66Kubernetes 系列（六十一）Pipy：保护 Kubernetes 应用程序免受 SQL 注入和 XSS 攻击

注入攻击在 2021 年下滑至第三位，多年来一直位居 OWASP 前 10 名，SQL 注入 (SQLi) 是一种用于攻击网站和 Web 应用程序的常见注入技术。没有将用户输入与数据库命令完全分开的应用程序存在将恶意输入作为 SQL 命令执行的风险。成功的注入攻击可能导致未经授权访问敏感数据，例如密码、信用卡详细信息和个人用户信息。最近许多引人注目的数据泄露事件都是由 SQL 注入攻击引起的，导致声誉受损和监管罚款。

您可能在其他解决方案中看到的常见和正常的解决方法是使用正则表达式列表来过滤掉流量，这在某些情况下有效，但在某些复杂输入或转义输入时效果不佳。我们不是要抨击 `Regular Expressions`，它们很好并且有自己的用例，但它们不适合此类用例。

这篇博文将演示如何使用开源可编程代理 Pipy，通过为您的应用程序添加额外的防护层来加强安全性，否则可能容易受到此类攻击。

我们将使用 Pipy NMI 开发一个模块来集成成熟稳定的开放-source library libinject 在传入流量到达应用程序之前针对 SQLi 和 XSS 攻击扫描传入流量。

对于演示应用程序，我们将使用一个开源的经典 LAMP 堆栈 Vulnerable Mama Shop (VMS)，它是有意开发为具有 SQLi 缺陷的。我们将通过首先破解基本应用程序来演示 SQLi 攻击来一起玩得开心，然后我们将通过添加 Pipy 作为 sidecar 来阻止某些 SQLi 攻击来加强应用程序安全性。

先决条件

这篇博文假设您有权访问：

- 运行 Kubernetes 集群
- kubectl

> 演示源代码在 GitHub 上可用，可以从 pipy-sqli-demo（https://github.com/flomesh-io/pipy-demos/tree/main/pipy-sqli-demo）存储库下载。

部署 Kubernetes 集群和 Vulnerable Mama Shop App

要在本地运行演示，我们建议使用 k3d 轻量级包装器在 docker 中运行 k3s（Rancher Lab 的最小 Kubernetes 发行版）。

```
$ k3d cluster create my-cluster -p 8080:30060@server:0
```

在上面的命令中，我们创建了一个带有单个服务器的集群，并将 k3d 容器的端口 30060 映射到本地 8080 端口，这个端口的用法将在本教程的后续步骤中变得清晰。

部署易受攻击的 Mama Shop 应用程序

我们将部署一个简单的在线商店应用程序，该应用程序随附安装

- Apache Webserver
- MariaDB
- 在 Apache 上运行并连接到 MariaDB 数据库的 PHP 应用程序

1. 使用您选择的文本编辑器并创建一个名为 1-app.yaml 的 YAML 文件，其中包含以下内容：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vms
spec:
  selector:
    matchLabels:
      app: vms
  template:
    metadata:
      labels:
        app: vms
    spec:
      containers:
        - name: vms
          image: naqvis/mamashop
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: vms
spec:
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30060
  selector:
    app: vms
  type: NodePort
```

2. 部署应用

```
$ kubectl apply -f 1-app.yaml
```

3. 确认 pod 已启动并正在运行，如 STATUS 列中的值 Running 所示。它们可能需要 30-40 秒才能完全部署，因此在继续下一步之前再次运行命令以确认所有 pod 都在运行是很有用的。.

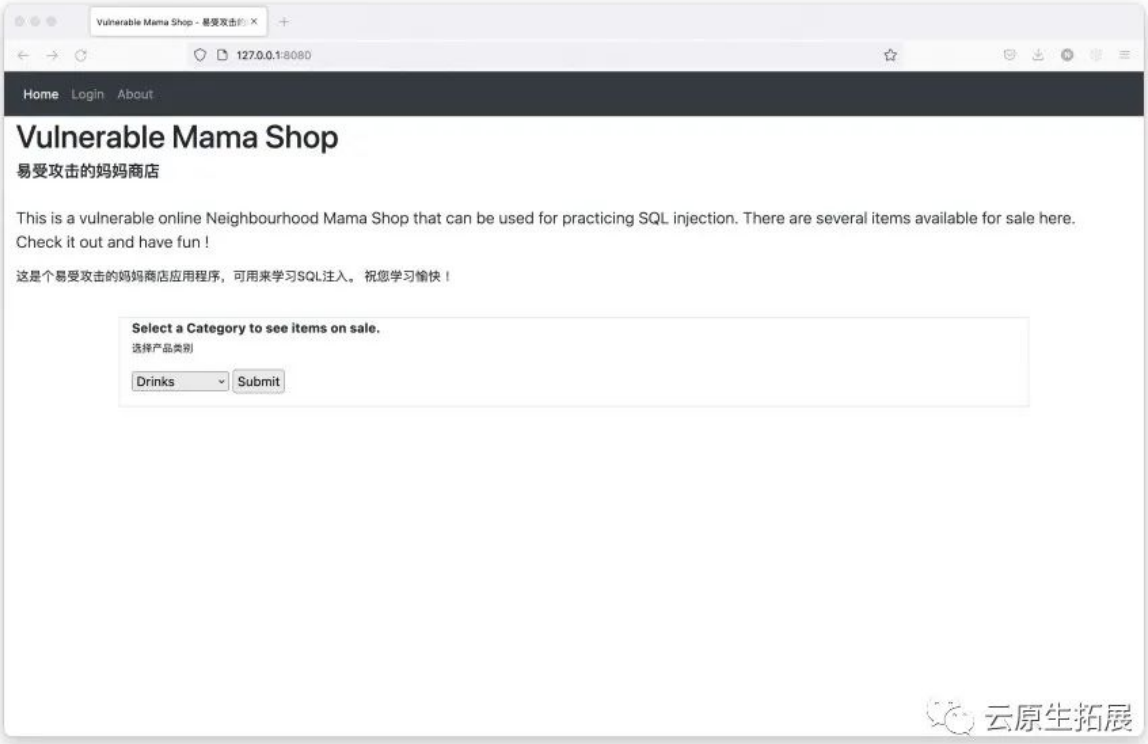
```
$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
vms-6658dd4478-hn246   1/1     Running   0          2m12s
```

4. 查询 Servcice

