Tony Bai

一个程序员的心路历程

HTTPS服务的Kubernetes ingress配置实践

- 六月 25, 2018
- 0 条评论

在公有云被广泛接纳的今天，数据传输安全问题日益凸显，因为在公有云提供商的经典网络（二层互通）中，即便是内部网络通信也要考虑网络嗅探等hack手段，这也是公有云主推所谓“专用网络（二层隔离）”的原因之一。从应用的角度，我们应该尽量通过技术手段保证数据通信的安全性。而目前最常用的方式就是基于SSL/TLS的安全通信方式了，在七层，对应的就是https了。

这样，下面的仅在负载均衡/反向代理入口做加密通信的传统模型越来越无法满足数据安全性的需要了(nginx与backend service之间是基于明文的http通信)：

传统安全通信模型：

```
client --- (via https) ---> nginx ---- (via http) ----> upstream backend services
```

我们需要下面的模型：

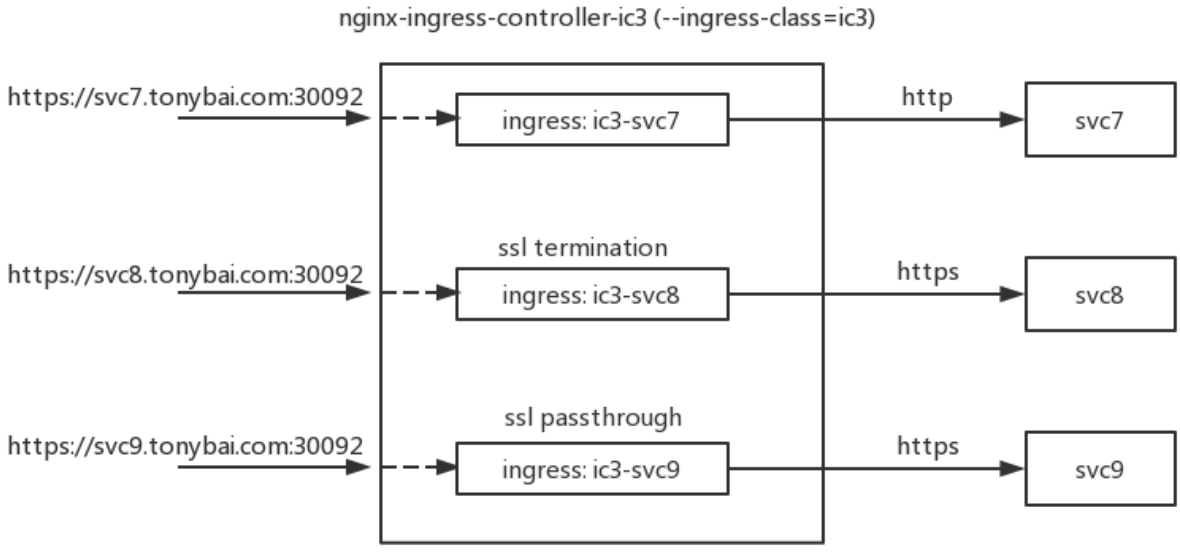
更为安全的通信模型：

```
client --- (via https) ---> nginx ---- (via https) ----> upstream backend services
```

在Kubernetes集群中，这种情况稍好些，首先，业务负载运行在集群的“虚拟网络”中，其次，一些K8s的网络插件实现是支持跨节点网络加密的（有一定的网络性能损耗），比如weave。但**永远没有绝对的安全**，作为业务应用的设计和实现人员，我们要尽可能的保证数据的通信安全，因此在面向七层的应用中，要尽可能的使用基于HTTPS的通信模型。本篇就来实践一下如何为Kubernetes集群内的HTTPS服务进行ingress的配置。

