Software Development Report -To Exposys Data Labs

Abstract

User sends a text or file from his phone to his friend. The risk of the data being accessed by a third party is high when the data enters the cloud platform. Hence a new method to secure the messages/file is to be done either by building a new algorithm or by modifying the existing one. We achieve this using a Plugin System designed using Python. We also provide a Testing Environment to test the results

Table OF Content

1.Designing a Plugin

2.Testing to ensure security guarantees

Introduction

We're going to use a Docker container through [Docker SDK for Python](https://docker-py.readthedocs.io/en/stable/) for creating The software instead of Conventional SDK methods. Requirements Are

Plugins are Python files

Prevent access to application's code (RCE)

Prevent access to network resources (SSRF)

Prevent side-effects of plugin executions (data)

Prevent exhausting system resources (DoS)

Prevent to run forever (DoS)

Prevent side-effects of the solution

Existing method

SDK which is abbreviated to System Development Kit is a set of programmes used to develop software. It is a software development tool used to create applications that perform a specific task. These may include the creation of applications for a computer system, operating system, software package, software framework, hardware platform, video game console etc. SDKs come with a sample that helps the developer to learn how to build basic programmes which eventually helps them to build more complex ones. These are known as Application Programming Interface (APIs) that are a set of instructions used during programming to access the database available online. An SDK contains  API and sometimes even may have several APIs and so we can further say that an API is a part of SDK. Most companies want that the applications created are for their own platform and so provide the SDKs free of cost

Proposed Method And Implementation

We're going to use a docker container through Docker SDK for Python

For the Given Advantages Over Conventional SDK, Code is :

1)First Advantage

* Prevent access to application's code
* Prevent side-effects of plugin executions

1. import docker
2. client = docker.from\_env()
3. input\_including\_plugin\_code = ...
4. container = client.containers.run(
5. "my-image",
6. input\_including\_plugin\_code,
7. detach=True,
8. )

2)Second Advantage

* Prevent access to network resources

1. ...
2. container = client.containers.run(
3. "my-image",
4. input\_including\_plugin\_code,
5. detach=True,
6. network\_disabled=True,
7. )

3)Third Advantage

* Prevent exhausting system resources

1. ...
2. container = client.containers.run(
3. "my-image",
4. input\_including\_plugin\_code,
5. detach=True,
6. network\_disabled=True,
7. nano\_cpus=25 \* 10\*\*7,
8. mem\_limit="128m",
9. )

4)Fourth Advantage

* Prevent side-effects of the solution

1. import requests
2. ...
3. MAX\_SECONDS\_TO\_RUN = 10
4. try:
5. container.wait(timeout=MAX\_SECONDS\_TO\_RUN)
6. except requests.exceptions.ConnectionError:
7. container.stop(timeout=0)
8. # process container output
9. ...
10. container.remove()

Testing and Conclusion

Using Django + DRF, Let's create an app where a user have orders

1)First, Defining an Order model

1. from django.contrib.auth import get\_user\_model
2. from django.db import models
3. class Order(models.Model):
4. price = models.IntegerField()
5. user = models.ForeignKey(
6. get\_user\_model(),
7. on\_delete=models.CASCADE
8. )
9. secret\_code = models.CharField(max\_length=20)

2)Second, Django settings

1. from django.contrib.auth import get\_user\_model
2. from django.db import models
3. class Order(models.Model):
4. price = models.IntegerField()
5. user = models.ForeignKey(
6. get\_user\_model(),
7. on\_delete=models.CASCADE
8. )
9. secret\_code = models.CharField(max\_length=20)

3) Defining the endpoint

1. from rest\_framework import serializers, viewsets
2. from .models import Order
3. class OrderSerializer(serializers.ModelSerializer):
4. class Meta:
5. model = Order
6. fields = ["id", "price"]
7. class OrderViewSet(viewsets.ReadOnlyModelViewSet):
8. def get\_queryset(self):
9. return Order.objects.filter(user=self.request.user)
10. serializer\_class = OrderSerializer

CONCLUSION:

* Only authenticated users can access it
* Order's secret\_code is never exposed
* Orders can be only listed and retrieved
* Users can only access their own orders