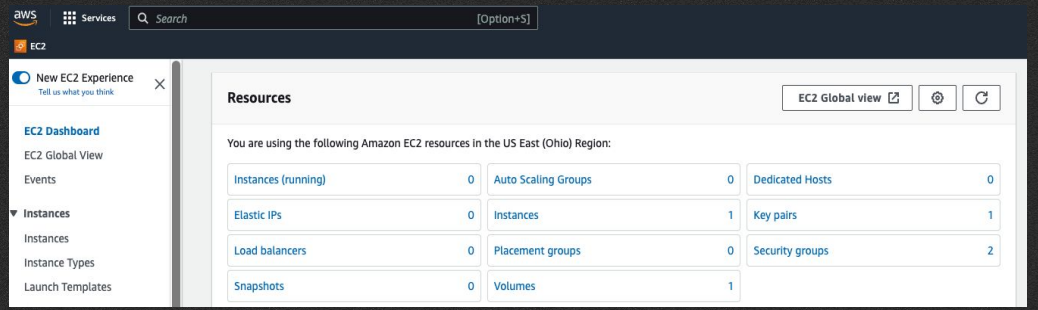
Let’s set up the development environment

- Set up AWS account
- Create EC2 Instance
- Default settings
- Default VPC
- No IAM Group
- Allow all TCP access
- Launch Instance
- Connect via SSH or EC2-User

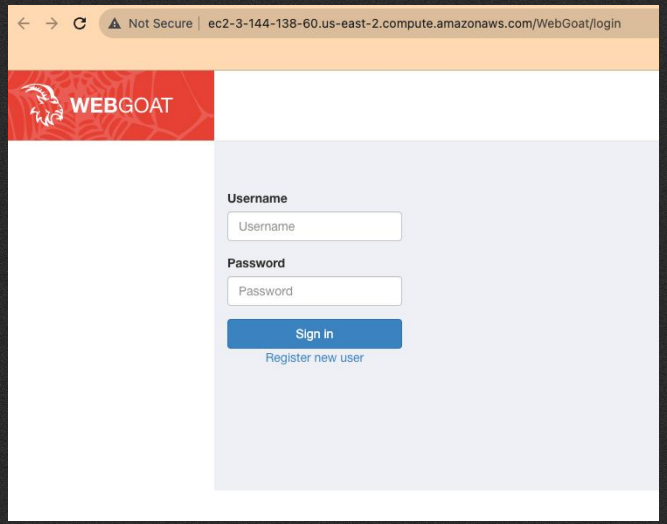
Sudo yum update -y

- sudo yum install -y docker
- sudo service docker start
- sudo usermod -a -G docker ec20user
- sudo docker pull webgoat/webgoat-8.0
- sudo docker run -p 80:8080
- (opt) docker run -p 80:8080
- Confirm access: curl -kv http://localhost
- Access WebGoat:
  http://ec2-publicip4.compute.amazonaws.com/WebGoat/Login

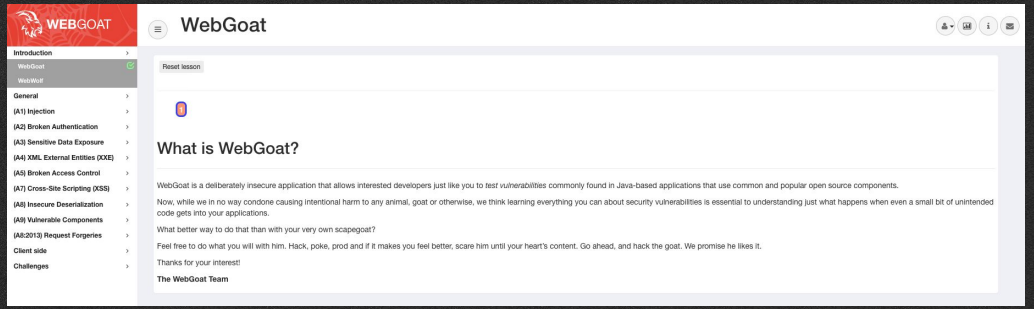
Let's set up the development environment