一. 例子概述与环境准备

在《实践kubernetes ingress controller的四个例子》一文中，我讲解了四种基本的kubernetes ingress配置方式。在这些例子中，有些例子的ingress controller(nginx)与backend service之间使用的是https，但client到ingress controller之间的通信却一直是基于http的。在本文中，我们的目标就是上面提到的那个**更为安全的通信模型**，即client与ingress controller(nginx)、nginx与backend service之间均使用的是https通信。这里在《实践kubernetes ingress controller的四个例子》一文例子的基础上，我们创建一个新的nginx ingress controller: **nginx-ingress-controller-ic3**，并将后端的svc7~svc9三个不同类型的服务暴露给client，如下图所示：



- svc7: 是对传统通信模型的“复现”，即client与ingress controller(nginx)间采用https加密通信，但ingress controller(nginx)与svc7间则是明文的http通信；
- svc8: 是ssl-termination的安全配置模型，即client与svc8的https通信分为“两段”，client与nginx建立https连接后，nginx将client提交的加密请求解密后，再向svc8发起https请求，并重新加密请求数据。这种client端ssl的过程在反向代理或负载均衡器终结的https通信方式被称为“ssl-termination”。
- svc9: 是ssl-passthrough的安全配置模型，即nginx不会对client的https request进行解密，而是直接转发给backend的svc9服务，client端的ssl过程不会终结于nginx，而是在svc9对应的pod中终结。这种https通信方式被称为”ssl-passthrough”。这种配置模型尤其适合backend service对client端进行client certificate验证的情况，同时也降低了nginx加解密的性能负担。

本文基于下面环境进行实验：kubernetes 1.10.3、weave networks 2.3.0、nginx-ingress-controller:0.15.0。关于本文涉及的例子的源码、chart包以及ingress controllers的yaml源文件可以在这里下载到。

二. 建立新的ingress-nginx-controller：nginx-ingress-controller-ic3

为了更好地进行例子说明，我们建立一个新的ingress-nginx-controller：**nginx-ingress-controller-ic3**，svc7~svc9都通过该ingress controller进行服务入口的暴露管理。要创建**nginx-ingress-controller-ic3**，我们首先需要在ic-common.yaml中为Role: nginx-ingress-role添加一个resourceName： “ingress-controller-leader-ic3”，并apply生效：

```
// ic-common.yaml
... ...
    resourceNames:
      # Defaults to "<election-id>-<ingress-class>"
      # Here: "<ingress-controller-leader>-<nginx>"
      # This has to be adapted if you change either parameter
      # when launching the nginx-ingress-controller.
      - "ingress-controller-leader-ic1"
      - "ingress-controller-leader-ic2"
      - "ingress-controller-leader-ic3"
... ...

# kubectl apply -f ic-common.yaml
```

我们为**nginx-ingress-controller-ic3**创建nodeport service，新nodeport为：30092：

```
// ic3-service-nodeport.yaml
apiVersion: v1
kind: Service
metadata:
  name: ingress-nginx-ic3
  namespace: ingress-nginx-demo
spec:
  type: NodePort
  ports:
  - name: https
    port: 443
    targetPort: 443
    nodePort: 30092
    protocol: TCP
  selector:
    app: ingress-nginx-ic3
```

注意：ingress-nginx-ic3 service的nodeport映射到ic3 ingress controller的443端口，也就是支持安全通信的端口，而不是明文的80端口。

最后创建nginx-ingress-controller-ic3 pod，可以复制一份ic2-mandatory.yaml，然后将内容中的ic2全部修改为ic3即可：

```
# kubectl apply -f ic3-mandatory.yaml
```

如无意外，**nginx-ingress-controller-ic3**应该已经正常地运行在你的k8s cluster中了。

三. svc7: 使用ssl termination，但nginx与backend服务之间采用明文传输（http）

加密Web流量有两个主要配置方案：SSL termination和SSL passthrough。

使用SSL termination时，客户端的SSL请求在负载均衡器/反向代理中解密，解密操作将增加负载均衡器的工作负担，较为耗费CPU，但简化了SSL证书的管理。至于负载均衡器和后端之间的流量是否加密，需要nginx另行配置。

SSL Passthrough，意味着client端将直接将SSL连接发送到后端(backend)。与SSL termination不同，请求始终保持加密，并且解密负载分布在后端服务器上。但是，这种情况的SSL证书管理略复杂，证书必须在每台服务器上自行管理。另外，在这种