```
$ kubectl get svc vms
NAME    TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
vms     NodePort   10.43.244.253   <none>        80:30060/TCP   5m43s
```

还记得一开始我们为K3d容器做了端口映射： `8080=>30060` 吗？敏锐的用户可能还会发现，我们为我们的 Vulnerable Mama Shop Web 应用程序创建了一个 NodePort 服务，节点端口是 `30060` 。

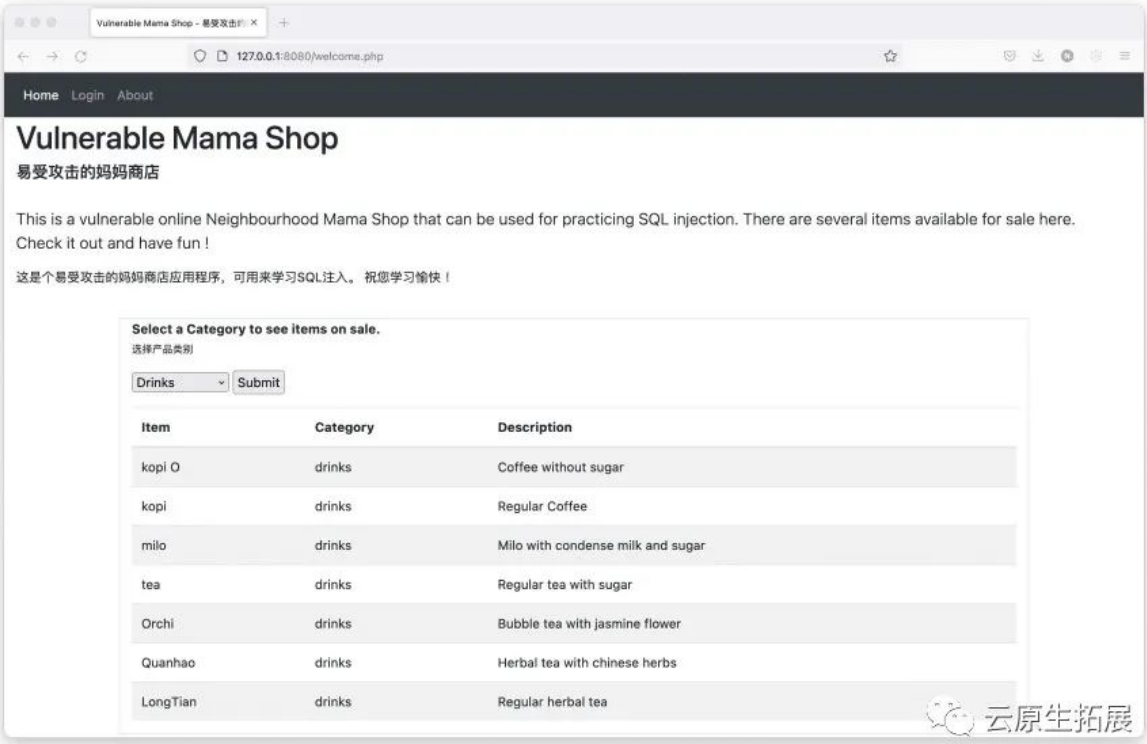
通过访问 URL **http://localhost:8080** 在浏览器中打开应用程序



攻破 应用程序

Mama Shop 非常基础，只有 3 页。主页（如上所示）是用户可以通过下拉框、客户登录页面和关于页面查询待售商品的地方。试用这 3 个页面，看看每个页面的作用以及每个页面的正常工作方式。

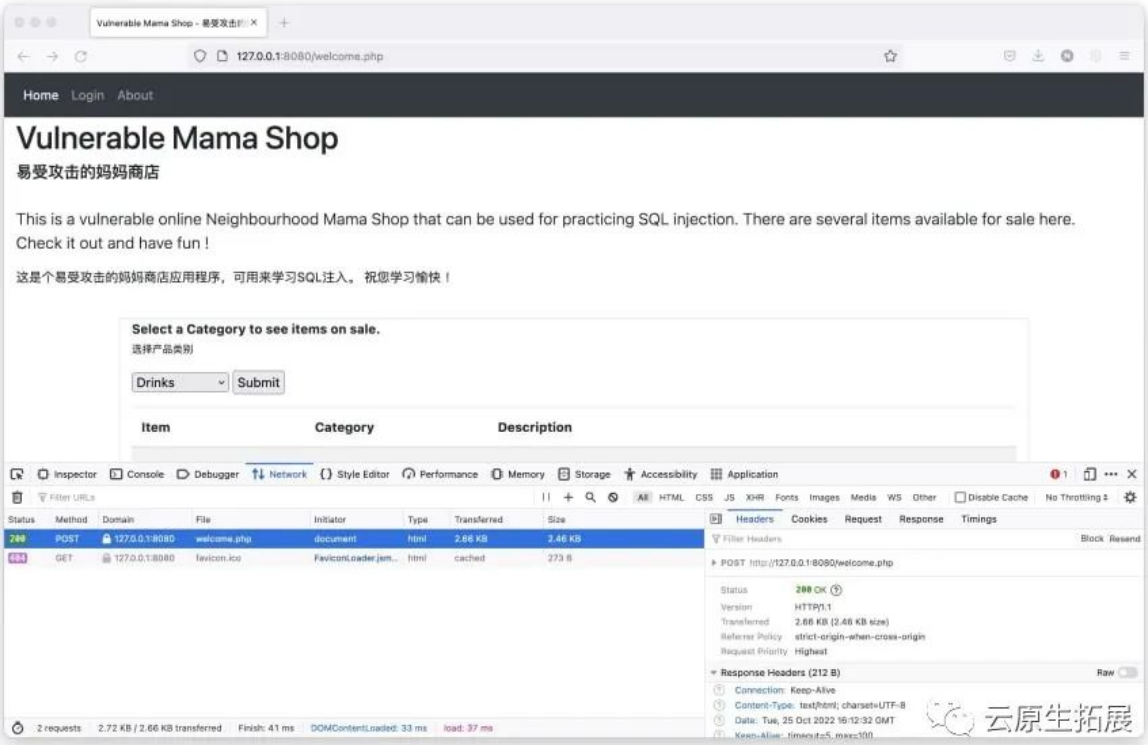
尝试提交一个类别并查看项目是如何列出的。下面显示了 Drinks 类别中的项目列表。



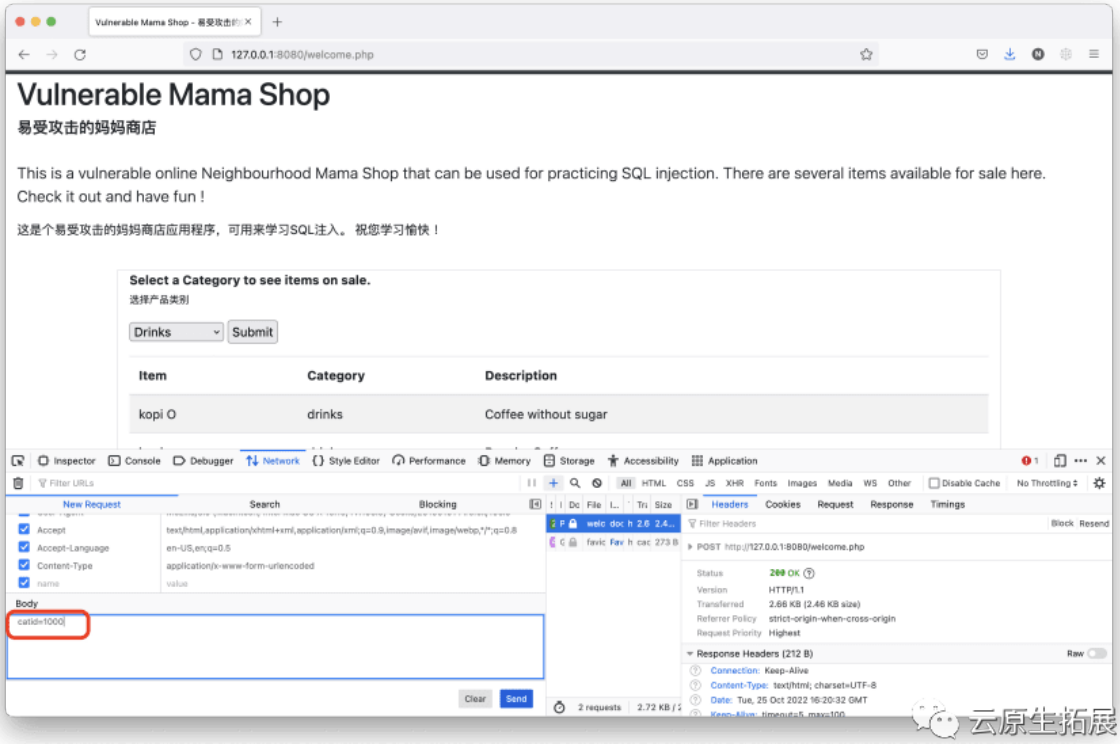
您可能已经注意到，选择类别并点击提交不会更改 URL，因此应用程序正在执行 `POST` 请求，我们将需要一些工具来拦截或查看网页生成的流量。

通常，您将使用 Blurp 套件或 OWASP ZAP 等代理工具来拦截和修改对应用程序的请求。但为简单起见，我们将使用 Firefox 浏览器开发人员控制台来查看和修改请求。

返回您的浏览器（本演示中为 Firefox）并通过 `Tools | Browser Tools | Web Developer Tools` 菜单选项或通过 Mac 上的快捷键 `Option+Cmd+I` 或操作系统上的类似功能打开 Web 开发人员工具。移至 `Network` 选项卡，然后点击网页上的提交按钮以查看网络流量。



我们可以看到所有请求以及所有详细信息，网页使网络服务器显示该页面。点击右下角显示的请求详细信息窗口右上角的 `Resend` 按钮。



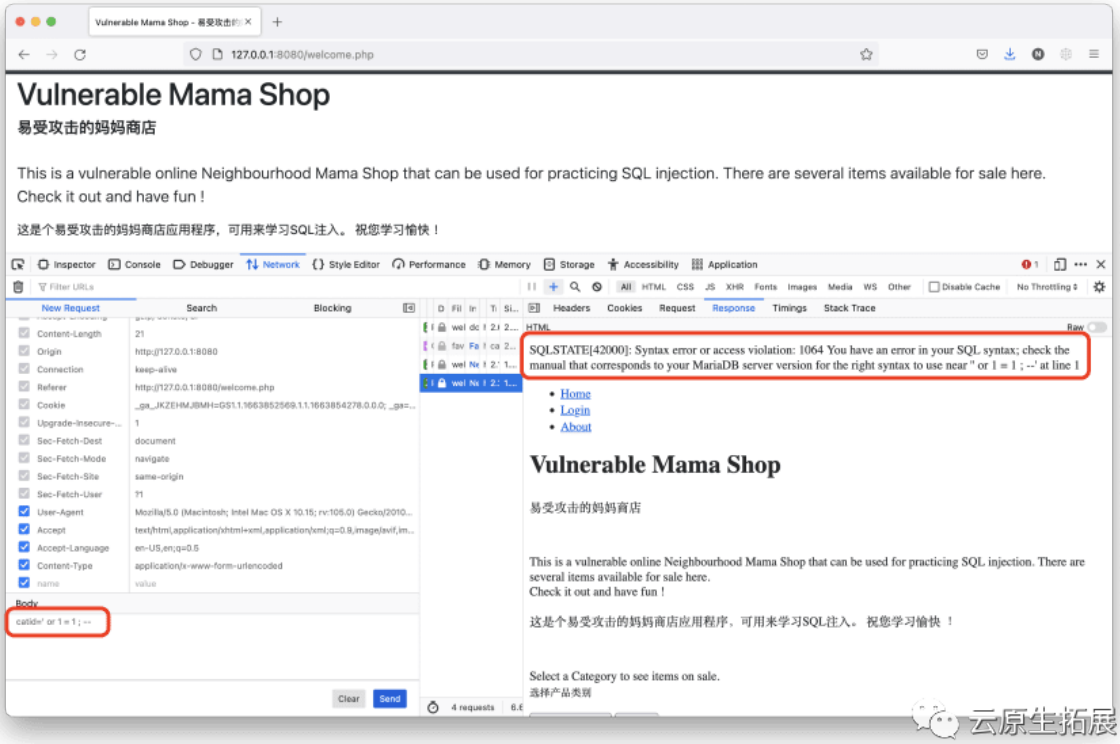
网络操作窗口显示，要浏览类别中的项目，会将带有单个参数 `catid` 的 URL 编码 HTTP POST 发送到 `welcome.php` 。要测试 SQL 注入，通常会修改用户输入并发送一个单引号，如 `'` 。

修改表单参数

我们将通过一个简单的实验来确定输入是否被正确转义：将 `catid` 更改为 SQL 查询字符串，以查看它是否会导致可能在网页上显示的语法错误。

