Blogs Subscribe

ENGINEERING

Authenticating S3 Proxy

Paul Saunders

01/03/17

Using nginx and oauth2_proxy to deliver S3 based content (and azure blob store)

tl;dr

1. Making oauth2_proxy work with nginx, and redirecting the output was hard.
2. It is possible to have an oauth2 authenticated website that serves static content from Azure Blob Stores, or S3 buckets.

S3 static sites

AWS S3 buckets allows you to serve up static html as a website – see here. But what if you want to control access to the website?

Our setup has us using AWS for everything except our authentication, which we are doing via Azure Active Directory. So the way to secure a static website would be to host it behind a proxy, and force authentication via AzureAD. For the authentication piece, we chosen to use oauth2_proxy. This could also be the proxy, but we already use nginx in a number of places, so we leverage that.

To set it up in AWS, we use Elastic Beanstalk to host a multi-container setup. The deployment basically takes a json'ified version of a docker-compose file. So what does

```
version: '2'
services:

  oauth2_proxy:
    image: 0/oauth2_proxy:latest
    environment:
    - OAUTH2_PROXY_CLIENT_ID=0
    - OAUTH2_PROXY_CLIENT_SECRET=0
    - OAUTH2_PROXY_COOKIE_SECRET=0
    - OAUTH2_PROXY_EMAIL_DOMAIN=finbourne.com
    - OAUTH2_PROXY_TENANT_ID=0
    command:

  nginx:
    image: openresty/openresty:alpine
    ports:
      - "80:80"
      - "443:443"
    links:
      - oauth2_proxy
    volumes:
     - ./aws/nginx.conf:/usr/local/openresty/nginx/conf/nginx.conf
     - ./.certs:/etc/ssl/certs
```

Since Elastic Beanstalk multidocker container deployments can't pass command arguments to containers, we've had to roll our own version of oauth2_proxy. This requires the dockerfile, and a startup.sh script:

`dockerfile`

```
FROM alpine:latest

ENV OAUTH2_PROXY_VERSION 2.1
ENV GO_VERSION 1.6

RUN apk --update add curl

RUN curl -sL -o oauth2_proxy.tar.gz \
    "https://github.com/bitly/oauth2_proxy/releases/download/v$OAUTH2_PROXY_VERSION/
  && tar xzvf oauth2_proxy.tar.gz \
  && mv oauth2_proxy-*/oauth2_proxy /bin/ \
  && chmod +x /bin/oauth2_proxy \
  && rm -r oauth2_proxy*
```

```
CMD ./startup.sh
```

`startup.sh`

```
#!/bin/ash

oauth2_proxy --email-domain=$OAUTH2_PROXY_EMAIL_DOMAIN \
             --provider=azure \
             --azure-tenant=$OAUTH2_PROXY_TENANT_ID \
             --upstream=https://0.0.0.0 \
             --http-address=0.0.0.0:4180
```

That just leaves getting nginx to play nicely with S3. In Azure it was dead simple, just proxy pass to a SAS tokenised url. S3 on the other hand is quite a bit different. But a bit of googling later, many failed attempts and then eventually thanks to https://github.com/lovelysystems/nginx-examples, I managed to cobble together the following:

`nginx.conf` – once again, substitute your own values into vars.

```
#user  nginx;
worker_processes  1;

error_log  /usr/local/openresty/nginx/logs/error.log warn;
pid        /usr/local/openresty/nginx/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /usr/local/openresty/nginx/conf/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /usr/local/openresty/nginx/logs/access.log  main;

    sendfile        on;
    #gzip  on;


server {
    listen 80;

    location /health {
      return 200 'Healthy!';
      add_header Content-Type text/plain;
    }

    location / {
      return 301 https://$host$request_uri;
    }
}

server {
    listen 443;

  proxy_intercept_errors off;
  proxy_send_timeout 120;
  proxy_read_timeout 300;

    client_max_body_size 3G;

    ssl_certificate           /etc/ssl/certs/cert.crt;
    ssl_certificate_key       /etc/ssl/certs/cert.key;

    ssl on;
    ssl_session_cache  builtin:1000  shared:SSL:10m;
    ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers HIGH:!aNULL:!eNULL:!EXPORT:!CAMELLIA:!DES:!MD5:!PSK:!RC4;
    ssl_prefer_server_ciphers on;

    access_log            /usr/local/openresty/nginx/logs/access.log;

    keepalive_timeout  5 5;
    proxy_buffering    off;

    location = /oauth2/auth {
        internal;
        proxy_pass http://oauth2_proxy:4180;
    }

    location /oauth2/ {
        proxy_pass http://oauth2_proxy:4180;
    }

    location / {
        auth_request /oauth2/auth;
        error_page 401 = /oauth2/start?rd=$request_uri;  # required to get redirecti

        set $s3_bucket '';

        # Setup AWS Authorization header
        set $aws_signature '';

        # the only reason we need lua is to get the current date
        set_by_lua $now "return ngx.cookie_time(ngx.time())";

        #the  access key
        set $aws_access_key '';
        set $aws_secret_key '';

        # the actual string to be signed
        # see: http://docs.amazonwebservices.com/AmazonS3/latest/dev/RESTAuthenticat
        set $string_to_sign "$request_method\n\n\n\nx-amz-date:$now\n/$s3_bucket/tes

        # create the hmac signature
        set_hmac_sha1 $aws_signature $aws_secret_key $string_to_sign;
        # encode the signature with base64
        set_encode_base64 $aws_signature $aws_signature;
        proxy_set_header x-amz-date $now;
        proxy_set_header Authorization "AWS $aws_access_key:$aws_signature";

        proxy_http_version     1.1;
        proxy_set_header       Host $s3_bucket.s3.amazonaws.com;
        proxy_redirect off;
        proxy_set_header       X-Real-IP $remote_addr;
        proxy_set_header       X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_hide_header      Content-Type;
        proxy_hide_header      x-amz-id-2;
        proxy_hide_header      x-amz-request-id;

        resolver               8.8.8.8 valid=300s;
        proxy_pass             https://$s3_bucket.s3.amazonaws.com/test/unit-results
        proxy_read_timeout     90;
    }
  }
}
```

And that's it for S3.

And what about Azure??

Azure was very similar except for the `nginx.config`, the salient difference outlined here:

```
location / {
    auth_request /oauth2/auth;
    error_page 401 = /oauth2/start?rd=$request_uri;  # required to get redirection

    proxy_hide_header Content-Type;
    set $args $args&st=2017-01-12T17%3A13%3A00Z&se=2017-03-13T17%3A13%3A00Z&sp=r&s
    proxy_pass          https://.blob.core.windows.net/tests/;
    proxy_read_timeout  90;
}
```

Conclusion

The devil is almost always in the detail. I didn't have redirect working after sign-in, and desperately wanted it to work. I couldn't find any help online with it. I did open an issue on the oauth2_proxy but there was no immediate help, so I soldiered on by myself, eventually posting back the solution to fulfil my duties as an upstanding netizen.

In order to test it with AzureAD, the code had to be deployed. That doesn't make for fast iteration devops. I must have burned a day and a half getting that one line right. For me, worth it. I burned a few hours getting the S3 authentication bit right too. Also worth it.

Paul Saunders

01/03/17

Subscribe

Get our top content on data strategy for the investment world in your inbox regularly

Subscribe

Engineering (11)
Investment Technology (21)
News (8)
Product (4)

Subscribe to our newsletter

Get stories like this in your inbox

Sign up

Related articles

The Mysterious Hanging Client & TCP Keep Alives

Michael McGarry 07/05/20

You should work from home unless it is impossible for you to do so

Chris Brook 06/04/20

HFM
EUROPEAN TECHNOLOGY SUMMIT
Technology Leaders: Plotting the future of hedge fund tech

Plotting the future of hedge fund technology

Thomas McHugh 11/02/20

What a Naval Historian Can Teach Us About Data

Steve Cheng 08/03/18

ABOUT FINBOURNE

About Finbourne

Blogs

News

Events

Careers

Contact us

SOLUTIONS

FOR DEVELOPERS

Knowledge Base

Help Centre