方式下可能无法添加或修改HTTP header，可能会丢失X-forwarded-* header中包含的客户端的IP地址，端口和其他信息。

我们先来看一种并不那么“安全”的“传统模型”：**在nginx上暴露https，但nginx到backend service(svc7)采用http。**

我们先来创建相关的密钥和公钥证书，并以一个Secret：ingress-controller-demo-tls-secret存储密钥和证书数据：

```
// ingress-controller-demo/manifests下面

# openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ic3.key -out ic3.crt -subj "/CN=*.tonybai.com/O=tonybai.com"
# kubectl create secret tls ingress-controller-demo-tls-secret --key  ic3.key --cert ic3.crt
```

svc7几乎是和svc1一样的程序（输出的字符串标识不同），但svc7的ingress与svc1大不相同，因为我们需要通过https访问svc7的ingress：

```
// svc7的values.yaml
... ...
replicaCount: 1

image:
  repository: bigwhite/ingress-controller-demo-svc7
  tag: v0.1
  pullPolicy: Always

service:
  type: ClusterIP
  port: 443

ingress:
  enabled: true
  annotations:
    kubernetes.io/ingress.class: ic3
  path: /
  hosts:
    - svc7.tonybai.com
  tls:
    - secretName: ingress-controller-demo-tls-secret
      hosts:
        - svc7.tonybai.com
... ...
```

与svc1的values.yaml不同的是，我们使用的ingress controller是ic3，我们开启了tls，secret用的就是我们上面创建的那个secret：ingress-controller-demo-tls-secret。创建ic3-svc7后，我们看到ingress controller内部的nginx.conf中有关svc7的配置输出如下：

```
# kubectl exec nginx-ingress-controller-ic3-67f7cf7845-2tnc9 -n ingress-nginx-demo -- cat /etc/nginx/nginx.conf

        # map port 442 to 443 for header X-Forwarded-Port
        map $pass_server_port $pass_port {
                442             443;
                default         $pass_server_port;
        }

        upstream default-ic3-svc7-http {
                least_conn;

                keepalive 32;

                server 192.168.28.13:8080 max_fails=0 fail_timeout=0;

        }

## start server svc7.tonybai.com
        server {
                server_name svc7.tonybai.com ;

                listen 80;

                listen [::]:80;

                set $proxy_upstream_name "-";

                listen 442 proxy_protocol   ssl http2;

                listen [::]:442 proxy_protocol  ssl http2;

                # PEM sha: 248951b75535e0824c1a7f74dc382be3447057b7
                ssl_certificate                         /ingress-controller/ssl/default-ingress-controller-demo-tls-secret.pem;
                ssl_certificate_key                     /ingress-controller/ssl/default-ingress-controller-demo-tls-secret.pem;

                ssl_trusted_certificate                 /ingress-controller/ssl/default-ingress-controller-demo-tls-secret-full-chain.pem;
                ssl_stapling                            on;
                ssl_stapling_verify                     on;

                location / {
```

```
                    ... ...
                    proxy_pass http://default-ic3-svc7-http;

                    proxy_redirect                          off;

            }
        ... ...
    }
    ## end server svc7.tonybai.com
```

可以看到30092(nodeport) 映射的ingress controller的443端口在svc7.tonybai.com这个server域名下已经有了ssl标识，并且ssl_certificate和ssl_certificate_key对应的值就是我们之前创建的ingress-controller-demo-tls-secret。

我们通过curl访问以下svc7服务：

```
# curl -k https://svc7.tonybai.com:30092
Hello, I am svc7 for ingress-controller demo!
```

此时，如果再用http方式去访问svc7，你会得到下面错误结果：

```
# curl http://svc7.tonybai.com:30092
<html>
<head><title>400 The plain HTTP request was sent to HTTPS port</title></head>
<body bgcolor="white">
<center><h1>400 Bad Request</h1></center>
<center>The plain HTTP request was sent to HTTPS port</center>
<hr><center>nginx/1.13.12</center>
</body>
</html>
```