```
catid=' or 1 = 1 ; --
```

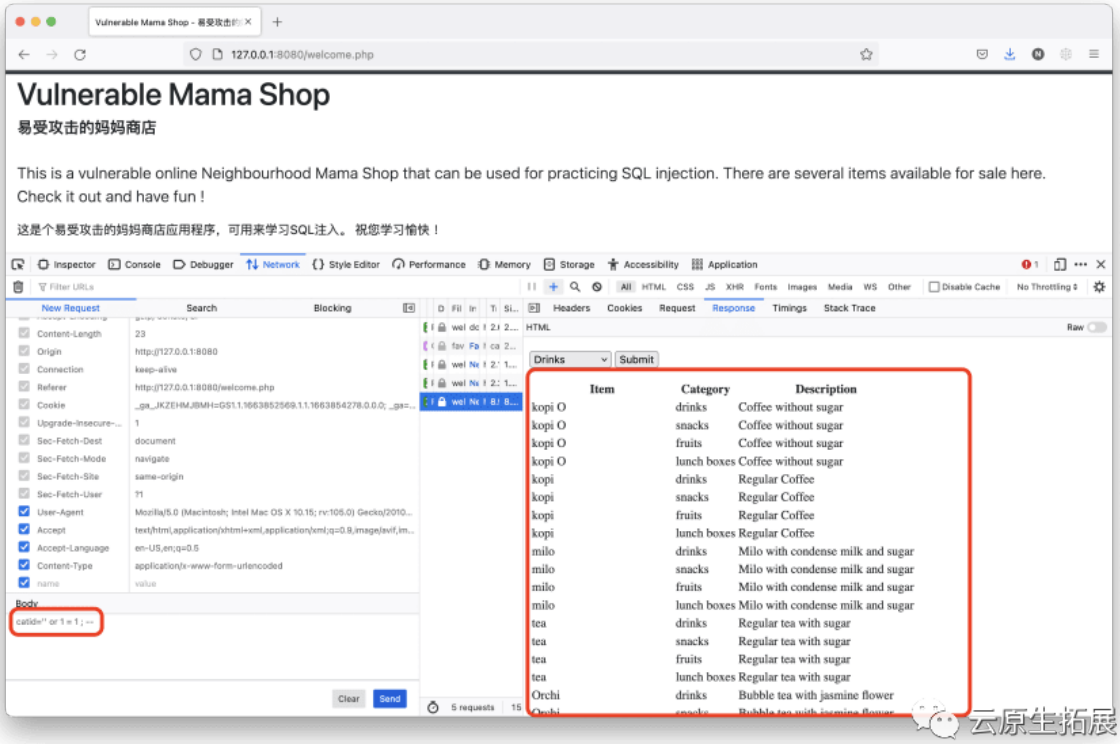
进行上述更改并点击网络操作窗口右下角的 `send` 按钮。



哇，我们搞砸了，用于显示类别项目的 `welcome.php` 函数存在 SQL 注入漏洞。错误消息还告诉我们数据库是 MariaDB。错误告诉我们 `or 1 = 1` 附近存在语法问题。发挥我们的想象力，我们可以假设数据库查询是这样的

