

Agloe: What the map makers of the 1930s can teach us about protecting data in 2018

Captains log star date 47634.44...

So recently (read: in August) I did a talk on the PyCon Australia Security and Privacy track about Canary Tokens which you can watch on [YouTube](#). More recently, in October I gave the same talk (but better) at The Open Web Application Security Project (OWASP) AppSec Day in Melbourne.

I thought it would be a nice follow up to talk more about some of the technical and non-technical points I addressed there, as well as talk a bit more about deploying the Thinkst Canary Tokens in your own environment.

Paper Towns

Phantom settlements or "Paper Towns" are settlements, towns or locations that appear in maps but don't actually exist. They are typically created by accident, or more often than not are created as copyright traps. This is done because the very nature of maps describes "what is" and copyright infringement can be hard to prove.

Throughout history there have been a number of notable examples including:

- Agloe, Colchester, NY - first appeared in 1930
- Argleton, Lancashire, UK - first identified in 2008 by Mike Nolan
- Beatosu and Gobl, Ohio, US - first appeared in the 1978–1979 official state map of Michigan

However, many more of these locations exist. At least on paper.

Agloe came to exist after the General Drafting Company was founded by Otto G. Lindberg in 1909. In 1930 Lindberg and his assistant Ernest Alpers assigned an anagram of their initials to a dirt-road intersection (AGLOE) as a copyright trap to catch out map thieves.



In the 1950s a small general store was built on the intersection on the map and named "Agloe General Store" because the name was on the Esso maps. At the time the General Drafting Company was the sole provider of maps for Esso.

A number of years later Rand McNally a competing map company added Agloe to their maps. This revealed the trap and Esso tried to sue McNally for copyright infringement. However, McNally had got the name of the "hamlet" from the Delaware County administration - as it was pointed out because the Agloe General Store existed, so did the town, and no infringement could be shown.

The store eventually went out of business, but the town continued to appear on maps, Google Maps listed Agloe as a town until it was silently removed in 2014.

So, what does the little town of Agloe, NY have in common with modern day data protection?

When doing assessments, we often find that organisations put considerable effort into perimeter security - firewalls, application security etc. However, very often once you get past the hard-crunchy outside, you find yourself in their soft squishy centre, in some cases you're dropped right in the middle of their network.

In other cases, you may not be as concerned about your perimeter as you are about malicious actors inside your companies' network.

That's where canary tokens come into the picture.