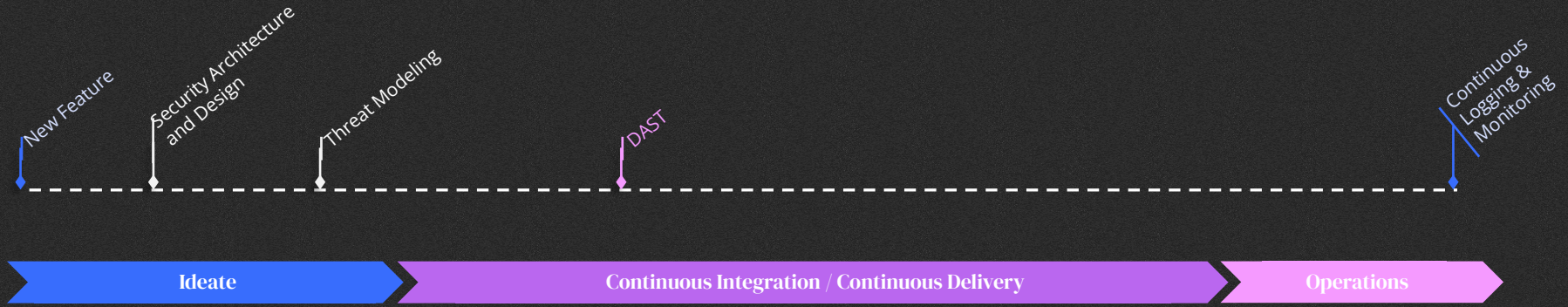


Let's set up the development environment



Let's set up the development environment ctn.

