Bash Shell使用curl POST向Web API提交JSON

Published on Feb 4, 2022 UTC by Wang Ye <https://wangye.org>

一些项目中对外服务需要开放一些API接口，常见也是最易用的就是使用GET方式向URL传参查询字符串（Query String），这种方式的缺点主要有①传输的数据需要进行URL编码（URL Encode）；②受到URL最大长度限制，超过限制则有可能被Web服务器拒绝；③一种可能的安全缺陷就是Web服务器的日志有可能暴露敏感信息，比如查询字符串传递了密码，虽然使用HTTPS加密传输的URL能够避免被中间人拦截识别敏感信息，但是无法避免服务器端Web应用日志记录的问题，敏感信息可能会通过日志暴露给审计人员或者攻击者。

可能有读者会考虑使用POST方法是不是能够避免这些问题，确实对于复杂数据传输场景使用POST方式更为合适，因为POST请求的载荷（Payload）主要位于HTTP请求的主体（Body）部分，除非特殊配置，否则有效避免了②和③，对于①，常见的POST主体编码是表单模式，常见提交表单就是这种，主要向Web服务器申明主体是`x-www-form-urlencoded`编码格式，这样的编码属于把查询字符串（Query String）放在请求主体部分，避免URL最大长度限制，但是对于Web API来说仍然不够RESTful，RESTful的方式就是直接向Web服务器提交JSON数据。

常规的编程语言都有一些相关的实现，我就不一一举例，本文主要讨论如何在Bash Shell环境下调用curl向Web API提交JSON数据。

1 一种简单的实现

1.1 调用curl命令直接提交

比如我们的Web API地址位于https://localhost/api，那么最直接的提交方式就是使用下面的命令：

```
curl    -X POST \
           -H "Content-Type: application/json" \
```

```
           -d '{"key":"val"}' https://localhost/api
```

上面的命令将向https://localhost/api提交`{"key":"val"}`的JSON数据，另外一种实现同样功能的curl表述方式如下：

```
curl --header "Content-Type: application/json" \
       --request POST \
       --data '{"key":"val"}'  https://localhost/api
```

1.2 存在的问题

对于固定格式的JSON提交，1.1的方式无任何问题，但是需要注意的是，如果你提交的JSON数据是拼接的，那么就要小心了，对于转义字符处理不好将会破坏JSON的格式规范，导致提交无效的JSON数据，简单的例子，比如`$val`这个Shell变量作为JSON数据的一部分：

```
curl --header "Content-Type: application/json" \
       --request POST \
       --data '{"key":"$val"}'  https://localhost/api
```

如果`$val`为空，将会提交`{"key":}`，这个和我们期望的`{"key":""}`不一致，也是无效的JSON数据，如果`$val`的内容包含单双引号（" '），例如`val="\"'aaa"`这个赋值，必然会导致这种错误的JSON格式`'{key:"'aaa}'`，这个对于业务上可能是灾难性的，尤其是服务器端如果没有处理好这个异常的话。

2 稳妥的实现方式

为了避免转义字符带来的灾难性后果，我们必须对JSON的数据进行预先处理，这里就要使用jq这个小工具了，jq是一个轻量和灵活的基于命令行的JSON处理程序。

2.1 安装jq

对于Debian系列Linux系统可以通过`apt-get install jq`安装，如果apt包管理工具找不到安装选项的话，可以直接下载安装，如下命令将安装64位1.6版本的jq：

```
wget https://github.com/stedolan/jq/releases/download/jq-1.6/jq-linux64 -O jq && \
cp ./jq   /usr/bin/jq && chmod +x /usr/bin/jq
```

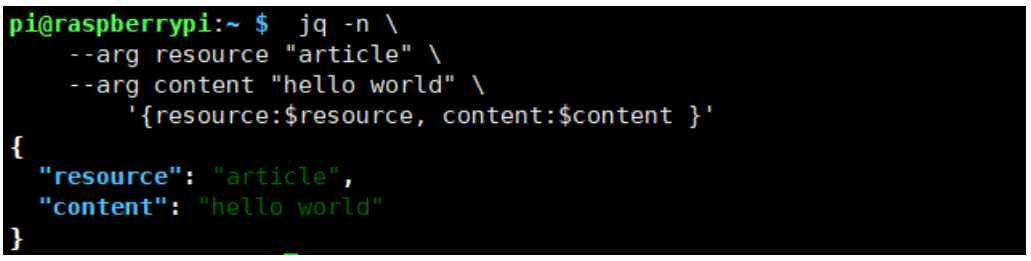
2.2 使用jq工具编码数据

首先我们尝试使用下面的命令生成一个简单的JSON数据：

```
jq -n \
    --arg resource "article" \
    --arg content "hello world" \
        '{resource:$resource, content:$content }'
```