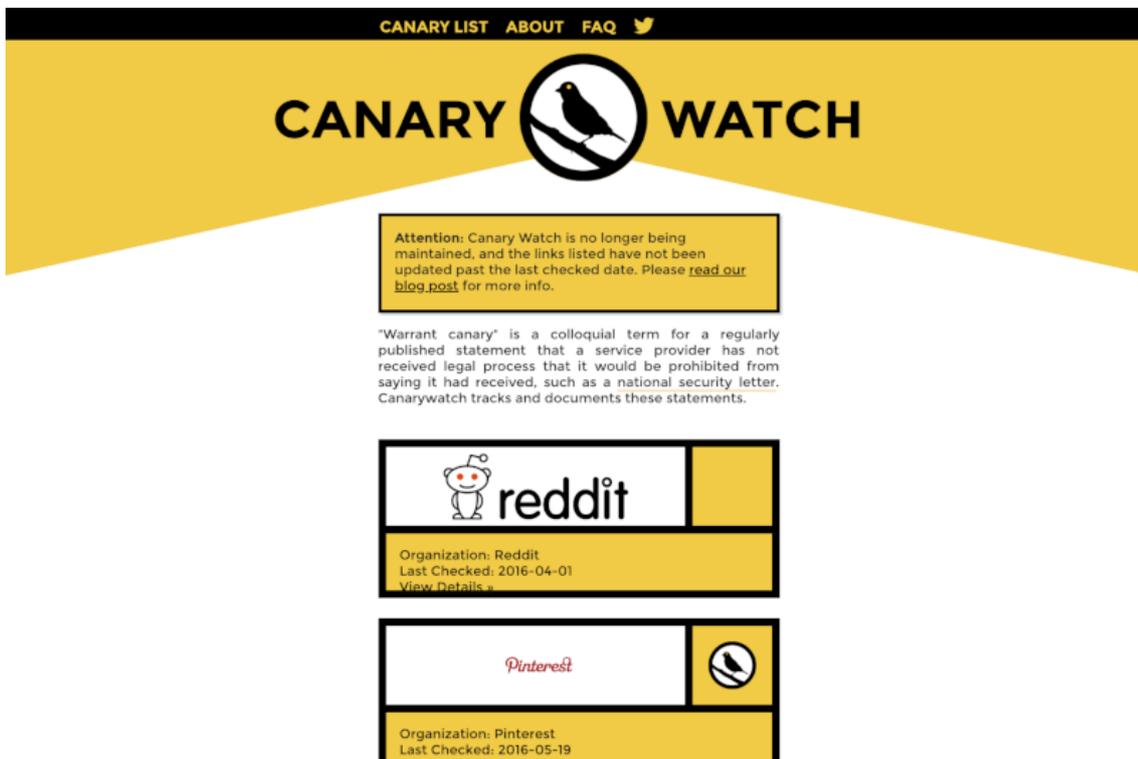
Canary Tokens

Like Agloe, canary tokens can be a piece of information that allow you to implant traps in your systems. When an unsuspecting hacker or malicious user comes along and interacts with the canary you get a notification. It's important to note though that for canary tokens to be useful to you they should be enticing to an attacker - you want them to find them and trigger them, tokens should also be detectable, if their usage goes unnoticed because of a lack of logging and monitoring canaries lose all value. Finally, canary tokens should have no value other than detecting their use - if there is any meaningful value to the token if an attacker gets hold of one it potentially spells game over.

This technology is not new, and canaries and tripwires have also been discussed at length by others.

- In 1994 Eugene H. Spafford & Gene H. Kim from Purdue University wrote a paper on [Using Integrity Checkers for Intrusion Detection](#).
- In 2003 Lance Spitzner from Symantec wrote about [Canary Tokens](#)

Additionally, a number of canary and canary like solutions exist, [Canary Watch](#) was developed by a group of organisations who helped monitor organisations published statements about receiving legal process that it would otherwise be prohibited from saying it had received.



Today though, I'll be writing about [Canary Tokens](#) created by Thinkst Applied Research.

Why Canary Tokens?

For small-to-medium businesses canary tokens are cost effective, leveraging existing logging and monitoring infrastructure as well as providing cross-platform tokens. Additionally, when deployed correctly canaries have a low signal to noise ratio, meaning that unlike canaries in coal mines, you shouldn't hear a peep from them unless someone has forgotten their security training, or something has gone wrong and you have an intruder on your network.

The downside to this though is they only provide a microscopic view of what is going on, so just because you have deployed some tokens throughout the system, it doesn't mean other security logging and monitoring can be disabled or ignored.

The reality is to get the most value out of canary tokens, they are best used in conjunction with other security best practices, as I mentioned previously, if you are not logging and alerting, canaries will provide you no benefit.

Additionally, if you are running SlackOps you will be disappointed to find out that the Thinkst solution does not currently support Slack Web Hooks.

Canary Tokens by Thinkst

As of August 2018, Canary Tokens offers 15 types of tokens including:

- **Web bug / URL token** - Alert when a URL is visited
- **DNS token** - Alert when a hostname is requested
- **Unique email address** - Alert when an email is sent to a unique address
- **Custom Image Web bug** - Alert when an image you uploaded is viewed
- **Microsoft Word Document** - Get alerted when a document is open in Microsoft Word
- **Acrobat Reader PDF Document** - Get Alerted when a PDF document is opened in Acrobat Reader
- **Windows Folder** - Be notified when a Windows Folder is browsed in Windows Explorer
- **Custom exe / binary** - Fire an alert when an EXE or DLL is executed
- **Cloned Website** - Trigger and alert when your website is cloned
- **SQL server** - Get alerted when MS SQL Server databases are accessed
- **QR code** - Generate a QR code for physical tokens
- **SVN** - Alert when someone checks out an SVN repository
- **AWS Keys** - Alert when AWS keys are used
- **Fast Redirect** - Alert when a URL is visited (user is redirected)
- **Slow Redirect** - Alert when a URL is visited (user is redirected, and fingerprinting is done)

While they do not currently have a token for Git based version control, they tweeted on August 24, 2018 that they are working on a newer implementation:



Following

Replying to @teknetia

We used to have a git token and recently retired it. We are working on a newer one. Watch this space!

(Also: thanks for using our stuff!)