四. svc8: 使用ssl termination，但nginx与backend服务之间采用加密传输(https)

前面说过，SSL termination配置场景中，负载均衡器和后端之间的流量是否加密，需要nginx另行配置。svc7采用了未加密的方式，nginx -> backend service存在安全风险，我们要将其改造为也通过https进行数据加密传输，于是有了svc8这个例子。

svc8对应的程序本身其实是上一篇文章《实践kubernetes ingress controller的四个例子》中的svc2的clone（唯一修改就是输出的log中的标识)。

在svc8对应的chart中，我们将values.yaml改为：

```
// ingress-controller-demo/charts/svc8/values.yaml

replicaCount: 1

image:
  repository: bigwhite/ingress-controller-demo-svc8
  tag: v0.1
  pullPolicy: Always

service:
  type: ClusterIP
  port: 443

ingress:
  enabled: true
  annotations:
    # kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/secure-backends: "true"
    kubernetes.io/ingress.class: ic3
  path: /
  hosts:
    - svc8.tonybai.com
  tls:
    - secretName: ingress-controller-demo-tls-secret
      hosts:
        - svc8.tonybai.com

... ...
```

与svc7不同点在于values.yaml中的新annotation： **nginx.ingress.kubernetes.io/secure-backends: “true”** 。这个annotation让nginx以https的方式去访问backend service: svc8。安装svc8 chart后，ingress nginx controller为svc8生成的配置如下：

```
## start server svc8.tonybai.com
        server {
```

```
                server_name svc8.tonybai.com ;

                listen 80;

                listen [::]:80;

                set $proxy_upstream_name "-";

                listen 442 proxy_protocol   ssl http2;

                listen [::]:442 proxy_protocol  ssl http2;

                # PEM sha: 248951b75535e0824c1a7f74dc382be3447057b7
                ssl_certificate                         /ingress-controller/ssl/default-ingress-controller-demo-tls-secret.pem;
                ssl_certificate_key                     /ingress-controller/ssl/default-ingress-controller-demo-tls-secret.pem;

                ssl_trusted_certificate                 /ingress-controller/ssl/default-ingress-controller-demo-tls-secret-full-chain.pem;
                ssl_stapling                            on;
                ssl_stapling_verify                     on;

                location / {
                     ... ...
                        proxy_pass https://default-ic3-svc8-https;

                        proxy_redirect                          off;

                }

        }
        ## end server svc8.tonybai.com

        upstream default-ic3-svc8-https {
                least_conn;

                keepalive 32;

                server 192.168.28.14:8080 max_fails=0 fail_timeout=0;

        }
```

使用curl访问svc8服务（-k: 忽略对server端证书的校验):

```
# curl -k https://svc8.tonybai.com:30092
Hello, I am svc8 for ingress-controller demo!
```

五. svc9: 使用ssl passthrough, termination at pod

某些服务需要通过对client端的证书进行校验的方式，进行身份验证和授权，svc9就是这样一个对client certification进行校验的双向https校验的service。针对这种情况，ssl termination的配置方法无法满足需求，我们需要使用ssl passthrough的方案。

在ingress nginx controller开启ssl passthrough方案需要在ingress controller和ingress中都做一些改动。

首先我们需要为**nginx-ingress-controller-ic3**添加一个新的命令行参数：**–enable-ssl-passthrough**，并重新apply生效:

```
// ic3-mandatory.yaml
... ...
spec:
      serviceAccountName: nginx-ingress-serviceaccount
      containers:
        - name: nginx-ingress-controller-ic3
          image: quay.io/kubernetes-ingress-controller/nginx-ingress-controller:0.15.0
          args:
            - /nginx-ingress-controller
            - --default-backend-service=$(POD_NAMESPACE)/default-http-backend
            - --configmap=$(POD_NAMESPACE)/nginx-configuration-ic3
            - --tcp-services-configmap=$(POD_NAMESPACE)/tcp-services-ic3
            - --udp-services-configmap=$(POD_NAMESPACE)/udp-services-ic3
            - --publish-service=$(POD_NAMESPACE)/ingress-nginx-ic3
            - --annotations-prefix=nginx.ingress.kubernetes.io
            - --enable-ssl-passthrough
            - --ingress-class=ic3
... ...
```

