1. apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: allow-egress-for-service

namespace: Security

spec:

podSelector:

matchLabels:

nev: security

policyTypes:

- Egress

egress:

- to:

- ipBlock:

cidr: 10.10.10.10/32

ports:

- protocol: TCP

port: 32000

2. განვიხილოთ მიკროსერვისების არქიტექტურის დაცვა და უსაფრთხოების ზომების განხორციელება დეველოპმენტის სასიცოცხლო ციკლის სხვადასხვა ეტაპზე.

1. წყარო კოდის საცავი (GitHub):

*წვდომის კონტროლი:*

გამოიყენეთ GitHub-ის წვდომის კონტროლის ფუნქციები, რათა შეზღუდოთ რეპოზიტორიზე წვდომა როლების მიხედვით (წაკითხვა, ჩაწერა, ადმინისტრატორი).

დანერგეთ ორფაქტორიანი ავთენტიფიკაცია (2FA) მომხმარებლის ანგარიშებისთვის.

რეგულარულად გადახედეთ და შეამოწმეთ წვდომის ჟურნალები.

*კოდების სკანირება:*

კოდის სკანირების ხელსაწყოების ინტეგრირება (მაგ., GitHub CodeQL, SonarQube), რათა დადგინდეს უსაფრთხოების ხარვეზები დეველოპმენტის პროცესისას.

2. კონტეინერიზაცია (დოკერი):

*კონტეინერის უსაფრთხოების სკანირება:*

დანერგეთ კონტეინერის უსაფრთხოების სკანირების ხელსაწყოები (მაგ. Trivy, Anchore), რათა Docker იმიჯებში მოხდეს დაუცველობის იდენტიფიცირება.

გამოიყენეთ სანდო ბაზის იმიჯები და რეგულარულად განაახლეთ ისინი.

*მინიმალური პრივილეგიის პრინციპი:*

გამოიყენეთ მინიმალური პრივილეგიის პრინციპი კონტეინერის იმიჯებზე, მხოლოდ საჭირო ნებართვების მინიჭებით.

მოერიდეთ კონტეინერების გაშვებას, root იუზერით; გამოიყენეთ არა root მომხმარებლები.

*Docker Content Trust:*

ჩართეთ Docker Content Trust სურათების მთლიანობისა და ავთენტურობის უზრუნველსაყოფად.

3. კონტეინერის ორკესტრირება (Kubernetes):

*RBAC (როლზე დაფუძნებული წვდომის კონტროლი):*

დანერგეთ RBAC Kubernetes, რათა გააკონტროლოთ რესურსებზე წვდომა კლასტერში.

რეგულარულად გადახედეთ და განაახლეთ RBAC პოლისები.

*პოდის უსაფრთხოების პოლიტიკა:*

განსაზღვრეთ პოდის უსაფრთხოების წესები პოდებზე უსაფრთხოების შეზღუდვების დასადგენად.

შეზღუდეთ პრივილეგიების ესკალაცია და გამოიყენეთ AppArmor/SELinux პროფილები.

*ქსელის პოლიტიკა:*

გამოიყენეთ Kubernetes secrets მნიშვნელოვანი ინფორმაციების დასაცავად.

გამოიყენეთ ინსტრუმენტები, როგორიცაა HashiCorp Vault sekretebis ცენტრალიზებული მართვისთვის.

4. CI/CD:

*უსაფრთხოების ავტომატური ტესტირება:*

danergeT უსაფრთხოების ტესტირების ხელსაწყოები (მაგ. OWASP ZAP, Bandit) CI/CD ფაიფლაინში.

ჩაშალეთ დაბილდვის ან გაშვების პროცესი უსაფრთხოების დაუცველობების აღმოჩენისას.

*უცვლელი ინფრასტრუქტურა:*

დააპრიორიტეტეთ უცვლელი ინფრასტრუქტურის გამოყენება თავდასხმის შესამცირებლად და გამეორებადობის გასაუმჯობესებლად.

5. ღრუბლოვანი სერვისები (AWS):

*IAM:*

განახორციელეთ მკაცრი IAM პოლიტიკა AWS რესურსებზე წვდომის გასაკონტროლებლად.

გამოიყენეთ AWS Identity Federation ერთჯერადი შესვლისთვის (SSO).

*VPC და უსაფრთხოების ჯგუფები:*

დააკონფიგურირეთ Virtual Private Cloud (VPC) უსაფრთხოების შესაბამისი ჯგუფებით და ქსელის ACL-ებით.

გამოიყენეთ AWS WAF ვებ აპლიკაციის firewall-ის დაცვისთვის.

*მონიტორინგი და ლოგირება:*

დანერგეთ AWS CloudTrail აუდიტისთვის და AWS Config რესურსების ინვენტარისთვის.

გამოიყენეთ Amazon CloudWatch მონიტორინგისა და შეტყობინებებისთვის.

*დაშიფვრა:*

განახორციელეთ მონაცემთა დაშიფვრა ტრანზიტში AWS სერტიფიკატების მენეჯერისა და AWS გასაღების მართვის სერვისის (KMS) გამოყენებით.

ჩართეთ დაშიფვრა EBS ვოლიუმებზე და გამოიყენეთ AWS Secrets Manager სთორიჯის უსაფრთხოებისთვის.