5:21 PM - 24 Aug 2018

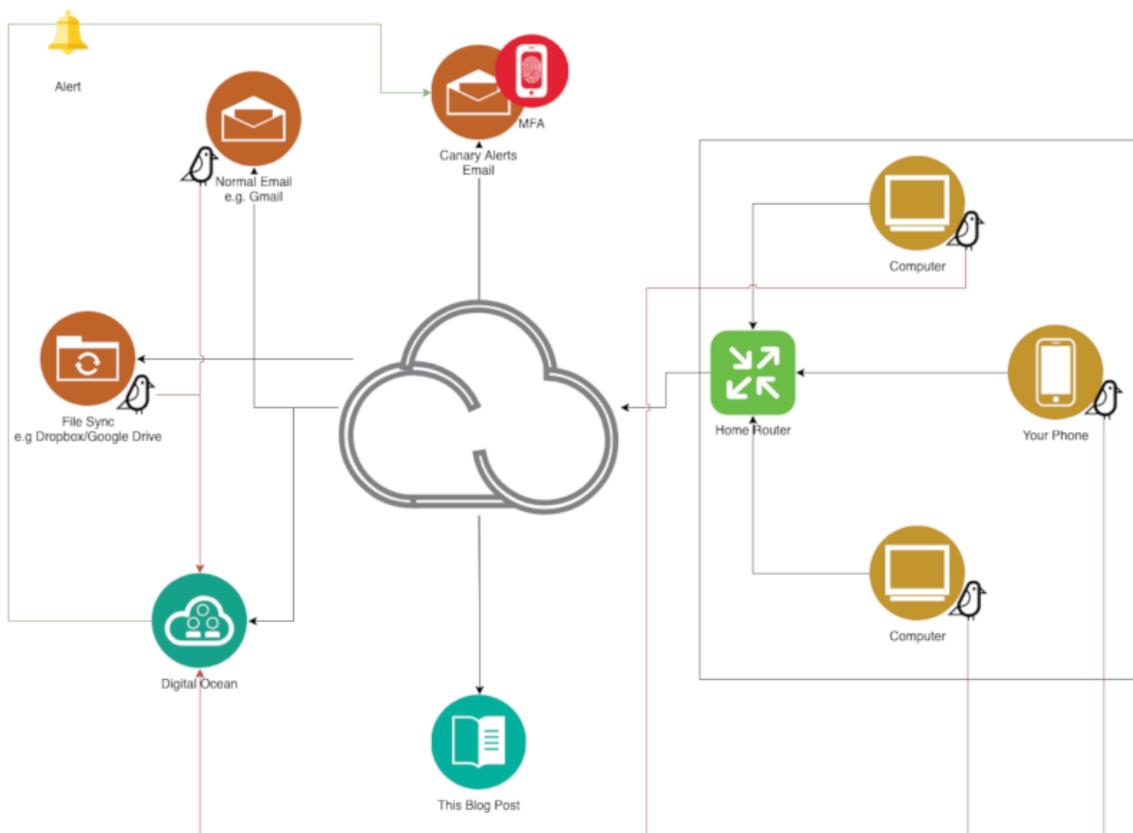
1 Retweet



How can we deploy Canary Tokens?

How can you deploy canaries you may ask? And I will reply "That is a great question reader! It's so great I drew a little picture!" now of course your network is going to be a little different. So my example today is going to be a small flat network with a mobile device, a couple of laptops, externally hosted mail (like Gmail) and externally hosted file hosting (like Dropbox or Google Drive). The proof is left as an exercise to the reader.

I preface this diagram by saying, I am not good at network diagrams.



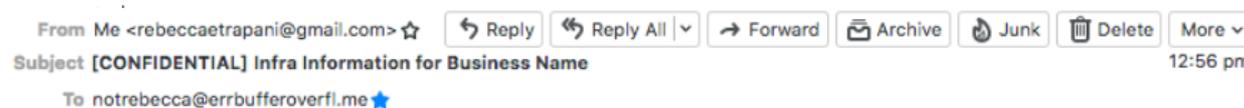
The idea is that everything in some way or another is connected to the Internet and when the canary is triggered it will call back to the Digital Ocean droplet and send you an email alerting you to the breach.

It's hard to give a "one-size-fits-all" response to what sort of tokens you can deploy in this sort of network configuration because it *depends*. Instead I'll endeavour to give a "one-size-fits-a-few", you'll notice I've put a little canary symbol next to the locations I think would make good locations for a/many canary tokens:

Web Bug/URL

In both of my talks I have found this to be the best example to give, for placing canaries in e-mail/file syncs/ mobile phones, basically anywhere. Web Bugs and URLs are versatile and easy to deploy no matter what you do or how you do it.

An example of what a Web Bug/URL canary might look like:



Hi Rebecca,

Below is a link to view the infra information.
Please be aware of the data classification.

[Link to View Infra Information](#)

Regards,

Not Rebecca

Simple, elegant, schön.

E-Mail addresses

Like Web Bugs, e-mail addresses are versatile little buggers. You can save them in your contacts on your phone, or on your computer and if they ever get e-mailed you know something has gone wrong and someone is sneaking through your e-mails or your contacts.

QR Codes

Normally I like to give the example of putting a QR code inside a phone case - like in the actual phone, but with modern manufacturing and the way manufacturers now approach right to repair, by making phone as impenetrable as Fort Knox this becomes a little harder - but placing QR codes inside phone covers or other sneaky locations - wallets, stuck to the bottom of your laptop - you provide a little tantalising "I'm not what you were expecting... scan me!".



<script async

src="//embedr.flickr.com/assets/client-code.js" charset="utf-8"></script>

Amazon Web Services (AWS) Tokens

I feel like when ever I mention AWS Tokens I always have to do a shout-out to PROJECT SPACECRAB by Atlassian - because this is really a great tool that allows you to easy deploy AWS tokens.

The first line of the README says it all if you ask me:

Bootstraps an AWS account with everything you need to generate, manage, and distribute and alert on AWS honey tokens. Made with breakfast roti by the Atlassian security team.