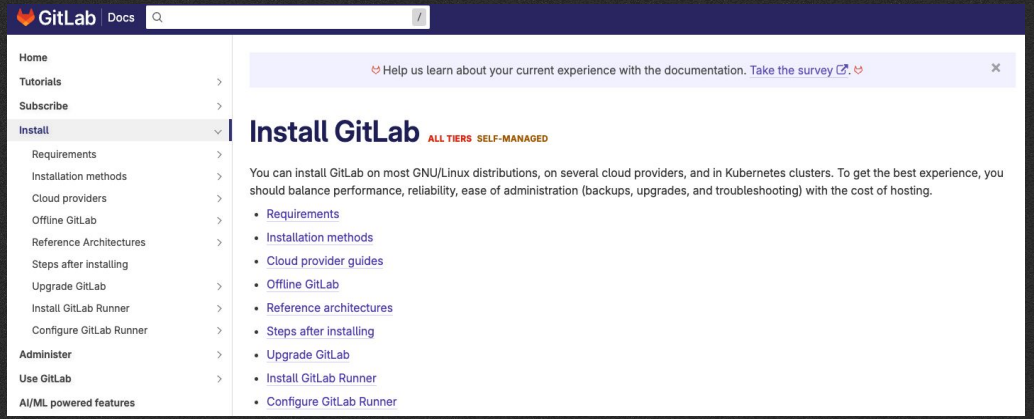




Continuous Integration / Continuous Delivery

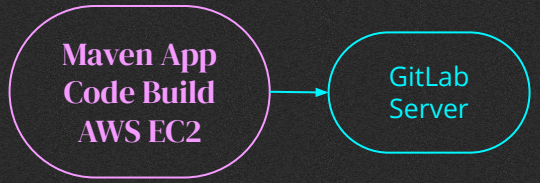
Demo

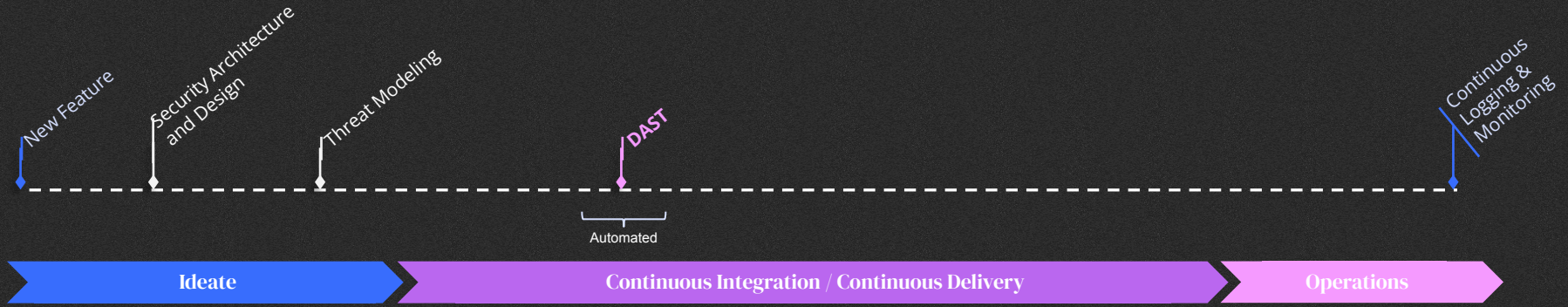
Let's set up Gitlab



Setting Up and Using a GitLab Server

- Install and configure the necessary dependencies
    - sudo yum install -y curl policycoreutils-python openssh-server openssh-clients perl
- Add the GitLab package repository and install the package
    - curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ee/script.rpm.sh | sudo bash
    - sudo EXTERNAL_URL="https://gitlab.example.com" yum install -y gitlab-ee



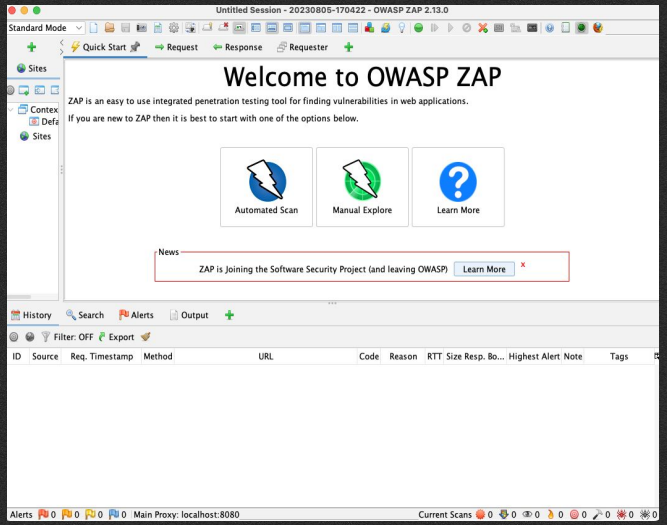


Dynamic Application Security Testing (DAST)

Demo

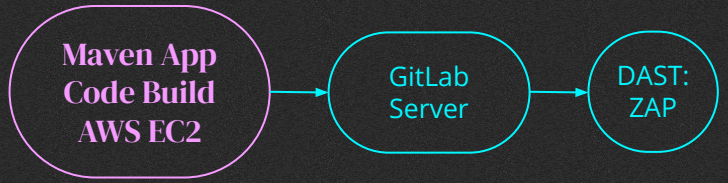
Let’s talk about automation

Let's set up DAST



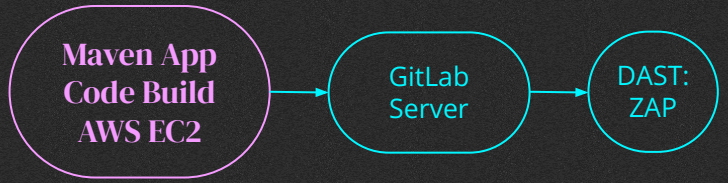
Authenticate to Web Application Scans with ZAP