```
SELECT item_name, item_category, item_description from items_table where item_category = ' or 1 = 1 ; -- ;
```

如果我们将 `catid` 结尾更改为 `" or 1 = 1 -- ;` ，那应该可以解决引用问题。它应该从数据库中选择所有行，这在 hack 中很有用。



假定的 SQL 查询的结构可能是正确的，尽管它不一定是开发人员使用的确切查询。这足以应对倾销客户信息等破坏性攻击。

提取用户数据

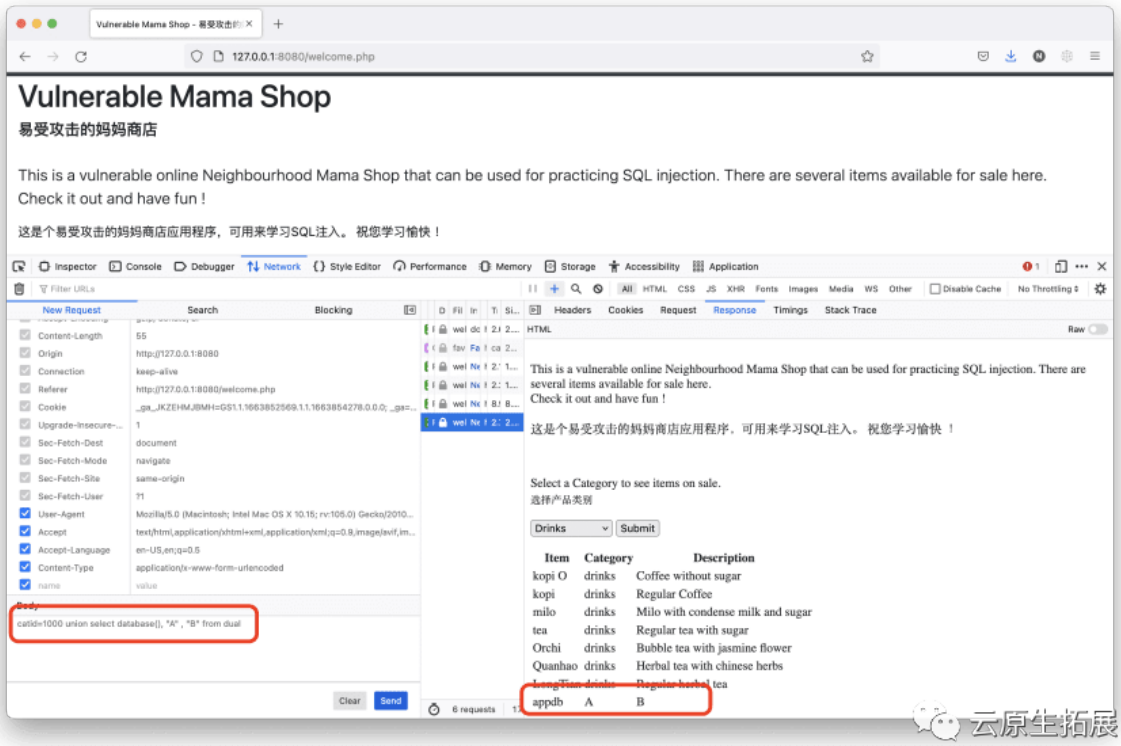
客户登录页面告诉我们该应用程序包含客户数据，这可能存储在某种客户或用户表中。

根据目前收集到的信息，我们可以使用 MariaDB INFORMATION_SCHEMA 和 SQL Union 运算符来检索数据库和表的详细信息。

将 `catid` 参数更改为以下以检索数据库名称，该名称将进一步用于检索表详细信息。