PROJECT SPACECRAB by Atlassian

<https://bitbucket.org/asecurityteam/spacecrab>

PROJECT SPACECRAB is available for download over at [BitBucket](https://bitbucket.org/asecurityteam/spacecrab).

Like PROJECT SPACECRAB Thinkst generates new user tokens with a deny-all policy. The only action these tokens can perform is the 'sts get-caller-identity' action, which you literally cannot stop a token from doing. You can then hide these keys away in your `~/.aws/config`.

BACK TO IT

Now depending on your configuration and whether you have the facilities to set up an external monitoring service (like Pager Duty) I'm going to presume that you are using e-mail. In which case I'd suggest setting up a separate account that isn't tied to your everyday account. Maybe something through [Protonmail](https://protonmail.com).

Now because I am presuming, you're going to have a pretty standard router, I'd suggest looking into Digital Oceans itty bitty droplets using their One-Click apps to help quickly deploy Docker. They are spec'd as follows (as of the 12 November 2018):

- 1 GB Memory
- 1vCPU
- 25GB SSD Disk
- 1TB Transfer
- Cost \$5.00/month (~\$0.007/hr)

However, if you're confident in your networking capabilities and your router, you could always set it up on your perimeter...

Installing Dockerized Canarytokens

For the tl;dr version see the [thinkst/canarytokens-docker README.md](https://github.com/thinkst/canarytokens-docker).

Pre-Installation

Before getting started you will need at least one domain name, if you want to enable PDF opening tracking you will need at least two.

You will also need that Docker host setup.

Since the October 17, 2018 you can also configure an AWS ID token if you want to configure AWS canaries.

Getting Started

DNS QuickStart

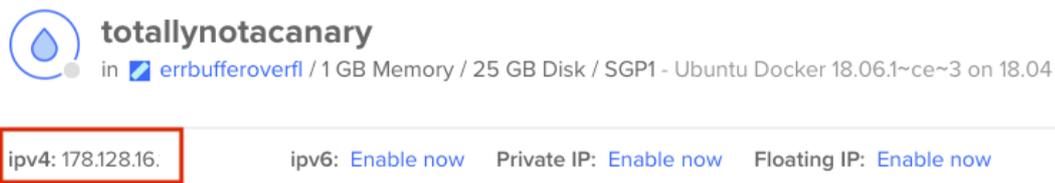
I'm not going to go into too much detail about how to get DNS configured with Digital Ocean, 1. Because it's not really the point of this blog post, 2. DNS is terrible and 3. It took me many months to work out why DNS wasn't working for me.

The first thing you will want to do is configure your DNS registrar to point to DigitalOceans name servers. This is different for every registrar, however, Kathleen Juell has written an easy to understand "[How to Point to DigitalOcean Nameservers From Common Domain Registrars](#)" post (Last Updated: February 22, 2018). This covers popular registrars including:

- GoDaddy
- HostGator
- Namecheap
- 1&1
- Name.com
- Network Solutions
- eNom
- Gandi
- Register.com
- A Small Orange
- iwantmyname
- Google Domains beta

You can then follow DigitalOcean's QuickStart guide to [add the domain to the droplet](#).

Once DNS shenanigans are over you can SSH onto your newly created Digital Ocean droplet - you can find the public IP address in the Droplet interface as shown in the screen capture below:



In your root directory clone the repository. The droplet should come with Git pre-installed, if not you can run:

```
$ sudo apt update
$ sudo apt install git
```

You can confirm Git has been installed correctly by running the following command:

```
$ git --version
git version 2.17.1
```

To clone the repository:

```
$ git clone https://github.com/thinkst/canarytokens-docker
$ cd canarytokens-docker
```

If you used the DigitalOcean One-Click apps Docker Compose should already be installed, however if not you can follow the mini-walk through below to get it installed.

Docker Compose is available from the official Ubuntu repositories, however it is several minor versions behind the latest release, so instead we'll install it from GitHub directly.

By using the `-o` flag to specify the output file first rather than redirecting the output, this syntax avoids running into a permission denied error caused when using `sudo`.

As of the 12 November 2018 the latest release was `1.23.1`.

```

$ sudo curl -L https://github.com/docker/compose/releases/download/1.23.1/docker-compose-`uname -s`-`uname -m` -o
/usr/local/bin/docker-compose
# Next set the permissions:
$ sudo chmod +x /usr/local/bin/docker-compose
# Verify that the installation was successful by checking the version:
$ docker-compose --version
docker-compose version 1.23.1, build xxx

```

There are two main configuration files that can be found in the root directory:

- **switchboard.env** - contain configuration options for MailGun, Mandrill, SendGrid, public IP address, public domain name, email configuration details and logging location.
- **frontend.env** - contains configuration options for canary domains, Google API key for location information, image upload path, and logging location.