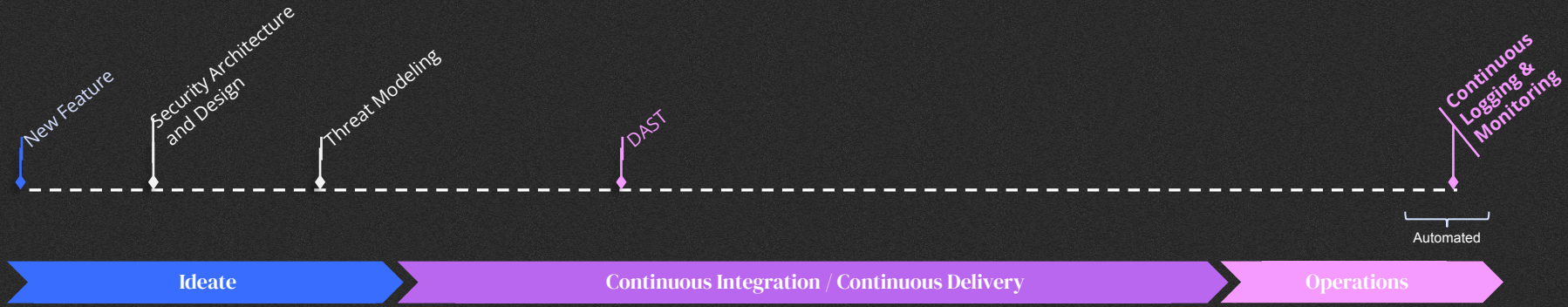
- Access ZAP
- Context > Session Properties > 1. Include in Text > Add url > http://ec2-publicip4.compute.amazonaws.com/WebGoat/Login > Add
- Record Zest script > title: authentication > type: authentication > server side script > start recording > capture logging into webgoat > stop recording > search for post requests > session property authentication > regex pattern identified in Logging in response message: admin > regex pattern identified in Logged out response message: Go to login page



Automate Web Application Scans with ZAP

- https://github.com/zaproxy/zap-api-python
    - pip install python-owasp-zap-v2.4
    - Create script to automate scans with ZAP API
- Next steps:
    - Trigger automated scan as part of the acceptance test script, run the CI/CD pipeline, and review results





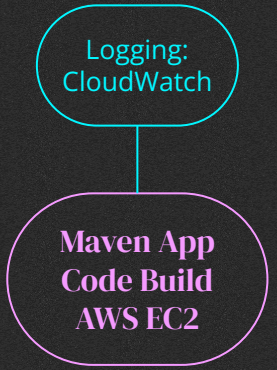
Continuous Logging and Monitoring

Demo

Let’s talk about configuration

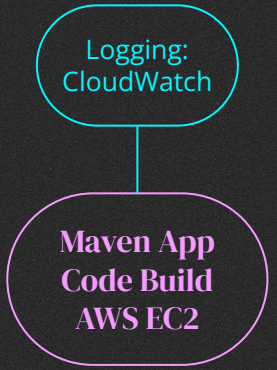
Implement CloudWatch Monitoring for an EC2 Instance

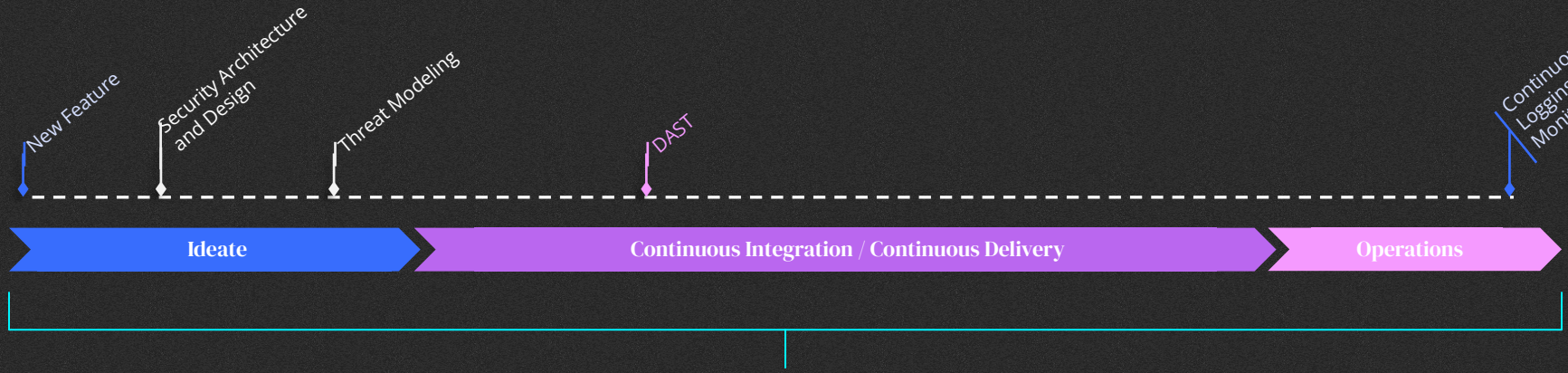
- Connect to instance
- wget -0 awslog-agent-setup.py https://s3.amazon.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py
- sudo python ./awslogs-agent-setup.py --region us-east-1
- Go to IAM > Users > cloud_user > permission policy allow_all > security credentials > create access key > select CLI > check then next > create access key > download .csv file > copy the access key ID > back to CLI > paste access key ID