```
catid=1000 union select database(), "A" , "B" from dual
```

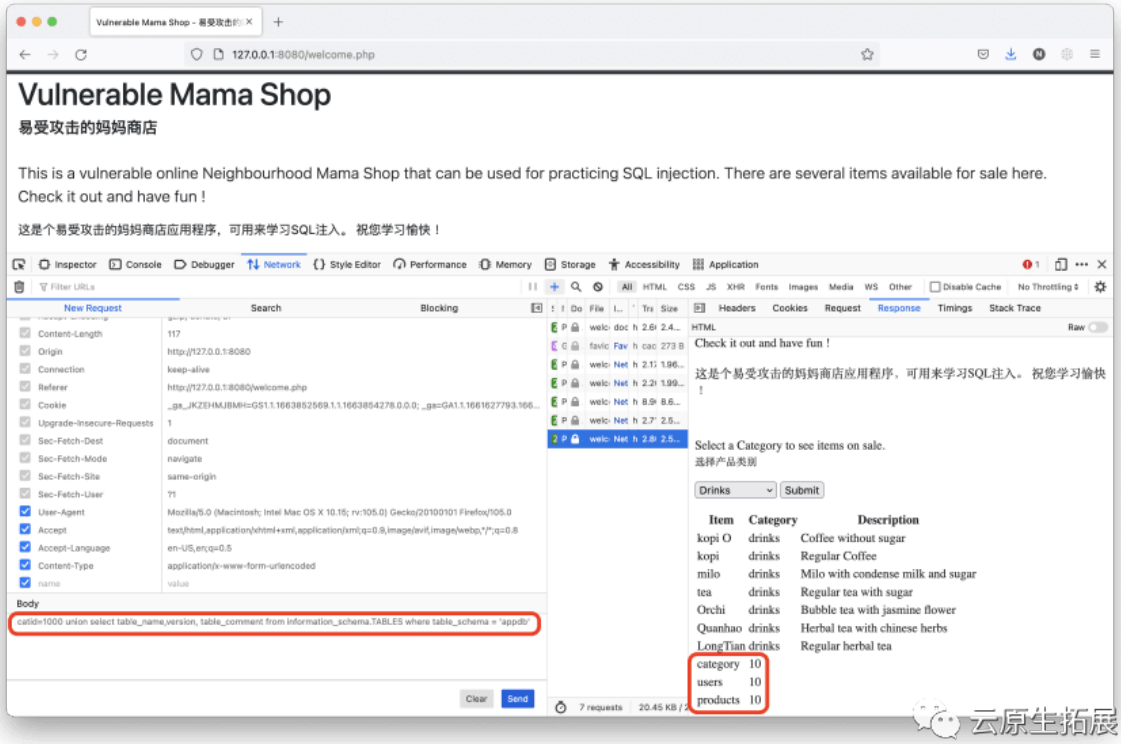
下面显示了来自 Vulnerable Mama Shop 的响应，其中包含数据库名称 `appdb` 。



现在我们有了数据库名，我们可以获取数据库中的表

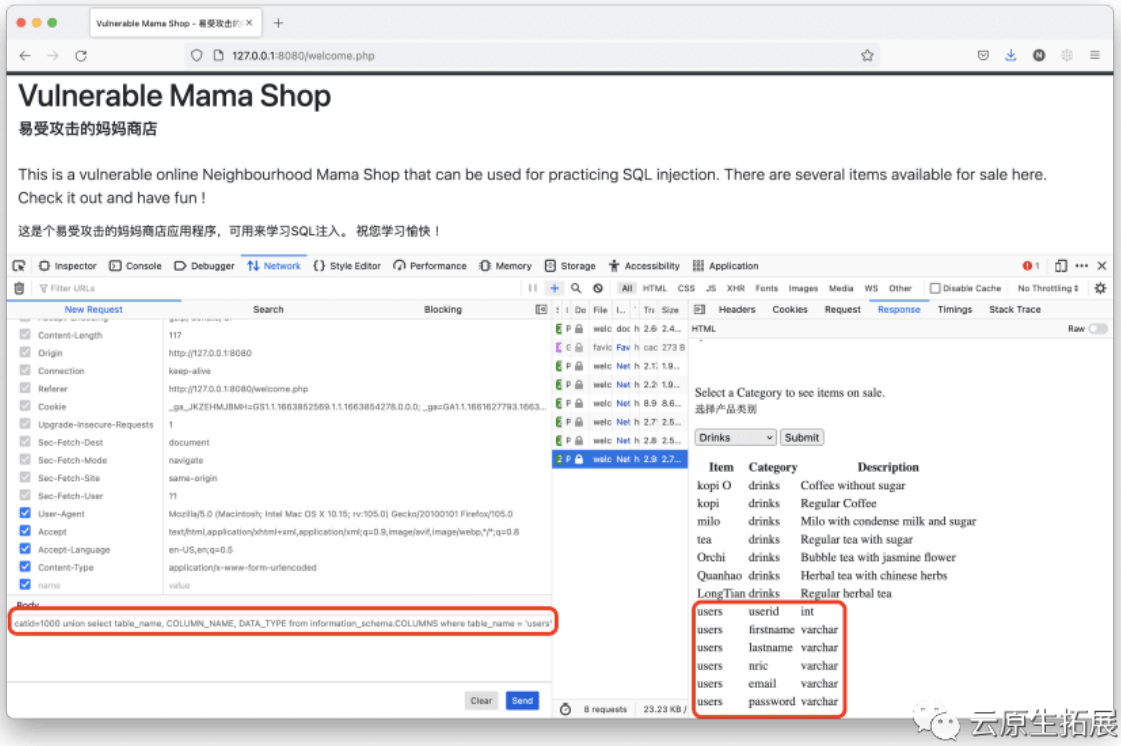
```
catid=1000 union select table_name,version, table_comment from information_schema.TABLES where table_schema = 'appdb'
```

所以我们得到了所有表的列表，我们可以安全地假设数据库中的 `users` 表包含用户名和密码。



我们需要检索 `users` 表具有的列列表，并且我们需要确保我们使用与从产品表中查询的相同数量的列以使 `UNION` 工作。