Configure the domains you will be using in the frontend.env first, an example is shown below:

```

#These domains are used for general purpose tokens
CANARY_DOMAINS=example1.com, example2.com

#These domains are only used for PDF tokens
CANARY_NXDOMAINS=example3.com

#Requires a Google API key to generate incident map
#CANARY_GOOGLE_API_KEY=

```

In the `switchboard.env` file, uncomment the `CANARY_PUBLIC_DOMAIN` and set it to one of the domains defined previously for the `CANARY_DOMAINS` in the `frontend.env` file. Keep in mind this will be the domain you will access the Canary Token Generation function from.

Additionally, if you **do not** uncomment and set this value the **Public IP** value will be used instead.

If you are using Mailgun to send emails, you will need to uncomment `CANARY_MAILGUN_DOMAIN_NAME` and `CANARY_MAILGUN_API_KEY` in the `switchboard.env` file and set the values.

If you are using Mandrill or Sendgrid instead, uncomment the appropriate API key setting and set it. An example is shown below. * You can generate tokens on `example1.com`, `example2.com` and `example3.com` (for PDF tokens) * This example would run on the public domain `my.domain.com` with a public IP of `1.1.1.1` * Mail is handled using the Mailgun domain name `mail.mailhost.com` and API key `mymailgunapikey`

Finally, initiate and download the images:

```
$ docker-compose up
```

During previous installations I have received a `502: Bad Gateway` error when attempting to fetch step 7, typically running Docker compose a few times gets the solution back on its way.

```

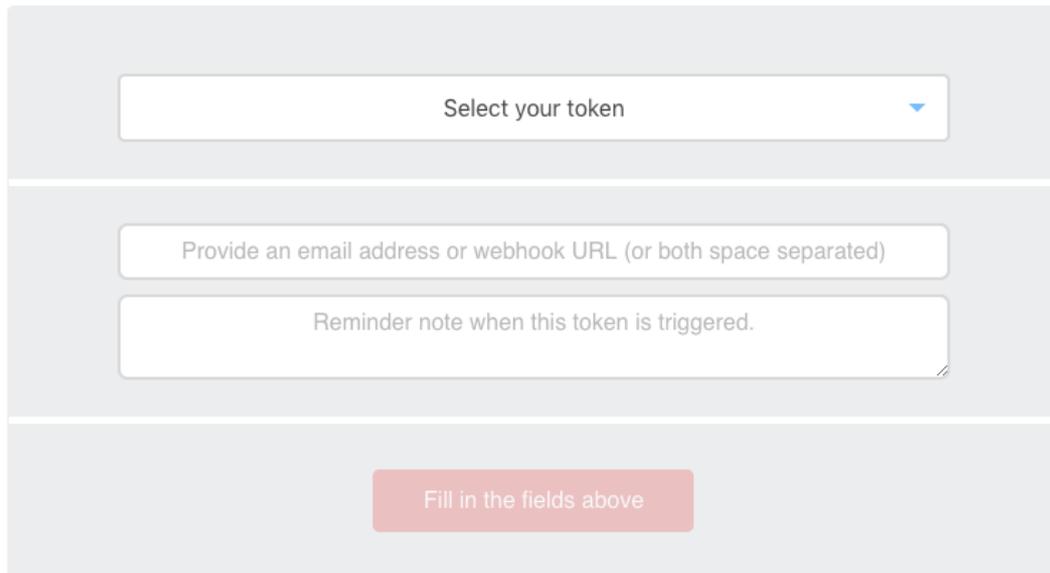
--2018-08-17 02:17:08-- https://github.com/thinkst/canarytokens/archive/master.zip?step=7
Resolving github.com (github.com)... 13.250.177.223, 13.229.188.59, 52.74.223.119
Connecting to github.com (github.com)|13.250.177.223|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/thinkst/canarytokens/zip/master [following]
--2018-08-17 02:17:09-- https://codeload.github.com/thinkst/canarytokens/zip/master
Resolving codeload.github.com (codeload.github.com)... 54.251.140.56, 13.229.189.0
Connecting to codeload.github.com (codeload.github.com)|54.251.140.56|:443... connected.
HTTP request sent, awaiting response... 502 Bad Gateway
2018-08-17 02:17:09 ERROR 502: Bad Gateway.

```

The frontend and switchboard will now be running in the foreground. The frontend is accessible at `http://{{ CANARY_PUBLIC_DOMAIN }}.com/generate`

Canarytokens by Thinkst

What is this and why should I care?



Select your token

Provide an email address or webhook URL (or both space separated)

Reminder note when this token is triggered.

Fill in the fields above

Brought to you by [Thinkst Canary](#), our insanely easy-to-use honeypot solution that deploys in just four minutes. **Know. When it matters.**

© [Thinkst Applied Research](#) 2015–2018

This [Canarytokens](#) installation is unaffiliated with Thinkst Applied Research.