然后在svc9的chart中，为ingress添加新的annotation nginx.ingress.kubernetes.io/ssl-passthrough: “true”

```
// ingress-controller-demo/charts/svc9/values.yaml

replicaCount: 1

image:
  repository: bigwhite/ingress-controller-demo-svc9
```

```
    tag: v0.1
    pullPolicy: Always

  service:
    type: ClusterIP
    port: 443

  ingress:
    enabled: true
    annotations:
      kubernetes.io/ingress.class: ic3
      nginx.ingress.kubernetes.io/ssl-passthrough: "true"

    path: /
    hosts:
      - svc9.tonybai.com
    tls:
      - secretName: ingress-controller-demo-tls-secret
        hosts:
          - svc9.tonybai.com
... ...
```

isntall svc9 chart之后，我们用curl来访问以下svc9：

```
# curl -k  https://svc9.tonybai.com:30092
curl: (35) gnutls_handshake() failed: Certificate is bad
```

由于svc9程序对client端的certificate进行验证，没有提供client certificate的curl请求被拒绝了！svc9 pod的日志也证实了这一点：

```
2018/06/25 05:36:29 http: TLS handshake error from 192.168.31.10:38634: tls: client didn't provide a certificate
```

我们进入到ingress-controller-demo/src/svc9/client路径下，执行：

```
# curl -k --key ./client.key --cert ./client.crt https://svc9.tonybai.com:30092
Hello, I am svc9 for ingress-controller demo!
```

带上client.crt后，svc9通过了验证，返回了正确的应答。

client路径下是一个svc9专用的客户端，我们也可以执行该程序去访问svc9:

```
# go run client.go
Hello, I am svc9 for ingress-controller demo!
```

我们再看看采用ssl-passthrough方式下ingress-nginx controller的访问日志，当curl请求发出时，ingress-nginx controller并未有日志输出，因为没有在nginx处ssl termnination，从此也可以证实：nginx将client的ssl过程转发到pod中去了，即passthrough了。

51短信平台：企业级短信平台定制开发专家 https://51smspush.com/
smspush：可部署在企业内部的定制化短信平台，三网覆盖，不惧大并发接入，可定制扩展； 短信内容你来定，不再受约束，接口丰富，支持长短信，签名可选。

著名云主机服务厂商DigitalOcean发布最新的主机计划，入门级Droplet配置升级为：1 core CPU、1G内存、25G高速SSD，价格5$/月。有使用DigitalOcean需求的朋友，可以打开这个链接地址：https://m.do.co/c/bff6eed92687 开启你的DO主机之路。

我的联系方式：

微博：https://weibo.com/bigwhite20xx
微信公众号：iamtonybai
博客：tonybai.com
github: https://github.com/bigwhite

微信赞赏：

商务合作方式：撰稿、出书、培训、在线课程、合伙创业、咨询、广告合作。

© 2018, bigwhite. 版权所有.

Related posts:

1. 实践kubernetes ingress controller的四个例子
2. 一步步打造基于Kubeadm的高可用Kubernetes集群-第二部分
3. 在Kubernetes 1.10.3上以Hard模式搭建EFK日志分析平台
4. 为Kubernetes集群中服务部署Nginx入口服务
5. Kubernetes集群中Service的滚动更新