```
catid=1000 union select table_name, COLUMN_NAME, DATA_TYPE from information_schema.COLUMNS where table_name = 'users'
```

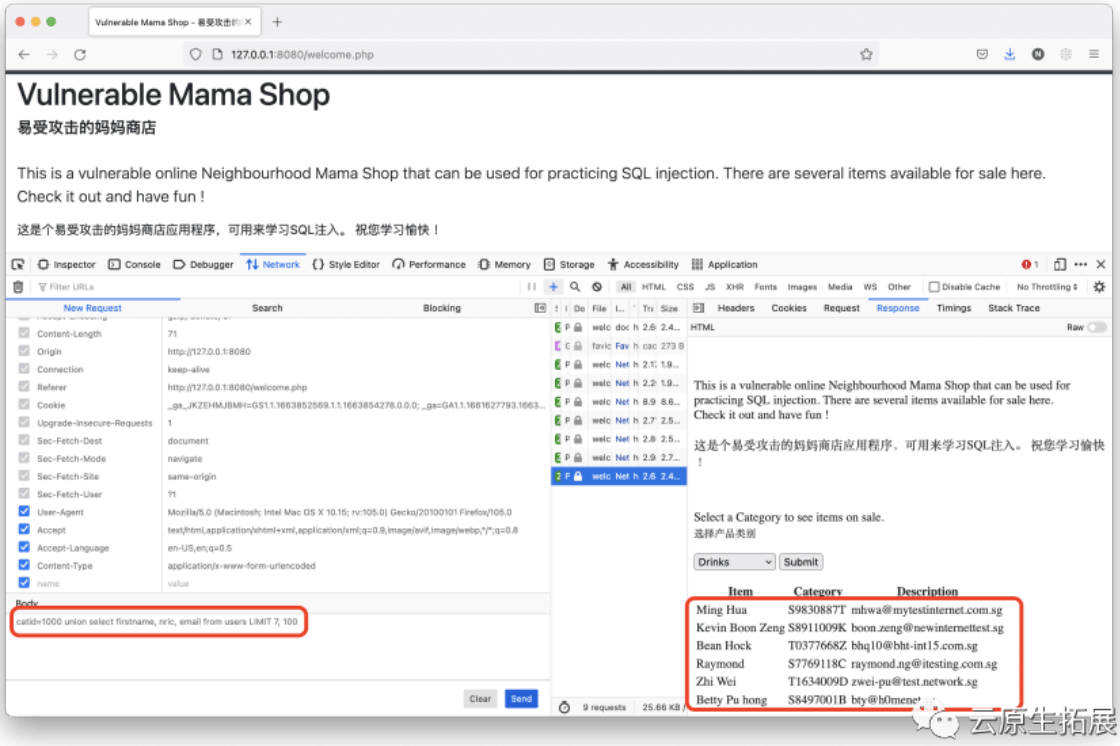


现在我们知道 users 表中共有六列，包含用户表中可用的名字、姓氏、nric、密码、电子邮件等详细信息。

我们已经收集到足够的信息来转储用户列表。以下输入可用于导出用户列表及其名字、密码和电子邮件地址。