Persisting Data

The tokens are saved in a Redis database file which exists outside of the Docker containers.

In the `canarytokens-docker/data` directory you should find a `dump.rdb` database file. To dump all existing token delete `dump.rdb`.

Enabling HTTPS

There is a separate `docker-compose` file which will mostly automate getting you up and running with HTTPS. You will need to do the following:

Edit the `certbot.env`. You will need to provide your domain and email address (these are necessary for the certbot's registration process). as shown in the example below:

```
MY_DOMAIN_NAME=example.com
EMAIL_ADDRESS=jay@example.com
```

Now when you want to bring up your server, you will use:

```
$ docker-compose -f docker-compose-letsencrypt.yml up
```

Which will run the server in the foreground, so you can make sure everything gets started alright.

If everything is running, you may want to CTRL+C and run the following:

```
run docker-compose -f docker-compose-letsencrypt.yml down
```

to get to a clean slate and then rerun

```
docker-compose -f docker-compose-letsencrypt.yml up -d
```

`-d` will run the server in daemon mode.

Please keep in mind that using the HTTPS method will use the email you specified and the domain name to register the certificate. You can read about the Let's Encrypt process (using `certbot`) over here. The process involves verifying that you are the owner of the domain you have specified and registering you with Let's Encrypt.

It's important to remember there is a rate-limit so try not to bring the server up and down quickly otherwise you will hit the Let's Encrypt certificate generation limit.

While you are testing the installation, you can add:

```
--staging to the ./certbot-auto command
```

command in the `certbot-nginx/start.sh` which will test whether Let's Encrypt gives you the certificate.

Post-Installation

In some scenarios it may be required to host the Canary Tokens server on the Internet, in other cases you may want it to be inaccessible to certain users even when run internally. An easy way for this to be achieved is to modify the `nginx.conf`. This can be done before you run `docker-compose` or it can be done when the Docker container is running, however, these changes won't persist if the docker container is removed.

Disabling `/generate`

1). You will want to edit the `nginx.conf`, which can be found in `/canarytokens-docker/nginx/`. Using `vim`:

```
$ vim /canarytokens-docker/nginx/nginx.conf
```

2). You will be presented with a `nginx.conf` that will look something like the example below, which has been clipped for brevity.

nginx.conf

```
...
# Proxying connections to application servers
location / {
    proxy_pass          http://frontend:8082/;
    proxy_redirect     off;
    proxy_set_header   Host $host;
    proxy_set_header   X-Real-IP $remote_addr;
    proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header   X-Forwarded-Host $server_name;
}

location /generate {
    proxy_pass          http://frontend:8082/generate;
    proxy_redirect     off;
    proxy_set_header   Host $host;
    proxy_set_header   X-Real-IP $remote_addr;
    proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header   X-Forwarded-Host $server_name;
}

location /download {
    proxy_pass          http://frontend:8082/download;
    proxy_redirect     off;
    proxy_set_header   Host $host;
    proxy_set_header   X-Real-IP $remote_addr;
    proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header   X-Forwarded-Host $server_name;
}

location = /robots.txt {
    proxy_pass          http://frontend:8082/robots.txt;
```

```

    proxy_redirect    off;
    proxy_set_header  Host $host;
    proxy_set_header  X-Real-IP $remote_addr;
    proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header  X-Forwarded-Host $server_name;
}

location ^~ /resources {
    proxy_pass         http://frontend:8082/resources;
    proxy_redirect    off;
    proxy_set_header  Host $host;
    proxy_set_header  X-Real-IP $remote_addr;
    proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header  X-Forwarded-Host $server_name;
}
...

```

3). For each location that `proxy_pass` is configured to `frontend:8082` you will want to add two new options `allow` and `deny`. The `allow` option should be configured to the static IP address you wish to allow to view and use the interfaces. `deny` can be configured to `all` if you wish no one except those on the allow list to view the interface, the alternative is to deny only particular static IP addresses, however, I would recommend the latter option. Your configuration may now look something like below, where `1.2.3.4` is your actual static IP:

nginx.conf

```

...
# Proxying connections to application servers
location / {
    proxy_pass         http://frontend:8082/;
    proxy_redirect    off;
    proxy_set_header  Host $host;
    proxy_set_header  X-Real-IP $remote_addr;
    proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header  X-Forwarded-Host $server_name;
    allow             1.2.3.4;
    deny              all;
}

location /generate {
    proxy_pass         http://frontend:8082/generate;
    proxy_redirect    off;
    proxy_set_header  Host $host;
    proxy_set_header  X-Real-IP $remote_addr;
    proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header  X-Forwarded-Host $server_name;
    allow             1.2.3.4;
    deny              all;
}

location /download {
    proxy_pass         http://frontend:8082/download;
    proxy_redirect    off;
    proxy_set_header  Host $host;
    proxy_set_header  X-Real-IP $remote_addr;
    proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header  X-Forwarded-Host $server_name;
    allow             1.2.3.4;
    deny              all;
}

location = /robots.txt {
    proxy_pass         http://frontend:8082/robots.txt;
    proxy_redirect    off;
    proxy_set_header  Host $host;
    proxy_set_header  X-Real-IP $remote_addr;
    proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header  X-Forwarded-Host $server_name;
    allow             1.2.3.4;
    deny              all;
}

location ^~ /resources {
    proxy_pass         http://frontend:8082/resources;
    proxy_redirect    off;
}

```

```

    proxy_set_header    Host $host;
    proxy_set_header    X-Real-IP $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Host $server_name;
    allow                1.2.3.4;
    deny                 all;
}
...