上面的命令将生成一个如下的JSON数据内容如下：

```
{
  "resource": "article",
  "content": "hello world"
}
```



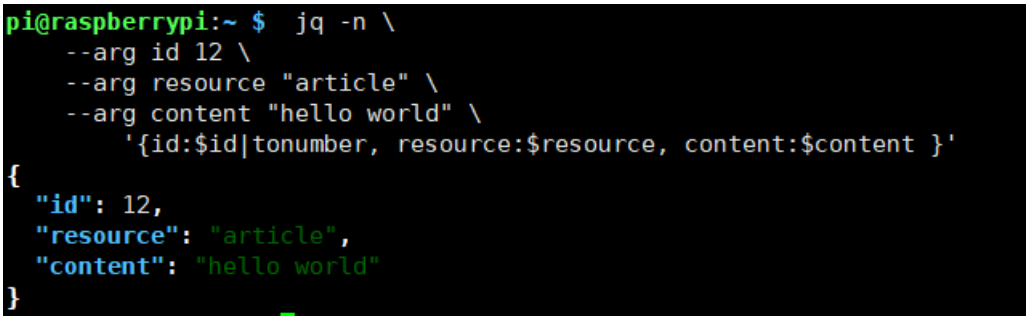
上面命令中`$resource`和`$content`不是Shell脚本的变量，只能算是模板`{resource:$resource, content:$content }`的占位符（placeholder），具体的内容将会由命令参数`--arg`所指示的数据替代，如果传入的数据需要转义，那

么会被自动转义以符合JSON格式的要求。 到这里可能读者尝试了一些使用jq命令的JSON转换，发现无论如何默认的转换都是引号包含的字符串格式，如果需要传入其他格式，比如整数，那么应该怎么办呢？比如下面的JSON数据：

```
{
  "id": 12,
  "resource": "article",
  "content": "hello world"
}
```

注意上面"id":12，通常默认转换是"id": "12"，这里的12变成了字符串，如果需要整形数字，那么只需要修改模板参数即可，例如：

```
jq -n \
   --arg id 12 \
   --arg resource "article" \
   --arg content "hello world" \
       '{id:$id|tonumber, resource:$resource, content:$content }'
```



这里模板使用了|tonumber修饰内部变量，以标明这是个整数型，更多的使用例子可以参考[jq的官方网站]，这里只做抛砖引玉的介绍。

2.3 使用Here Document插入的方式调用curl

原本准备配合jq工具修改并使用1.1的方式提交JSON，但转义这边处理还是比较麻烦，后来想到可以使用Here Document方式直接插入JSON数据，比如下面这样：

```
# 下面参数可以通过外部传入，这里为了直观起见直接赋值了
ID=12
RESOURCE="article"
CONTENT="hello world"
SERVER_URL="https://localhost/api"

JSON_STRING=$( jq -n \
        --arg id $ID \
        --arg resource "$RESOURCE" \
        --arg content "$CONTENT" \
          '{id:$id|tonumber, resource:$resource, content:$content }'
)

echo $JSON_STRING

curl -0 -X POST  $SERVER_URL \
      -H 'Content-Type: application/json' \
      --data-binary @- << EOF
$JSON_STRING
EOF
```

上面的命令通过<<EOF和EOF的Here Document方式直接插入了要提交的JSON数据。

3 总结

对于任何问题来说我们不能简单粗暴的看待，也许1.1节的方式可能短时间内能够很好的工作，但如果遇到1.2节的问题，那么将会是灾难性的，尤其是如果拼接的是用户传入的数据，是否用户可能构造恶意命令？还是那句话：永远不要相信任何外部输入的数据，就和零信任（Zero Trust，ZT）机制一样，认证认证再认证，校验校验再校验。

External References (Links)

jq这个小工具

https://stedolan.github.io/jq/

零信任（Zero Trust，ZT）机制

https://en.wikipedia.org/wiki/Zero_trust_security_model