```
catid=1000 union select firstname, nric, email from users LIMIT 7, 100
```

LIMIT 和 offset 的值可以通过观察正常请求中有多少项目条目来计算。偏移值应删除这些项目。可以将 LIMIT 设置为足够大的值，以便可以转储客户记录。



尝试并使用这些收集到的信息来查看您是否可以使用这些凭据登录。

使用 Pipy Sidecar 来阻止某些 SQLi 攻击

当未经验证的用户输入与 SQL 指令混合时会发生 SQL 注入，因此开发人员应更加注意转义用户输入（例如使用参数化查询），但您 - Kubernetes 工程师 - 也可以通过防止这种攻击来帮助避免 SQL 注入到达应用程序。这样，即使应用程序易受攻击，仍然可以阻止攻击。

保护您的应用程序有多种选择，但对于这篇博文，我们将重点关注注入一个 sidecar 容器来代理所有流量并拒绝任何被检测为 SQLi 攻击的请求。

> 出于演示目的，我们遵循手动注入 sidecar 的方法，但实际上，手动将代理部署为 sidecar 并不是最佳解决方案。您的应用程序和架构的复杂性可能需要更精细的控制。如果您的组织需要零信任并且需要端到端加密，请考虑像 osm-edge 这样的服务网格，它是一种轻量级、超快、低资源、高度可扩展、服务网格接口 (SMI) 兼容、专为边缘构建的服务网格和云计算服务网格。

部署 Pipy Sidecar

1. 使用以下内容创建一个名为 2-app-sidecar.yaml 的 YAML 文件，并检查这些值得注意的组件：

- 运行 Pipy 的边车容器在端口 8000 上启动。
- Pipy 进程将所有流量转发到应用程序。
- 包含 SQLi 的请求 URI 或 `POST` 正文被拒绝
- 该应用程序的服务首先将所有流量路由到 Pipy sidecar 容器。

```
apiVersion: apps/v1
kind: Deployment
metadata:
```

```yaml
metadata:
  name: vms
spec:
  selector:
    matchLabels:
      app: vms
  template:
    metadata:
      labels:
        app: vms
    spec:
      containers:
        - name: vms
          image: naqvis/mamashop
          ports:
            - containerPort: 80
        - name: pipy # <-- sidecar
          image: naqvis/pipy-nmi
          env:
            - name: PIPY_CONFIG_FILE
              value: /etc/pipy/nmi/nmi.js
          ports:
            - containerPort: 8000
          volumeMounts:
            - mountPath: /etc/pipy/nmi
              name: pipy-pjs
      volumes:
        - name: pipy-pjs
          configMap:
            name: sidecar
---
apiVersion: v1
kind: Service
metadata:
  name: vms
spec:
  ports:
    - port: 80
      targetPort: 8000 # <-- the traffic is routed to the proxy
      nodePort: 30060
  selector:
    app: vms
  type: NodePort
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: sidecar
data:
  nmi.js: |-
    pipy({
      _rejected: undefined,
    })
      .import({
        __is_sqli: 'lib-inject',
        __is_xss: 'lib-inject',
        __sqli_fingerprint: 'lib-inject',
      })
      .listen(8000)
      .demuxHTTP().to(
```

```
    $=>$
        .use('/etc/pipy/modules/inject-nmi.so')
        .handleMessage(() => _rejected = (__is_sqli || __is_xss))
        .branch(
            () => _rejected === true, (
            $=>$
            .handleMessageStart(_ =>
              console.log(`SQL Injection found with Fingerprint: ${__sqli_fingerprint}`))
            .replaceMessage(new Message({ status: 403 }, 'Forbidden'))
            ),
            () => _rejected === false, (
              $=>$.muxHTTP().to(
                  $=>$.connect('localhost:80')
              )
            )
        )
  )
```

2. 部署 sidecar

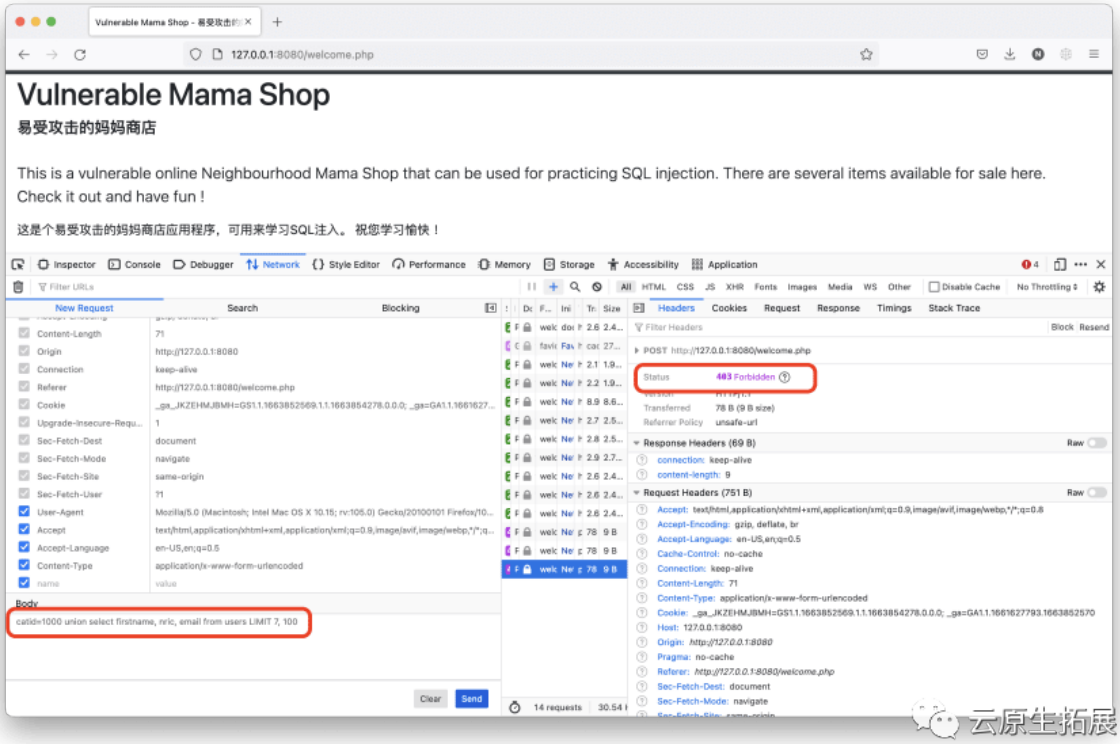
```
$ kubectl apply -f 2-app-sidecar.yaml
```

等待 pod 启动并准备就绪