```

4). If you have run `docker-compose up` already you may need to remove the previously built containers by running `docker-compose rm` and then run `docker-compose build`.

5). Finally, once the container is up and running you should now get a `403 Forbidden` error anytime a user tries to access the site. Of course, you can always make this nicer by redirecting to another service etc. but we won't be covering that here.

Pre-Installation

If you already have your Docker Container running you can follow the instructions below to disable the `/generate` interface using Nginx. Keep in mind that if you ever need to remove the Docker containers the changes will not persist and when you run `docker-compose` the next time these settings will not be configured. Therefore this option is better suited to testing your configuration is correct.

1). Before starting you will want to find the container ID or the name of the Docker container you wish to connect to, this can be done by running `docker ps`. The Thinkst solution is broken up into four different components, you will want to ensure you are connecting to the `thinkst/canarytokens_nginx` container:

```

$ docker ps
CONTAINER ID   IMAGE                                COMMAND                                  CREATED        STATUS        PORTS
NAMES
a67e3314ec1d   thinkst/canarytokens_nginx          "/usr/sbin/nginx -c ..."             7 days ago    Up 7 days
0.0.0.0:80->80/tcp
4449440dd899   thinkst/canarytokens               "bash -c 'rm switchb..."             7 days ago    Up 7 days
0.0.0.0:25->25/tcp, 0.0.0.0:53->53/tcp, 0.0.0.0:53->53/udp
04195bdf162   thinkst/canarytokens               "bash -c 'rm fronten..."             7 days ago    Up 7 days
frontend
c3abeae4e9e1   redis                                "docker-entrypoint.s..."             7 days ago    Up 7 days    6379/tcp
redis

```

2). `docker exec` will let you run arbitrary commands inside an existing container. We will want to connect to the `thinkst/canarytokens_nginx` this can be done by running the following command, ensuring to replace `a67e3314ec1d` with the ID of your container:

```
$ docker exec -it a67e3314ec1d bash
```

- `-i`, `--interactive[=false]` : tells Docker to keep `STDIN` open even if not attached. This is required if you want to view information that is provided by `STDIN` - for example when installing packages.
- `-t`, `--tty[=false]` : tells Docker to allocate a pseudo-TTY.

3). You should be dropped into a session on the container, however, there is currently no way to modify the Nginx file, so you will need to install `vim` in the container. This can be done by just running `apt`, as follows:

```
$ apt-get update
$ apt-get install vim
```

4). Once you have a text editor installed, you will want to edit the `nginx.conf`, which can be found in `/etc/nginx/`. Using `vim`:

```
$ vim /etc/nginx/nginx.conf
```

5). You will be presented with a `nginx.conf` that will look something like the example below, which has been clipped for brevity.

nginx.conf

```

...
# Proxying connections to application servers
location / {

```

```

    proxy_pass      http://frontend:8082/;
    proxy_redirect  off;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Host $server_name;
}

location /generate {
    proxy_pass      http://frontend:8082/generate;
    proxy_redirect  off;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Host $server_name;
}

location /download {
    proxy_pass      http://frontend:8082/download;
    proxy_redirect  off;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Host $server_name;
}

location = /robots.txt {
    proxy_pass      http://frontend:8082/robots.txt;
    proxy_redirect  off;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Host $server_name;
}

location ^~ /resources {
    proxy_pass      http://frontend:8082/resources;
    proxy_redirect  off;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Host $server_name;
}
...

```

6). For each location that `proxy_pass` is configured to `frontend:8082` you will want to add two new options `allow` and `deny`. The `allow` option should be configured to the static IP address you wish to allow to view and use the interfaces. `deny` can be configured to `all` if you wish no one except those on the allow list to view the interface, the alternative is to deny only particular static IP addresses, however, I would recommend the latter option. Your configuration may now look something like below, where `1.2.3.4` is your actual static IP:

nginx.conf

```

...
# Proxying connections to application servers
location / {
    proxy_pass      http://frontend:8082/;
    proxy_redirect  off;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Host $server_name;
    allow           1.2.3.4;
    deny            all;
}

location /generate {
    proxy_pass      http://frontend:8082/generate;
    proxy_redirect  off;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Host $server_name;
    allow           1.2.3.4;
}

