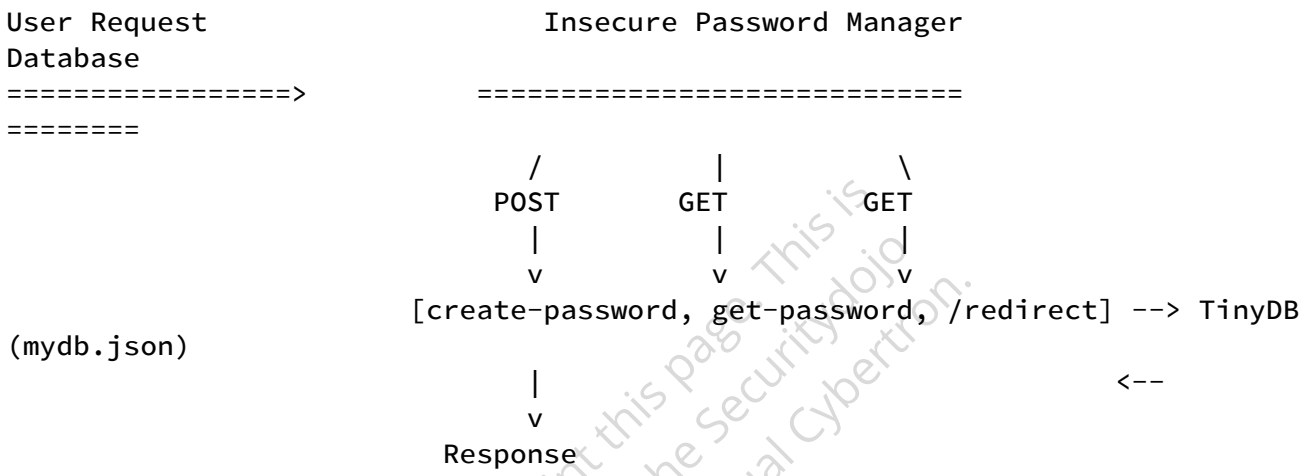
Theory: Working of Sample Application

Application Flow

The code is a Insecure Python Application using flask that creates an insecure password manager with security vulnerabilities such as SSRF and command execution.



Libraries

The following libraries are imported at the beginning of the code:

- Flask: a Python web framework for creating web applications
- request: a Flask library for handling HTTP requests
- jsonify: a Flask library for returning JSON responses
- b64encode and b64decode: Python libraries for encoding and decoding Base64 strings
- jwt: a Python library for encoding and decoding JSON Web Tokens (JWT)
- os: a Python library for interacting with the operating system
- TinyDB: a lightweight database management system for storing JSON documents
- AES: a Python library for encrypting and decrypting data using the Advanced Encryption Standard (AES)
- Padding: a Python library for padding and unpadding data
- logging: a Python library for logging messages
- subprocess: a Python library for running shell commands
- json: a Python library for working with JSON data

- urllib.request: a Python library for making HTTP requests

Endpoints

The Insecure Password Manager application has four endpoints:

- GET /: returns the homepage of the application
- POST /create-password: saves an email and password to the password manager
- GET /get-password/: retrieves a password for a given email from the password manager
- GET /redirect: retrieves the contents of a given URL using the urllib.request library, indicating an SSRF vulnerability
- GET /date: runs a shell command using the subprocess library, indicating a Command Injection vulnerability

---

Refer to source-code

Kubernetes Yaml

What is `namespace.yaml`?

```
apiVersion: v1
kind: Namespace
metadata:
  name: vulapp-namespace-cl
```

- `namespace.yaml` file is used to create a namespace within the kubernetes cluster.

What is `clusterrole.yaml`?

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: vulapp-clusterrole
  namespace: vulapp-namespace-cl
rules:
 - apiGroups: [""]
   resources: ["namespaces"]
   verbs: ["list", "get"]
```

- A ClusterRole resource is created in the rbac.authorization.k8s.io/v1 API version via the clusterrole.yaml file. At the cluster level, this ClusterRole resource defines a set of permissions that may be given to a user, group, or ServiceAccount.
- Users, groups, or ServiceAccounts with this role can list and access information about namespaces in the cluster thanks to the ClusterRole described in this YAML file. Utilising a RoleBinding or ClusterRoleBinding resource, it may be applied to a user or ServiceAccount. To give the vulapp-clusterrole to a particular user or ServiceAccount in the vulapp-namespace-cl namespace, for instance, a RoleBinding can be made.

What is `clusterrolebinding.yaml`?

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: vulapp-clusterrolebinding
  namespace: vulapp-namespace-cl
subjects:
- kind: ServiceAccount
  name: vulapp-sa-cl
  namespace: vulapp-namespace-cl
roleRef:
  kind: ClusterRole
  name: vulapp-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

- A ClusterRoleBinding resource is created in the rbac.authorization.k8s.io/v1 API version via the clusterrolebinding.yaml file. A ClusterRole is bound to a user, group, or ServiceAccount in a namespace using the ClusterRoleBinding resource.
- The vulapp-clusterrole ClusterRole is bound to the vulapp-sa-cl ServiceAccount in the vulapp-namespace-cl namespace via the ClusterRoleBinding. This indicates that the permissions specified in the vulapp-clusterrole ClusterRole are now granted to the ServiceAccount.
- It enables users or applications to carry out cluster-wide operations that call for the authorizations specified by the vulapp-clusterrole ClusterRole.

What is `deployment.yaml`?

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vulapp-deployment-cl
  namespace: vulapp-namespace-cl
spec:
  replicas: 1
  selector:
    matchLabels:
      app: vulapp-cl
  template:
    metadata:
      labels:
        app: vulapp-cl
    spec:
      serviceAccountName: vulapp-sa-cl
      containers:
      - name: vulapp-container-cl
        image: justmorpheu5/insecure-python-app
        ports:
        - containerPort: 8000
```


- A Deployment resource is created in the apps/v1 API version via the deployment.yaml file, the deployment of containerized apps is controlled by this Deployment resource.
- The justmorpheu5/insecure-python-app Docker image is used to deploy a containerized application according to the Deployment outlined in this YAML file. The vulapp-sa-cl ServiceAccount in the vulapp-namespace-cl namespace is used to deploy the application & within the cluster, the application is reachable on port 8000.

What is `service.yaml`?

```
apiVersion: v1
kind: Service
metadata:
  name: vulapp-service-cl
  namespace: vulapp-namespace-cl
spec:
  selector:
    app: vulapp-cl
  ports:
  - name: http
    port: 8000
    targetPort: 8000
  type: NodePort
```

- The `service.yaml` file establishes a Service resource in the v1 API version. This Service resource is used to make a Kubernetes deployment available from both within and outside the cluster by exposing it as a network service.
- The vulapp-cl Deployment is exposed as a network service by the Service described in this YAML file. Based on their labels, it directs traffic to the Pods that the Deployment manages. Traffic is sent to the Pod container operating on port 8000 by the Service, which listens on port 8000. Using a dynamically issued port in the range of 30000-32767 on each cluster node's IP address, it is reachable from outside the cluster.

However in our case kind is running worker node within the container hence the application won't be accessible directly via internet, we need to portforward to access the application from outside the cluster.

What is `serviceaccount.yaml`?

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: vulapp-sa-cl
  namespace: vulapp-namespace-cl
```


- The serviceaccount.yaml file is a manifest that creates a ServiceAccount resource in the v1 API version. This ServiceAccount resource is used to provide an identity for processes that run in a Pod.