```
$ kubectl get pods
NAME                 READY   STATUS    RESTARTS   AGE
vms-945f6f85c-8v7sb   2/2     Running   0          8m48s
```

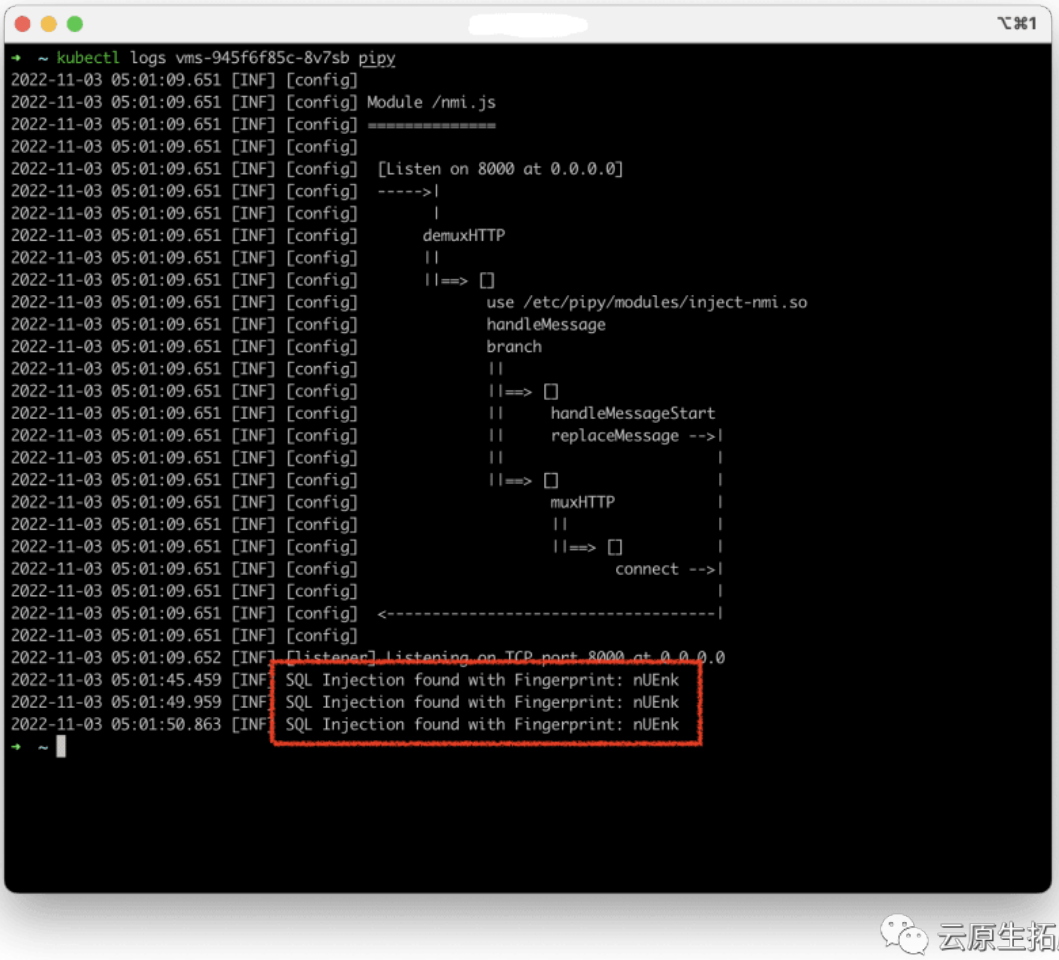
测试 Sidecar

通过返回应用程序并再次尝试 SQL 注入来测试 Sidecar 是否正在过滤流量。 Pipy sidecar 在请求到达应用程序之前将其阻止！



Sidecar 通过返回 403 Forbidden 来阻塞流量。使用我们之前进行的任何 SQLi 尝试再运行几个请求，我们将得到 403 作为响应。

我们可以通过查看 Pipy side car 的日志来验证这一点。



总结

学习攻击技术的目的是使开发人员、防御者和安全专业人员能够保护关键信息资产和服务的机密性、完整性和可用性。

默认情况下，Kubernetes 并不安全。这篇博文使用了 Vulnerable Mama Shop，这是一个基于 LAMP 堆栈的简单应用程序，用于演示 SQL 注入攻击以及如何使用 SideCar 等技术来抵御这些攻击。

但随着应用程序和架构的复杂性增加，您可能需要对服务进行更精细的控制。如果您的组织需要零信任并且需要像 mTLS 这样的端到端加密，您应该考虑像 osm-edge 这样的服务网格，它是一种轻量级、超快、低资源、高度可扩展的服务网格接口 (SMI)兼容，专为边缘和云计算服务网格而构建。当您在服务（东西向流量）之间进行通信时，服务网格允许您控制该级别的流量。