```

```

    deny                all;
}

location /download {
    proxy_pass          http://frontend:8082/download;
    proxy_redirect      off;
    proxy_set_header    Host $host;
    proxy_set_header    X-Real-IP $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Host $server_name;
    allow               1.2.3.4;
    deny                all;
}

location = /robots.txt {
    proxy_pass          http://frontend:8082/robots.txt;
    proxy_redirect      off;
    proxy_set_header    Host $host;
    proxy_set_header    X-Real-IP $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Host $server_name;
    allow               1.2.3.4;
    deny                all;
}

location ^~ /resources {
    proxy_pass          http://frontend:8082/resources;
    proxy_redirect      off;
    proxy_set_header    Host $host;
    proxy_set_header    X-Real-IP $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Host $server_name;
    allow               1.2.3.4;
    deny                all;
}
...

```

7). Finally, restart the Nginx web-server by running

```
$ /etc/init.d/nginx reload
```

8). You should now get a `403 Forbidden` error anytime a user tries to access the site. Of course you can always make this nicer by redirecting to another service etc. but we won't be covering that here, however, [nixCraft](#) has a reasonably good tutorial on [Creating Custom 404 / 403 Error Pages on Linux or Unix](#).



Further Improvements

One issue I often notice while testing websites is unnecessary server headers in HTTP responses, unless there is a business requirement for these details, they should ultimately be removed as they can provide helpful information to attackers about the software used by your infrastructure.

The important thing to note is whether you select option one or option two, there is always more than one way to fingerprint a server, so removing the Server version number doesn't mean people won't be able to fingerprint the server, it just means they will need to do OTHER things to fingerprint it.

Also note that if you decide to set a bogus server header, the default Nginx 403 page will still tell people it is Nginx.

You will first need to add an entry to the `canarytokens-Docker\nginx\ Dockerfile` entry like so:

Dockerfile

```
RUN apt-get -y update \
&& apt-get -q -y -o "DPkg::Options::=--force-confold" -o "DPkg::Options::=--force-confdef" install nginx-extras
```

I replaced the original `apt-get update` commands as I found this much easier to read.

You will note the `-o "DPkg::Options::=--force-confold" -o "DPkg::Options::=--force-confdef"` this is added because typically when `nginx-extras` is installed you are asked if you wish to override your original configuration file. Originally, when I was working out the details of this, it was something I kept getting caught up on because `docker-compse build` kept failing with the status code `100`. Adding this option will install `nginx-extras` with as the flag suggests configuration defaults - which if you check is to not replace the configuration file. So keep this in mind if you plan on rolling in any other packages.

You will also need to make some additional changes to your `nginx.conf` file, if you just want the server header to return `nginx` with no version number you just need to add the following line to the `Dockerfile` like so:

nginx.conf

```
...
http {
    server_tokens    off;
    ...
}
```

If you wish to remove the server header completely, or replace it with something more humorous you will need to make the following changes:

nginx.conf

```
# Nginx 1.9.11 introduced dynamic modules, which allow modules to be loaded and unloaded as necessary.
# This means that in the Docker container the module is compiled, but not loaded.
# If you don't add this line you Nginx will fail to reload. Don't say I didn't warn you.

# You can put this as the second or third line in the config.
load_module modules/nginx_http_headers_more_filter_module.so;

...

http {
    more_clear_headers Server;
    # Additionally, if you want to annoy those pesky hackers
    # you can also add in a fake Server entry like so
    add_header 'Server' 'lol sorry pal' always;

    ...
}
```

Wrapping Up

During OWASP AppSec day a few important things were brought up during the Q+A that don't really fit into the blog post but I think are important to address:

- Some Gmail users have reported that PDF attachments in e-mails caused alerts about once a month, reported IP address came from within Google's IP address range. Now before you put on your tin-foil hat and claim Google are reading your e-mails this can probably be attributed as a number of things including Google scanning for malicious attachments in PDF files.
- @synick informed us that "Next Generation Anti-Virus" software (like CrowdStrike) can detect canaries - take from that what you will.
- I had a lot of questions about managing canaries in an ~~large~~ organisation - and it's an interesting and difficult question about what you are trying to achieve. If you are trying to find an insider threat - educating users on canaries and their locations is probably not the best idea. But if they are there as a general stop the baddies measure - educating your users about what canaries are and potentially their placement is not a horrible idea. Keep in mind it might be worth documenting their locations, however it's important to remember if a Bad Hacker (tm) comes across this sheet they can use this information to their benefit - so manage these records in the same way you would secrets.

Future Developments:

- Google Maps API integration
- In-depth AWS Tokens
- Changing `CANARY_TOKEN_RETURN=fortune`

Slide Deck

[You can download my updated OWASP Melbourne AppSec day slide deck from my website \(that you are currently viewing\).](#)

I think that's everything (at least for now)! I hope you enjoyed this blog post~

Signing out, errbufferoverfl.