Implement CloudWatch Monitoring for an EC2 Instance

- Path of log file to upload /var/log/syslog: /var/log/apache2/error.log
- Destination Log Group name: 1
- Choose log stream name: Enter Choice 1
- Choose log event timestamp form: Enter choice 1
- Choose initial position of upload: n
- Setting up agent as a daemon
- Go to CloudWatch > Log groups > /var/log/apache2.error.log > Log stream > Log events
- cat /var/log/apache2.error.log
- Command line: ‘usr/sbin/apache2’

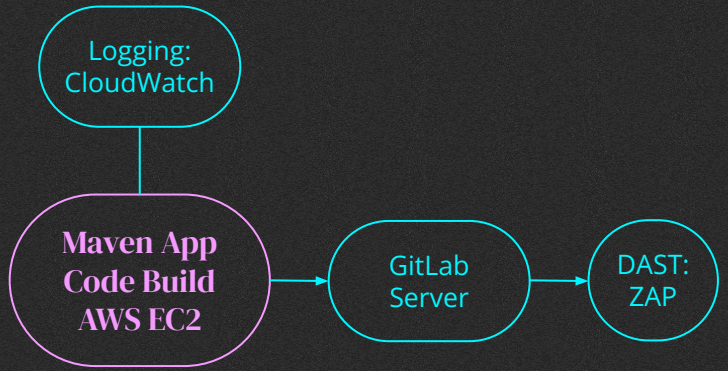




Bringing it all together for Continuous Integration / Continuous Delivery

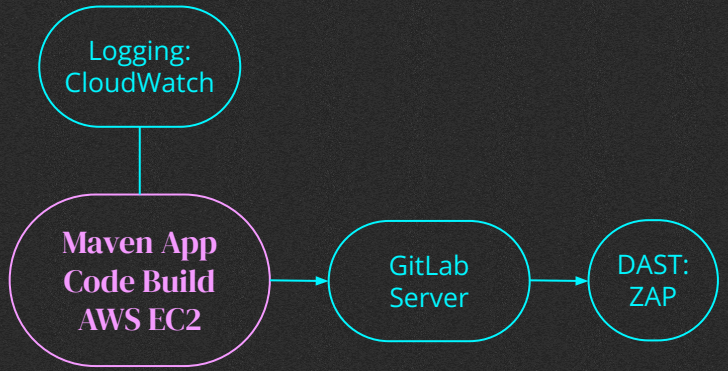
Connecting the GitLab Server to automated scans and continuous logging