添加新评论

称呼

邮箱

网站 http://example.com

提交评论

Capatcha

输入关键字搜索 搜索

如发现本站页面被黑，比如：挂载广告、挖矿等恶意代码，请朋友们及时联系我。十分感谢！

欢迎使用邮件订阅我的博客

输入邮箱订阅本站，只要有新文章发布，就会第一时间发送邮件通知你哦！

名字： 请输入您的名字

邮箱: 请输入您的电子邮箱

马上订阅

这里是 Tony Bai的个人Blog，欢迎访问、订阅和留言！ 订阅Feed请点击上面图片。

如果您觉得这里的文章对您有帮助，请扫描上方二维码进行捐赠 ， 加油后的Tony Bai将会为您呈现更多精彩的文章，谢谢！

如果您希望通过微信捐赠，请用微信客户端扫描下方赞赏码：

如果您希望通过比特币或以太币捐赠，可以扫描下方二维码：

比特币：

以太币：

如果您喜欢通过微信浏览本站内容，可以扫描下方二维码，订阅本站官方微信订阅号“iamtonybai”；点击二维码，可直达本

人官方微博主页^_^：

本站Powered by Digital Ocean VPS。
选择Digital Ocean VPS主机，即可获得10美元现金充值，可 免费使用两个月哟！ 著名主机提供商Linode 10$优惠码：linode10，在 这里注册即可免费获 得。阿里云推荐码： 1WFZ0V， 立享9折！

bigwhite.cn@Gmail.com
my Linked in profile

我的业余项目

- smspush短信发送平台

文章

- 初窥Go module
- HTTPS服务的Kubernetes ingress配置实践
- 实践kubernetes ingress controller的四个例子
- 使用kubectl访问Kubernetes集群时的身份验证和授权
- 在Kubernetes 1.10.3上以Hard模式搭建EFK日志分析平台
- 对一段有关Go Code Block和变量作用域的代码的简要分析
- 慕课网免费课“Kubernetes：开启云原生之门”上线
- 写Go代码时遇到的那些问题[第3期]
- defer函数参数求值简要分析
- 对一段Go语言代码输出结果的简要分析

评论

- 正在加载...
-
-

分类

- 光影汇 (7)
- 影音坊 (36)
- 思考控 (66)
- 技术志 (561)
- 教育记 (1)
- 杂货铺 (75)
- 生活簿 (154)
- 职场录 (14)
- 读书吧 (14)
- 运动迷 (107)
- 驴友秀 (40)

标签

Blog Blogger C Cpp docker English GCC github GNU Go Golang Google Java k8s Kernel Kubernetes Linux M10 Opensource Programmer Python Solaris Subversion Ubuntu Unix Windows 世界杯 博客 学习 容器 工作 巴萨 开

源 思考 感悟 摄影 旅游 梅西 球王 生活 程序员 编译器 西甲 足球 驴友

归档

- 2008 年四月 (4)
- 2008 年三月 (9)
- 2008 年二月 (11)
- 2008 年一月 (15)
- 2007 年十二月 (11)
- 2007 年十一月 (14)
- 2007 年十月 (4)
- 2007 年九月 (5)
- 2007 年八月 (1)
- 2007 年七月 (10)
- 2007 年六月 (10)
- 2007 年五月 (10)
- 2007 年四月 (8)
- 2007 年三月 (15)
- 2007 年二月 (4)
- 2007 年一月 (17)
- 2006 年十二月 (18)
- 2006 年十一月 (9)
- 2006 年十月 (11)
- 2006 年九月 (6)
- 2006 年八月 (5)
- 2006 年七月 (22)
- 2006 年六月 (35)
- 2006 年五月 (24)
- 2006 年四月 (26)
- 2006 年三月 (25)
- 2006 年二月 (18)
- 2006 年一月 (15)
- 2005 年十二月 (10)
- 2005 年十一月 (10)
- 2005 年九月 (13)
- 2005 年八月 (11)
- 2005 年七月 (6)
- 2005 年六月 (2)
- 2005 年五月 (3)
- 2005 年四月 (6)
- 2005 年三月 (1)
- 2005 年一月 (15)
- 2004 年十二月 (9)
- 2004 年十一月 (14)
- 2004 年十月 (2)
- 2004 年九月 (2)

私人

- 我的女儿

链接

- @douban
- @flickr
- @github
- @googlecode
- @picasa
- @slideshare
- @twitter
- @weibo
- Hoterran
- Lionel Messi
- Puras He
- 梦想风暴
- 过眼云烟

开源项目

- buildc
- cbehave
- lcut

翻译项目

- C语言编码风格和标准
- 《Programming in Haskell》中文翻译项目