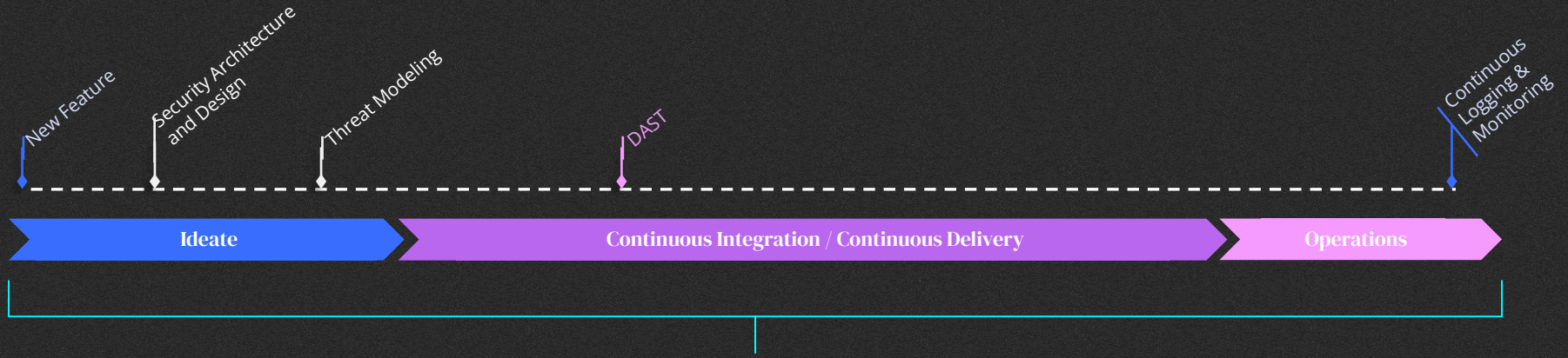
- With the GitLab Server EC2 instance up go ahead and configure GitLab Runner
- Install and configure GitLab Runner on the EC2 Instance. GitLab Runner will be responsible for executing the automated OWASP ZAP scans
- Create AWS IAM Roles
    - Create an IAM role that allows the EC2 instance running the webapp to access necessary AWS services, attack to EC2 instance
- Install and Configure ZAP on the WebGoat EC2 Install
- Create ZAP scanning scripts utilizing ZAP API



Connecting the GitLab Server to automated scans and continuous logging ctn.

- Configure GitLab CI/CD pipeline
  - The GitLab server with the WebGoat code repo create a `.gitlab-ci.yml` file to define the CI/CD pipeline
- Register WebGoat EC2 Instance as GitLab Runner
  - On the GitLab Server EC2 instance, register the WebGoat EC2 instance as a GitLab runner so that it can execute the CI/CD pipeline jobs
- Test the Setup
  - Push changes to your web application's repository to trigger GitLab CI/CD pipeline.
  - The pipeline should perform the automated OWASP ZAP scanning and send logs to AWS CloudWatch





Continuous Improvements and Lessons Learned

Demo

Implement continuous improvements, lessons learned, next steps, feedback, suggestions

Lessons learned:

- Don't bite off more than you can chew...
- Overestimate and buffer in more troubleshooting time than you anticipated!
- This pipeline is not exhaustive...always more to add. Threat model and focus in on different things to learn and incorporate:

Amazon Web Services:

- Lock down sec groups (inbound rules), IAM roles, restrictive VPC, EC2 Elastic Beanstalk, EKS
- Cloudwatch: more alerts, Cloudformation to trigger setup, billing alerts

WebGoat:

- Focus the automation to target OWASP Top 10 web vulns and third party components
- Try other vulnerable Web Applications like OWASP Juice Shop, AWSGoat, and Damn Vulnerable Web App on different cloud platforms like Azure and GCP

Implement continuous improvements, lessons learned, next steps, feedback, suggestions ctn.

And beyond:

- Policy as Code, Secrets Management, Third Party Component Scanning...
- AI??

Questions?

Shoutouts

It takes a village

My board of directors:

- Joseph Halpin, a passion for graphic design
- Daniel Stroie

Technical troubleshooting and mentorship:

- Robert Beltran

Resources

Resources and References

- https://www.sans.org/blog/from-devoops-to-devsecops/
- https://resources.snyk.io/changing-times-in-devsecops-playbook-typ/code-to-cloud-wp?_gl=1*ldx0gp*_ga*NDE0OTEwMzgzLjE2OTEwNzU4OTI.*_ga_X9SH3KP7B4*MTY5MTA4MTAzOC4zLjEuMTY5MTA4MTA0MS4wLjAuMA..&&
- https://www2.deloitte.com/content/dam/Deloitte/us/Documents/risk/us-ruisk-ssdl-PoV.pdf
- https://www.sans.org/cyber-security-courses/cloud-security-devsecops-automation/
- https://www.sans.org/blog/from-devoops-to-devsecops/
- https://about.gitlab.com/blog/2023/07/24/how-devsecops-drives-business-success/
- https://about.gitlab.com/blog/2023/04/20/gitlab-survey-highlights-wins-challenges-as-orgs-adopt-devsecops/
- https://about.gitlab.com/blog/2023/01/26/whats-next-for-devsecops/
- https://about.gitlab.com/blog/2023/02/02/its-time-to-put-the-sec-in-devsecops/
- https://about.gitlab.com/blog/2023/03/16/what-the-ml-ai/
- https://dodcio.defense.gov/Portals/0/Documents/Library/DevSecOps%20Playbook_DoD-CIO_20211019.pdf
- https://assets.contentstack.io/v3/assets/blt36c2e63521272fdc/bltf9dc689ee95dd752/5e320e658f7e217daef67590/continuous-security-exploring-the-devops-toolchain.pdf

Tools Used

- https://aws.amazon.com/
- https://aws.amazon.com/ec2/
- https://aws.amazon.com/cloudwatch/
- https://owasp.org/www-project-webgoat/
- https://owasp.org/www-project-threat-dragon/
- https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats
- https://www.zaproxy.org/
- https://www.docker.com/
- https://about.gitlab.com/install/#amazonlinux-2

Appendix

Threat Modeling with STRIDE

Spoofing

- Involves illegally accessing and then using another user's authentication information, such as username and password

Tampering

- Involves illegally accessing and theInvolves the malicious modification of data. Examples include unauthorized changes made to persistent data, such as that held in a database, and the alteration of data as it flows between two computers over an open network, such as the Internet

Repudiation

- Associated with users who deny performing an action without other parties having any way to prove otherwise—for example, a user performs an illegal operation in a system that lacks the ability to trace the prohibited operations. Non-Repudiation refers to the ability of a system to counter repudiation threats. For example, a user who purchases an item might have to sign for the item upon receipt. The vendor can then use the signed receipt as evidence that the user did receive the package

Threat Modeling with STRIDE ctn.

Information Disclosure

- Involves the exposure of information to individuals who are not supposed to have access to it—for example, the ability of users to read a file that they were not granted access to, or the ability of an intruder to read data in transit between two computers

DoS

- Denial of service (DoS) attacks deny service to valid users—for example, by making a Web server temporarily unavailable or unusable. You must protect against certain types of DoS threats simply to improve system availability and reliability

Elevation of Privilege

- An unprivileged user gains privileged access and thereby has sufficient access to compromise or destroy the entire system. Elevation of privilege threats include those situations in which an attacker has effectively penetrated all system defenses and become part of the trusted system itself, a dangerous situation indeed

Enterprise and open source tools used in DevSecOps

Code reviews

- Manual: Gitlab

Infrastructure as Code

- Manual: Terraform (tFlint), BridgeCrew

SAST

- Automated: Veracode, Microfocus / Coverity Scan Static Analysis, Bandit

Continuous Third-Party component Scanning

- Automated: Whitesource, Synk / Node Security Project, RetireJS

Continuous Vulnerability Scanning

- Automated: Nessus / OpenVas

DAST

- Automated: Veracode, Microfocus, Skipfish / ZAP, Archani

Enterprise and open source tools used in DevSecOps ctn.

Threat Modeling

- Manual: SecuriCAD, MS SDL Threat Modeling Modeling Tool / OWASP Threat Dragon, OWASP Purple Team

Secrets Management

- Automated: AWS Secrets Management / Hashicorp Vault

Repository Scanning for Credentials

- Manual: GitGuardian \ Gitleaks, Trufflehog

Continuous Logging and Monitoring

- Automated: Splunk, CloudTrail, CloudWatch / Graylog, Elastic Search

Thank you!